8164950

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

October 06, 2021

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: ***11/477,807***
FILING DATE: ***June 30, 2006***
PATENT NUMBER: ***8418250***
ISSUE DATE: ***April 09, 2013***

IPR2023-00556
Open Text Inc. EX-2019
Page 1 of 1187

Certified by

Performing the Functions and Duties of the
Under Secretary of Commerce
for Intellectual Property
and Director of the United States
Patent and Trademark Office

PTO/SB/06 (12-04)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD
Substitute for Form PTO-875

Application or Docket Number: 10/647,807

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	300
SEARCH FEE (37 CFR 1.16(k), (l), or (m))	N/A	N/A	N/A			N/A	500
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	200
TOTAL CLAIMS (37 CFR 1.16(i))	9 minus 20 =		x =		OR	x =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	1 minus 3 =		x =			x =	
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))			N/A			N/A	
			TOTAL			TOTAL	1,000

* If the difference in column 1 is less than zero, enter "0" in column 2.

APPLICATION AS AMENDED – PART II

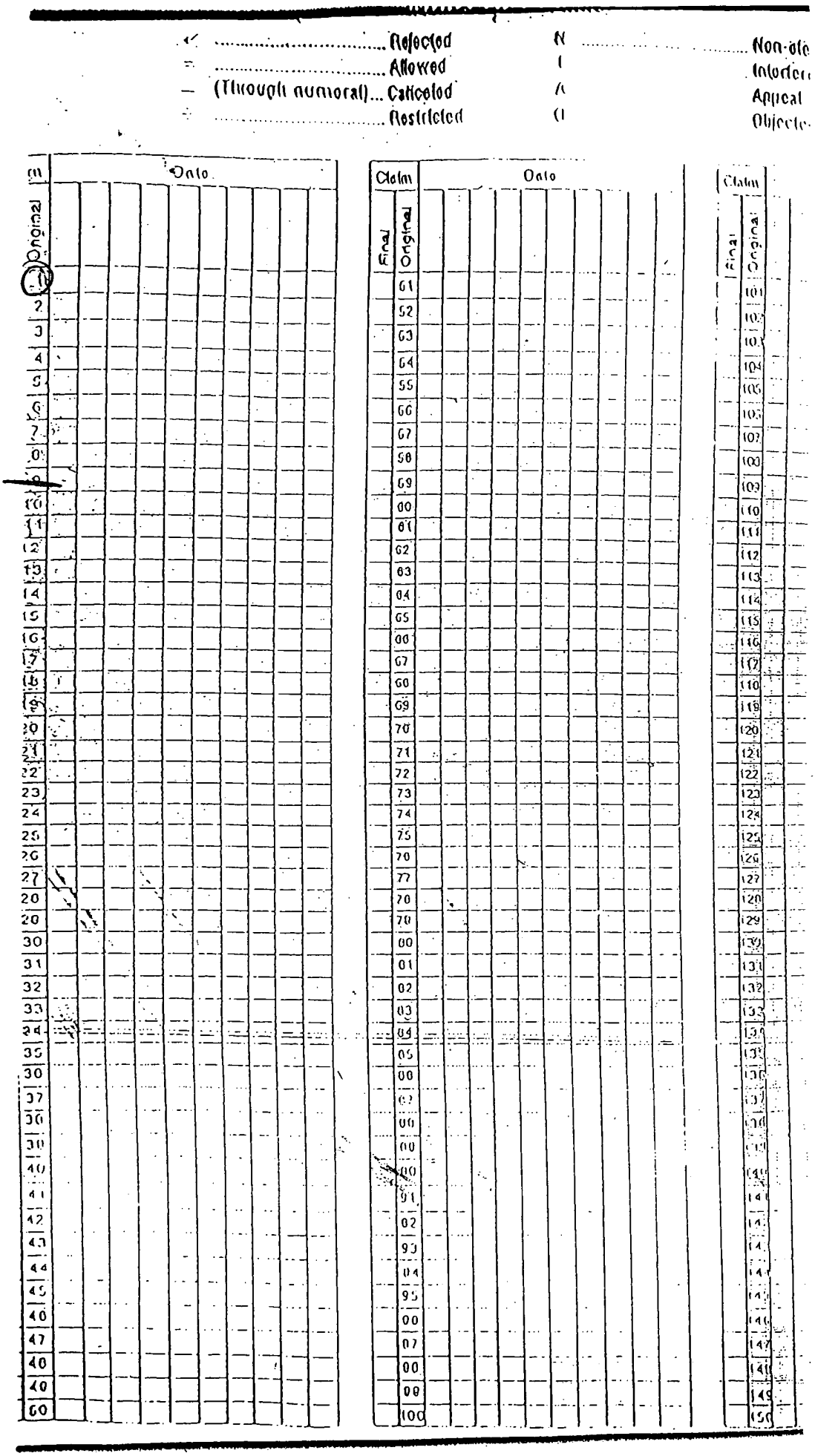
		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
AMENDMENT A	Total (37 CFR 1.16(i))	* 9	Minus	** 20	= 0	x =		OR	x =	
	Independent (37 CFR 1.16(h))	* 1	Minus	*** 3	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)	OR	RATE ($)	ADDITIONAL FEE ($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to comp including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comm on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. P and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.


R ..... Rejected
A ..... Allowed
C (Through numeral) ... Cancelled
P ..... Restricted
N ..... Non-ele
I Interfer
A Appeal
O Objecte
Date
Claim
Original
Final
Date
Claim
Original
Final

17712 U.S.PTO
063006

PTO/SB/05 (09-04)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office. U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No.	011765-0355986
First Inventor	MELVYN MORRISj
Title	See attached addendum
Express Mail Label No.	

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. ☐ **Fee Transmittal Form** (e.g., PTO/SB/17) *(Submit an original and a duplicate for fee processing)*
2. ☐ **Applicant claims small entity status.** See 37 CFR 1.27.
3. ☒ **Specification** [*Total Pages* 58] Both the claims and abstract must start on a new page *(For information on the preferred arrangement, see MPEP 608.01 (a))*
4. ☒ Drawing(s) (*35 U.S.C. 113*) [*Total Sheets* 3]
5. **Oath or Declaration** [*Total Sheets*]
   a. ☐ Newly executed (original or copy)
   b. ☐ Copy from a prior application (37 CFR 1.63(d)) *(for continuation/divisional with Box 18 completed)*
      i. ☐ DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) name in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
6. ☒ **Application Data Sheet.** See 37 CFR 1.76
7. ☐ **CD-ROM or CD-R** in duplicate, large table or Computer Program *(Appendix)*
   ☐ Landscape table on CD
8. **Nucleotide and/or Amino Acid Sequence Submission** *(if applicable, all necessary)*
   a. ☐ Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. ☐ CD-ROM or CD-R (2 copies); or
      ii. ☐ Paper
   c. ☐ Statements verifying identity of above copies

ADDRESS TO: **Commissioner for Patents, P.O. Box 1450, Alexandria VA 22313-1450**

113261 U.S.PTO
11/477807
063006

ACCOMPANYING APPLICATION PARTS

9. ☐ **Assignment Papers** (cover sheet & document(s)) Name of Assignee
10. ☐ **37 CFR 3.73(b) Statement** *(when there is an assignee)* ☐ Power of Attorney
11. ☐ **English Translation Document** *(if applicable)*
12. ☐ **Information Disclosure Statement** (PTO/SB/08 or PTO-1449 ☐ Copies of Citations attached
13. ☐ **Preliminary Amendment**
14. ☐ **Return Receipt Postcard** (MPEP 503) *(Should be specifically itemized)*
15. ☐ **Certified Copy of Priority Document(s)** *(if foreign priority is claimed)*
16. ☐ **Nonpublication Request** under 35 U.S.C. 122 b)(2)(B)(i). Applicant must attach form PTO/SB/35 or equivalent.
17. ☐ Other: .....

18. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in the first sentence of the specification following the title, or in an Application Data Sheet under 37 CFR 1.76:*

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.: .....

Prior application information: *Examiner* _____ *Art Unit:* _____

19. CORRESPONDENCE ADDRESS

☒ The address associated with Customer Number: 00909 *OR* ☐ Correspondence address below

Name					
Address					
City		*State*		*Zip Code*	
Country		*Telephone*		*Fax*	

Signature		*Date*	June 30, 2006
Name (Print/Type)	Christine H. McCarthy	*Registration No. (Attorney/Agent)*	41844

This collection of information is required by 37 CFR 1.53(b). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Addendum

Invention Title

METHODS AND APPARATUS FOR DEALING WITH MALWARE

METHODS AND APPARATUS FOR DEALING WITH MALWARE

The present invention relates generally to methods and apparatus for dealing with malware. In one aspect, the present invention relates to a method and apparatus for classifying a computer object as malware. In another aspect, the present invention relates to a method and apparatus for determining the protection that a remote computer has from malware. In another aspect, the present invention relates to a method and apparatus for classifying a computer object as malware or as safe. In another aspect, the present invention relates to a method of installing software on a computer.

The term "malware" is used herein to refer generally to any executable computer file or, more generally "object", that is or contains malicious code, and thus includes viruses, Trojans, worms, spyware, adware, etc. and the like.

A typical anti- malware product, such as anti-virus scanning software, scans objects or the results of an algorithm applied to the object or part thereof to look for signatures in the object that are known to be indicative of the presence of a virus. Generally, the method of dealing with malware is that when new types of malware are released, for example via the Internet, these are eventually detected. Once new items of malware have been detected, then the service providers in the field generate signatures that attempt to deal with these and these signatures are then released as updates to their anti-

malware programs. Heuristic methods have also been employed.

These systems work well for protecting against known malicious objects. However, since they rely on signature files being generated and/or updated, there is inevitably a delay between a new piece of malware coming into existence or being released and the signatures for combating that malware being generated or updated and supplied to users. Thus, users are at risk from new malware for a certain period of time which might be up to a week or even more. Moreover, in order to try to defeat anti-virus products, the malware writers use obfuscation techniques in order to attempt to hide the signature or signature base data of the virus code from detection. Typically, the obfuscation involves encrypting or packing the viral code.

WO-A-2004/097602 describes a system that analyses computer files received or generated by a local computer and compares these with a database of known files to determine whether a particular file is known and if so whether it has been known about long enough that it can be regarded as "safe". However, in practice, on its own this is not likely to provide for adequate protection because, for example, the active payload of a virus or Trojan may only be programmed to activate at a particular date, or upon receiving a message or instruction from a local or remote system or process, or on the occurrence of a particular event that may be many months or even years after the process has been first run or is released. Thus, just looking at the age of a file is an unsatisfactory way of determining whether it is properly safe and will remain so.

In the system of WO-A-2004/0083408, a worm in a file is detected by examining connection attempts made by the specific file running on a computer.

US-A-6944772, US-A-6772346, EP-A-1549012 and EP-A-1280040 all disclose "community-based" anti-malware systems in which a plurality of "local" computers all connect via a network (which may be a LAN or the Internet, for example) to a central computer. On encountering a file that is not already known to them, the local computers send a request to the central computer for authorisation to run the file. If the file is recognised at the central computer, then the central computer can send permission for the local computer to run the file if the file is known to be safe or send a "deny" command if the file is known to be malicious. However, in each of these prior art proposals, if the file is not known at the central computer, then the whole file is sent to the central computer where it can be analysed to determine whether it should be regarded as safe or malware. Such analysis is typically carried out manually or "semi-manually" by subjecting the file to detailed analysis, for example by emulation or interpretation, which can still take days given the human involvement that is typically required. There is therefore still a considerable period of time before a new file is classified as safe or as malware. In the case of these prior art systems, the request for authorisation to run the file that is sent by a local computer to the central computer may comprise sending a checksum or "signature" or "key" that uniquely represents the file.

A similar community-based anti-malware system is disclosed in WO-A-02/33525. In this system, in the case that a local computer is seeking clearance to run a file that is not known by the central computer to be safe or malware, some limited audit information about the prevalence of the file on other local computers can be sent to a human system administrator associated with the local computer that is seeking permission to run the file. The human system administrator can therefore make a better informed though still "manual" decision as to whether or not the file is safe to run.

In the system of US-A-2004/0073810, a metafile containing data about an attachment or other transmitted file is sent to a central computer. The data about that file is analysed to determine a likelihood of the transmitted file being malware. A specific example given is that if the transmitted file has been transmitted via at least a certain number of servers, then it should be treated as malware.

In the systems disclosed in US-A-2005/0021994 and US-A-2004/0153644, pre-approved files, which may be certified as safe by for example the software vendor associated with the files, may be permitted always to run without further checking. In one embodiment of the system of US-A-2004/0153644, monitoring is carried out to decide that a file is malicious if an abnormally high number of requests by that file is received at a central authority from plural local computers in a time period or if an abnormally high number of requests by that file on a single computer is received from the single local computer in a time period.

In the system of US-A-2004/0006704, a comparison is made between installed versions of software on a computer with a database of software versions and their known vulnerabilities. A user of the computer can therefore be informed of specific risks and how to minimise those risks by updating existing or installing new software.

In the system of WO-A-03/021402, a central database holds a virtual image of all files stored on each of plural local computers. If a threat in one local computer is identified, other local computers with a similar configuration can be notified of the risk.

Thus, the prior art systems either rely on deep analysis of a new object in order to determine whether or not the object is malicious, which introduces delay and therefore risk to users during the period that the file is analysed and new anti-malware signatures distributed, or limited analysis of the operation of the particular object or its method of transmission to a computer is carried out to decide a likelihood of the object being malicious.

According to a first aspect of the present invention, there is provided a method of classifying a computer object as malware, the method comprising:

at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored;

comparing in the base computer the data about the computer object received from the plural computers; and,

classifying the computer object as malware on the basis of said comparison.

Compared to the prior art that relies solely on signature matching, this aspect allows a comparison to be made between the objects and/or their effects on the different remote computers to determine whether or not a particular object should be classed as good or as malware. Sophisticated pattern analysis can be carried out. This allows a rapid determination of the nature of the object to be made, without requiring detailed analysis of the object itself as such to determine whether it malware and also avoids the need to generate new signatures to be used for signature matching as in the conventional prior art anti-virus software.

In a preferred embodiment, the data about the computer object that is sent from the plural remote computers to the base computer and that is used in the comparison includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

Preferably, the comparing identifies relationships between the object and other objects. In an example, this can be used immediately and automatically to mark a child object as bad (or good) if the or a parent or other related object is bad (or good). Thus, if at least one other

object to which said object is related is classed as malware, then the method may comprise classifying said object as malware. Said other objects include the object or similar objects stored on at least some of the remote computers. Said other objects may include other objects that are parent objects or child objects or otherwise process-related objects to said object.

In a most preferred embodiment, the data is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on the respective remote computers. A major advantage of using such a key is that it keeps down the volume of data that needs to be transmitted to the base computer. Given that there may be thousands or even millions of connected remote computers and further given that each may send details about very many objects, this can be an important advantage.

The key preferably has at least one component that represents executable instructions contained within or constituted by the object. This important preferred feature allows a comparison to be made at the base computer of only the executable instructions of the object. This means for example that differently named objects that basically have the same executable instructions, which is often an indicator that the objects are malware, can nevertheless be regarded as the "same" object for this purpose. As another example, a new version of a program may be released which has minor changes compared to a previous version already known to the base computer and which in substance, at least in respect of the executable instructions, can be regarded as being the same as the previous version. In that case, the minor differences can

be ignored and the objects regarded as being the same. Not only is this useful in distinguishing between malware and for example revised versions of previous software, it also keeps down the data transmission and storage requirements because the base computer can inform the remote computers that an apparently new object is for this purpose the same as a previously known object, thus avoiding having the remote computers send full details about the object or the object itself to the base computer.

The key preferably has at least one component that represents data about said object. Said data about said object may include at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

The key preferably has at least one component that represents the physical size of the object.

Where more than one of these components are present in the key, the plural components are preferably severable.

The method may comprise initially classifying an object as not malware, generating a mask for said object that defines acceptable behaviour for the object, and comprising monitoring operation of the object on at least one of the remote computers and reclassifying the object as

malware if the actual monitored behaviour extends beyond that permitted by the mask. This provides an efficient and effective way of monitoring the behaviour of an object that has been classified or regarded as good and allows the object to be reclassified quickly as malware if the behaviour of the object warrants it.

According to a second aspect of the present invention, there is provided apparatus for classifying a computer object as malware, the apparatus comprising:

a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored;

the base computer being constructed and arranged to compare the data about the computer object received from said plural computers; and,

the base computer being constructed and arranged to classify the computer object as malware on the basis of said comparison.

According to a third aspect of the present invention, there is provided a method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers, the method comprising:

providing from a remote computer to a base computer data about a computer object that is stored on the remote computer;

the data including one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the

physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers;

the data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer.

This method, which may be carried out by so-called agent software running on the remote computer, allows for efficient sending of data to the base computer, which minimises data transmission and storage requirements and also permits rapid analysis to be made at the base computer.

The key preferably has at least one component that represents executable instructions contained within or constituted by the object.

The key preferably has at least one component that represents data about said object. Said data about said object may include at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the

object is created, configured or runs on the respective remote computers.

The key preferably has at least one component that represents the physical size of the object.

According to a fourth aspect of the present invention, there is provided a method of determining the protection that a remote computer has from malware, the method comprising:

receiving at a base computer details of all or selected security products operating at a point in time on said remote computer;

receiving similar information from other remote computers connected to the base computer; and,

identifying any malware processes that were not identified by said other remote computers having that particular combination of security products.

In this way, the base computer can be used to obtain information as to whether for example a particular, specific combination of operating system and various security products, including settings and signature files existing at a point in time, renders a particular computer having those products and settings susceptible or vulnerable to any particular malware object. The user can be advised accordingly and for example provided with recommendations for remedying the situation.

The method may therefore comprise providing information to the user of said remote computer that said remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

The details of all or selected security products preferably includes the name of the security products, versions, and loaded signature files.

According to a fifth aspect of the present invention, there is provided apparatus for determining the protection that a remote computer has from malware, the apparatus comprising:

a base computer constructed and arranged to receive computer details of all or selected security products operating at a point in time on said remote computer;

the base computer being constructed and arranged to receive similar information from other remote computers connected to the base computer; and,

the base computer being constructed and arranged to identify any malware processes that were not identified by said other remote computers having that particular combination of security products.

According to a sixth aspect of the present invention, there is provided a method of classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the method comprising:

classifying a first computer object as malware or as safe;

identifying in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and,

classifying said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

This aspect uses the concept of ancestry to enable objects to be marked as malware. For example, any particular process may spawn child processes which are therefore related. The key relating to the first object may be inspected to identify a component that uniquely identifies the first object and that is inherited or otherwise present in the key of a descendant or other related object of the first object.

The method may comprise monitoring activities of said first computer object and reclassifying the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the method further comprising automatically classifying as malware any computer object that has a key in which said unique identifier component is present.

According to a seventh aspect of the present invention, there is provided apparatus for classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the apparatus comprising:

a computer constructed and arranged to classify a first computer object as malware or as safe;

the computer being constructed and arranged to identify in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the

key of a descendant or other related computer object of the first computer object; and,

the computer being constructed and arranged to classify said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

According to an eighth aspect of the present invention, there is provided a method of installing software on a computer, the method comprising:

on initiation of installation of software on a computer, providing a computer-generated prompt on the computer to a user to ascertain whether the user authorises the installation; and,

ceasing the installation if a user authorisation is not received, else:

receiving at the computer the user's authorisation to proceed with the installation;

proceeding with the installation;

obtaining data about computer objects that are created or used during the installation;

storing said data at the local computer.

This provides for security when a user is installing new software and is not for example connected to a base computer having a community database of the type mentioned above. In that case, the method, which may be implemented in agent software running on the local computer, allows the user to permit the installation to proceed whilst at the same time gathering data about the objects (such as processes, new files, etc.) that are created during the installation.

Preferably, the locally stored data is referred to during the installation to ensure that all objects created or used during the installation are part of the installation process, and, if it is found that objects created or used during the installation are not part of the installation process, either or both of: (i) ceasing the installation and (ii) providing a computer-generated prompt on the computer to the user accordingly. This allows the method to ensure that only those objects that are required for the installation are permitted to be created or used and thus avoids unwittingly allowing malware to install (given that malware often creates objects that are not expected as part of a normal installation of new software).

In a preferred embodiment, the method comprises connecting the computer to a community database that is connectable to a plurality of computers, and uploading the stored data to the community database for comparison with similar data provided by other computers.

The method may comprise downloading data about trusted installers to the computer, said data about trusted installers being referred to during the installation such that any objects relating to or created by the trusted installer are automatically authorised to proceed. This facilitates installation of software that is known a priori to be trustworthy.

Said data about trusted installers may be referred to only for a predetermined time period following receipt at the computer of the user's authorisation to proceed with the installation.

The present invention also includes computer programs comprising program instructions for causing a computer to perform any of the methods described above.

Although the embodiments of the invention described with reference to the drawings comprise computer processes performed in computer apparatus and computer apparatus itself, the invention also extends to computer programs, particularly computer programs on or in a carrier, adapted for putting the invention into practice. The program may be in the form of source code, object code, a code intermediate source and object code such as in partially compiled form, or in any other form suitable for use in the implementation of the processes according to the invention. The carrier be any entity or device capable of carrying the program. For example, the carrier may comprise a storage medium, such as a ROM, for example a CD ROM or a semiconductor ROM, or a magnetic recording medium, for example a floppy disk or hard disk. Further, the carrier may be a transmissible carrier such as an electrical or optical signal which may be conveyed via electrical or optical cable or by radio or other means.

Embodiments of the present invention will now be described by way of example with reference to the accompanying drawings, in which:

Figure 1 shows schematically apparatus in which an embodiment of the present invention may be implemented;

Figure 2 is a flowchart showing schematically the operation of an example of a method according to an embodiment of the present invention;

Figure 3 is a flowchart showing schematically the operation of another example of a method according to an embodiment of the present invention; and,

Figure 4 is a flowchart showing schematically an information obtaining stage.

Referring to Figure 1, a computer network is generally shown as being based around a distributed network such as the Internet 1. The present invention may however be implemented across or use other types of network, such as a LAN. Plural local or "remote" computers 2 are connected via the Internet 1 to a "central" or "base" computer 3. The computers 2 may each be variously a personal computer, a server of any type, a PDA, mobile phone, an interactive television, or any other device capable of loading and operating computer objects. An object in this sense may be a computer file, part of a file or a sub-program, macro, web page or any other piece of code to be operated by or on the computer, or any other event whether executed, emulated, simulated or interpreted. An object 4 is shown schematically in the figure and may for example be downloaded to a remote computer 2 via the Internet 1 as shown by lines 5 or applied directly as shown by line 6.

In one preferred embodiment, the base computer 3 holds a database 7 with which the remote computers 2 can interact when the remote computers 2 run an object 4 to determine whether the object 4 is safe or unsafe. The community database 7 is populated, over time, with information relating to each object run on all of the connected remote computers 2. As will be discussed further below, data

representative of each object 4 preferably takes the form of a so-called signature or key relating to the object and its effects. As will also be discussed further below, the database 7 may further include a mask for the object 4 that sets out the parameters of the object's performance and operation.

Referring now to Figure 2, at the start point 21, a computer object 4 such as a process is run at a remote computer 2. At step 22, by operation of local "agent" software running on the remote computer 2, the operation of the process is hooked so that the agent software can search a local database stored at the remote computer 2 to search for a signature or key representing that particular process, its related objects and/or the event. If the local signature is present, it will indicate either that the process is considered to be safe or will indicate that that process is considered unsafe. An unsafe process might be one that has been found to be malware or to have unforeseen or known unsafe or malevolent results arising from its running. If the signature indicates that the process is safe, then that process or event is allowed by the local agent software on the remote computer 2 to run at step 23. If the signature indicates that the process is not safe, then the process or event is stopped at step 24.

It will be understood that there may be more than two states than "safe" or "not-safe" and choices may be given to the user. For example, if an object is considered locally to be not safe, the user may be presented with an option to allow the related process to run nevertheless. It is also possible for different states to be presented to each remote computer 2. The state can be varied by the

central system to take account of the location, status or ownership of the remote computer or timeframe.

If the object is unknown locally, then details of the object are passed over the Internet 1 or other network to the base computer 3 for storing in the community database 7 and preferably for further analysis at the base computer 3. In that case, the community database 7 is then searched at step 25 for a signature for that object that has already been stored in the community database 7. The community database 7 is supplied with signatures representative of objects, such as programs or processes, run by each monitored remote computer 2. In a typical implementation in the field, there may be several thousands or even millions of remote computers 2 connected or connectable to the base computer 3 and so any objects that are newly released upon the Internet 1 or that otherwise are found on any of these remote computers 2 will soon be found and signatures created and sent to the base computer 3 by the respective remote computers 2.

When the community database 7 is searched for the signature of the object that was not previously known at the remote computer 2 concerned, then if the signature is found and indicates that that object is safe, then a copy of the signature or at least a message that the object is safe is sent to the local database of the remote computer 2 concerned at step 26 to populate the local database. In this way, the remote computer 2 has this information immediately to hand the next time the object 4 is encountered. A separate message is also passed back to the remote computer 2 to allow the object to run in the current instance.

If the signature is found in the community database 7 and this indicates for some reason that the object is unsafe, then again the signature is copied back to the local database and marked "unsafe" at step 27, and/or a message is sent to the remote computer 2 so that running of the object is stopped (or it is not allowed to run) and/or the user given an informed choice whether to run it or not.

If after the entire community database 7 has been searched the object is still unknown, then it is assumed that this is an entirely new object which has never been seen before in the field. A signature is therefore created representative of the object at step 28, or a signature sent by the remote computer 2 is used for this purpose, and this signature is initially marked as bad or unsafe community database 7 at step 29. The signature is copied to the local database of the remote computer 2 that first ran the object at step 30. A message may then be passed to the remote computer 2 to instruct the remote computer 2 not to run the object or alternatively the user may be given informed consent as to whether to allow the object to run or not. In addition, a copy of the object itself may be requested at step 31 by the community database 7 from the remote computer 2.

If the user at the remote computer 2 chooses to run a process that is considered unsafe because it is too new, then that process may be monitored by the remote computer 2 and/or community database 7 and, if no ill effect occurs or is exhibited after a period of time of n days for example, it may then be considered to be safe. Alternatively, the community database 7 may keep a log of each instance of the

process which is found by the many remote computers 2 forming part of the network and after a particular number of instances have been recorded, possibly with another particular number of instances or the process being allowed to run and running safely, the signature in the community database 7 may then be marked as safe rather than unsafe. Many other variations of monitoring safety may be done within this concept.

The details of an object 4 that are passed to the base computer 3 are preferably in the form of a signature or "key" that uniquely identifies the object 4. This is mainly to keep down the data storage and transmission requirements. This key may be formed by a hashing function operating on the object at the remote computer 2.

The key in the preferred embodiment is specially arranged to have at least three severable components, a first of said components representing executable instructions contained within or constituted by the object, a second of said components representing data about said object, and a third of said components representing the physical size of the object. The data about the object in the second component may be any or all of the other forms of identity such as the file's name, its physical and folder location on disk, its original file name, its creation and modification dates, vendor, product and version and any other information stored within the object, its file header or header held by the remote computer 2 about it; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers. In general, the information provided in the key may include at least one of these

elements or any two or more of these elements in any combination.

In one preferred embodiment, a check sum is created for all executable files, such as (but not limited to) .exe and .dll files, which are of the type PE (Portable Executable file as defined by Microsoft). Three types of checksums are generated depending on the nature of the file:

Type 1: five different sections of the file are check summed. These include the import table, a section at the beginning and a section at the end of the code section, and a section at the beginning and a section at the end of the entire file. This type applies to the vast majority of files that are analysed;

Type 2: for old DOS or 16 bit executable files, the entire file is check summed;

Type 3: for files over a certain predefined size, the file is sampled into chunks which are then check summed. For files less than a certain predefined size, the whole file is check summed.

For the check summing process, in principle any technique is possible. The MD5 (Message-Digest algorithm 5) is a widely-used cryptographic hash function that may be used for this purpose.

This allows a core checksum to be generated by viewing only the executable elements of the checksum and making a

comparison between two executables that share common executable code.

For the type 1 checksum mentioned above, three signature processes may be used. The first defines the entire file and will change with almost any change to the file's content. The second attempts to define only the processing instructions of the process which changes much less. The third utilises the file's size, which massively reduces the potential of collisions for objects of differing sizes. By tracking the occurrences of all signatures individually appearing with different counterparts, it is possible to identify processes that have been changed or have been created from a common point but that have been edited to perform new, possibly malevolent functionality.

This "meta data" enables current and newly devised heuristics to be run on the data in the community database 7.

The data stored in the community database 7 provides an extensive corollary of an object's creation, configuration, execution, behaviour, identities and relationships to other objects that either act upon it or are acted upon by it.

The preferred central heuristics use five distinct processes to establish if an object is safe, unsafe or suspicious.

The first of the said processes utilises the singularity or plurality of names, locations, vendor,

product and version information captured and correlated from all of the remote computers 2 that have seen the object. By considering the plurality of this information for a single object, a score can be determined which can be used as a measure of the authenticity and/or credibility of the object. Most safe objects tend not to use a large plurality of identifying information or locations. Rules can be established to consider this information in respect of the type of object and its location. For example, temporary files often utilise a plurality of system generated file names which may differ on each remote computer for the same object. Where an object has little plurality, then it provides a reference point to consider its behaviour in comparison to the known behaviours of other objects that have previously used that identifying information. For example, a new object that purports to be a version of notepad.exe can have its behaviour compared with the behaviour of one or more other objects that are also known as notepad.exe. This comparison may be against a single other object or multiple other objects that use the same or even similar identifying information. In this way, new patterns of behaviour can be identified for the new object. Also it allows the preferred embodiment to police an object's behaviour over time to identify new behavioural patterns that may cause an object that was previously considered safe to have its status reconsidered. Alternatively, the score based on identities may be considered along with scores for other objects or scores created by other processes on this object to be considered in combinations.

The second of the said processes utilises an object's relationship to other objects that act upon it or upon

which it acts. For example, analysis can be made of which object created this object, which objects this object created, which objects created a registry key to run or configure this object, which objects were configured by or had registry keys created by this object, etc. In this regard an object is considered to have a relationship based on the event performed by it upon another object, or upon it by another object. This simple 1-to-1 relationship chain provides a complex series of correlation points, allowing ancestral relationships to be considered for any object and its predecessors by event or its issue (i.e. child and sub-child processes) by event. This allows a score to be developed that describes its relationships and associations with known, unknown, known safe or known bad objects or a combination thereof. Objects that have specific relationships, volumes of relationships or mixes of relationships to one type or another may be judged safe or unsafe accordingly. Alternatively, the relationship-based score may be considered along with other scores to arrive at a determination of safe or unsafe. This data can also be used to deduce a number of factors about objects related directly or via other objects and their behaviours. For example it is possible to deduce how one object's behaviour can be influenced or changed by its association or linkage to another. Consider for example notepad.exe as supplied by Microsoft with the Windows series of operating systems. It has a limited range of functionality and would not be expected therefore to perform a wide variety of events, such as transmitting data to another computer or running other programs etc. However, the behaviour of notepad.exe could be modified by injecting new code into it, such as via dynamic link library injection (DLL injection). In this case notepad.exe would now have new

capabilities derived by the code injection or linkage to another object. Using the data that defines the relationships between objects it is possible to deduce that the new behaviours of a program can be attributed to the association with another object. If that new behaviour is malevolent, then it is possible to mark either or all processes as unsafe as appropriate.

The combination of behaviours captured provide a basis to determine if the object is safe or unsafe. Malware typically exhibit certain behaviour and characteristics. For example, malware frequently has a need to self-persist. This manifests itself in the need to automatically restart on system restarts or upon certain events. Creating objects in specific locations to auto restart or trigger execution is a typical characteristic of malware. Replacing core objects of the Windows system environment are another example of typical characteristics of malware. By providing a pattern of behaviour, the determination of objects to be unsafe or safe can be automated. The centralisation of the community data in the community database 7 provides the ability to rapidly assimilate object behaviours, allowing for the rapid identification and determination of malware. Objects may also perform events upon themselves which can be considered in deriving a score.

The third said process involves time and volumes. Relevant data includes when the object was first seen, when it was last seen, how many times it has been seen, how many times it has been seen in a given interval of time, and the increase or decrease of acceleration in it being seen. This information is highly relevant in determining the

prevalence of an object in the community of remote computers. A score is developed based on these metrics which can be used to determine if an object is safe, unsafe or too prevalent to allow it to execute or propagate without very thorough examination. In this case, the object can be temporarily held or blocked from executing pending further information about its behaviour or relationships. This score may also be used in combination with scores from other processes. Time is also highly relevant in combination with other information, including but not limited to behaviours and identities. For example in the case of polymorphic or randomly named objects, time is a powerful qualifier. (A polymorphic virus changes its encryption algorithm and the corresponding encryption keys each time it replicates from one computer to another and so can be difficult to detect by conventional measures.) A program that creates other programs can often be considered normal or abnormal based on its activity over time.

The fourth said process considers the behaviour of an object. This allows a score to be developed based on the types of events performed by an object or events performed on it by itself or other objects. The centralised system of the community database 7 allows for an unlimited number of event types and can consider the object performing the event or the object having the event performed upon it, or both. Some event types also relate to external information other than objects, for example a program performing an event to connect with an Internet Chat Relay site, or a program modifying a non-executable file such as the Windows hosts file. The behavioural events of an object, be they as "actor" (i.e. the object doing something to another object) or as "victim" (i.e. the object has something done

to it by another object) of any event can be considered in many ways, such as in combination, in sequence, in volume, in presence or absence, or in any combination thereof. The behavioural events in the preferred embodiment may have been provided by a remote computer 2 or from other external sources. The process can consider these in isolation or in combination. Furthermore it is a feature of the preferred embodiment that the behavioural events can be considered in combination with the status of other objects upon which the object acts or that act upon the object. For example, creating a program may have a different score if the program being created is safe, unsafe, new, unknown or suspicious. Similarly, a program that is created by a known bad program will likely have a different score attributed to its creation event depending on the status of the object creating it.

The fifth process considers the behaviour of a web page or script. In this respect, the web page and url combination is assigned a unique identity which allows its behaviour to be tracked as if it were an object like any other executable file. In this example, the web page may perform events that would normally be seen as events performed by the web browser (e.g. IExplore.exe or Firefox.exe). The preferred embodiment substitutes the identifying details and signatures of the web browser for the "pseudo" object identity associated with the web page being displayed or executing within the browser. In this respect, the status of the web page and/or web site to which it relates may be determined as safe, unsafe, unknown or suspicious in the same way as any other object. The web page's "pseudo" object identity also allows the preferred embodiment to block, interject or limit the functionality

of that web page or web site to prevent some or all of its potentially unsafe behaviour or to provide the remote user with qualifying information to guide them about the safety of the web site, web page or their content.

Amongst other types, the types of meta data captured might be:

"Events": these define the actions or behaviours of an object acting upon another object or some other entity. The event has three principal components: the key of the object performing the act (the "Actor"), the act being performed (the "Event Type"), and the key of the object or identity of an other entity upon which the act is being performed (the "Victim"). While simple, this structure allows a limitless series of behaviours and relationships to be defined. Examples of the three components of an event might be:

Actor	Event Type	Victim
Object 1	Creates Program	Object 2
Object 1	Sends data	IP Address 3
Object 1	Deletes Program	Object 4
Object 1	Executes	Object 2
Object 2	Creates registry key	Object4

"Identities": these define the attributes of an object. They include items such as the file's name, its physical location on the disk or in memory, its logical location on the disk within the file system (its path), the file's header details which include when the file was created, when it was last accessed, when it was last modified, the information stored as the vendor, the product

it is part of and the version number of the file and it contents, its original file name, and its file size.

"Genesisactor" - the key of an object that is not the direct Actor of an event but which is the ultimate parent of the event being performed. For example in the case of a software installation, this would be the key of the object that the user or system first executed and that initiated the software installation process, e.g. Setup.exe.

"Ancillary data": many events may require ancillary data, for example an event such as that used to record the creation of a registry run key. In this situation the "event" would identify the Actor object creating the registry run key, the event type itself (e.g. "regrunkey"), and the Victim or subject of the registry run key. The ancillary data in this case would define the run key entry itself; the Hive, Key name and Value.

"Event Checksums": because the event data can be quite large extending to several hundred bytes of information for a single event, its identities for the Actor and Victim and any ancillary data, the system allows for this data itself to be summarised by the Event Checksums. Two event checksums are used utilising a variety of algorithms, such as CRC and Adler. The checksums are of the core data for an event. This allows the remote computer 2 to send the checksums of the data to the central computer 3 which may already have the data relating to those checksums stored. In this case, it does not require further information from the remote computer 2. Only if the central computer 3 has never received the checksums will it request the associated data from the

remote computer 2. This affords a considerable improvement in performance for both the remote and central computers 2,3 allowing much more effective scaling.

Thus, the meta data derived from the remote computers 2 can be used at the community database 7 to define the behaviour of a process across the community. As mentioned, the data may include at least one of the elements mentioned above (file size, location, etc.) or two or three or four or five or six or all seven (or more elements not specifically mentioned here). This may be used accordingly to model, test and create new automated rules for use in the community database 7 and as rules that may be added to those held and used in the local database of the remote computers 2 to identify and determine the response of the remote computers 2 to new or unknown processes and process activity.

Moreover, it is possible to monitor a process along with any optional sub-processes as an homogenous entity and then compare the activities of the top level process throughout the community and deduce that certain, potentially malevolent practices only occur when one or more specific sub-processes are also loaded. This allows effective monitoring (without unnecessary blocking) of programs, such as Internet Explorer or other browsers, whose functionality may be easily altered by downloadable optional code that users acquire from the Internet, which is of course the principal source of malevolent code today.

The potentially high volume of active users gives a high probability of at least one of them being infected by new malware. The speed of propagation can be detected and

recorded so that the propagation of malware can be detected and malware designated as bad on the basis of the speed of propagation, optionally in combination with the other factors discussed above, such as file size, location and name. The simple volume of infection can also be used as a trigger. In a further embodiment, difference of naming of an otherwise identical piece of code combined with acceleration of first attempts to execute the code within the community allows pattern matching that will show up an otherwise identically signatured piece of code as bad.

This feature allows the statement in some embodiments that "nothing will propagate in our community faster than X without being locked down", so that if any process or event propagates more quickly over a given duration, it is marked as bad. This is for reasons of safety given that if for example an object is propagating quickly enough, then it might infect computers before it can be analysed to determine whether or not it is malware.

This process can be automated by the identification of the vector of propagation in the community (i.e. the source of type of propagation), from timestamp data held in the community database and the marking of a piece of code that has these attributes as bad. By comparison, it is believed that all other anti-malware providers rely on a simplistic known bad model and therefore are reliant primarily on malware infection actually occurring on terminals and being reported.

Thus, the community database 7 can be used to make early diagnosis, or simply to take precautionary measures, and thus stop potentially fast propagating worms and other

malware very, very early in their life cycle. Given that it is possible to create a worm that can infect every computer connected to the Internet within a matter of a few minutes, this feature is highly desirable.

Even faster determination may be made by combining data defining the speed of propagation of a new piece of software with metadata collected by the agent software from the remote computers 2 and fed to the community database 7. This includes monitoring processes that attempt to conceal themselves from the user by randomly changing name and/or location on the remote computers 2. It also includes a process's attempt to create an identical copy (i.e. with identical code contents) on the computer but with a different name. This is a classic attribute of a worm.

The signature of an object may comprise or be associated with a mask which can be built up with use of that object and which indicates the particular types of behaviour to be expected from the object. If an object is allowed to run on a remote computer 2, even if the initial signature search 22 indicates that the object is safe, then operation of that object may be monitored within the parameters of the mask. The mask might indicate for example, the expected behaviour of the object; any external requests or Internet connections that that object might legitimately have to make or call upon the remote computer 2 to make, including details of any ports or interfaces that might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, that may be expected to be interrogated by that object; and so on.

Thus, the mask can give an overall picture of the expected "normal" behaviour of that object.

In practice, therefore, in one embodiment the behaviour of the object is continually monitored at the remote computer(s) 2 and information relating to that object continually sent to and from the community database 7 to determine whether the object is running within its expected mask. Any behaviour that extends beyond the mask is identified and can be used to continually assess whether the object continues to be safe or not. Thus, if for example the object, on a regular basis (say monthly or yearly) opens a new port to update itself or to obtain regular data, then this information is flagged. If it is found that the object has done this on other remote computers and has had no ill effects, or this behaviour is known from other objects and known to be safe, then this behaviour might be considered as safe behaviour and the mask is then modified to allow for this. If it has been found previously that this new behaviour in fact causes unsafe or malevolent results, then the object can then be marked as unsafe even if previously it was considered safe. Similarly, if the object attempts to connect to a known unsafe website, database or to take action that is known as generally being action only taken by unsafe programs, then again the object may be considered to be unsafe.

This is shown schematically in Figure 3. Figure 3 also shows the concept that any object can be pre-authorised by, for example a trusted partner, such as a major software company, a verification authority, a Government department, and so on. Pre-authorisation enables a supplier of a new object, which has not been

released before, to get pre-authorisation for that object, and optionally includes the provision by that supplier of a mask detailing the expected and allowable behaviour of that object.

Referring to Figure 3 for example, when a process is run, the local and/or community databases are searched as before at step 31. If the process is not a pre-authorised one, then the steps of Figure 2 may be taken and the process might be allowed to run or not at step 32. If the process is pre-authorised, as determined at step 33, then it is immediately allowed to run, step 34. This may terminate the operation of the method. However, in a preferred variation, the process is then monitored whilst running, and is monitored each time it is run in the future in a monitoring state step 35 to determine whether its behaviour falls within its pre-authorised mask 36. If the behaviour falls within the pre-authorised behaviour, then the process is allowed to continue to run. If the behaviour extends beyond the allowed mask, such as by trying to instigate further processes or connections that have not been pre-authorised, then this behaviour is flagged at an alert step 37. Various actions could be taken at this stage. The process might simply not be allowed to run. Alternatively, the trusted authority that initially enabled pre-authorisation might be contacted, who may be able to confirm that this behaviour is acceptable or not. If it is acceptable, then the mask could be modified accordingly. If not acceptable, then the process might be marked as unsafe. Many other actions may be taken upon the noting of such an alert state.

If the process has been found not to be pre-authorised at step 33 but is nevertheless allowed to run, then the process is monitored at step 38 in order to generate a mask 39 representative of the normal behaviour of that process. Data representative of this mask might be sent to the community database 7 for scanning when other computers run that process. By continually monitoring a process each time it is run or during running of the process, any behaviour that differs from previous behaviour of the process can be noted and the mask can be modified, or this behaviour might be used to determine that a process that was once considered safe should now be considered to be unsafe.

In another embodiment, a computer 2 may have agent software installed that periodically or on-demand provides information to the community database 7 that is representative of all or selected ones of the software products loaded on or available the computer 2. In particular, this may be information on one or more of: all the locally-loaded security products (such as anti-malware systems including anti-virus software, anti-spyware, anti-adware and so on), firewall products, specific settings and details of which signature files are currently loaded, version details for the operating system and other software, and also information such as which files are operating and the particular version and software settings at any time. (It will be understood from the following that auditing and testing for a match for more of these criteria increases the likelihood of computers being very similarly arranged and thus reduces the rate of false negatives and positives during the match search.)

The information relating to these software products, etc. may be provided individually to the community database. Preferably however, again for reasons of data quantity for storage and transmission, the information is provided as a signature or key representative of the information (e.g. by a hashing or compression function at the computer 2). Figure 4 shows schematically how the details of all local security products, versions, signature files, firewall settings, etc. 40 are used to create a key 41. The key is transmitted to the community database 7. Since the community database 7 is provided with such information from many, possibly millions, of users' computers 2, it is likely to hold corresponding information for other computers 2 that have the same or a similar configuration of security products, etc. Thus, the community database 7 can be searched at step 42 for other computers 2 having the same or a similar combination of security products including the same setting, signature files loaded and so on.

The community database 7 in this embodiment is also provided by the agent software with details of processes run by every computer 2 and thus knows whether or not a process has been detected by each computer 2.

In this way, the community database 7 can be used to obtain information as to whether for example a particular, specific combination of operating system and various security products, including settings and signature files existing at a point in time, renders a particular computer 2 having those products and settings susceptible or vulnerable to any particular malware object.

In a simple example, if for example the database knows that a computer in the past has version A of anti-virus product B with downloaded signature update C, and also has a firewall D with particular settings E, and perhaps anti-spyware software F with signature updates G, but that a particular malware process P was not detected by this combination of programs at that point in time, then this information can be provided to a computer 2 that is known as having that combination of security programs/settings, and can be used to indicate that that computer 2 is vulnerable in the short term to attack by that particular malware process. This information can be presented to the user either by displaying a window 43 on the screen display of the computer 2 or by directing the user to a particular website which explains the position in more detail. The user might be informed for example that their particular combination of security products, etc., exposes their computer to a risk of being infected by the Sobig virus as that virus is not detectable by their computer. The user might be offered specific advice (e.g. to update a particular anti-virus program with a particular signature file) or software to download and install to remove the risk.

Thus, the community database 7, when provided with information relating to all the security products, etc. on a particular computer at a particular time, is searched for events for processes marked as "bad" that occurred on computers with that particular mix of security products and that were not locally detected. This information can then be fed back to the user of the particular computer, for example directly or by directing the user to a website. This information can be provided virtually in real-time,

allowing a new user or a user of a new computer to be able to increase the computer's effective security very quickly.

The preferred method also tracks which objects are related to each other and uses the concept of ancestry to enable objects to be marked as malware. For example, any particular process may spawn child processes which are therefore related. The key relating to the first object may be inspected to identify a component that uniquely identifies the first object and that is inherited or otherwise present in the key of a descendant or other related object of the first object. This component is referred to herein as a "gene". This general technique may be used in a number of ways:

a) A known and trusted parent process is afforded the ability to create child processes which may be automatically marked as safe to run on the local computer. It is also possible that this "inherited" property may be passed down to grand children processes and so on. This safe status is passed to the parent's child processes and possibly, through them, further child processes (referred to here as "issue"), such signatures for the issue can all automatically be recorded in the local database as good. This allows the issue processes to be quickly marked as good, even if a connection to the community database 7 is not available.

b) By monitoring activity of the parent process, if it is later found that the parent process is malware, then all of the issue processes can all automatically be recorded in the local database as bad.

C) Similarly, by monitoring activity of the issue processes, if it is later found that one of the issue processes is malware, then one or more of the parent process and all of the other issue processes (i.e. all of the related processes in this context) can all automatically be recorded in the local database as bad.

d) Parental creation of a signature for a child or children including the ability for these to be automatically marked as either good or bad depending on the parent's behaviour and determination. Note that in some embodiments the product can "watch" or monitor the birth of a child process and automatically create the signature upon arrival. This provides the ability to monitor the creation of a bad program by another bad program. It is possible therefore to monitor the ancestry of a program so if for example the grandfather creates a program (the father) and this in turn creates a bad program (the son), it is possible automatically to determine the father as a bad program.

e) A feature may be included that allows for automatic forfeiture of a child's inherited ability to trigger the automatic creation of signatures on any further births because the child, as parent, has already produced bad offspring. Preferably, a rule is that if a file has one bad offspring then the inherited ability can be automatically removed.

f) An ability to watch clones or identical twins of objects (e.g. the same process running on other systems in the community) to compare the pattern of their issue and to

make decisions as to whether or not to treat any particular process as malware.

One or more of these features a) to f) can be used to provide a solution to the problem of producing a security product that can be used effectively without 100% reliance on being permanently connected to the Internet, which is often impractical. Examples of this are Windows Update and other processes used more and more by vendors who wish to be able to roll out product updates automatically across the worldwide web.

Possible benefits of these types of features above conventional software are as follows. Antivirus software tends to have a cache of known bad signatures. The problem is keeping this up to date. Take the simple example of someone buying a new computer. The computer comes with an antivirus product preloaded with a signature cache. Between the time when the PC was built, shipped to the store and bought by the user several days or weeks will have passed. The user starts the PC and is exposed to any new virus or malware which was created after the PC was built. Full protection requires the user to connect to the internet and download updates. This cannot be guaranteed to occur ahead of other activities by the user on the internet (almost physically impossible to guarantee). With a local cache of known good processes, as in the embodiments of the present invention, it is possible to ship the computer/terminal preloaded with a pre-generated cache of signatures for all of the good (clean) software preloaded by the computer manufacturer. In this case the user can connect to the internet knowing that any new or updated programs will be immediately detected and verified.

Also any auto-updating software can function forcing signatures to be automatically built for its children and more remote off-spring (i.e. grandchildren, great-grandchildren, etc).

Reference is now made to "Defeating Polymorphism: Beyond Emulation" by Adrian Stepan of Microsoft Corporation published in "Virus Bulletin Conference October 2005" and also to US 60/789156 filed on 5th April 2006, the entire content of which are hereby incorporated by reference. In that paper and patent application, there are disclosed methods of decrypting files to allow the files to be analysed to determine whether or not the file actually is malware. In US 60/789156 in particular, there is disclosed a method of unpacking/decrypting an executable computer file using a host computer, the method comprising: partitioning the executable computer file into plural basic blocks of code; translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer; linking at least some of the translated basic blocks of code in memory of the host computer; and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware. There is also disclosed in US 60/789156 a method of unpacking/decrypting an executable computer file, the method comprising: partitioning the executable computer file into plural basic blocks of code; creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached

real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and, emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware; wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

The techniques disclosed in these documents can be used in the present context when it is desired to analyse a file in detail. More generally however, the techniques disclosed in these papers, and particularly the enhanced techniques disclosed in US 60/789156, can be used to provide information about the activity of a file when it is run on a computer because the techniques disclosed in these papers emulate the running of the file and therefore allow the file's activity to be interpreted.

A further situation arises when users wish to install software while offline. In a preferred embodiment, when a user attempts to install new software while offline, agent software running on the user's computer 2 prompts the user for authorisation to allow the installation process to proceed such that the execution of the installation can

itself be "authorised" by the user. This authorisation by the user is treated as a "Genesis" event and will be so termed hereinafter. However there are some processes commonly used in installation of software that communicate with other existing programs on the installation machine, e.g. Microsoft's MSIEXEC.

The Genesis approach involves a process that generates signatures as a result of the user's authorisation on the user's computer 2. Those signatures are stored in the local database on the user's computer 2. In one embodiment, those locally stored signatures are referred to as necessary by the agent software during the installation process so that the installation can proceed. Alternatively, the security checks made by the agent software can be switched off during the installation process. The switching off may only be for a limited duration, such as a few minutes which should be sufficient to allow most software to be installed, the off time optionally being user-configurable.

In any event, once the installation has been completed and the user's computer 2 connected to the community database 7, the agent software on the user's computer 2 can upload the signatures relating to the installation from the user's local database to the community database 7. With corresponding data from other users' computers 2, the community database 7 can then be used to make a rapid determination that the installation of this particular software is benign.

As a variant to this embodiment, when a user's computer 2 is at some point in time on-line, the agent

software on the user's computer 2 may download signatures of a so-called "trusted installer" or "licensed installer". This allows the operation of a method such that a "licensed installer" and any child processes of the licensed installer are permitted to execute while a Genesis event is "current", e.g. within a period of minutes after an authorisation from the user. Preferably signatures of licensed installers are always downloaded, as and when added to the community database 7, to a remote computer 2 while online.

There may be further refinements to this method, such as to prevent licensed installers executing if a media change has occurred during the currency of a Genesis event. However, any unknown processes, which may be malware, may still be detected and blocked. Having a small number of licensed installers facilitates download, as minimal data is required, especially compared to downloading virus signature files. As another example, super-trusted installers, such as for example "Windows Update", may be employed whereby all new processes created by the super-trusted installer are marked immediately as safe.

In the case of an installation occurring when the remote computer 2 is connected to the community database 7, another option is that if the software that is about to be installed is not known to the community database 7, then the system will block the installation or alert the user. For example, a message may be displayed at the user's computer 2 to the effect that "You are about to install some software. This software is not known to [the community]. Are you sure you wish to proceed?".

Embodiments of the present invention have been described with particular reference to the examples illustrated. However, it will be appreciated that variations and modifications may be made to the examples described within the scope of the present invention.

CLAIMS

1. A method of classifying a computer object as malware, the method comprising:

at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored;

comparing in the base computer the data about the computer object received from the plural computers; and,

classifying the computer object as malware on the basis of said comparison.

2. A method according to claim 1, wherein the data about the computer object that is sent from the plural remote computers to the base computer and that is used in the comparison includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

3. A method according to claim 1, wherein the comparing identifies relationships between the object and other objects.

4. A method according to claim 3, wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware.

5. A method according to claim 3, wherein said other objects include the object or similar objects stored on at least some of the remote computers.

6. A method according to claim 3, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

7. A method according to claim 1, wherein the data is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on the respective remote computers.

8. A method according to claim 7, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

9. A method according to claim 7, wherein the key has at least one component that represents data about said object.

10. A method according to claim 9, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the

object is created, configured or runs on the respective remote computers.

11. A method according to claim 7, wherein the key has at least one component that represents the physical size of the object.

12. A method according to claim 1, comprising initially classifying an object as not malware, generating a mask for said object that defines acceptable behaviour for the object, and comprising monitoring operation of the object on at least one of the remote computers and reclassifying the object as malware if the actual monitored behaviour extends beyond that permitted by the mask.

13. Apparatus for classifying a computer object as malware, the apparatus comprising:

a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored;

the base computer being constructed and arranged to compare the data about the computer object received from said plural computers; and,

the base computer being constructed and arranged to classify the computer object as malware on the basis of said comparison.

14. Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to compare the data where the data includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the

physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

15. Apparatus according to claim 13, wherein the base computer is constructed and arranged so that the comparing identifies relationships between the object and other objects.

16. Apparatus according to claim 15, the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.

17. Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least some of said remote computers.

18. Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

19. Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of key that is

obtained by a hashing process carried out in respect of the objects on said respective remote computers.

20. Apparatus according to claim 19, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

21. Apparatus according to claim 19, wherein the key has at least one component that represents data about said object.

22. Apparatus according to claim 21, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

23. Apparatus according to claim 19, wherein the key has at least one component that represents the physical size of the object.

24. Apparatus according to claim 19, wherein the base computer is constructed and arranged to generate a mask for an object that is initially classed not as malware, said mask defining acceptable behaviour for the object, and the base computer is constructed and arranged to monitor operation of the object on at least one of the remote computers and to reclassify the object as malware if the

actual monitored behaviour extends beyond that permitted by the mask.

25. A method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers, the method comprising:

providing from a remote computer to a base computer data about a computer object that is stored on the remote computer;

the data including one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers;

the data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer.

26. A method according to claim 25, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

27. A method according to claim 25, wherein the key has at least one component that represents data about said object.

28. A method according to claim 27, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

29. A method according to claim 25, wherein the key has at least one component that represents the physical size of the object.

30. A computer program comprising program instructions for causing a computer to perform the method of claim 25.

31. A method of determining the protection that a remote computer has from malware, the method comprising:

receiving at a base computer details of all or selected security products operating at a point in time on said remote computer;

receiving similar information from other remote computers connected to the base computer; and,

identifying any malware processes that were not identified by said other remote computers having that particular combination of security products.

32. A method according to claim 31, comprising providing information to the user of said remote computer that said remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

33. A method according to claim 31, wherein the details of all or selected security products includes the name of the security products, versions, and loaded signature files.

34. Apparatus for determining the protection that a remote computer has from malware, the apparatus comprising:

a base computer constructed and arranged to receive computer details of all or selected security products operating at a point in time on said remote computer;

the base computer being constructed and arranged to receive similar information from other remote computers connected to the base computer; and,

the base computer being constructed and arranged to identify any malware processes that were not identified by said other remote computers having that particular combination of security products.

35. Apparatus according to claim 34, wherein the base computer is constructed and arranged to provide information to the user of said remote computer that said remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

36. Apparatus according to claim 34, wherein the details of all or selected security products includes the name of the security products, versions, and loaded signature files.

37. A method of classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the method comprising:

classifying a first computer object as malware or as safe;

identifying in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and,

classifying said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

38. A method according to claim 37, comprising monitoring activities of said first computer object and reclassifying the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the method further comprising automatically classifying as malware any computer object that has a key in which said unique identifier component is present.

39. A computer program comprising program instructions for causing a computer to perform the method of claim 38.

40. Apparatus for classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the apparatus comprising:

a computer constructed and arranged to classify a first computer object as malware or as safe;

the computer being constructed and arranged to identify in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the

key of a descendant or other related computer object of the first computer object; and,

the computer being constructed and arranged to classify said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

41. Apparatus according to claim 40, wherein the computer is constructed and arranged to monitor activities of said first computer object and reclassify the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the computer further being constructed and arranged to automatically classify as malware any computer object that has a key in which said unique identifier component is present.

42. A method of installing software on a computer, the method comprising:

on initiation of installation of software on a computer, providing a computer-generated prompt on the computer to a user to ascertain whether the user authorises the installation; and,

ceasing the installation if a user authorisation is not received, else:

receiving at the computer the user's authorisation to proceed with the installation;

proceeding with the installation;

obtaining data about computer objects that are created or used during the installation;

storing said data at the local computer.

43. A method according to claim 42, wherein the locally stored data is referred to during the installation to ensure that all objects created or used during the installation are part of the installation process, and, if it is found that objects created or used during the installation are not part of the installation process, either or both of: (i) ceasing the installation and (ii) providing a computer-generated prompt on the computer to the user accordingly.

44. A method according to claim 42, comprising connecting the computer to a community database that is connectable to a plurality of computers, and uploading the stored data to the community database for comparison with similar data provided by other computers.

45. A method according to claim 42, comprising downloading data about trusted installers to the computer, said data about trusted installers being referred to during the installation such that any objects relating to or created by the trusted installer are automatically authorised to proceed.

46. A method according to claim 45, wherein said data about trusted installers is referred to only for a predetermined time period following receipt at the computer of the user's authorisation to proceed with the installation.

47. A computer program comprising program instructions for causing a computer to perform the method of claim 42.

METHODS AND APPARATUS FOR DEALING WITH MALWARE

ABSTRACT

In one aspect, a method of classifying a computer object as malware includes receiving at a base computer data about a computer object from each of plural remote computers on which the object or similar objects are stored. The data about the computer object received from the plural computers is compared in the base computer. The computer object is classified as malware on the basis of said comparison. In one embodiment, the data about the computer object includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the name of the object; the logical storage location or path of the object on the respective remote computers; the vendor of the object; the software product and version associated with the object; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

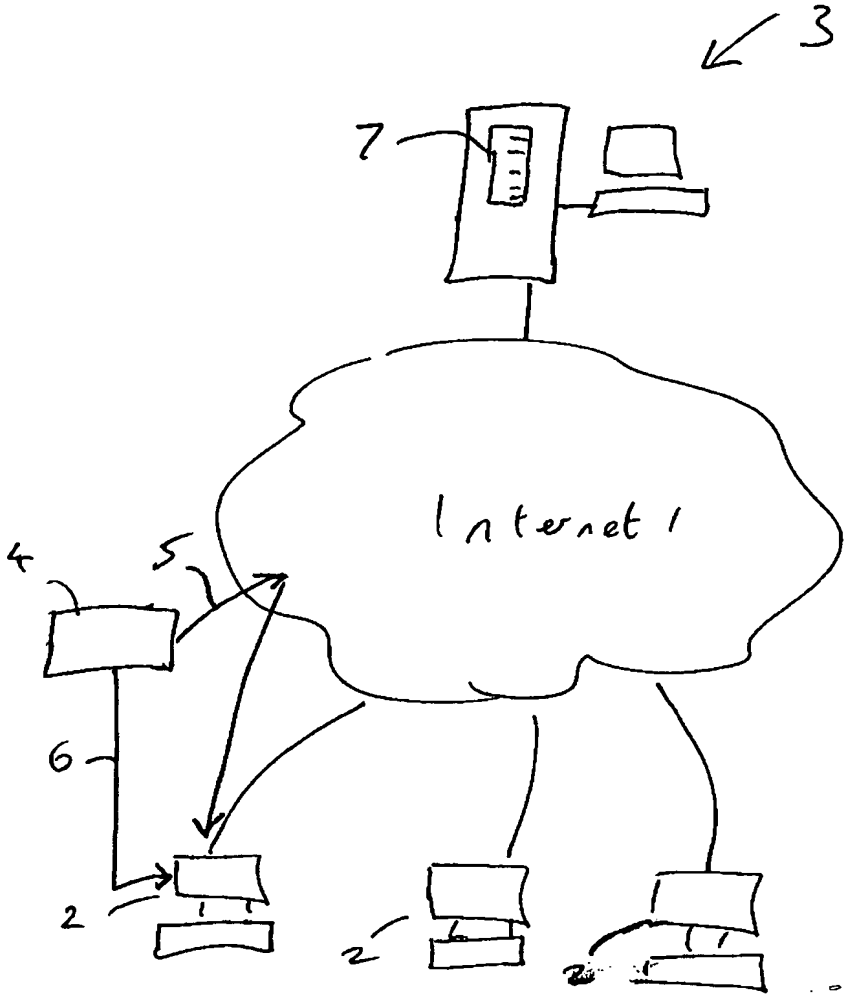


FIG. 1

REMOTE COMPUTER

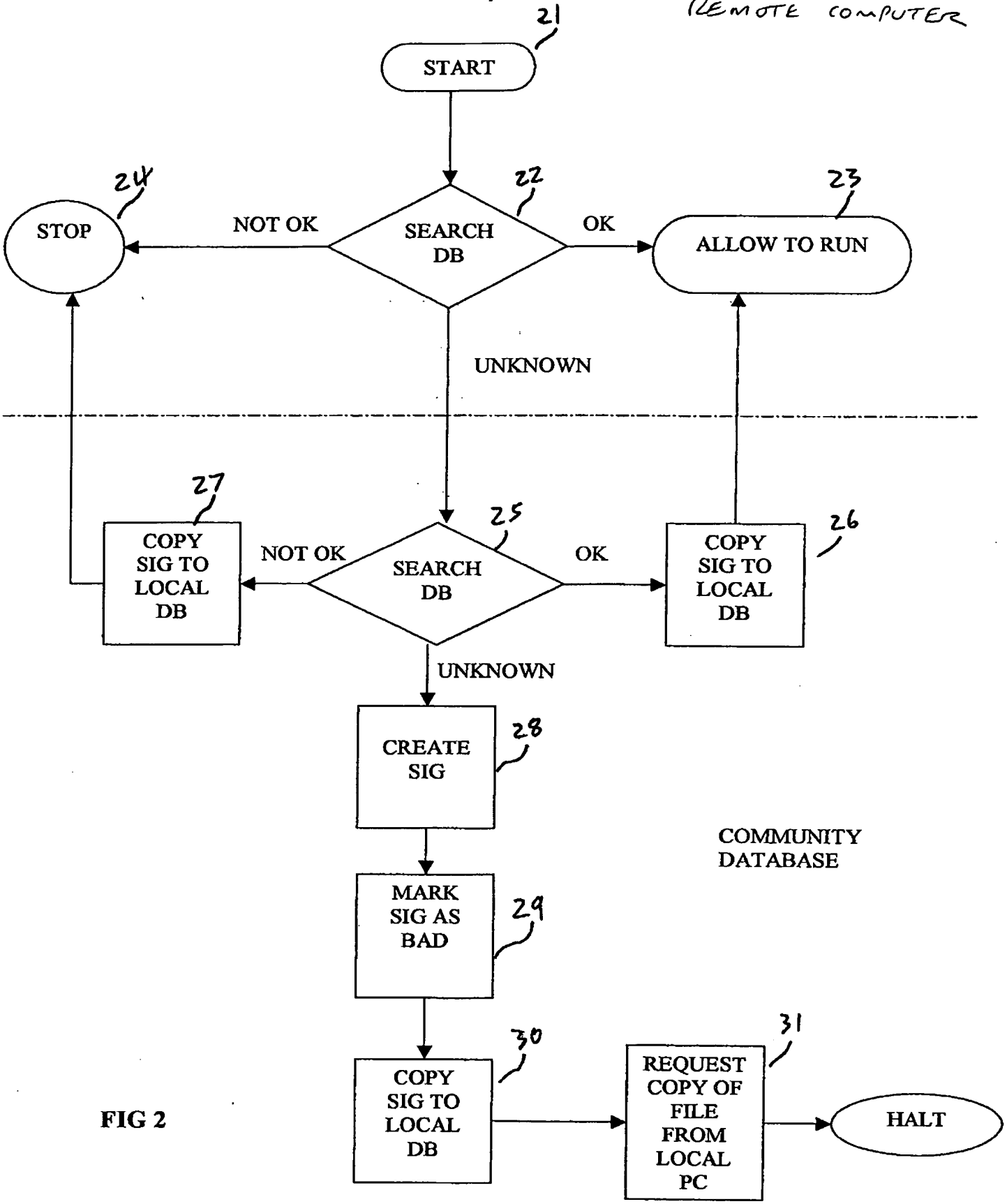


FIG 2

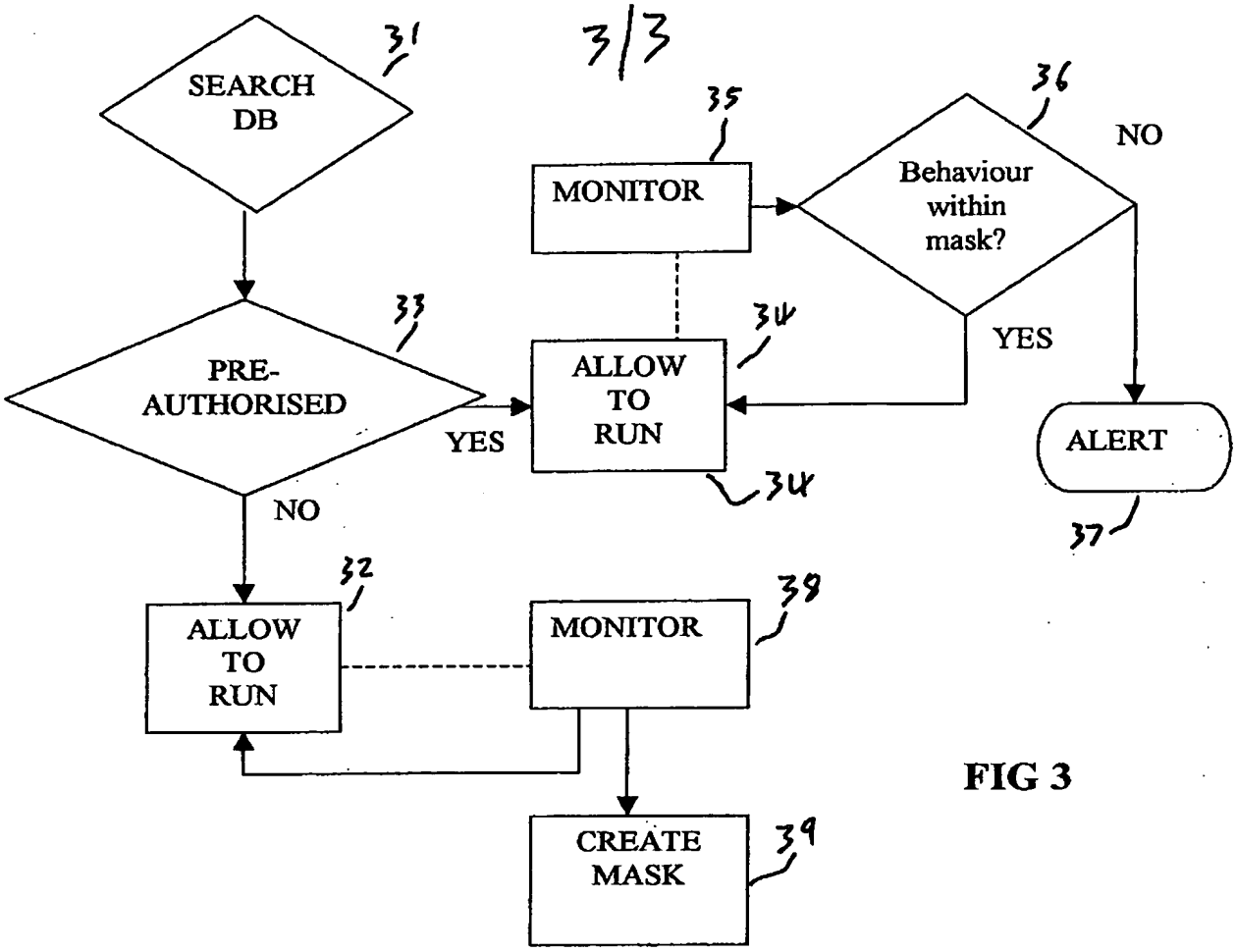

SEARCH DB
31
PRE-AUTHORISED
33
YES
NO
ALLOW TO RUN
32
MONITOR
35
Behaviour within mask?
36
NO
YES
ALLOW TO RUN
34
ALERT
37
MONITOR
38
CREATE MASK
39

FIG 3

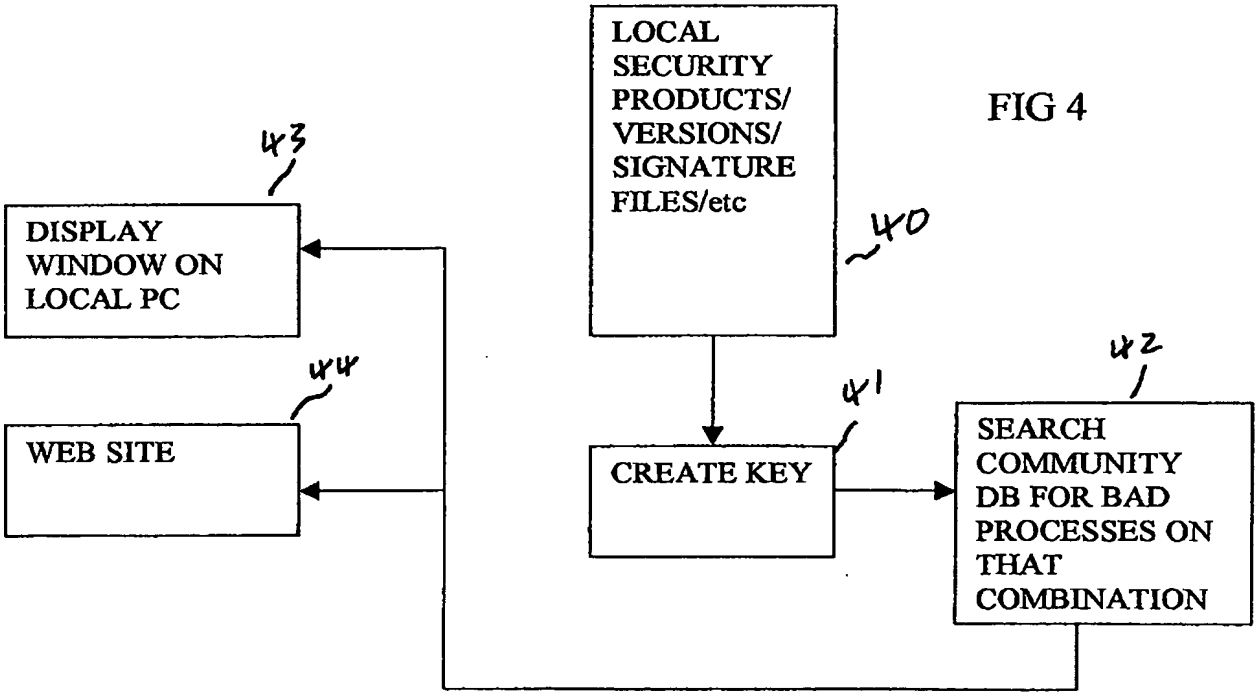

LOCAL SECURITY PRODUCTS/ VERSIONS/ SIGNATURE FILES/etc
40
DISPLAY WINDOW ON LOCAL PC
43
WEB SITE
44
CREATE KEY
41
SEARCH COMMUNITY DB FOR BAD PROCESSES ON THAT COMBINATION
42

FIG 4

Attorney Docket No. 011765-0355986 *PATENT*
Client Reference: AF/JG/P10717US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: MELVYN MORRIS, PAUL STUBBS, MARKUS HARTWIG and DARREN HARTER

Application No.:	NEW	Confirmation No:	Unknown
Filed:	June 30, 2006	Group No.:	TBA
		Examiner	Unknown

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPLICATION DATA SHEET
37 C.F.R. § 1.76

BIBLIOGRAPHIC DATA

1. Applicant information

First applicant: MELVYN MORRIS
Citizenship: Great Britain
Residence: Redmire Gap, Intake Lane, Turnditch, Derbyshire DE56 2LU, United Kingdom

Second applicant: PAUL STUBBS
Citizenship: Great Britain
Residence: 7 Rookery Road, Wyboston, Bedfordshire MK44 3AX, United Kingdom

Third applicant: MARKUS HARTWIG
Citizenship: Germany
Residence: Exchange House, 494 Midsummer Boulevard, Milton Keynes, Buckinghamshire MK9 2EA, United Kingdom

Fourth applicant: DARREN HARTER
Citizenship: Great Britain
Residence: 178 Barnwood Road, Hucclecote, Gloucester, Gloucestershire GL4 3JZ, United Kingdom

2. Correspondence information

Correspondence for this application should be addressed as follows:

Customer No.: 00909

3. Application information

Title of Invention: METHODS AND APPARATUS FOR DEALING WITH MALWARE

Docket number assigned to this application: 011765-0355986
Suggested Classification: Class:
Subclass:
Technology Center to which subject matter is assigned:

Total number of drawing sheets: 3
Type of application: Utility
Application is to be published. Suggested drawing figure for publication: Figure 1

Secrecy order under § 5.2: This application does not disclose subject matter of an application which is under a secrecy order pursuant to § 5.2.

4. Representative information

The following have a power of attorney or authorization of agent in this application:

Customer No.: 00909

5. Foreign Priority information

Foreign priority for this application is claimed as follows:

35 U.S.C. § 365(c):

Application No.: GB 0513375.6
Filed: June 30, 2005
Status: Pending

6. Assignee information

The assignee(s) of this application is/are:

PREVX LIMITED
3 New Park Place
Pride Park
Derby DE24 8DZ
United Kingdom

Extent of interest of assignee in application: Entire right, title and interest.

Date: June 30, 2006

Christine M. McCarthy
Registration No. 41844
PILLSBURY WINTHROP SHAW PITTMAN LLP
P.O. Box 10500
McLean, VA 22102
(703.) 770.7743
Customer No. 00909

PTO/SB/06 (12-04)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number: 11,477,807

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))				150			300
SEARCH FEE (37 CFR 1.16(k), (i), or (m))				250			500
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))				100			200
TOTAL CLAIMS (37 CFR 1.16(i))	47 minus 20 =	27	X$ 25		OR	X$50	1350
INDEPENDENT CLAIMS (37 CFR 1.16(h))	8 minus 3 =	5	X$100			X$200	1000
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR						
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))			180			360	
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	3350

APPLICATION AS AMENDED – PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDI-TIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDI- TIONAL FEE ($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					180		OR	360	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDI-TIONAL FEE ($)	OR	RATE ($)	ADDI- TIONAL FEE ($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/477,807	06/30/2006	Melvyn Morris	011765-0355986

CONFIRMATION NO. 4785

00909
PILLSBURY WINTHROP SHAW PITTMAN, LLP
P.O. BOX 10500
MCLEAN, VA 22102

FORMALITIES LETTER

Date Mailed: 07/27/2006

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
  Applicant must submit $ **300** *to complete the basic filing fee for a non-small entity. If appropriate, applicant may make a written assertion of entitlement to small entity status and pay the small entity filing fee (37 CFR 1.27).*
- The oath or declaration is missing. *A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.*
  Note: If a petition under 37 CFR 1.47 is being filed, an oath or declaration in compliance with 37 CFR 1.63 signed by all available joint inventors, or if no inventor is available by a party with sufficient proprietary interest, is required.

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- Additional claim fees of **$2350** as a non-small entity, including any required multiple dependent claim fee, are required. Applicant must submit the additional claim fees or cancel the additional claims for which fees are due.
- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of $130 for a non-small entity, must be submitted with the missing items identified in this letter.

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is **$3480** for a Large Entity

- **$300** Statutory basic filing fee.
- **$130** Surcharge.

- The application search fee has not been paid. Applicant must submit **$500** to complete the search fee.
- The application examination fee has not been paid. Applicant must submit **$200** to complete the examination fee for a large entity

- Total additional claim fee(s) for this application is **$2350**

  - **$1000** for **5** independent claims over 3.
  - **$1350** for **27** total claims over 20.

Replies should be mailed to: Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

A copy of this notice MUST be returned with the reply.

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199, or 1-800-972-6382
PART 3 - OFFICE COPY



Attorney Docket: 011765-0355986
Client Reference: AF/JG/P10717US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re PATENT APPLICATION of: MELVYN MORRIS ET AL.
Application No.: 11/477,807 Confirmation No: 4785

Filed: June 30, 2006 Group Art Unit: 2131

Examiner: Unassigned

Title: METHODS AND APPARATUS FOR DEALING WITH MALWARE

RESPONSE TO NOTICE TO FILE MISSING PARTS

Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Notice to File Missing Parts of Application mailed July 27, 2006, a copy of which is enclosed, Applicants herewith submit the following:

- Executed Combined Declaration and Power of Attorney of Inventors
- Assignment and Recordation Cover Sheet

Please see attached PTO/SB/17 for payment of fees.

Respectfully submitted,
PILLSBURY WINTHROP SHAW PITTMAN LLP

By _____
Christine H. McCarthy
Registration No. 41844
Telephone: (703) 770.7743

Date: September 26, 2006
P.O. Box 10500
McLean, VA 22102
Customer No.: 00909

400462530v1

SEP 26 2006
PATENT & TRADEMARK OFFICE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/477,807	06/30/2006	Melvyn Morris	011765-0355986

CONFIRMATION NO. 4785

00909
PILLSBURY WINTHROP SHAW PITTMAN, LLP
P.O. BOX 10500
MCLEAN, VA 22102

FORMALITIES LETTER

Date Mailed: 07/27/2006

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
  Applicant must submit $ **300** *to complete the basic filing fee for a non-small entity. If appropriate, applicant may make a written assertion of entitlement to small entity status and pay the small entity filing fee (37 CFR 1.27).*
- The oath or declaration is missing. *A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.*
  Note: If a petition under 37 CFR 1.47 is being filed, an oath or declaration in compliance with 37 CFR 1.63 signed by all available joint inventors, or if no inventor is available by a party with sufficient proprietary interest, is required.

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- Additional claim fees of **$2350** as a non-small entity, including any required multiple dependent claim fee, are required. Applicant must submit the additional claim fees or cancel the additional claims for which fees are due.
- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of $130 for a non-small entity, must be submitted with the missing items identified in this letter.

09/27/2006 CCHAU1 00000111 033975 11477807

01 FC:1011 300.00 DA
02 FC:1111 500.00 DA
03 FC:1311 200.00 DA
04 FC:1051 130.00 DA
05 FC:1202 1350.00 DA
06 FC:1201 1000.00 DA

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is **$3480** for a Large Entity

- **$300** Statutory basic filing fee.
- **$130** Surcharge.

- The application search fee has not been paid. Applicant must submit **$500** to complete the search fee.
- The application examination fee has not been paid. Applicant must submit **$200** to complete the examination fee for a large entity

- Total additional claim fee(s) for this application is **$2350**

  - **$1000** for **5** independent claims over 3.
  - **$1350** for **27** total claims over 20.

Replies should be mailed to: Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

A copy of this notice MUST be returned with the reply.

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199, or 1-800-972-6382
PART 1 - ATTORNEY/APPLICANT COPY

OIPE IAP56 SEP 26 2006 PATENT & TRADEMARK OFFICE

PTO/SB/17 (01-06)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL For FY 2006

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 3,520.00

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	MELVYN MORRIS
Examiner Name	
Art Unit	2131
Attorney Docket No.	011765-0355986

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): _____

☒ Deposit Account Deposit Account Number: 033975 Deposit Account Name: PILLSBURY WINTHROP SHAW PITTMAN LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☒ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

☒ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☒ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

FEE CALCULATION (All the fees below are due upon filing or may be subject to a surcharge.)

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
		Small Entity		Small Entity		Small Entity	
Application Type	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	1,000.00
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

Fee Description	Fee ($)	Small Entity Fee ($)
Each claim over 20 (including Reissues)	50	25
Each independent claim over 3 (including Reissues)	200	100
Multiple dependent claims	360	180

Total Claims		Extra Claims	Fee ($)		Fee Paid ($)
47	- 20 or HP =	27	x 50.00	=	1,350.00

HP = highest number of total claims paid for, if greater than 20.

Indep. Claims		Extra Claims	Fee ($)		Fee Paid ($)
8	- 3 or HP =	5	x 200.00	=	1,000.00

HP = highest number of independent claims paid for, if greater than 3.

Multiple Dependent Claims	
Fee ($)	Fee Paid ($)

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper (excluding electronically filed sequence or computer listings under 37 CFR 1.52(e)), the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets		Extra Sheets		Number of each additional 50 or fraction thereof	Fee ($)		Fee Paid ($)
61	- 100 =	-39	/ 50 =	_____ (round up to a whole number)	x 250.00	=	0.00

4. OTHER FEE(S)

	Fees Paid ($)
NLate Declaration Surchage	130.00
Other (e.g., late filing surcharge): Assignment Recordation Fee	40.00

SUBMITTED BY

Signature		Registration No. (Attorney/Agent) 41844	Telephone 703.770.7743
Name (Print/Type)	Christine McCarthy		Date September 26, 2006

This collection of information is required by 37 CFR 1.136. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 30 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

SEP 26 2006
PATENT & TRADEMARK OFFICE

FOR UTILITY
ORIGINAL
DECLARATION

RULE 63 (37 C.F.R. 1.63)
DECLARATION AND POWER OF ATTORNEY
FOR PATENT APPLICATION
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

As a below named inventor, I hereby declare that my residence, post office address and citizenship are as stated below next to my name, and I believe I am an original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the INVENTION ENTITLED Methods and Apparatus for Dealing with Malware, the specification of which was filed on 30th June 2006 as U.S. Application No. 11/477807.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose all information known to me to be material to patentability as defined in 37 C.F.R. 1.56. Except as noted below, I hereby claim foreign priority benefits under 35 U.S.C. 119(a)-(d) or 365(b) of any foreign application(s) for patent or inventor's certificate, or 365(a) of any PCT International Application which designated at least one other country than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate, or PCT International Application, filed by me or my assignee disclosing the subject matter claimed in this application and having a filing date (1) before that of the application on which priority is claimed, or (2) if no priority claimed, before the filing date of this application:

PRIOR FOREIGN APPLICATION(S) Number	Country	Filed	Date First Laid Open Or Published	Date Patented or Granted	Priority Claimed
0513375.6	GB	30th June 2005			Yes

Except as noted below, I hereby claim domestic priority benefit under 35 U.S.C. 119(e) or 120 and/or 365(c) of the indicated United States applications listed below and PCT International applications listed above or below and, if this is a continuation-in-part (CIP) application, insofar as the subject matter disclosed and claimed in this application is in addition to that disclosed in such prior applications, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in 37 C.F.R. 1.56 which became available between the filing date of each such prior application and the national or PCT international filing date of this application.

PRIOR U.S. PROVISIONAL, NONPROVISIONAL AND/OR PCT APPLICATION(S) Application Number	Filed	Status pending, abandoned, patented	Priority Claimed

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

And I hereby appoint Pillsbury Winthrop Shaw Pittman LLP, Intellectual Property Group, (to whom all communications are to be directed), and persons of that firm who are associated with USPTO Customer No. 00909 individually and collectively my attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith and with the resulting patent, and I hereby authorize them to delete from that Customer No. names of persons no longer with their firm, to add new persons of their Firm to that Customer No., and to act and rely on instructions from and communicate directly with the person/assignee/attorney/firm/organization who/which first sends/sent this case to them and by whom/which I hereby declare that I have consented after full disclosure to be represented unless/until I instruct the above Firm and/or an attorney of that Firm in writing to the contrary.

Power of Attorney to Customer Number 00909

INVENTOR'S SIGNATURE: Date:

Name	Melvyn		MORRIS
	First	Middle Initial	Family Name
Residence	Turnditch	United Kingdom	GB
	City	State/Foreign Country	Country of Citizenship
Mailing Address	Redmire Gap, Intake Lane, Turnditch, Derbyshire DE56 2LU, United Kingdom		

INVENTOR'S SIGNATURE: Date:

Name	Paul		Stubbs
	First	Middle Initial	Family Name
Residence	Wyboston	United Kingdom	GB
	City	State/Foreign Country	Country of Citizenship
Mailing Address	7 Rookery Road, Wyboston, Bedfordshire MK44 3AX, United Kingdom		

BEST AVAILABLE COPY

INVENTOR'S SIGNATURE: M. Hartwig Date:

Name	Markus		HARTWIG
	First	Middle Initial	Family Name
Residence	Milton Keynes	United Kingdom	DE
	City	State/Foreign Country	Country of Citizenship
Mailing Address	Exchange House, 494 Midsummer Boulevard, Milton Keynes, Buckinghamshire MK9 2EA, United Kingdom		

INVENTOR'S SIGNATURE: D Harter Date:

Name	Darren		Harter
	First	Middle Initial	Family Name
Residence	Hucclecote	United Kingdom	GB
	City	State/Foreign Country	Country of Citizenship
Mailing Address	178 Barnwood Road, Hucclecote, Gloucester Gloucestershire GL4 3JZ, United Kingdom		

BEST AVAILABLE COPY



Attorney Docket: 011765-0355986
Client Reference: AF/JG/P10717US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re PATENT APPLICATION of: MELVYN MORRIS ET AL.
Application No.: 11/477,807 Confirmation No: 4785

Filed: June 30, 2006 Group Art Unit: 2131

Examiner: Unassigned

Title: METHODS AND APPARATUS FOR DEALING WITH MALWARE

RESPONSE TO NOTICE TO FILE MISSING PARTS

Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Notice to File Missing Parts of Application mailed July 27, 2006, a copy of which is enclosed, Applicants herewith submit the following:

- Executed Combined Declaration and Power of Attorney of Inventors
- Assignment and Recordation Cover Sheet

Please see attached PTO/SB/17 for payment of fees.

Respectfully submitted,
PILLSBURY WINTHROP SHAW PITTMAN LLP

By _____
Christine H. McCarthy
Registration No. 41844
Telephone: (703) 770.7743

Date: September 26, 2006
P.O. Box 10500
McLean, VA 22102
Customer No.: 00909

400462530v1

SEP 26 2006
PATENT & TRADEMARK OFFICE
IAP56

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NUMBER
11/477,807	06/30/2006	Melvyn Morris	011765-0355986

CONFIRMATION NO. 4785

00909
PILLSBURY WINTHROP SHAW PITTMAN, LLP
P.O. BOX 10500
MCLEAN, VA 22102

FORMALITIES LETTER

Date Mailed: 07/27/2006

NOTICE TO FILE MISSING PARTS OF NONPROVISIONAL APPLICATION

FILED UNDER 37 CFR 1.53(b)

Filing Date Granted

Items Required To Avoid Abandonment:

An application number and filing date have been accorded to this application. The item(s) indicated below, however, are missing. Applicant is given **TWO MONTHS** from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The statutory basic filing fee is missing.
  Applicant must submit $ **300** *to complete the basic filing fee for a non-small entity. If appropriate, applicant may make a written assertion of entitlement to small entity status and pay the small entity filing fee (37 CFR 1.27).*
- The oath or declaration is missing. *A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.*
  Note: If a petition under 37 CFR 1.47 is being filed, an oath or declaration in compliance with 37 CFR 1.63 signed by all available joint inventors, or if no inventor is available by a party with sufficient proprietary interest, is required.

The applicant needs to satisfy supplemental fees problems indicated below.

The required item(s) identified below must be timely submitted to avoid abandonment:

- Additional claim fees of **$2350** as a non-small entity, including any required multiple dependent claim fee, are required. Applicant must submit the additional claim fees or cancel the additional claims for which fees are due.
- To avoid abandonment, a surcharge (for late submission of filing fee, search fee, examination fee or oath or declaration) as set forth in 37 CFR 1.16(f) of $130 for a non-small entity, must be submitted with the missing items identified in this letter.

09/27/2006 CCHAU1 00000111 033975 11477807

01 FC:1011 300.00 DA
02 FC:1111 500.00 DA
03 FC:1311 200.00 DA
04 FC:1051 130.00 DA
05 FC:1202 1350.00 DA
06 FC:1201 1000.00 DA

SUMMARY OF FEES DUE:

Total additional fee(s) required for this application is **$3480** for a Large Entity

- **$300** Statutory basic filing fee.
- **$130** Surcharge.

- The application search fee has not been paid. Applicant must submit **$500** to complete the search fee.
- The application examination fee has not been paid. Applicant must submit **$200** to complete the examination fee for a large entity

- Total additional claim fee(s) for this application is **$2350**

  - **$1000** for **5** independent claims over 3.
  - **$1350** for **27** total claims over 20.

Replies should be mailed to: Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

A copy of this notice MUST be returned with the reply.

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199, or 1-800-972-6382
PART 1 - ATTORNEY/APPLICANT COPY

OIPE IAP56 SEP 26 2006 PATENT & TRADEMARK OFFICE

PTO/SB/17 (01-06)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL
For FY 2006

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 3,520.00

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	MELVYN MORRIS
Examiner Name	
Art Unit	2131
Attorney Docket No.	011765-0355986

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): _____

☒ Deposit Account Deposit Account Number: 033975 Deposit Account Name: PILLSBURY WINTHROP SHAW PITTMAN LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☒ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

☒ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☒ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

FEE CALCULATION (All the fees below are due upon filing or may be subject to a surcharge.)

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
		Small Entity		Small Entity		Small Entity	
Application Type	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	1,000.00
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

Fee Description	Fee ($)	Small Entity Fee ($)
Each claim over 20 (including Reissues)	50	25
Each independent claim over 3 (including Reissues)	200	100
Multiple dependent claims	360	180

Total Claims		Extra Claims	Fee ($)		Fee Paid ($)
47	- 20 or HP =	27	x 50.00	=	1,350.00

HP = highest number of total claims paid for, if greater than 20.

Indep. Claims		Extra Claims	Fee ($)		Fee Paid ($)
8	- 3 or HP =	5	x 200.00	=	1,000.00

HP = highest number of independent claims paid for, if greater than 3.

Multiple Dependent Claims	
Fee ($)	Fee Paid ($)

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper (excluding electronically filed sequence or computer listings under 37 CFR 1.52(e)), the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets		Extra Sheets		Number of each additional 50 or fraction thereof	Fee ($)		Fee Paid ($)
61	- 100 =	-39	/ 50 =	_____ (round up to a whole number)	x 250.00	=	0.00

4. OTHER FEE(S)

	Fees Paid ($)
NLate Declaration Surchage	130.00
Other (e.g., late filing surcharge): Assignment Recordation Fee	40.00

SUBMITTED BY

Signature		Registration No. (Attorney/Agent) 41844	Telephone 703.770.7743
Name (Print/Type)	Christine McCarthy		Date September 26, 2006

This collection of information is required by 37 CFR 1.136. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 30 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

OIPE IAP44
OCT 31 2006
PATENT & TRADEMARK OFFICE

Attorney Docket: 011765-0355986
Client Reference: AF/JG/P10717US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re PATENT APPLICATION of: MELVYN MORRIS , ET AL.
Application No.: 11/477,807

Confirmation Number: 4785

Group Art Unit: 2131

Filed: June 30, 2006

Examiner: Unassigned

Title: METHODS AND APPARATUS FOR DEALING WITH MALWARE

PRELIMINARY AMENDMENT

Commissioner for Patents
P.O. Box 1450
Alexandria,VA 22313-1450

Sir:

Prior to examination of the above-identified application, please amend the application as outlined herein. There are no amendments to the claims.

IN THE SPECIFICATION:

Page 3, please amend the first paragraph as indicated below:

In the system of ~~WO-A-2004/0083408~~ US-A-2004/0083408, a worm in a file is detected by examining connection attempts made by the specific file running on a computer.

REMARKS

The foregoing amendment to the specification corrects an inadvertent typographical error in the application country designation code of a cited patent reference. The country code "WO" has been amended to correctly reflect the correct country code for the application referenced, i.e., "US". No new matter is added, and entry of the amendment is therefore respectfully requested and deemed in order.

Please charge any fees associated with the submission of this paper to Deposit Account Number 033975. The Commissioner for Patents is also authorized to credit any over payments to the above-referenced Deposit Account.

Respectfully submitted,

PILLSBURY WINTHROP SHAW PITTMAN LLP
CHRISTINE H. MCCARTHY
Reg. No. 41844
Tel. No. 703.770.7743
Fax No.

Date: October 31, 2006
P.O. Box 10500
McLean, VA 22102
(703) 770-7900

OCT 31 2006
PATENT & TRADEMARK OFFICE

Docket Number: 011765-0355986
Client Reference: AF/JG/P10717US

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re the Application of

MELVYN MORRIS, et al.

Group Art Unit: 2131

Application No.: 11/477,807

Examiner:

Filed: June 30, 2006

Confirmation No.: 4785

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

Pursuant to 37 CFR 1.56, the attention of the Patent and Trademark Office is hereby directed to the reference(s) listed on the attached PTO-SB/08b. Copies of the references are provided herewith. It is respectfully requested that the information be expressly considered during the prosecution of this application, and that the references be made of record therein and appear among the "References Cited" on any patent to issue therefrom. Applicants respectfully request the Examiner return an initialed copy of the enclosed Form PTO-SB/08b to Applicants with the next Office communication to indicate that the references have been considered, per MPEP § 609.

This Information Disclosure Statement is being filed before the mailing date of the first Office Action on the merits in the present application. No certification or fee is required.

Respectfully Submitted,
PILLSBURY WINTHROP SHAW PITTMAN LLP

Christine H. McCarthy
Registration Number 41844
Customer Number 00909

Date: October 30, 2006
Telephone: (703) 770-7900
Facsimile: (703) 770-7901
P.O. Box 10500
McLean, VA 22102

OIPE IAP44
OCT 31 2006
PATENT & TRADEMARK OFFICE

PTO/SB/08a (07-05)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	11/477,807
Filing Date	06/30/2006
First Named Inventor	MELVYN MORRIS
Art Unit	2131
Examiner Name	
Attorney Docket Number	011765-0355986

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 02/33525 A2	04/25/2002			
		WO 03/021402 A2	03/13/2003			
		WO 2004/097602 A2	11/11/2004			
		EP 1 280 040 A3	03/24/2004			
		EP 1 549 012 A1	06/29/2005			

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 April 2002 (25.04.2002)

PCT

(10) International Publication Number
WO 02/33525 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/SG01/00213

(22) International Filing Date: 17 October 2001 (17.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
200005973-3 17 October 2000 (17.10.2000) SG

(71) Applicant and
(72) Inventor: CHUANG, Shyne-Song [SG/SG]; Singapore Post Centre, P.O. Box 493, Singapore 914017 (SG).

(74) Agent: DREW & NAPIER LLC; 20 Raffles Place, #17-00 Ocean Towers, Singapore 048620 (SG).

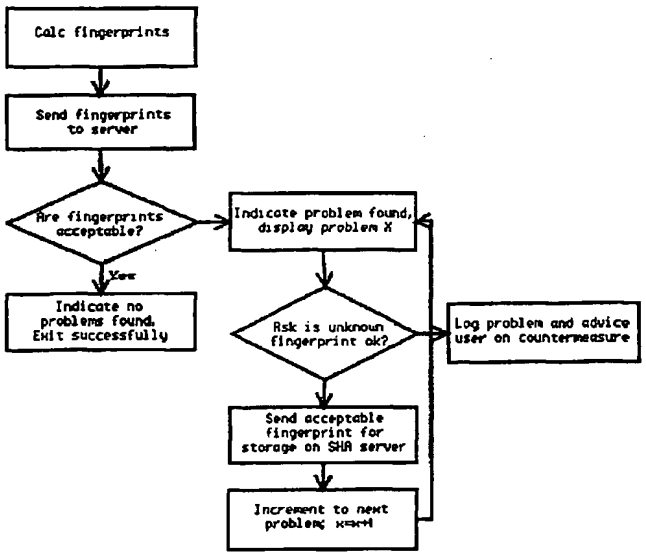
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A METHOD AND SYSTEM FOR DETECTING ROGUE SOFTWARE



(57) Abstract: A method of detecting rogue software includes the step of creating a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, wherein the pre-calculated fingerprints are calculated using one or more cryptographic formulae. The one or more cryptographic formulae are then used to calculate fingerprints of files on a computer system which is to be scanned for rogue software. The fingerprints calculated for the files on the computer system are compared with the fingerprints which are contained in the first database of pre-calculated fingerprints. Files on the computer system which may contain rogue software are identified by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

WO 02/33525 A2

BEST AVAILABLE COPY

A METHOD AND SYSTEM FOR DETECTING ROGUE SOFTWARE

Field of the invention

The present invention relates to a method and system for detecting rogue software such as trojan horses, root-kits, viruses and other unauthorized software which masquerades as valid software) on a computer system or data processing device such as a personal digital assistant. It relates particularly but not exclusively to a method and system for calculating and comparing fingerprints for files which are used either on a stand-alone computer system or on a computer system which is part of a computer network.

Back-ground to the Invention

Undesired rogue software is a nuisance and security threat. As computer systems and other information devices become even more interconnected with modern day networking technology and the Internet, the danger from rogue software has magnified considerably. Instead of being programmed to do damage once, today's rogue software can continue to receive commands and do the bidding of an unauthorized intruder for an extended period of time, effectively giving the creator of the rogue software continuous illegal access to a computer system.

One example of rogue software is the so-called trojan horse. Such software may be installed by innocent users unknowingly (whether via social engineering or otherwise) or it may be installed by an attacker when a system has been broken into. These trojan horses are back doors which allow an attacker to reconnect back into the compromised system and illegally access files and make unauthorized changes.

A trojan horse typically consists of new software and has new functionality. It is installed on a compromised system and disguised to look like original system software whenever necessary, so as to avoid detection. Sometimes, the trojan

horse is a modified piece of original system software and is almost identical to the one it replaced. However, other techniques are also used to obfuscate its existence.

Once the trojan horse is installed, a user continues operating his/her system without knowing that an intruder is now able to access his/her data illegally whilst remaining hidden. These are highly relevant problems which are encountered on a day to day basis. Trojan horses such as "Back Orifice" or "Netbus" hit PC systems in the late 1990s, and "root-kits" are a concern for various UNIX systems.

Normally, according to current practice, little is done to prevent or detect such rogue software. Anti-virus vendors maintain databases of rogue software signatures, and their software searches for files on a system associated with all known rogue software. Unfortunately, this technique has inherent scaling problems - the more signatures there are, the slower the scan process for each file. More importantly, the only rogue software types that can be detected are the ones that the anti-virus vendors know about. If the rogue software is, as an example, a custom trojan horse built by an expert professional hacker for penetrating a specific target, none of the anti-virus vendors will know about it. Therefore none of the anti-virus tools will be looking for it and the attacker and their trojan horse will exist completely undetected.

A more recent incremental innovation with this technology involves smarter scanning engines. Aside from looking for signatures of known rogue software, they are also able to look for software code that appears to be doing unusual things. This allows the scanning engine to detect additional rogue software that may not be known and whose signatures may not be in the database. However, this approach also has limitations. Trojan horses can be encrypted or compressed using special proprietary algorithms or encryption keying material. The rogue software is shipped in an encrypted and/or compressed format where it appears to be gibberish to a scanner. This rogue software is then decompressed or decrypted upon execution on the victim's computer system. A single trojan horse can thus be encrypted or compressed into thousands of

possibilities, each with its own unique signature. Traditional scanning technology will fail miserably when attempting to detect this type of rogue software, since there is no way that anti-virus engineers can keep track of thousands of mutations of the same piece of rogue software.

To summarise, today's scanning techniques for detecting rogue software fail for two main reasons:

1. They cannot detect unknown rogue software that has not already been identified. This is a serious problem because it is this kind of rogue software that may involve professional hackers and therefore warrant serious attention.
2. They cannot efficiently detect rogue software which has mutated, using new methods of compression and/or encryption. This problem exists to a large extent even for rogue software that is already known.

Another approach for detecting rogue software is to ensure that a system's files have not been altered, rather than looking for signatures of rogue software. If a system has no added files and all files remain unchanged from their original, unaltered state, it is clear that no rogue software is present on the system.

Academic work at Purdue University by Gene Kim and Gene Spafford resulted in a product called Tripwire which is now commercially sold. The product requires users to generate a database containing fingerprints of files on a system when the system is freshly loaded and in a pristine state. Subsequently, fingerprints can be recalculated and compared with the database of original pristine fingerprints and detect changes which have been made to the computer system.

This technology requires users to generate a database of fingerprints of a system's files while it is still pristine and free from alteration. This is not always feasible because many systems would already have been placed on public networks and exposed to risk for some time (often years). Since changes can be detected only by calculating new fingerprints and comparing them with the

database of original fingerprints, any rogue software which already exists when the original fingerprint database was generated will not be detected.

Existing products do not use a central database of fingerprints which are acceptable for a broad collection of system and application software. Therefore users need to make the following tedious and expensive steps when installing a software upgrade:

1. Schedule downtime in which to create the new database of fingerprints;
2. Re-calculate fingerprints to ensure that no rogue software has been added;
3. Install the software upgrade; and
4. Generate a new fingerprint database.

This is often a time-consuming and costly exercise.

Summary Of The Invention

Therefore it is an object of the present invention to provide a more reliable and effective method of identifying rogue software on a computer system or device, especially rogue software with unknown signatures or characteristics. The invention is preferably usable on systems or devices that have already been exposed to risk of intrusion by rogue software, and in cases where no fingerprints for the files on the system were calculated or archived when the system or device was known to be in a pristine state.

According to a first aspect of the invention, there is provided a method for detecting rogue software including the steps of:

(a) creating a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, wherein the pre-calculated fingerprints are calculated using one or more cryptographic formulae;

(b) using the one or more cryptographic formulae to calculate fingerprints of files on a computer system which is to be scanned for rogue software;

(c) comparing the fingerprints calculated for the files on the computer system with the fingerprints which are contained in the first database of pre-calculated fingerprints; and

(d) identifying files on the computer system which may contain rogue software by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

According to a second aspect of the invention, there is provided a system for detecting rogue software including:

(a) a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, the fingerprints having been calculated using one or more cryptographic formulae;

(b) a software component which uses one or more cryptographic formulae to calculate fingerprints for files on a computer system; and

(c) a software component which compares the calculated fingerprints for the files on the computer system with corresponding pre-calculated fingerprints stored in the first database, such that files on the computer system which may contain rogue software are identified by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

Brief Description Of The Drawings

The invention will now be described in greater detail by reference to the drawings which show an example form of the invention. It is to be understood that the particularity of the drawings does not supersede the generality of the foregoing description of the invention.

Figure 1 is a schematic representation of a client portion and server portion of a security system on a Redhat Linux platform connected via a network according to a preferred embodiment of the present invention.

Figure 2 illustrates a more detailed data flow diagram relating to the schematic representation of Figure 1.

Detailed Description Of A Preferred Embodiment Of The Invention

Figure 1 is a schematic representation of a client portion and server portion of a security system on a Redhat Linux platform connected via a network 10 according to a preferred embodiment of the present invention. The system includes a client 12, a server 14 and a database of acceptable file fingerprints 16. Communication between the client 12 and server 14 may be via the Internet 18, using the TCP/IP protocol. The system is first set up by calculating and archiving fingerprints for all files relating to operating system or application software used in a typical Redhat Linux system, perhaps from original Redhat CDs or other secure software distribution methods. This software can be installed on test systems (not shown in Figure 1) so that the new files added or replaced can be fingerprinted and profiled. These new fingerprints, the file location of each file added or replaced, and other information, can then be stored in the database of acceptable file fingerprints 16. An alternative method that eradicates the need for software installation on a test system involves the use of a custom developed program that understands the RPM (Redhat Package Manager) format of software packages on the Redhat Linux CD. By examining each RPM software package's installation instructions, the program determines the file location and calculates the fingerprints of each file to be installed. This information and other information is then stored in the database of acceptable file fingerprints 16.

The fingerprints are preferably calculated using one or more cryptographic formulae. In the preferred embodiment, such cryptographic formulae may include hash functions to generate hash values for each file, or asymmetric cryptographic functions to generate digital signatures for each file. The original version of the files as well as patches, updates/upgrades of all types of operating system or application software should be fingerprinted. System performance and reliability will improve as more operating system and application software is fingerprinted and archived.

Hashing is a contraction of the file contents created by a cryptographic hash function. A hash value (or simply hash) is the output when an arbitrary input is passed into a hash function. The hash is substantially smaller than the input itself, and is generated by a formula in such a way that it is extremely unlikely that slight modifications of the input will result in the same hash. Hashes conventionally play a role in security systems where they are used to ensure that transmitted messages have not been tampered with. As an illustration, a sender generates a hash of the message and sends it with the message itself. The recipient then calculates another hash from the received message, and compares the two hashes. If they are the same, there is a very high probability that the message was transmitted intact. There may be other equivalent methods for calculating fingerprints that may be implemented as the relevant technology develops.

The system's client component is installed on the client 12 that requires file integrity protection. During the first time the client component is executed, the client software recurses through the file system and calculates and stores the cryptographic hash of every single file on the system. When the file system has been completely traversed, the client software makes a secure TCP/IP connection via the Internet 18 to the server component on the server, which usually resides on premises remote from the client component. However, the client component need not be physically located remote from the server component.

For security purposes, it is preferable that bi-directional authentication takes place between the client component and the server component before any further communication and this can be done with SSL (Secure Socket Layer) or TLS (Transport Layer Security). In a nutshell, the server presents its digital certificate to the client software and the client uses its hardwired CA (Certificate Authority) public credentials to verify the CA signature on the server's certificate. If the signature is authentic and the server's address matches the machine which the certificate was issued to, the client can be certain that the server is who it claims to be. Subsequently, the same thing happens in the reverse

direction. The client presents the server with its digital certificate and the server goes through the same process to verify that the client is who it claims to be. This practice is very common today and is an industry standard method of mutually authenticating two nodes communicating with one another. Other authentication methods may also be used.

The calculated hash results and gathered basic client system information from the client 12 are then transferred to the server 14 for validation. On the server 14, each hash result for each file on the client system is compared against what are the expected hash values given certain parameters such as the client system's operating system version and software patch/update level. This expected hash information is fetched from the database of acceptable file fingerprints 16 which houses all the pre-calculated hash values for all files in various operating systems and applications.

A report is then generated on the fly and returned to the client 12. This report lists the files on the client which are possibly unsafe since they do not represent authentic software from the vendor. There are 3 possible results for each file:

(a) the hash result matches so the given file on the client is definitely authentic;

(b) the database of acceptable fingerprints 16 has no information on such a file in the database and it is uncertain if the file is authentic;

(c) the hash result does not match the fingerprint in the database of acceptable fingerprints 16 and the file is suspicious.

Armed with this report, the systems administrator for the client server 12 can then verify each of the files in categories (b) and (c). Outcomes in categories (b) and (c) are typically from files that are part of an internal customer specific application that the database 16 will not contain. If the administrator verifies the hash with the owners of the application, the authenticity of the file can be determined. This should be done for all questionable files in the report so that a client system can be certified as 100% authentic. If some of the questionable files cannot be resolved via these means, it is likely that they have been

augmented by rogue software and should be replaced or the system should be reinstalled.

Using additional management software, the administrator can then check off all remaining questionable files as acceptable and the security system will take the additional hashes into account in all subsequent runs. These additional hashes can then be stored in a second database (not shown in Figure 1) so that they can be considered when checking other systems from the same customer - this is a configurable feature.

Figure 2 illustrates a more detailed data flow diagram relating to the schematic representation of Figure 1. Using the database of pre-calculated acceptable fingerprints 16, the system will be able to determine if any given file on a client's system is authentic, i.e. not invaded by rogue software. When comparisons are done, file location, time stamps, platform information, user preferences and other parameters can also be taken into consideration.

The system should be continuously updated with new fingerprint information in the database of acceptable file fingerprints 16 as new software and updates become available. The system thus provides pristine fingerprint information that is made available to the file integrity checking software installed on a client's computer system. Instead of identifying bad files, the system therefore ensures that the data is good. Instead of requiring users to have generated a fingerprint database some time back, the system provides pre-calculated fingerprints and greatly reduces the barriers to adoption of this important file integrity technology.

In addition to the above, the system may also store fingerprints of various customers' files in a separate database (not shown in Figure 1) so that the system can provide heuristic, statistically based best effort guesses on whether a certain fingerprint is acceptable for a given file.

In the above example, in the event of a questionable outcome (categories (b) and (c)) where the system does not have pre-calculated hash information on a

certain file on the client, the system may also render a heuristic result on whether a file is safe. This result can be provided by accessing the second database (not shown in Figure 1) which contains hashes that the customer's administrators have confirmed to be acceptable. For example, if the system does not know about whether a file such as "/usr/bin/myspecialprogram" should have a hash result of "xyz", it can inform the administrator, and also point out one of the following:

1) no other systems in the client's class have such a file so authenticity is uncertain;
2) other systems in the client's class which have such a file do not have a hash of "xyz" so the file is suspicious;
3) other systems in the client's class which have such a file have the same hash of "xyz" so the file is probably safe.

This information can also be provided with a percentage figure so that administrators have a best guess of where they stand before engaging in manual verification as described above.

As the client base grows, this information will allow the system to make increasingly improved guesses. The system can render an opinion along the lines of "10,000 other customers have this file and 9,985 of them have the same fingerprint, so your file is probably safe" - perhaps a common application whose fingerprint that does not already exist in the first database 16. Such information, while not substantive, allows users to zoom into more critical anomalies on their systems sooner. For example, consider this other response: "10,000 other customers have this file and no one has the same fingerprint as you. Worse yet, all these 10,000 customers have the same fingerprint so your file is most probably unsafe." The system can thus provide a percentage or quantifiable risk rating in either a numeric fashion or with the use of colours.

From the client's point of view, the advantage is that even systems currently deployed in risky public network environments can be easily reliably scanned and put onto a file integrity protection regime without re-installation to assure a pristine state and with significantly reduced downtime. As customers apply upgrades to their systems, the system will similarly be able to verify that new

software being installed is authentic since the fingerprints of the new software should be in the system's database 16. Conversely, the system can be programmed to warn the user if the update contains software the system does not believe is authentic.

While a particular embodiment of the invention has been shown and described, it will be obvious to those skilled in the art that changes and modifications of the present invention may be made without departing from the invention in its broader aspects. As such, the scope of the invention should not be limited by the particular embodiment and specific construction described herein but should be defined by the appended claims and equivalents thereof. Accordingly, the aim in the appended claims is to cover all such changes and modifications as fall within the spirit and scope of the invention.

Claims:

1. A method of detecting rogue software including the steps of:

(a) creating a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, wherein the pre-calculated fingerprints are calculated using one or more cryptographic formulae;

(b) using the one or more cryptographic formulae to calculate fingerprints of files on a computer system which is to be scanned for rogue software;

(c) comparing the fingerprints calculated for the files on the computer system with the fingerprints which are contained in the first database of pre-calculated fingerprints;

(d) identifying files on the computer system which may contain rogue software by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

2. A method according to claim 1 including the further step of generating a list of questionable files on the computer system for which the calculated fingerprints do not correspond to pre-calculated fingerprints stored in the first database.

3. A method according to claim 1, wherein the cryptographic formulae used in the pre-calculation of fingerprints which are stored in the first database and in the calculation of fingerprints for files on the computer system, use one or more hash functions to generate hash values for each file.

4. A method according to claim 1, wherein cryptographic formulae used in the pre-calculation of fingerprints which are stored in the first database and in the calculation of fingerprints for files on the computer system, use one or more asymmetric cryptographic functions to generate digital signatures for each file.

5. A method according to claim 2 wherein questionable files are considered by a system administrator and may be marked by the system administrator as acceptable.

6. A method according to claim 5 wherein the fingerprints for questionable files which are accepted by the system administrator are stored in a second database.

7. A method according to claim 6 which includes the step of calculating the probability that a questionable file has been corrupted by rogue software, by comparing its fingerprint with fingerprints for similar files that have previously been accepted by the system administrator and stored in the second database.

8. A method according to claim 7 wherein the system provides statistical information regarding:

(a) the number of fingerprints in the second database which represent files with the same characteristics as the questionable file;

(b) the number of fingerprints in the second database which are identical to the fingerprint of the questionable file;

(c) the number of fingerprints in the second database which are different to the fingerprint of the questionable file.

9. A method according to claim 5 wherein files that are not acceptable are replaced or reinstalled.

10. A method according to claim 6 wherein the step of calculating fingerprints for files on the computer system and the step of comparing fingerprints on the computer system with corresponding pre-calculated fingerprints stored on the first database are both implemented by the computer system and wherein verification of questionable files takes place before fingerprints from the computer system are added to the second database.

11. A method according to claim 10 wherein the computer system is physically remote from the first database and communication between the

computer system and the first database takes place over a network such as the Internet.

12. A method according to any one of claims 6 to 8 wherein the step of calculating fingerprints for the files on the computer system is implemented by the computer system and the step of comparing the fingerprints which represent files on the client system with corresponding pre-calculated fingerprints stored in the database is implemented by a server, and wherein verification of questionable files takes place between the computer system and the server before the corresponding fingerprints are transferred from the computer system to the second database.

13. A method according to claim 12 wherein the computer system is physically remote from the server and communication between them takes place over a communications network such as the Internet.

14. A system for detecting rogue software including:

(a) a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, the fingerprints having been calculated using one or more cryptographic formulae;

(b) a software component which uses one or more cryptographic formulae to calculate fingerprints for files on a computer system; and

(c) a software component which compares the calculated fingerprints for the files on the computer system with corresponding pre-calculated fingerprints stored in the first database, such that files on the computer system which may contain rogue software are identified by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

15. A system according to claim 14 including a software component which generates a list of questionable files for which the calculated fingerprints do not correspond to the pre-calculated fingerprints which are stored in the first database.

16. A system according to claim 14 or 15 wherein the software components are installed on the computer system.

17. A system according to claim 14 wherein the pre-calculation of fingerprints which are stored in the first database and calculation of fingerprints for files on the computer system use one or more hash functions to generate hash values for each file.

18. A system according to claim 14 wherein the pre-calculation of fingerprints which are stored in the database and calculation of fingerprints for files on the computer system use one or more asymmetric cryptographic functions to generate digital signatures for each file.

19. A system according to claim 15 further including a second database in which fingerprints of questionable files which are found to be acceptable by a system administrator are stored.

20. A system according to claim 15 wherein the system calculates the probability that a questionable file is a file that has been corrupted by rogue software, by comparing its fingerprint with fingerprints for similar files that have been verified and stored in the second database.

21. A system according to claim 20 wherein the system produces statistical information regarding:

(a) the number of fingerprints in the second database which represent files with the same characteristics as the questionable file;

(b) the number of fingerprints in the second database which are identical to the fingerprint of the questionable file;

(c) the number of fingerprints in the second database which are different to the fingerprint of the questionable file.

22. A system according to claim 19 wherein files that are not acceptable are replaced or reinstalled.

23. A system according to any one of claims 14 to 22 wherein the step of calculating fingerprints of files on the computer system and the step of comparing the fingerprints on the computer system with corresponding pre-calculated fingerprints stored on the database are both implemented by the computer system and wherein verification of questionable files takes place before fingerprints from the computer system are added to the second database.

24. A system according to claim 23 wherein the computer system is physically remote from the first database and communication between the computer system and the first database takes place over a network such as the Internet.

25. A system according to any one of claims 14 to 20 wherein the step of calculating fingerprints for the files on the computer system is implemented by the computer system and the step of comparing the fingerprints which represent files on the computer system with corresponding pre-calculated fingerprints stored in the first database is implemented by a server and wherein verification of questionable files takes place between the computer system and the server before the corresponding fingerprints are transferred between them.

26. A system according to claim 25 wherein the computer system is physically remote from the server and communication between them takes place over a communications network such as the Internet.

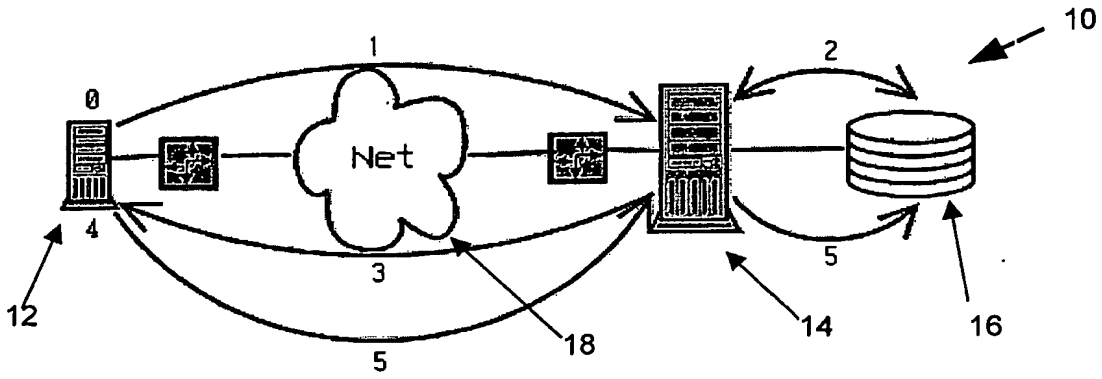


FIGURE 1

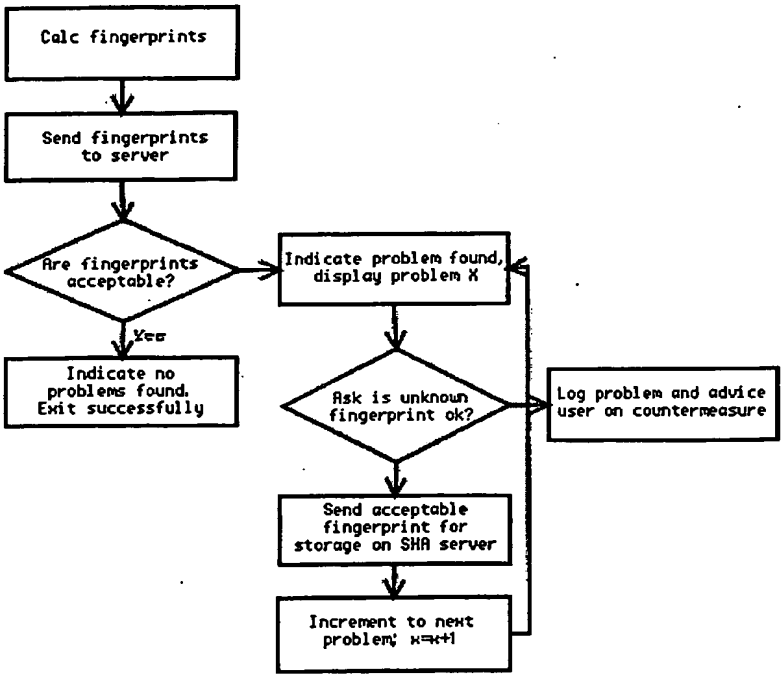


FIGURE 2

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 March 2003 (13.03.2003)

PCT

(10) International Publication Number
WO 03/021402 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/GB02/04059

(22) International Filing Date:
5 September 2002 (05.09.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0121497.2 5 September 2001 (05.09.2001) GB

(71) Applicant *(for all designated States except US)*: **CRYPTIC SOFTWARE LIMITED** [GB/GB]; The Business Exchange, Rockingham Road, Kettering, Northants NN16 8JX (GB).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **DUKE, David, John** [GB/GB]; 3 The Saplings, Kettering, Northants NN16 9UL (GB).

(74) Agent: **CRAWFORD, Andrew, B.**; AA Thornton & Co., 235 High Holborn, London WC1V 7LE (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 03/021402 A2

(54) Title: NETWORK SECURITY

(57) Abstract: A computer network comprises a server computer and a plurality of local computers. Each local computer is provided with an engine program which monitors all files and peripherals and detects charges in the files or configuration. The results of the monitoring are used to create compressed information relating to the files and peripherals which is communicated to the server computer which stores previous information relating to the files and peripherals of the local computers. The server computer compares the received information with the stored information in order to detect the presence of altered and unwanted files or changes in configuration. In this way both known and unknown viruses can be detected as can un-authorised use of the local computers.

BEST AVAILABLE COPY

NETWORK SECURITY

The present invention relates to an arrangement for monitoring the security of a computer especially computers on a network and particularly individual devices and information contained therein.

It is well known to provide individual computers with virus checking software but it is also well known that such software consumes resources when it is run and also takes time. Additionally, updating such software is somewhat cumbersome as each individual computer has to be updated separately.

Further, it is known to provide firewalls to protect computers and computer systems which are connected to a network or to the outside world but again these firewalls have limitations and they do not provide a completely effective defence to potential hackers.

Both of these methodologies are reactive and only detect threats to a network which have been seen and indexed previously.

It is an object of the present invention to provide an arrangement whereby protection from attack by viruses is improved as well as providing a more effective firewall in the event that the computer or computer system is provided with such. It is a further object that the invention is deployed in such a way that it is itself not vulnerable to attack.

From one aspect, the present invention is based on a concept whereby a main memory holds a record of all files used by a local computer, is provided with update information on all existing files as well as all new files and determines whether the updated files or new files represent a security threat.

In a preferred embodiment, as data is created, modified or deleted on the local computer, events are monitored and collected by an engine program and forwarded to the main memory. Preferably, the engine program itself consists of a

single file which is arranged to run a program in parallel with the normal operating system of the local computer thus making the file easier to hide so that the presence of the engine program will not be noticed either by the user or by a hacker. Further, as far as the user of the local computer is concerned, the engine program has no noticeable effect on the system. Preferably, the main memory is provided by a central computer (hereafter referred to as the console) and the local computer communicates with the console over a network or in some other convenient manner.

The engine program monitors each and every file and by inspecting preselected parts of each file can create a highly compressed accurate image of the file which can then be transmitted to the central database. It will be appreciated that all files can be handled in this way. Consequently, when the term "file" is used we mean all files associated with the operating system, data, registries, directories, hardware, software and such like. In this way, a complete virtual image of a local computer can be stored within the central database and the console can have management software for monitoring changes in the data files, programs or hardware of the local computer and thus warn of tampering with such files, programs and/or hardware.

From another aspect, the present invention provides a method of monitoring the security of a computer system comprising monitoring a file as it is created or updated by inspecting preselected portions of the file selected from the whole file, storing information derived from the preselected portions and transmitting the stored information to a main memory location. The main memory location is preferably a central database of a network.

From a further aspect the present invention provides a method of improving the security of computer apparatus by providing two copies of a security program and interconnecting them such that as one is switched off, either deliberately or

inadvertently, the other is automatically switched on and vice versa. It is to be understood that this aspect is not limited to any particular type of security program and is of general application. However, it has particular application with the present invention where the security program is relatively small and easy to hide.

In order that the present invention be more readily understood, an embodiment thereof will now be described with reference to the accompanying drawings in which: -

Fig 1 shows a block diagram of a network according to the present invention;

Fig 2 shows a block diagram representing the main functions of an engine program arranged to be installed on each computer to be protected; and

Fig 3 is a flow chart for explaining the operation of the engine program of Fig 1.

The present invention will be described in relation to its use for network security but it will be appreciated that the invention can be implemented in a number of different ways independently of any connection technology. In fact, the totality of the invention could be located within a single computer or a local computer could be in communication with a central server over conventional telephone lines rather than utilising any network protocol.

As shown in Fig 1, a central server 1 communicates via transmit/receive ports 2 with a number of local computers 3, each of which is provided with a monitoring engine program 4. The central server is provided with a plurality of databases, namely an archive of threats database 5, a virtual image of each computer database 6 and an audit database 7.

The basis of the present embodiment is that a virtual copy of a local computer and all its files will be kept in memory at the database to the central server computer with each file having its own individual characteristics stored at

the central computer. The virtual copies are created by the small engine programs 4 loaded on the local computers 3 which monitor all files by inspecting preselected portions of the files in order to create a so-called "fingerprint" of each file which is in fact a highly compressed version of the file. The fingerprint can then be very speedily forwarded to the central location where the current fingerprint can be compared with a previous fingerprint and any changes detected. The changes are then evaluated by the central computer 1 in order to determine the level of threat, if any, to the security of the local computer system and an appropriate signal sent back from the central server to the local computer 3 in the event that a particular file should not be opened or a particular program should not be run. Simultaneously, an event is displayed at the central computer and/or forwarded to other programs.

It will be appreciated that by the use of a virtual copy of the local computer and all its files, the central server can then monitor and log all changes to files, programs and/or hardware in order to provide evidence of breaches of security at the local computer. As an example of this, one could consider a computer fitted with several pieces of memory. Removal of all or part of this memory is an event which can be detected by the engine program and thus signalled to the central computer which can log the event.

In further explanation of the evidential facilities of this invention on the triggering of specified events, biometric information can be gathered in real time and forwarded to the central database. For example, if an event is detected, a screen shot of the local computer is gathered for evidence and at the same time the engine program can gather biometric information such as a photograph of the user of the local computer.

The above description represents a simple brief overall view of the arrangement in accordance with the present invention but for a more detailed

description we will now separate the operation of the engine program at the local computer and firstly describe this before we move onto a description of the operation of the monitoring program at the central server.

The engine program is shown diagrammatically in Fig 2 where all file data is monitored at monitoring section 11 under the control of a program section 12 which determines which parts of the data will be sampled. The selected parts of the data are then stored as indicated at 14 in order to create a virtual compressed copy of the file data for transmission as indicated at 16.

The engine program is arranged to commence running as soon as the computer on which it is loaded is powered up. The program then monitors all files as they are opened and in particular it monitors all changes to a file as indicated by the flow chart shown in Fig 3. Each and every change is inspected in order to determine whether the change is the creation of a new file. It also determines whether the change is a valid change. Each file is also checked by inspecting the file from a plurality of points of view by sampling predetermined portions of the file. This in turn creates a fingerprint or virtual copy of the file which is then assembled for transmission to a central database either over the network or via some other communications link. Among the predetermined portions of the file which the engine program inspects is the tag indicating whether or not the file is an executable. It also determines whether the file is a manipulated file, eg a zipped or encrypted file. It looks for the presence of capabilities such as keystroke logging, FTP server capability, IP notification, joystick controls, game libraries, etc

The engine program has a memory capability 12 so that it can store instructions received from the central processor as described below and then carry out those instructions in the event that the fingerprint it has assembled of a file indicates that action is necessary. As an example, if a known threat is present in a

file as indicated by the fingerprint, the engine program can immediately kill the threat in view of the fact that it has the necessary instruction previously received from the central server. In certain circumstances, the file in question can be automatically dealt with.

A particular feature of the engine program is that it does not wait to be polled by a central computer. Rather, it itself generates a message for transmission to the central computer. This has the advantage that the engine program is immune to attack from a hacker because there is no "listening" port waiting for an incoming transmission. It is not until the engine program has communicated with the central server that a two way communication is possible and only after the engine program's communication with the central server can the central server transmit modified information to the engine program. However, the monitoring and modification of the files on the local computer occur in real time while the local computer is in operation either by being started up or while the local computer is running programs. Further, the engine program can itself run other software in order to transmit and/or record data as a result of the detection of a particular event or change to a file. As an example of this, if the local computer is fitted with a web cam, should particular files be modified, the engine program can take a picture of the user of the machine at the appropriate time and correlate the event with the actual user as evidence for subsequent use.

In an especially secure system, it is possible to provide two identical engine programs which are interlinked with each other so that in the event that one is disabled for any reason, eg being switched off, the other is activated before the first is terminated and vice versa. This ensures security in the event that one of the engine programs is detected by a hacker or other person seeking to tamper with the local computer.

Turning now to the central server 1, this contains a reception port 2 for receiving data transmitted from the local computer. As fingerprints are stored, the central computer builds up a virtual picture of each local computer, its hardware, programs and files generally in its database 6. In addition, the central server maintains a database 5 of all known security threats and viruses. As fingerprints of files are received these are compared with the information in the threats database 5 so that any threat can be detected instantaneously. Additionally, because the central server maintains an up to date virtual copy of each local computer with which it is in communication, in the event that it identifies a security threat in relation to one computer 3, it can then inspect all other local computers for similar configurations and then be ready to communicate the necessary instruction to another local computer 3 as soon as the other local computer communicates with the central server for any reason. In this way, security threats and the mechanism of deleting files or repairing infected files can be undertaken much more quickly and in fact even before a local computer might have been subjected to attack or infection. In other words, when a new threat is identified, it is added to the server database so that the central processor can check its data relating to the local computers to which it is connected ot identify other infected systems without the need to communicate with them. Therefore even if a laptop is disconnected or machines are powered down, threats can still be identified and remedial action taken once such machines are back in communication with the central server.

In the same way, the central computer can monitor changes to hardware and software using the information in the database 7 and consequently know when a security breach might have taken place. In this case, the central computer can log the event in question or signal the local computer to take the necessary action to

provide evidence of the possible security breach, eg by taking a web cam picture as well as recording the user name and password.

Another sophisticated ability of the server is to construct a dynamic accurate focused detection formula which can be custom built to detect dynamic changing threats which avoid typical security fingerprinting techniques.

The central server can communicate back to the local computer in the event of a security breach or potential security breach or it can communicate with some other communications device such as a mobile telephone or personal digital assistant by sending an e-mail SMS or fax to indicate the existence of the security breach or potential security breach. In this way, users of portable computers could be warned of security breaches relating to their machine without the need to actually switch it on and establish communications with the central server.

CLAIMS:

1. A computer arrangement comprising a central processor, first memory means for storing an operating system program for the central processor, further memory means for storing data relating to all files used by the computer arrangement, storage means separate from the first storage memory means for storing an engine program for monitoring files used or created by the computer arrangement, and means for communicating the results of the monitoring to the further memory means.

2. A computer arrangement according to claim 1, wherein the engine program is arranged to inspect preselected parts of each file and create a compressed version of the file.

3. A computer arrangement according to claim 1, wherein the further memory means is located remote from the central processor and first memory means.

4. A computer arrangement according to claim 3, wherein the means for communicating results to the further memory means comprises a network connection.

5. A computer arrangement according to claim 1, wherein the storage means is arranged to store two engine programs each of which is capable of being switched off and which are interconnected such that as one is switched off, the other is switched on.

6. A computer arrangement according to claim 1, wherein the engine program is arranged to monitor peripheral devices connected to the arrangement in order to detect the presence of new devices and the absence of previously connected devices.

7. A computer arrangement according to claim 6, wherein the engine program is arranged to record data relating to any change in peripheral devices or files.

8. A computer arrangement according to claim 3, wherein the further memory means is located in a further computer which is arranged to analyse the results of the monitoring.

9. A computer arrangement according to claim 8, wherein the further computer is arranged to transmit instructions to the central processor depending on the results of the analysis.

1/2

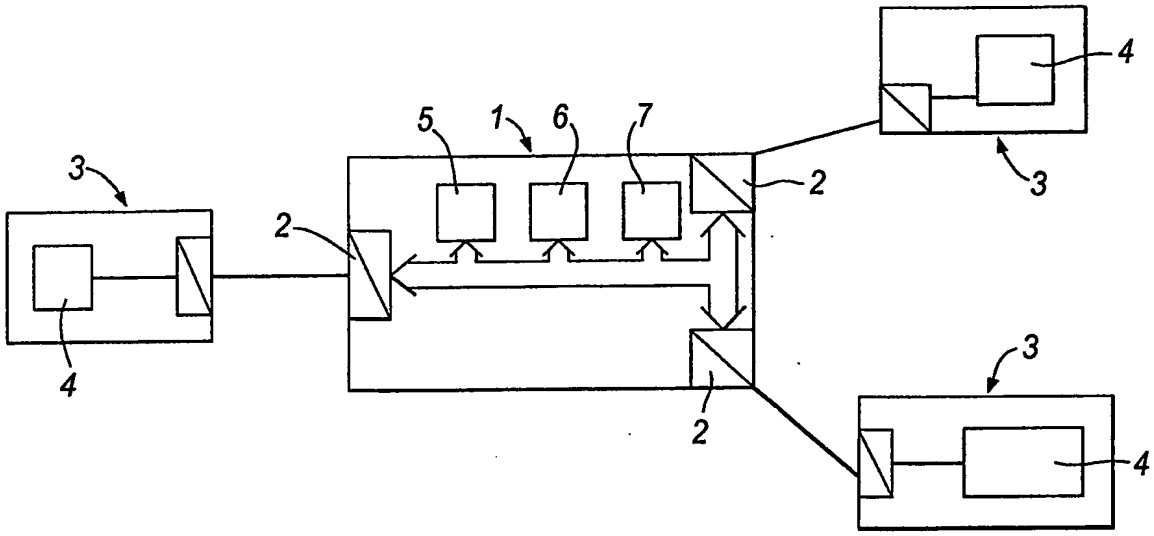


Fig.1

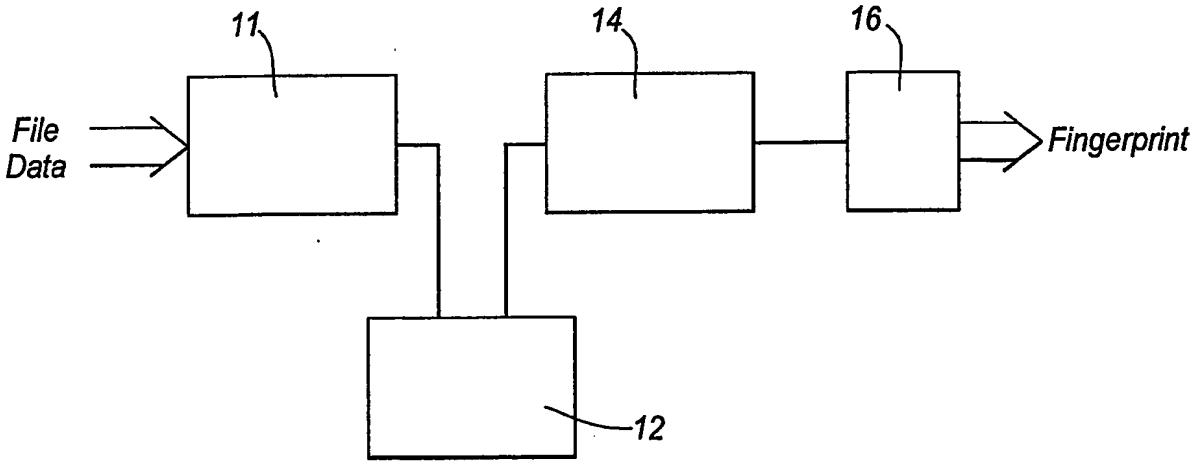


Fig.2

2/2

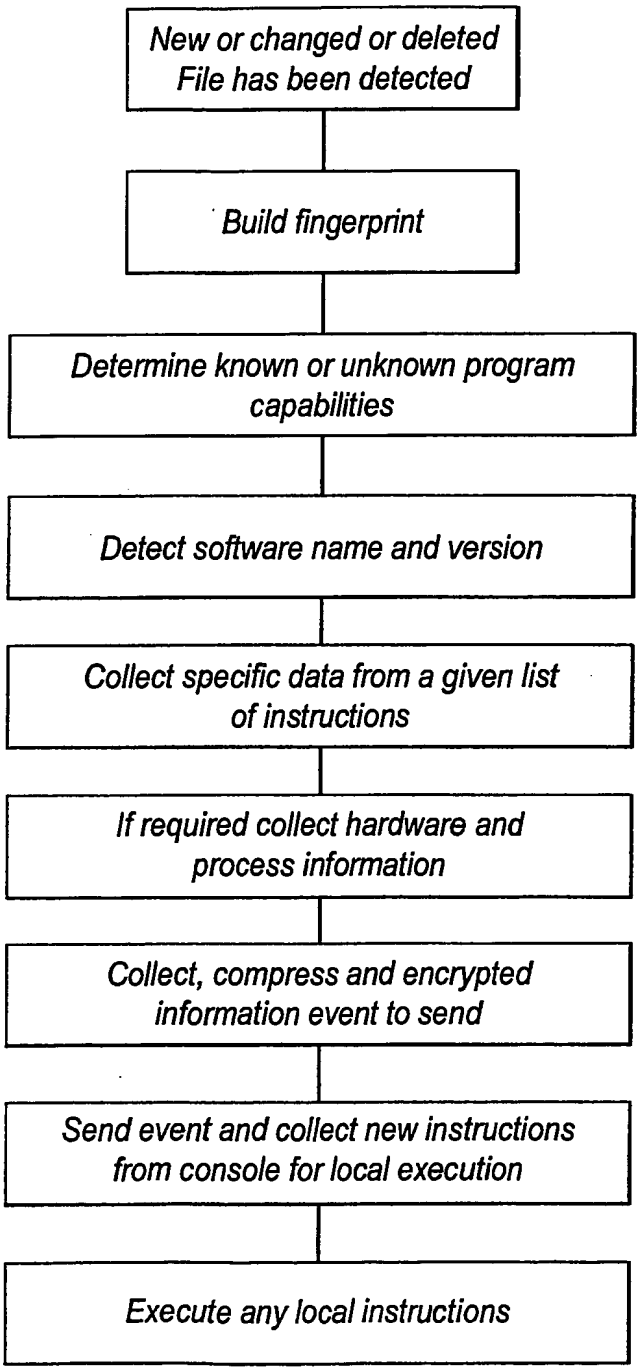

New or changed or deleted File has been detected
Build fingerprint
Determine known or unknown program capabilities
Detect software name and version
Collect specific data from a given list of instructions
If required collect hardware and process information
Collect, compress and encrypted information event to send
Send event and collect new instructions from console for local execution
Execute any local instructions

Fig.3

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 280 040 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
24.03.2004 Bulletin 2004/13

(43) Date of publication A2:
29.01.2003 Bulletin 2003/05

(21) Application number: **02253235.2**

(22) Date of filing: **09.05.2002**

(51) Int Cl.[7]: **G06F 1/00**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **26.07.2001 US 912392**

(71) Applicant: **Networks Associates Technology, Inc.**
Santa Clara, CA 95054 (US)

(72) Inventors:
- **Hinchliffe, Alexander James**
  Milton Keynes, Buckinghamshire MK7 7LL (GB)
- **Howard, Fraser Peter**
  Chipping Norton, Oxon OX7 3BT (GB)
- **Kemp, Andrew**
  Oxford, Oxon OX4 2QT (GB)
- **Rai, Bobby**
  Kettering, Northants NN15 5BP (GB)

(74) Representative:
Robinson, Nigel Alexander Julian
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Centrally managed virus scanning system

(57) A malware detecting system is described in which a client computer 8 sends a file access clearance request to an assessment computer 24 to determine whether or not access is permitted to that file. In this way, the different client computers on a network 2 may share their scan results and centralised control may be provided over access permissions to individual files or groups of files.

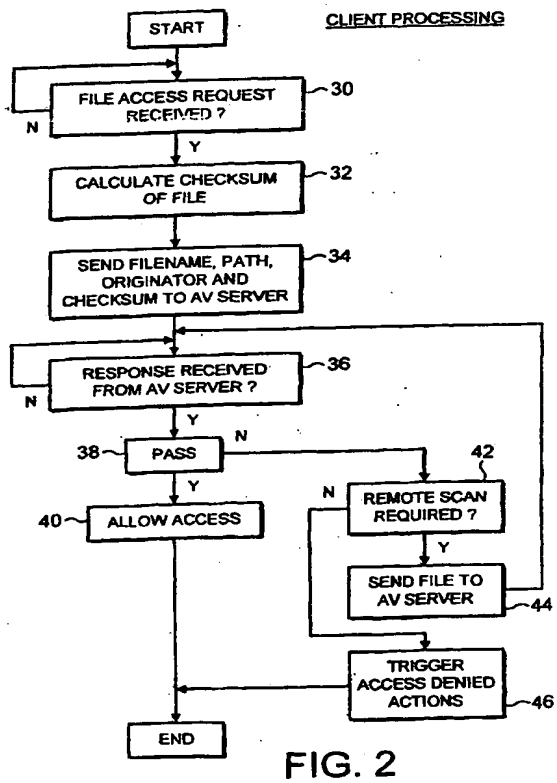


FIG. 2

BEST AVAILABLE COPY

EP 1 280 040 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 25 3235

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	US 6 195 587 B1 (HRUSKA JAN ET AL) 27 February 2001 (2001-02-27) * abstract * * column 1, line 1-9; figure 3 * * column 2, line 15-65; figures 1,2 * * column 3, line 5-25,40-67 * * column 5, line 1-60 * * column 4 *	1-13, 16-24, 28-41, 44-52, 55-68, 71-79,82	G06F1/00 G06F21/00
X	US 5 960 170 A (LAU STEVEN YUEN-LAM ET AL) 28 September 1999 (1999-09-28) * abstract * * column 1-10; figures 1-8 * * column 11, line 1-55 * * column 12-27 *	1,7,17, 29,35, 45,56, 62,72	
X	EP 1 111 507 A (SECURE OYJ F) 27 June 2001 (2001-06-27) * the whole document *	1,7,17, 29,35, 45,56, 62,72	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F

-/--

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	26 January 2004	Nazzaro, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 25 3235

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	MARMELSTEIN R E ET AL: "A distributed architecture for an adaptive computer virus immune system" SYSTEMS, MAN, AND CYBERNETICS, 1998. 1998 IEEE INTERNATIONAL CONFERENCE ON SAN DIEGO, CA, USA 11-14 OCT. 1998, NEW YORK, NY, USA,IEEE, US, 11 October 1998 (1998-10-11), pages 3838-3843, XP010310923 ISBN: 0-7803-4778-1 * abstract; figure 1 * * page 3841 * * page 3842 *	14,15, 25,26, 42,43, 53,54, 69,70, 80,81	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	26 January 2004	Nazzaro, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

European Patent Office

Application Number
EP 02 25 3235

CLAIMS INCURRING FEES

The present European patent application comprised at the time of filing more than ten claims.

☐ Only part of the claims have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims and for those claims for which claims fees have been paid, namely claim(s):

☐ No claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims.

LACK OF UNITY OF INVENTION

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

see sheet B

☒ All further search fees have been paid within the fixed time limit. The present European search report has been drawn up for all claims.

☐ As all searchable claims could be searched without effort justifying an additional fee, the Search Division did not invite payment of any additional fee.

☐ Only part of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the inventions in respect of which search fees have been paid, namely claims:

☐ None of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the invention first mentioned in the claims, namely claims:

European Patent Office

LACK OF UNITY OF INVENTION
SHEET B

Application Number
EP 02 25 3235

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

1. Claims: 1-13,16-24,28-41,44-52,55-68,71-79,82

Computer program , method and apparatus for detecting malware, comprising: detecting logic, file access clearance request generating and transmitting logic, file access clearance receiving logic at an assessment computer, file access clearance response generating and transmitting logic, file access clearance response receiving logic and file access permitting logic if file access clearance response indicates said file does not contain malware.

2. Claims: 14-15, 25-26, 42-43, 53-54, 69-70, 80-81

Computer program method and apparatus for detecting malware comprising detecting logic, file access clearance request generating and transmitting logic, file access clearance receiving logic at an assessment computer, file access clearance response generating and transmitting logic, file access clearance response receiving logic and file access permitting logic if file access clearance response indicates said file does not contain malware. The assessment computer is operable in a higher level security mode serving to deny access to greater range of computer files. The change from lower to higher level security mode is triggered from a remote computer.

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 02 25 3235

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

26-01-2004

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6195587 B1	27-02-2001	GB 2283341 A	03-05-1995
		AU 8000394 A	22-05-1995
		DE 69407812 D1	12-02-1998
		DE 69407812 T2	16-07-1998
		EP 0725951 A1	14-08-1996
		WO 9512162 A1	04-05-1995
		JP 9504395 T	28-04-1997
US 5960170 A	28-09-1999	AU 6537698 A	12-10-1998
		EP 0968466 A1	05-01-2000
		JP 2001515625 T	18-09-2001
		WO 9841919 A1	24-09-1998
EP 1111507 A	27-06-2001	GB 2353372 A	21-02-2001
		EP 1111507 A2	27-06-2001
		US 2001005889 A1	28-06-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 280 040 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.01.2003 Bulletin 2003/05

(51) Int Cl.[7]: **G06F 1/00**

(21) Application number: **02253235.2**

(22) Date of filing: **09.05.2002**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **26.07.2001 US 912392**

(71) Applicant: **Networks Associates Technology Inc.**
Santa Clara, CA 95054 (US)

(72) Inventors:
- **Hinchliffe, Alexander James**
  Milton Keynes, Buckinghamshire MK7 7LL (GB)
- **Howard, Fraser Peter**
  Chipping Norton, Oxon OX7 3BT (GB)
- **Kemp, Andrew**
  Oxford, Oxon OX4 2QT (GB)
- **Rai, Bobby**
  Kettering, Northants NN15 5BP (GB)

(74) Representative:
Robinson, Nigel Alexander Julian
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Centrally managed virus scanning system

(57) A malware detecting system is described in which a client computer 8 sends a file access clearance request to an assessment computer 24 to determine whether or not access is permitted to that file. In this way, the different client computers on a network 2 may share their scan results and centralised control may be provided over access permissions to individual files or groups of files.

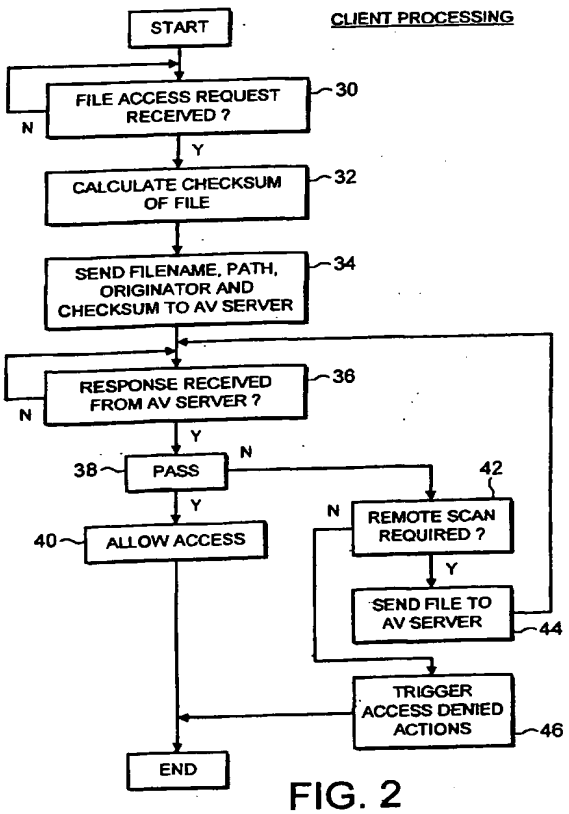


FIG. 2

EP 1 280 040 A2

Description

[0001] This invention relates to the field of data processing systems. More particularly, this invention relates to the field of malware scanning, such as, for example, scanning for computer viruses, Trojans, worms, banned computer files or E-mails containing banned words or content.

[0002] It is known to provide malware scanning systems and mechanisms for identifying malware within a computer file to be accessed. As the number of computer viruses and the like that are present in the wild increases the processing associated with scanning a computer file to identify the presence of any of those viruses similarly increases. This increase in the processing required is disadvantageous. Furthermore, with the increasing levels of network connectivity between computer systems and the use of E-mail and other fast messaging systems, the spread of computer viruses has become increasingly rapid. In a number of recent cases mass mailing viruses have spread at such a speed that considerable damage has been caused before appropriate countermeasures have been able to be put in place. The delay in deploying such countermeasures is further increased by the need in many cases to update virus definition data on every individual node computer to be protected. The speed of spread of recent viruses is such that the download and installation delays associated with installing such new countermeasures, even when they have been developed and are available, is a significant disadvantage.

[0003] An anti-virus system produced by Sophos uses a locally held record of previously conducted on-access scans on the individual computer in question to determine whether or not a scan should be conducted for a computer file or a previously determined result used instead when this is available.

[0004] Viewed from one aspect the present invention provides a computer program product for controlling a computer to detect malware, said computer program product comprising:

detecting logic operable to detect a file access request to a computer file by a requesting computer;
file access clearance request generating logic operable to generate a file access clearance request including data identifying said computer file;
file access clearance request transmitting logic operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
file access clearance response receiving logic operable to receive at said requesting computer a file access clearance response from said assessment computer; and
file access permitting logic operable if said file access clearance response indicates said computer file does not contain malware, to permit said file access request by said requesting computer.

[0005] The invention recognises that an individual computer requiring a file access may pass the task of determining whether or not that access should be allowed to a different assessment computer. Whilst at first sight this may seem that it would slow down the file access, in practice there are considerable advantages. For example, the assessment computer to which the task is passed may be more rapidly updated with new virus definition data as this is released than would be possible for the requesting computer. Accordingly, the requesting computer can benefit from the most up-to-date virus definition data more rapidly. Furthermore, the invention is particularly well suited to systems in which a plurality of requesting computers share an assessment computer since in many cases the individual requesting computers will show a high degree of correlation in the computer files to which they are requesting access and for which a malware scan is needed. Thus, rather than each requesting computer individually scanning the computer files that are also being scanned by a large number of other computers, (e.g. the computer files associated with the operating system shared by all the requesting computers), these computer files can instead be scanned once by the assessment computer and then each requesting computer can check that the computer file has been scanned and cleared by sending an appropriate request to the assessment computer rather than having to scan the file itself. Thus, in exchange for the use of a small amount of bandwidth on the connections between the requesting computers and the assessment computer, a considerable processing burden may be lifted from the requesting computers. The additional processing burden on the assessment computer does not increase disproportionately since the high degree of correlation between the computer files accessed by the different requesting computers means that in many cases the assessment computer will be able to respond to a clearance request from a requesting computer on the basis that it has already scanned that file and without the need to rescan that file.

[0006] In order to identify the computer file to be scanned in a secure manner to the assessment computer, the data identifying the computer file preferably includes a checksum value. Checksum values may be made sufficiently specific for a computer file from which they have been calculated so as to be difficult to bypass as a way of uniquely identifying a computer file.

[0007] Additional data that is highly useful to the assessment computer in managing the requests it receives includes the filename of the computer file, data identifying the requesting computer and the storage location of the computer file. This provides useful audit information as well as providing for the possibility of files having the same name and same storage location on different computers in fact being different files with dif-

ferent checksums, both of which should be treated as separate entities when determining whether or not a request from another requesting computer relates to a file that has already scanned.

[0008] In circumstances when a requesting computer wishes to access a file that has not already been scanned, the assessment computer may send a scan request message back to the requesting computer such that the requesting computer may send a copy of the file to the assessment computer for scanning. It will be appreciated that this may be slower than scanning the file locally, but the benefit will be that should a different requesting computer later wish to access that same file, then the scanning result produced by this action can be shared with that other computer, so speeding the operation of that other computer.

[0009] If the assessment computer determines that access should be denied to a computer file, then this may be used to trigger denied access actions within the requesting computer, the assessment computer, or elsewhere. Such denied access actions may include deletion of the computer file, repair of the computer file, quarantining of the computer file, generation of user warning messages, generation of administrator warning messages and the like.

[0010] Viewed from another aspect the invention also provides a computer program product for controlling a computer to detect malware, said computer program product comprising:

file access request receiving logic operable to receive at an assessment computer a file access clearance request from a requesting computer, said file access clearance request including data identifying a computer file to be accessed by said requesting computer;

file access clearance response generating logic operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response; and

file access clearance response transmitting logic operable to transmit said file access clearance response to said requesting computer.

[0011] The assessment computer may advantageously store the results of its previous scans within a database. This database may include a variety of fields relating to each computer file that has been scanned. These fields may include an access flag indicating whether access is to be denied to that computer file and a persistence flag indicating whether or not the entry for that computer file should be purged during purge operations.

[0012] The access flag is particularly useful as in addition to allowing the recording that access should be denied due to malware being detected, it also allows the central management of whether or not a particular individual file or class or type of file should be permitted to be accessed by all of those requesting computers that seek their access permissions from that assessment computer. This centralised control is a powerful tool that may be used to implement techniques such as the triggering of a lock down mode of operation in which higher level security provisions are put in place by denying access to certain files or types of files. As an example, if a message was received indicating that the higher security mode should be entered, then the assessment computer may use its mechanisms to deny access to any newly encountered computer file that had not previously been scanned and cleared for use. This would typically allow the large majority of computer activity to continue whilst providing protection against newly released malware threats until the appropriate countermeasures could be put in place.

[0013] The persistence flag allows control of the flushing of entries from the assessment computer. Whilst one of the advantages of the invention is storing the results of previously conducted scans such that they need not be repeated, this has to be tempered by allowing the results to be refreshed at a later time for at least some files. It is possible to envisage that a particular computer file carrying a newly released virus may not be detected as carrying that virus when it is first scanned, but later when a new virus driver is available, that computer file would be detected and blocked. Accordingly, as an example, it may be that all previous scan results could be purged from the system whenever the virus definition data was updated.

[0014] Viewed from a further aspect the invention provides a computer program product for controlling a computer to detect malware, said computer program product comprising:

file access request detecting logic operable to detect a file access request to a computer file by a requesting computer;

file access clearance request generating logic operable to generate a file access clearance request including data identifying said computer file;

file access clearance request transmitting logic operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;

file access clearance request receiving logic operable to receive at said assessment computer said file access clearance request from a requesting computer;

file access clearance response generating logic operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response;

file access clearance response transmitting logic operable to transmit said file access clearance response to said requesting computer;
file access clearance response receiving logic operable to receive at said requesting computer said file access clearance response from said assessment computer; and
file access permitting logic operable if said file access clearance response indicates said computer file does not contain malware to permit said file access request by said requesting computer.

[0015] As well as the complementary aspects of the invention embodied in the form of the client software, the server software and the combination of the client and server software, the invention may also take the form of corresponding methods of malware detection and apparatus for malware detection.

[0016] Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 schematically illustrates a computer network to which the present technique may be applied;

Figure 2 is a flow diagram illustrating an example of the processing that may be performed on a client computer;

Figure 3 is a flow diagram illustrating an example of the processing that may be performed on a server computer;

Figure 4 is a flow diagram illustrating an example process that may be run on a server computer waiting for a lock down signal;

Figure 5 schematically illustrates a database entry relating to a computer file that has previously been scanned; and

Figure 6 is a diagram schematically illustrating the form of a general purpose computer that may be used to implement the above described techniques.

[0017] Figure 1 schematically illustrates a computer network 2 comprising a plurality of client computers 4, 6, 8, 10, 12, 14, 16, 18 connected to respective departmental servers 20, 22 and to a corporate anti-virus (assessment) server 24. The computer network 2 is also connected via a network link to an anti-virus providers FTP server 26 from which virus definition data may be downloaded and from which messages, such as lock down messages or messages relating to the availability of new virus definition data may be received.

[0018] In operation a client computer 8 conducts processing activity during which it seeks to make a file access to a particular computer file (in this example, winword.exe). Rather than scanning this file itself for malware such as computer viruses, Trojans, worms, banned content etc, the client computer 8 instead passes an access clearance request relating to the computer file through to the assessment computer 24 via the departmental server 20. The access clearance request includes the file name of the computer file being accessed, a checksum derived from that file in an effort to uniquely identify it (e.g. an MD5 checksum), data identifying the name of the client computer 8, and the path used by that client computer 8 to access the computer file. The assessment computer 24 receives the access clearance request and uses the data identifying the computer file to look up within an associated database 28 as to whether or not a malware scan has already been conducted for that particular computer file. The computer file is uniquely identified by its filename and checksum value. If the computer file in question has already been scanned, then the result of that scan may be reused rather than conducting the scan again. The assessment computer 24 can accordingly pass back to the client computer 8 a clearance request response indicating whether or not access to that file is permitted or some other further action should be taken. The client computer 8 can then use this access clearance response to either permit access to the file or take the further action specified.

[0019] The further action specified could be that the computer file has already been identified as containing malware and accordingly an appropriate anti-virus or other anti-malware response should be initiated on the client computer 8. A corresponding response could already have been initiated on the assessment computer 24. The denied access responses could take the form of deleting the computer file concerned, repairing the computer file concerned, quarantining the computer file concerned, issuing a user alerting message, issuing an administrator alerting message, issuing a message to an anti-virus provider or some other action.

[0020] A different type of further action that could be required by the client computer 8 in response to the access clearance response arises when the assessment computer 24 has not previously scanned that file and accordingly the client computer 8 should send a copy of that file through to the assessment computer 24 to be scanned. When such a copy of the computer file has been sent through to the assessment computer 24 and scanned, an appropriate pass or denied access action triggering response can be sent back from the assessment computer 24 to the client computer 8.

[0021] It will be appreciated that the client computers 4, 6, 8, 10, 12, 14, 16, 18 operate to share the results of the malware scanning that is performed on their behalf by the assessment computer 24. Since there is likely to be a high degree of correlation between the files being accessed by the different client computers, in many cases the scan that a client computer is request-

ing will already have been performed on behalf of another computer and that scan need not be run again by the assessment computer but instead merely the result of the previous scan returned. The use of a checksum allows the computer files to be reliably uniquely identified.

[0022] The co-ordination of file access permissions by the assessment computer 24 also allows central management of which computer files may be accessed upon the network 2 in a manner that allows the rapid implementation of any higher security level mode should this be desired. As an example, a lock down trigger message may be received from the anti-virus provider FTP server 26 that will trigger the assessment computer 24 to enter a higher security level mode compared to its normal mode of operation. In the higher security level mode, it may be that the access to whole variety of different types of files may be temporarily banned. As an example, the access to VBS files which are often a source of viruses may be banned across the network 2, as could access to e-mail attachments which are another potential source of virus propagation. Thus, when a new threat is identified by the anti-virus provider, lock down messages may be used to trigger predetermined, in accordance with user configuration and preferences, higher security level modes within connected assessment computers 24 to provide a degree of protection for the networks concerned whilst allowing the majority of their normal operations to continue.

[0023] Figure 2 is a flow diagram illustrating the processing performed by a client computer. At step 30 the computer waits to receive a file access request from a computer program executing on that client computer. At step 32 a checksum value is calculated in accordance with one of several different possible checksum algorithms, such as the MD5 algorithm. At step 34 the filename, path, originating computer name and checksum value are sent through to the assessment computer. At step 36 the client computer waits to receive a response from the assessment computer. When that response is received, step 38 serves to determine whether or not the response indicates that the computer file had passed its scanning. If the computer file had passed its scanning and access is to be permitted, then processing proceeds to step 40 at which the access is allowed.

[0024] If the response is not that the computer file had passed its scan, then processing proceeds to step 42 at which a test is made as to whether or not the response indicated that a remote scan was required. If a remote scan was required, then the computer file in question is sent to the assessment computer at step 44 and processing is returned to step 36.

[0025] If the test at step 42 does not identify a remote scan request, then since the response is neither a pass, or a remote scan request, then the computer file must be one to which access is denied and accordingly step 46 serves to trigger the denied access actions. These may include deletion, repair, quarantining or other actions upon the computer file in question as well as the generation of appropriate warning messages to a user or an administrator.

[0026] Figure 3 illustrates the processing performed by the assessment computer. At step 48 the assessment computer waits to receive a request from a client computer. When a request is received, step 50 serves to identify whether a copy of a computer file is being returned to the assessment computer for scanning following a remote scan request that had earlier been issued by the assessment computer. If a copy of a computer file is being returned for scanning, then processing proceeds to step 52 at which the necessary malware scanning is conducted. Step 54 determines whether or not the computer file passed this malware scanning. If the computer file did pass the malware scanning, then step 56 serves to add details of that computer file to the database of scanned files held at the assessment computer relating to scans already performed. Step 58 then returns the pass result to the client computer.

[0027] If the scanning at step 52 was detected as not being passed at step 54, then processing proceeds to step 60 at which malware detected actions are triggered within the assessment computer. These malware detected actions may be similar to those previously described in relation to the client computer file. In addition, an entry specifying the malware scan fail of that computer file may also be added to the database. At step 62 an access denied result is returned to the client computer.

[0028] If the test at step 50 did not indicate that a computer file was being returned for scan, then step 64 serves to compare the file details being passed to the server with the database entries of computer files that have already been scanned. The computer file may be uniquely identified by its filename and its checksum. If at step 66 a match within the database is detected, then step 68 determines whether or not this entry indicates that access should be allowed to that computer file. If access is not to be allowed, then processing proceeds to step 60. If access is allowed, then processing proceeds to step 70 at which a pass result is returned to the client computer.

[0029] If the test at step 66 indicated that a match was not found within the database, then step 72 serves to return a remote scan required result to the client computer in order to trigger the client computer to return a copy of the computer file to the assessment computer for scanning at step 52.

[0030] Figure 4 illustrates a process that may run on the assessment computer as a background task. At step 74 the assessment computer waits to receive a lock down trigger message from the anti-virus provider. If such a message is received, then step 76 serves to activate a lock down mode in the assessment computer. The lock down mode can switch on a user predetermined set of measures intended to provide a higher degree of security, normally at the cost of at least some

functionality. As an example, access to e-mail attachments or VBS files as a class may be denied. These are known to be particular vulnerabilities.

[0031] At step 78 the assessment computer waits for a user command to cancel the lock down mode. When such a command is received, then step 80 serves to cancel the lock down mode.

[0032] Figure 5 schematically illustrates the data that may be stored for a particular computer file within the database of previously conducted scan results held by the assessment computer 24. For an individual computer file, its filename, the originating computer for which the scan of that file was first conducted, the path to the file, the checksum value for the file, an allowed access flag and a persistence flag are all stored. The allowed access flag may be used to indicate whether or not that file passed its scan result. The allowed access flag may also be used as a powerful tool for switching off or on access to individual files or classes of files by an administrator. The persistence flag controls how the entry is flushed from the database on a regular interval, such as when new virus definition data is received. It will be appreciated that other fields could be added to the database relating to the particular file as required.

[0033] Figure 6 schematically illustrates a general purpose computer 200 of the type that may be used to implement the above techniques. The general purpose computer 200 includes a central processing unit 202, a random access memory 204, a read only memory 206, a hard disk drive 208, a display driver 210 and display 212, a user input/output circuit 214 and keyboard 216 and mouse 218 and a network interface unit 220 all connected via a common bus 222. In operation the central processing unit 202 executes program instructions stored within the random access memory 204, the read only memory 206 or the hard disK drive 208. The working memory is provided by the random access memory 204. The program instructions could take a variety of forms depending on the precise nature of the computer 200 and the programming language being used. The results of the processing are displayed to a user upon the display 212 driven by the display driver 210. User inputs for controlling the general purpose computer 200 are received from the keyboard 216 and the mouse 218 via the user input/output circuit 214. Communication with other computers, such as exchanging e-mails, downloading files or providing internet or other network access, is achieved via the network interface unit 220.

[0034] It will be appreciated that the general purpose computer 200 operating under control of a suitable computer program may perform the above described techniques and provide apparatus for performing the various tasks described. The general purpose computer 200 also executes the method described previously. The computer program product could take the form of a recordable medium bearing the computer program, such as a floppy disk, a compact disk or other recordable medium. Alternatively, the computer program could be dynamically downloaded via the network interface unit 220.

[0035] It will be appreciated that the general purpose computer 200 is only one example of the type of computer architecture that may be employed to carry out the above described techniques. Alternative architectures are envisaged and are capable of use with the above described techniques.

Claims

1. A computer program product for controlling a computer to detect malware, said computer program product comprising:

detecting logic operable to detect a file access request to a computer file by a requesting computer;
file access clearance request generating logic operable to generate a file access clearance request including data identifying said computer file;
file access clearance request transmitting logic operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
file access clearance response receiving logic operable to receive at said requesting computer a file access clearance response from said assessment computer; and
file access permitting logic operable if said file access clearance response indicates said computer file does not contain malware, to permit said file access request by said requesting computer.

2. A computer program product as claimed in claim 1, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

3. A computer program product as claimed in claims 1 and 2, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

4. A computer program product as claimed in claims 1, 2 and 3, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then computer file transmitting logic is operable to transmit said computer file from said requesting computer to said assessment computer.

5. A computer program product as claimed in any

one of the preceding claims, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

6. A computer program product as claimed in any one of the preceding claims, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

7. A computer program product for controlling a computer to detect malware, said computer program product comprising:

file access request receiving logic operable to receive at an assessment computer a file access clearance request from a requesting computer, said file access clearance request including data identifying a computer file to be accessed by said requesting computer;
file access clearance response generating logic operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response; and
file access clearance response transmitting logic operable to transmit said file access clearance response to said requesting computer.

8. A computer program product as claimed in claim 7, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

9. A computer program product as claimed in any one of claims 7 and 8, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

10. A computer program product as claimed in any one of claim 7, 8 and 9, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then computer file receiving logic is operable to receive at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

11. A computer program product as claimed in any one of claims 7 to 10, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

12. A computer program product as claimed in any one of claims 7 to 11, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

13. A computer program product as claimed in claim 12, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

14. A computer program product as claimed in any one of claims 7 to 13, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

15. A computer program product as claimed in claim 14, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

16. A computer program product as claimed in any one of claims 7 to 15, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

17. A computer program product for controlling a computer to detect malware, said computer program product comprising:

file access request detecting logic operable to detect a file access request to a computer file by a requesting computer;
file access clearance request generating logic operable to generate a file access clearance request including data identifying said computer file;
file access clearance request transmitting logic operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
file access clearance request receiving logic

operable to receive at said assessment computer said file access clearance request from a requesting computer;
file access clearance response generating logic operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response;
file access clearance response transmitting logic operable to transmit said file access clearance response to said requesting computer;
file access clearance response receiving logic operable to receive at said requesting computer said file access clearance response from said assessment computer; and
file access permitting logic operable if said file access clearance response indicates said computer file does not contain malware to permit said file access request by said requesting computer.

18. A computer program product as claimed in claim 17, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

19. A computer program product as claimed in claims 17 and 18, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

20. A computer program product as claimed in any one of claims 17 to 19, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then computer file transmitting logic is operable to transmit said computer file from said requesting computer to said assessment computer, receiving at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

21. A computer program product as claimed in any one of claims 17 to 20, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

22. A computer program product as claimed in any one of claims 17 to 21, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

23. A computer program product as claimed in any one of claims 17 to 22, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

24. A computer program product as claimed in claim 23, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

25. A computer program product as claimed in any one of claims 17 to 25, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

26. A computer program product as claimed in claim 25, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

28. A computer program product as claimed in any one of claims 17 to 27, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

29. A method of detecting malware, said method comprising the steps of:

detecting a file access request to a computer file by a requesting computer;
generating a file access clearance request including data identifying said computer file;
transmitting said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
receiving at said requesting computer a file access clearance response from said assessment computer; and
If said file access clearance response indicates said computer file does not contain malware, then permitting said file access request by said requesting computer.

30. A method as claimed in claim 29, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

31. A method as claimed in claims 29 and 30, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

32. A method as claimed in any one of claims 29 to 31, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then transmitting said computer file from said requesting computer to said assessment computer.

33. A method as claimed in any one of claims 29 to 31, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

34. A method as claimed in any one of claims 29 to 33, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

35. A method of detecting malware, said method comprising the steps of:

receiving at an assessment computer a file access clearance request from a requesting computer, said file access clearance request including data identifying a computer file to be accessed by said requesting computer;
in dependence upon said data identifying said computer file determining if said computer file has previously been assessed as not containing malware and generating a file access clearance response; and
transmitting said file access clearance response to said requesting computer.

36. A method as claimed in claim 35, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

37. A method as claimed in claims 35 and 36, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

38. A method as claimed in any one of claims 35 to 37, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then receiving at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

39. A method as claimed in any one of claims 35 to 38, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

40. A method as claimed in any one of claims 35 to 39, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

41. A method as claimed in claim 40, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

42. A method as claimed in any one of claims 35 to 41, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

43. A method as claimed in any one of claims 35 to 42, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

44. A method as claimed in any one of claims 35 to 43, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

45. A method of detecting malware, said method comprising the steps of:

detecting a file access request to a computer file by a requesting computer;
generating a file access clearance request including data identifying said computer file;
transmitting said file access clearance request

from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
receiving at said assessment computer said file access clearance request from a requesting computer;
in dependence upon said data identifying said computer file determining if said computer file has previously been assessed as not containing malware and generating a file access clearance response;
transmitting said file access clearance response to said requesting computer;
receiving at said requesting computer said file access clearance response from said assessment computer; and
if said file access clearance response indicates said computer file does not contain malware, then permitting said file access request by said requesting computer.

46. A method as claimed in claim 45, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

47. A method as claimed in claims 45 and 46, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

48. A method as claimed in any one of claims 45 to 47, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then transmitting said computer file from said requesting computer to said assessment computer, receiving at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

49. A method as claimed in any one of claims 45 to 48, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

50. A method as claimed in any one of claims 45 to 49, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

51. A method as claimed in any one of claims 45 to 50, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

52. A method as claimed in claim 51, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

53. A method as claimed in any one of claims 45 to 52, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

54. A method as claimed in claim 53, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

55. A method as claimed in any one of claims 45 to 54, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

56. Apparatus for detecting malware, said apparatus comprising:

a detector operable to detect a file access request to a computer file by a requesting computer;
a file access clearance request generator operable to generate a file access clearance request including data identifying said computer file;
a file access clearance request transmitter operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
a file access clearance response receiver operable to receive at said requesting computer a file access clearance response from said assessment computer; and
a file access permission unit operable if said file access clearance response indicates said computer file does not contain malware, to permit said file access request by said requesting

computer.

57. Apparatus as claimed in claim 56, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

58. Apparatus as claimed in claims 56 and 57, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

59. Apparatus as claimed in any one of claims 56 to 58, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then a computer file transmitter is operable to transmit said computer file from said requesting computer to said assessment computer.

60. Apparatus as claimed in any one of claims 56 to 59, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

61. Apparatus as claimed in any one of claims 56 to 60, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

62. Apparatus for controlling a computer to detect malware, said apparatus comprising:

a file access request receiver operable to receive at an assessment computer a file access clearance request from a requesting computer, said file access clearance request including data identifying a computer file to be accessed by said requesting computer;
a file access clearance response generator operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response; and
a file access clearance response transmitter operable to transmit said file access clearance response to said requesting computer.

63. Apparatus as claimed in claim 62, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

64. Apparatus as claimed in claims 62 and 63, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

65. Apparatus as claimed in any one of claims 62 to 64, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then a computer file receiver is operable to receive at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

66. Apparatus as claimed in any one of claims 62 to 65, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

67. Apparatus as claimed in any one of claims 62 to 66, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

68. Apparatus as claimed in claim 67, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

69. Apparatus as claimed in any one of claims 62 to 68, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

70. Apparatus as claimed in claim 69, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

71. Apparatus as claimed in any one of claims 62 to 70, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

72. Apparatus for controlling a computer to detect

malware, said apparatus comprising:

a file access request detector operable to detect a file access request to a computer file by a requesting computer;
a file access clearance request generator operable to generate a file access clearance request including data identifying said computer file;
a file access clearance request transmitter operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
a file access clearance request receiver operable to receive at said assessment computer said file access clearance request from a requesting computer;
a file access clearance response generator operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response;
a file access clearance response transmitter operable to transmit said file access clearance response to said requesting computer;
a file access clearance response receiver operable to receive at said requesting computer said file access clearance response from said assessment computer; and
a file access permission unit operable if said file access clearance response indicates said computer file does not contain malware to permit said file access request by said requesting computer.

73. Apparatus as claimed in claim 72, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

74. Apparatus as claimed in claims 72 and 73, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

75. Apparatus as claimed in any one of claims 72 to 74, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then a computer file transmitter is operable to transmit said computer file from said requesting computer to said assessment computer, receiving at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

76. Apparatus as claimed in any one of claims 72 to 75, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

77. Apparatus as claimed in any one of claims 72 to 76, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

78. Apparatus as claimed in any one of claims 72 to 77, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

79. Apparatus as claimed in claim 78, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

80. Apparatus as claimed in any one of claims 72 to 79, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

81. Apparatus as claimed in claim 80, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

82. Apparatus as claimed in any one of claims 72 to 81, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied

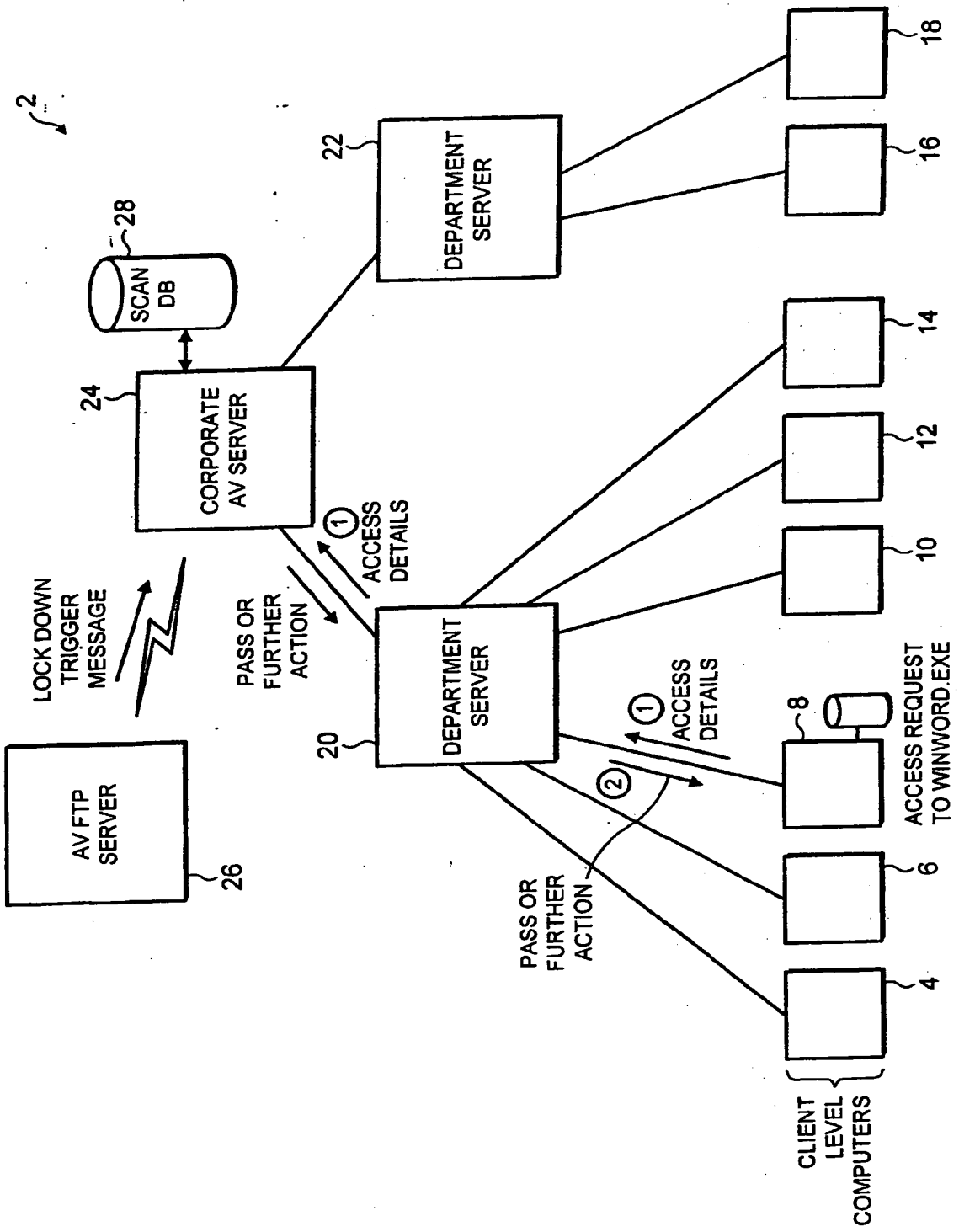

2
28
SCAN DB
24
CORPORATE AV SERVER
22
DEPARTMENT SERVER
18
16
14
12
10
LOCK DOWN TRIGGER MESSAGE
PASS OR FURTHER ACTION
1 ACCESS DETAILS
AV FTP SERVER
26
20
DEPARTMENT SERVER
1 ACCESS DETAILS
2
8
ACCESS REQUEST TO WINWORD.EXE
PASS OR FURTHER ACTION
6
4
CLIENT LEVEL COMPUTERS

FIG. 1

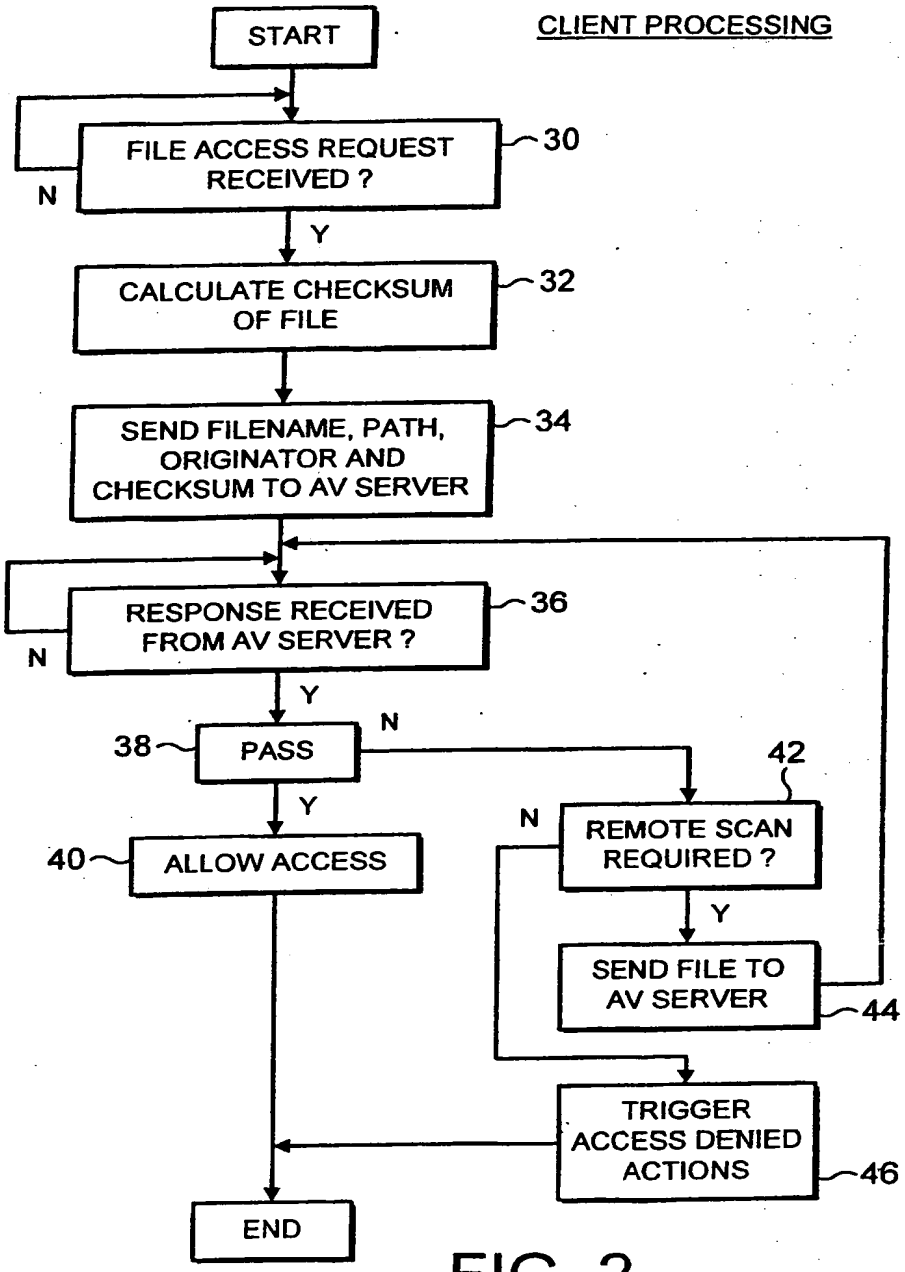

START
CLIENT PROCESSING
FILE ACCESS REQUEST RECEIVED ?
30
N
Y
CALCULATE CHECKSUM OF FILE
32
SEND FILENAME, PATH, ORIGINATOR AND CHECKSUM TO AV SERVER
34
RESPONSE RECEIVED FROM AV SERVER ?
36
N
Y
38
PASS
N
42
Y
N
REMOTE SCAN REQUIRED ?
40
ALLOW ACCESS
Y
SEND FILE TO AV SERVER
44
TRIGGER ACCESS DENIED ACTIONS
46
END

FIG. 2

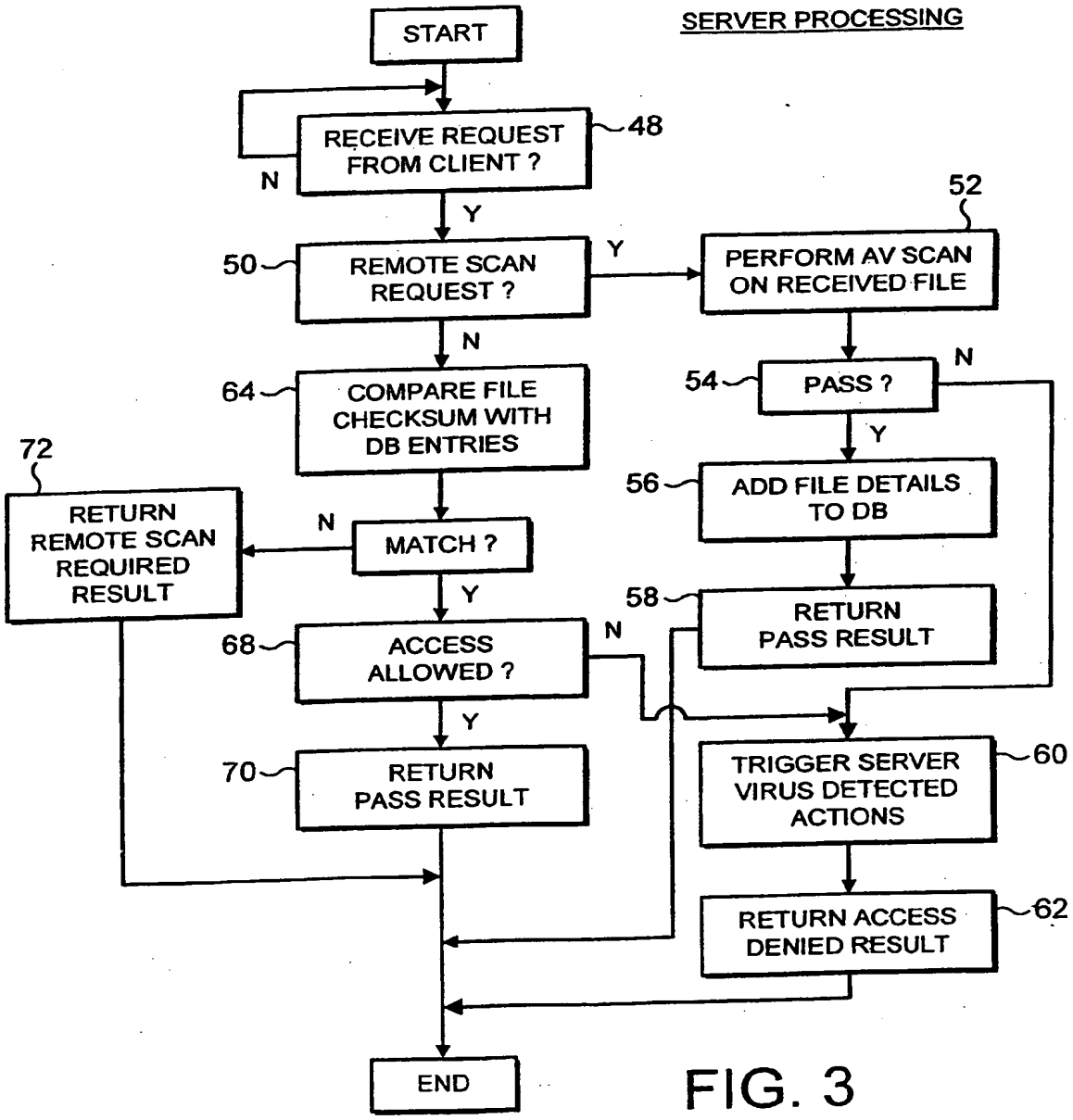

SERVER PROCESSING
START
RECEIVE REQUEST FROM CLIENT ?
48
N
Y
50
REMOTE SCAN REQUEST ?
Y
52
PERFORM AV SCAN ON RECEIVED FILE
N
64
COMPARE FILE CHECKSUM WITH DB ENTRIES
54
PASS ?
N
Y
72
RETURN REMOTE SCAN REQUIRED RESULT
N
MATCH ?
56
ADD FILE DETAILS TO DB
Y
58
RETURN PASS RESULT
68
ACCESS ALLOWED ?
N
Y
70
RETURN PASS RESULT
TRIGGER SERVER VIRUS DETECTED ACTIONS
60
RETURN ACCESS DENIED RESULT
62
END

FIG. 3

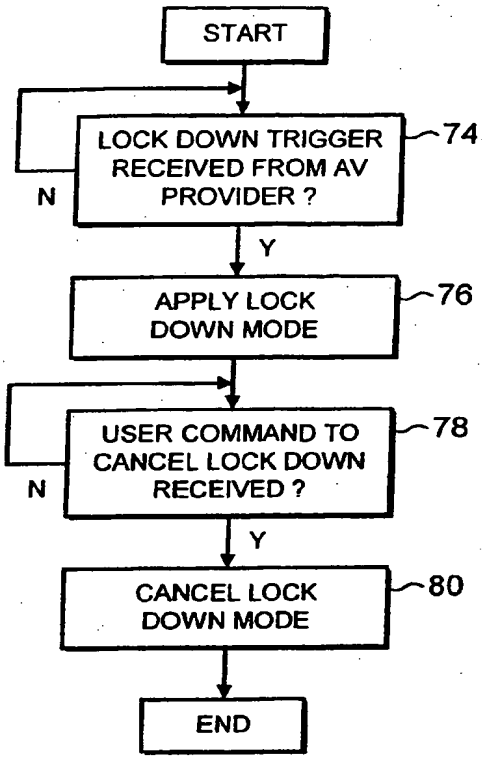

START
LOCK DOWN TRIGGER RECEIVED FROM AV PROVIDER ?
74
N
Y
APPLY LOCK DOWN MODE
76
USER COMMAND TO CANCEL LOCK DOWN RECEIVED ?
78
N
Y
CANCEL LOCK DOWN MODE
80
END

FIG. 4

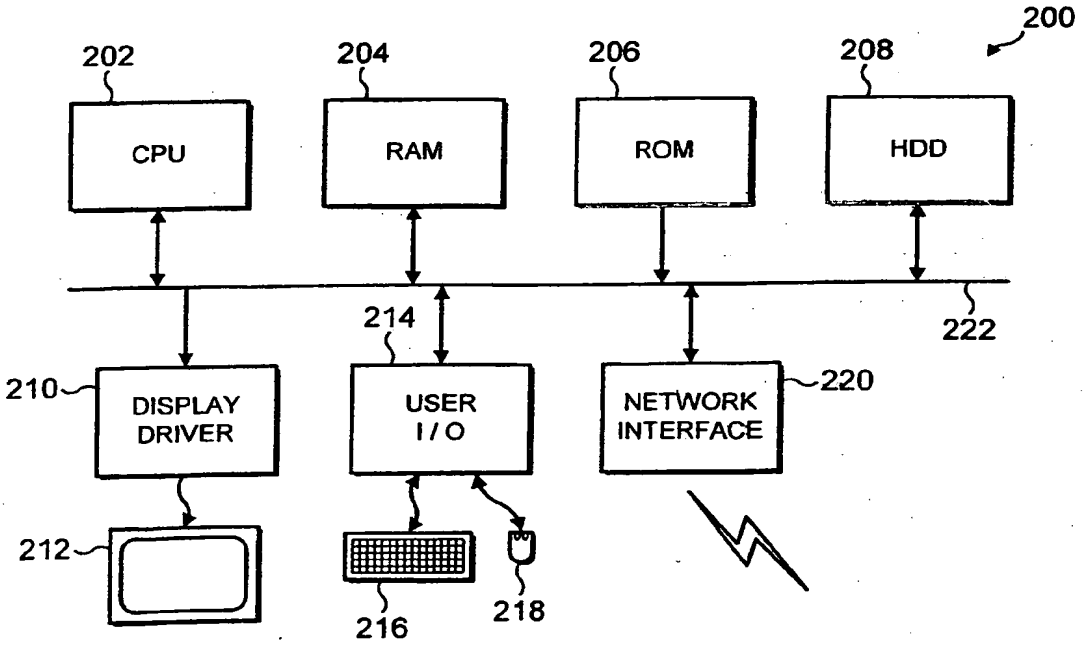

200
202
CPU
204
RAM
206
ROM
208
HDD
222
214
210
DISPLAY DRIVER
USER I/O
NETWORK INTERFACE
220
212
216
218

FIG. 6

FILENAME	ORIGINATOR	PATH	CHECKSUM	ACCESS ALLOWED	PERSISTENT
•	•	•	•	•	•
WINWORD.EXE	CLIENT N	C:\PROGRAM FILES\ MICROSOFT OFFICE\ OFFICE	[12345678]	Y	Y
•	•	•	•	•	•

FIG. 5

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 549 012 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.06.2005 Bulletin 2005/26

(51) Int Cl.[7]: **H04L 29/06**, G06F 1/00

(21) Application number: **03447310.8**

(22) Date of filing: **24.12.2003**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PT RO SE SI SK TR
Designated Extension States:
AL LT LV MK

(71) Applicant: **DataCenterTechnologies N.V.**
9080 Lochristi (BE)

(72) Inventor: **De Spiegeleer, Kristof**
8301 Knokke-Heist (BE)

(74) Representative: **Bird, William Edward et al**
Bird Goen & Co.,
Klein Dalenstraat 42A
3020 Winksele (BE)

(54) Method and system for identifying the content of files in a network

(57) A method and system for performing securing and controlling of a network using content identification of files in a network having a central infrastructure and local computing devices is presented. The method comprises calculating a hash value of a new file created or received on a local computing device, transmitting the hash value to the central infrastructure, comparing the hash value with a previously determined hash value stored in a database on the central infrastructure to determine whether the file is new to the network and if the file is new to the network, checking the file content with a content identifying engine, installed and updated on the central infrastructure. Content attributes are determined for the files which allow to perform appropriate actions on the local computing devices according to policy rules.

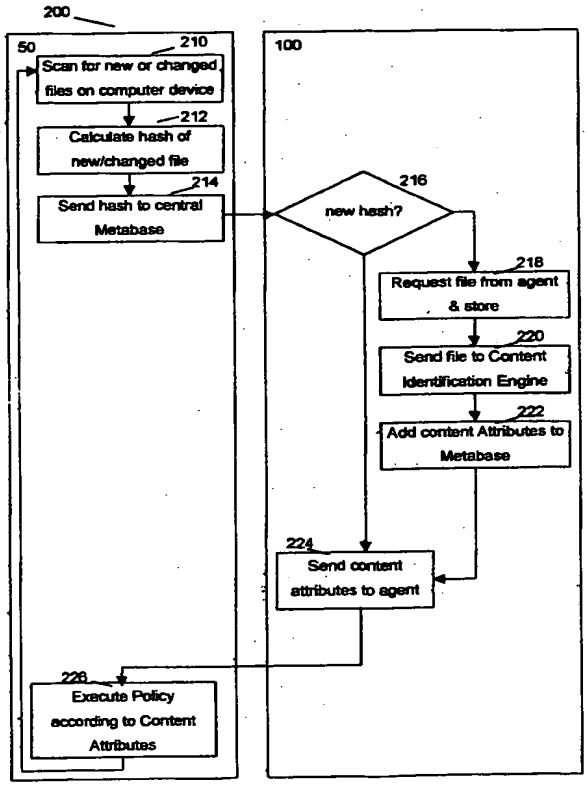


Fig. 3

EP 1 549 012 A1

BEST AVAILABLE COPY

Description

Technical field of the invention

[0001] The invention relates to a method and system to control the content of computer files, e.g. containing text or graphical data. More specifically, a method and system is described for checking and managing the security status and the content of computer files on a local computing device in a network environment.

Background of the invention

[0002] In today's world, computers are widely spread. Very often, especially in business environment, they are interconnected in small or larger networks. As software and data often are an important part of the investment goods of both private persons and firms, it is important to protect single computing devices and complete networks and their workstations against attacks from viruses, trojan horses, worms and malicious software. Another problem is related to the amount of files containing undesirable content such as explicit adult content. These files are often received on local computing devices uninvited and unwanted.

[0003] To solve the security problems associated with viruses, virus protection systems, also called virus checkers, have been developed. Some examples of conventional virus checkers are Norton AntiVirus, McAfee VirusScan, PC-cillin, Kaspersky Anti-Virus. Most of these conventional virus protection software packages can be configured so that they are continuously running in the background of the computing device and providing continuous protection. These virus protection systems compare codes of new or amended software with fingerprints (e.g. parts of code introduced in files by the viruses) of well known viruses. Other virus protection systems compare codes of all data available on the computing device. This leads to the use of a significant amount of central processing unit (CPU) time, which limits the capacity of the computing device for performing other tasks. Furthermore, the working principle of these virus checkers makes these software packages work rather reactive than proactive, as the fingerprint of the virus needs to be known in order for a virus scanning program to recognise it. This implies that the database of fingerprints needs to be updated very regularly in order to be secured against relatively new viruses. Consequently, the secure state of the computer is not only depending on external factors like the accurateness with which fingerprints of new viruses are made available by the suppliers of virus protection software packages, but also on the sense of duty of the user regarding performing updates regularly. If updates are provided centrally from a server automatically, then network capacity is reduced as these virus updates must be sent to each workstation.

[0004] In a network environment, the problem of updating such a database of fingerprints becomes significantly more important, as it implies that the responsibility is put to all users, who all have to update their virus checker database. Alternatively, the virus scanning could be performed by a central server, thus limiting the updating for new fingerprints to the central server. Nevertheless this implies that a large amount of data needs to be transferred over the network on a regular basis thereby utilising large amounts of expensive network bandwidth and possibly (depending on the number of clients for the server) overloading the network or server capacity for other activities.

[0005] In order to limit the amount of CPU time used, additional techniques have been developed to speed up the virus scanning process. These very often include hashing of the content of files. Hashing is one example of application of a "one-way-function". A one-way-function is an algorithm which when applied in one direction makes the reverse direction almost impossible to perform. A one-way-function generates a value such as a hash value by a calculation on the content of a file and can uniquely fingerprint this file if the one-way-function is complex enough to avoid duplicate values from different files. The uniqueness of a hashing function depends on the type of hashing function that is used, i.e. the size of the digest that is formed and the quality of function. Good hashing functions have the fewest collisions in a table, i.e. the chance of providing the same hash value for different files is the smallest. As mentioned, this is also determined by the size of the digest, i.e. hash value, that is calculated. If e.g. a 128-bit digest is used, the number of possible different values that can be obtained is 2^{128}.

[0006] It is known to use hashing for virus checking, possibly in a network environment. Typically, a hash of an application selected to run on a local computer is calculated, a stored hash from a database on a secured computer is retrieved on the local computer, whereby the secured computer can be a secured part of the local computer or a network server, and both values are compared. If there is a match, the application is executed, if there is no match, a security action is performed. This security action comprises loading a virus scanner on the local computer. It may also comprise alerting the network administrator. Furthermore it is also known to use this for differentiating accessibility to software from different workstations and as a way of checking whether software is licensed.

[0007] It is also known to use hashing in a method of identifying rogue software on a computer system or device. The method typically is applicable in a network environment. A hash value of a software application to be executed is calculated, this hash value is transferred to a server and compared with previously stored values. One of the essential features is that the method uses a database on a server, the server being a server with a large number of clients. The database on the server thereby is built up by adding information by different cli-

ents so that most software applications and their corresponding fingerprints are already stored in the database. The database is built up by checking software applications on authenticity with the owners of the application. If this is not possible, the system is also able to give a heuristic result, evaluating the occurrence of this application on local computers from other clients.

[0008] Methods for sending an electronic file by electronic mail, i.e. e-mail, including a file content and message content identifier are known. Depending on the message content identification, the message is delivered to a customer or not. The method may be used to organise e-mail delivery, but it has the disadvantage of being focussed on e-mail delivery and it does not allow to secure all files in a network.

[0009] Monitoring of electronic mail messages, to protect a computer system for protection against virus attacks and unsolicited commercial e-mail (UCE) is also known. Such a system is preferably installed on a mail server or an Internet Service Provider and checks specific parts of e-mails by calculating a digest and comparing it with stored digest values of e-mails previously received. In this way it is determined whether the e-mail has an approved digest or whether the e-mail is UCE or contains a e-mail worm. The system has the disadvantage that it is focussed on e-mail viruses and SPAM and that it does not allow to check all data files or executable files which are possibly infected, e.g. by files copied from external memory storage means like floppy disks or CD-ROMs or by e.g. Trojan horses.

[0010] Controlling the execution of software on different workstations according to certain policy rules by a network server is known, whereby an improved computer security system is obtained, by classifying software. It is suggested that this classification can be based on several forms of data one of which is e.g. the hash values of software data. This typically is performed by the calculation of hash values of a program if it is selected for loading and execution, and comparison of the hash value with a trusted value to determine the rule of execution. The classification also may be based on a hash of the content, a digital signature, the file system or network path or the URL zone.

[0011] The above mentioned methods and systems describe the use of hashing functions to check whether software applications are authentic or to regulate the execution of software applications. Nevertheless the problem of virus scanning all new files in a network using a conventional virus scanner whereby the necessity of updating the database of fingerprints of a conventional virus scanner on every local computer is limited is not discussed.

Summary of the Invention

[0012] It is an object of the present invention to provide a system and a method for identifying the content of files new on local computing devices in a network while limiting the necessity of updating the information needed by a content identifier on every local computing device.

[0013] It is an advantage of the present invention that, if the invention is used as a virus checker, the security level is further increased as the database of fingerprints of a conventional virus scanner does not have to be updated on every local computing device.

[0014] It is a specific advantage of the present invention that the content of a file new to a network is only identified once for the whole network.

[0015] It is furthermore a specific advantage of the present invention that the total processor (CPU) processing time in the network and the amount of network traffic is reduced.

[0016] The above described object and advantages are obtained with a method and system of content identification in a network according to the present invention.

[0017] The method for identifying the content of a data file in a network environment is used for a network having at least one local computing device linked to a remaining part of the network environment including a central infrastructure. The method and system comprises calculating a reference value for a new file on one of said at least one local computing devices using a one-way-function, transmitting said calculated reference value to said central infrastructure, comparing said calculated reference value with reference values previously stored within the remaining part of the network environment.

[0018] The method further comprises, after comparing, deciding that the content of the new file is already identified if a match between said calculated reference value and a previously stored reference value is found and retrieving the corresponding content attributes; or deciding that the content of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure identifying the content of said new file and determining content attributes corresponding with the content of the new file and storing a copy of said content attributes, after deciding, triggering an action on said local computing device in accordance with said content attributes.

[0019] In the method for identifying the content of a data file in a network environment, the reference value may be a hash value. The reference values previously stored may be stored within the central infrastructure. In the method and system for identifying the content of a data file in a network environment, identifying the content of the new file may comprise scanning the new file for viruses using an anti-virus checker means on a central infrastructure.

[0020] The method may furthermore comprise transferring the new file from the local computing device to the central infrastructure before said identifying the content of said new file is performed. Furthermore it may comprise storing a copy of the new file on the central

infrastructure after transferring the new file from the local computing device to the central infrastructure. An address of where the file is stored may be stored together with the hash value, as to be able to quickly track copies of the files stored on the central infrastructure.

[0021] In the method of the present invention, triggering an action on the local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with a copy of a previous version of said new file. Furthermore, triggering an action on the local computing device in accordance with said content attributes may also comprise replacement of the new file on the local computing device with another version of said new file restored from the remaining part of the network environment.

[0022] The method of the present invention furthermore may comprise sharing the new file on the local computing device to the central infrastructure before said identifying the content of said new file is performed and whereby said identifying the content of said new file is performed by remotely identifying the content over the network environment. The method may comprise checking the functioning of the local agent on the local computing device.

[0023] Furthermore, triggering an action on the local computing device may be performed after transmitting the content attributes corresponding to the new file to the local computing device.

[0024] In the method for identifying the content of a data file in a network environment, identifying the content of the new file may comprise one or more of the group of scanning for adult content, scanning for Self Promotional Advertising Messages or Unsolicited Commercial E-mail (UCE) and scanning for copyrighted information. The method may further relate to a method and system for providing a content firewall, whereby one local computing device is connected to the external network, which may e.g. be the internet, and the one local computing device is also connected to the network environment formed by the remaining local computing devices. The one local computing device thus links the network environment with an external network and is the only computing device that is directly connected to sources external from the network environment. The local computing device thus acts as a content firewall as to protect the network environment from attacks originating from places in the external network. The local computing device may act as a content firewall working in a promiscuous way, i.e. whereby the local computing device acts as a content firewall that sees all traffic passing by, executes the hashing and comparing functions and contacts the agents to enforce a policy.

[0025] The method may be specifically related to a method for checking the security status of a network and its components. In this embodiment, a method for determining the security status of a data file in a network environment is used in a network having at least one local computing device linked to a remaining part of the network environment including a central infrastructure. The method comprises calculating a reference value for a new file on one of the at least one local computing devices using a one-way-function, transmitting said calculated reference value to said central infrastructure, comparing said calculated reference value with reference values previously stored within the remaining part of the network environment and after comparing, deciding that the security status of the file has already been checked if a match between the calculated reference value and a previously stored reference value is found and retrieving the corresponding security status; or deciding that the security status of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure checking the security status of the new file and determining the security status corresponding with the new file and storing a copy of the security status, followed by

after deciding, triggering an action on said local computing device in accordance with the security status of the new file. This action may be e.g. making the file inaccessible for the user of the local computing device and for other users in the network or restoring the infected file.

[0026] The methods described above may be triggered by an action performed on the local agent. The triggering by an action performed on the local agent may be e.g. running an application or opening a file.

[0027] The invention also relates to a method for altering a system for identifying the content of a file in a network environment according to the systems described above, the network environment comprising means for calculating a one-way function, at least one local computing device linked to a remaining part of the network environment including a central infrastructure and means for identifying the content and, the method comprising altering said means for identifying the content or said means for calculating a one-way function, scanning the remaining part of the network environment for reference values calculated with a one-way function and for each of the reference values, requesting a file that corresponds with said reference value from said network environment, sending the file to means for identifying the content, identifying the content of said file and determining content attributes corresponding with the content of the file and storing a copy of said content attributes, sending the content attributes to every local computing device containing the file and after sending; triggering an action on said local computing device in accordance with said content attributes. Altering of a system for identifying the content of a file in a network environment may be triggered by the introduction of a new one-way function to calculate reference values or may be also triggered by the updating of the means for identifying the content of the files. The method may be used to scan only part of the hashing keys in the remain-

ing part of the network environment, e.g. hashing keys of files of which the content is identified after a certain date, as to minimise the actions to be performed. The date of the previous content identification may be retrieved from the content attributes.

[0028] The reference values may be hashing values. The means for identifying the content may be an anti-virus checker means, a means for scanning for adult content, a means for scanning for Self Promotional Advertising Messages or a means for scanning for copyrighted information. Triggering an action on the local computing device in accordance with said content attributes may comprise replacement of the file on the local computing device with another version of the file restored from the remaining part of the network environment or may comprise replacement of a file with a copy of a previous version of the file or may comprise putting the file in quarantine or removing the file.

[0029] The invention is also related to a computer program product for executing any of the above described methods, when executed on a network. The invention furthermore relates to a system for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part the network environment which includes a central infrastructure and, said remaining part including a stored database, whereby the system comprises means for calculating a reference value for a new file on said local computing device using a one-way-function, means for transmitting said calculated reference value to said central infrastructure and means for comparing said calculated reference value with previously stored reference values from the database. The system furthermore comprises means for deciding whether the content of the new file is already identified based on comparison of said calculated reference value and reference values previously stored within the remaining part, means located on the central infrastructure, for identifying the content of the new file and as to assign content attributes if the new file has not been identified yet and means for storing said content attributes within the remaining part, and means for triggering an action on said local computing device in accordance with content attributes for said new file.

[0030] In the system according to the present invention, the means for identifying the content of a file may comprise an anti-virus checker means on said central infrastructure. Furthermore, means for storing a copy of the new file within the remaining part. The means for identifying the content of a file may comprise one or more of the group of means for scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

[0031] The invention may also relate to a machine readable data storage device, storing the computer program product for executing any of the above described methods, when executed on a network. Furthermore, the invention may also relate to the transmission of the computer program product for executing any of the above described methods.

[0032] Although there has been constant improvement, change and evolution of methods of virus scanning and content identification of data files, the present concepts are believed to represent substantial new and novel improvements, including departures from prior practices, resulting in the provision of more efficient, stable and reliable methods of this nature.

[0033] These and other characteristics, features and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, which illustrate, by way of example, the principles of the invention. This description is given for the sake of example only, without limiting the scope of the invention. The reference figures quoted below refer to the attached drawings.

Brief description of the drawings

[0034]

Fig. 1 is a schematic representation of a computer network

Fig. 2 is a schematic representation of a central infrastructure and its basic software components

Fig. 3 is a schematic representation of a local agent-driven content identification process.

Fig. 4 is a schematic representation of a metabase-driven content identification process.

Fig. 5 is a schematic representation of a computer network to which the content firewall system and method can be applied.

[0035] In the different figures, the same reference figures refer to the same or analogous elements.

Description of illustrative embodiments

[0036] The present invention will be described with respect to particular embodiments and with reference to certain drawings but the invention is not limited thereto but only by the claims. The drawings described are only schematic and are non-limiting. In the drawings, the size of some of the elements may be exaggerated and not drawn on scale for illustrative purposes. Where the term "comprising" is used in the present description and claims, it does not exclude other elements or steps.

[0037] Furthermore, the terms first, second, third and the like in the description and in the claims, are used for distinguishing between similar elements and not necessarily for describing a sequential or chronological order. It is to be understood that the terms so used are interchangeable under appropriate circumstances and that the embodiments of the invention described herein are capable of operation in other sequences than described or illustrated herein.

[0038] In this description, the terms "file", "program",

"computer file", "computer program", "data file" and "data" are used interchangeably, and any one use may imply the other terms, according to the context used. The terms "hash" and "hashing" will be used as examples of the application of one-way-functions but the present invention is not limited to a particular form of one-way-function.

[0039] The term "computing device" should be interpreted widely to include any device capable of carrying out computations and/or executing algorithms. A computing device may be any of a laptop, workstation, personal computer, PDA, smart phone, router, network printer or any other device which has a processor and can be connected to a network such as e.g. faxing devices or copiers or any dedicated electronic device such as a so-called "hardware firewall" or a modem.

[0040] The method and system to secure and control a network by identifying the content of each new file in the network can be used on any type of network. This may be a private network which may be a virtual private network, a local area network (LAN) or a wide area network (WAN). This may also be within a part of a public wide area network such as the internet. If a part of a public wide area network is used, this may be performed by remotely providing the method and system for identifying the content of each file by a service provider using an ASP or XSP business model, wherein the central infrastructure is provided to a paying client operating a local computing device. An exemplary network 10 is shown in figure 1, showing several local computing devices 50a, 50b, ... , 50i and a central infrastructure 100, also called a server. The number of local computing devices 50 connected to the network 10 is not limiting for the method of securing and controlling a network 10 according to the current invention. In business environment this number of local computing devices 50 typically ranges from a few to a few thousands. The method and system for identifying the content of each new file present in the network 10 may be used with many different operating systems such as Microsoft DOS, Apple Macintosh OS, OS/2, Unix, DataCenter-Technologies' Operating Systems, ...

[0041] In order to provide a quick method of securing and determining content identification of files, the method and system according to the present invention will determine hash values of new files present on the local computing devices 50, compare them with previously stored hash values and file information on a central server and determine the content of files new to the network 10 using a content identifying engine on the central infrastructure 100. The content attributes describing the content of a new file are then send to the local computing device 50 where an appropriate action is performed. It is also possible that the content attributes are not sent to the local computing device 50 but that the appropriate action is triggered from the central infrastructure 100. New files typically are files wherein new content has been generated on a local computing device 50 or when an external file has been received. The wording "file" may refer to data as well as to software applications, also called software.

[0042] The central infrastructure 100 contains a database, also called metabase 110, which contains a record for every hash value that is calculated for a file that already exists on one of the local computing devices 50. Besides the hash value, this record also contains a number of other fields. In these fields, file source information is stored. The file source information corresponding with a specific hash value includes the file name, a list of local computing devices 50 where the files that correspond to this hash value are residing on, including the path to the file on the file system of the local computing devices 50 and the date of last modification. An example of file source information for a specific file is given in Table 1.

Table 1

Filename	Myexampleword.doc
Path	c:\data\
Assetname	PCmarketing001
ModDate	23/4/2002

[0043] In a further field, a list of content attributes that identifies the type of content that is enclosed by the file is stored. The content attributes can e.g. refer to a file that contains a virus, a file that is a copyrighted MP3 audio file, a file that is a copyrighted video file, a file that is a picture, a file that is a picture that might contain adult content, a file that is a Self Promotional Advertising Message (SPAM), a file that is a HOAX, a file containing explicit lyrics or a file containing pieces of executable code. This list is not limiting.

[0044] The central infrastructure 100 furthermore contains a content identification engine 120. This can be a software application 130 or a set of software applications 130a, 130b, 130c, 130d, ... that use the content of a file to determine which type of content the file contains. These software applications may be various :

- a virus scanner : this is a piece of software that scans the content of the presented file and compares it with a database of known fingerprints of viruses. This can be any conventional virus scanning software like e.g. Norton anti-virus by Symantec Corporation, McAfee by Network Associates Technologies Inc., PC-cillin by Trend Micro, Kapersky Anti-Virus by Kaspersky Lab, F-secure Anti-Virus by F-Secure Corporation, ...
- an adult content in pictures scanner : this is a piece of software that scans the content of the presented file for the presence of shading, colors, textures that might represent adult content. Scanning pictures for adult content is already known. Content adult can e.g. be determined by the amount of nude that is

shown. Skin tones have hue-saturation values that are in a specific range. Therefore, if an image is scanned, it is possible to determine the amount of pixels having a skin tone character and to compare it with the total number of pixels. The ratio of skin tone pixels to the total number of pixels allows to determine a ratio of possible adult content in an image. Thresholds often are introduced so that images can be classified according to their possible adult content. In similar way, video images can be categorised, whereby the video is split into its different frames and wherein the images are categorised according to the above method.

- A scanner for internet content ratings : A piece of software that scans objects for adult content based on the PICS, i.e. the Platform for Internet Content Selection, label system. On voluntary basis, internet content providers can provide internet objects with a PICS rating determining the adult content in the internet object. This PICS rating is stored in the meta data of the object. This data normally is not visible to the viewer of an internet object. The rating systems is well known and an example of a scanner for internet content ratings is provided in the Netscape web browser for scanning the content of web pages.
- A scanner for scanning an object for explicit lyrics which may indicate adult content. This is known for both text files and audio files. Audio files are first transferred to text files. Subsequently, the text files are scanned and compared with databases which contain explicit lyrics.
- a SPAM-engine: A piece of software that scans the content of e-mail messages for the presence of unalleged SPAM. Algorithms to recognize SPAM area already known. These are typically based on decomposing the text in an electronic mail message, associating statistics with the text using a statistical analyzer and coupling a neural network engine to the statistical analyzer to recognize unwanted messages based on statistical indicators.

Other examples of software applications that could be used in the content identification engine 120 are e.g. engines that scan for copyrighted content and that compares the content of the file to a database of copyrighted information, etc. In some adoptions, a human operator can pursue the role of content identification engine 120, where he manually tags a file with a content identification attribute. When the content identification engine 120 is activated, it will take a file from the local agent as input and produce a set of attributes that represent the detected content.

[0045] The content Identification engine 120 will thus be constructed as a piece of software aggregating the functionality of a set of third party engines.

[0046] In a further embodiment of the invention, a system and method in accordance with the above embodiment is described whereby the record corresponding with a specific hash value stored in the metabase 110 also comprises a field wherein the location of the file on the central infrastructure 100 corresponding with the hash value is stored. In this embodiment, a copy of all different files present on the local computing devices 50 in the network 10 is stored on the central infrastructure 100. So, the central infrastructure 100 of this embodiment also comprises a large amount of storing space. This preferably is a secured part of the central infrastructure 100, not directly connected to the network 10 so that these identical copies of the files present on the local computing devices 50 can be used in case the files on the local computing devices 50 are corrupted e.g. by a virus.

[0047] The hash value of the files are calculated using a hashing function. A hashing function typically is a one way function, i.e. given the digest, it is at least computationaly prohibitive to reconstruct the original data. Different types of hashing functions could be used : MD5, SHA-1 or ripemd all available from RSA Data Security Inc., haval which is designed at the University of Wollongong, snefru which is a Xerox secure hash function, etc.. The hashing functions most often used are MD5 and SHA-1. The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be 'compressed' in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem. The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. An alternative hashing function SHA-1, i.e. Secure Hashing Algorithm - 1, is a hashing algorithm generating a 160-bit hash. Newer versions of this algorithm also provide bit lengths of 256 and 512.

[0048] In the above mentioned embodiments describing a method and system to secure and/or control the network 10, a local agent is installed on the local computing device 50. The local agent is a piece of software that is running on a local computing device 50 and that performs certain algorithms and procedures. The local agent on the local computing device 50 is triggered typically in situations where new content is being generated on local computing devices 50. In order to avoid unnecessary hash value calculations and data transfer, a policy is setup to determine which actions will trigger the local agent and which actions do not trigger it. If e.g. a text document is being created, it is not necessary to check the file every time the document is saved. The policy for such a type of documents would preferably be that the document is checked e.g. if the file is both saved

and closed. Some examples of actions which could trigger the local agent and thus start the content identification process are opening or receiving e-mail messages, opening or receiving e-mail attachments, running executable files, running files with dll or piff extension ... Applying this policy thus allows to prevent from continuously checking and scanning of documents leading to a limitation of the number of unnecessary hash calculations and content identification operations and thus limiting the unnecessary use of CPU time and load on the network traffic. The method and system of content identification is not limited due to the type of application in which the file is made.

[0049] The content identification process can be either triggered by the local agent on the local computing device 50 or it can be triggered by the central infrastructure 100. The latter process typically occurs in situations wherein new algorithms or tools are being used for content identification. Such new algorithms or tools can either be optimised algorithms and tools or previously uninstalled tools. Some examples of these tools, without restricting to these functions, could be virus checking, checking whether a file is a copyrighted MP3 Audio File, checking whether a file is a copyrighted Video File, checking whether a file is a picture that might contain adult content, checking whether a file is tagged as being SPAM or HOAX, checking whether a file contains explicit lyrics or checking whether a file contains copyrighted pieces of executable code. Updating of these tools may influence the status of the files and thus may in principle influence the corresponding records in the metabase 110. Therefore, depending on the type of update of the content identification means 120, it may be interesting to update the corresponding records.

[0050] In a specific embodiment, the method relates to a virus checker for a network environment. The networks 10 on which this method can be applied are the same as those described for the previous embodiments. The local agent calculates the hash value of a new file on the local computing device 50. This new file may comprise new content generated on the local computing device 50 or an external file which is received on the local computing device 50. The hash value of the new file and the corresponding file information then is sent to a central infrastructure 100, also called server, where it is compared to previously stored hash values corresponding with files that are already present on the different local computing devices 50 of the network 10. This comparison allows to check whether the file is new or not in the entire network 10. Alternatively, the hash value may also be first compared with a local database of hash values and file information corresponding with the files present on that particular local computing device 50 and subsequently, if the file has been found not yet present on the local computing device 50, the hash value and the corresponding file information may be interchanged with the central infrastructure 100 so it can be checked whether the file is new or not in the entire network 10. Although transferring the file information and the hash value of every new file only corresponds with a very small fraction of the network traffic for a conventional central virus checker, this alternative could reduce the network traffic used for virus checking even further. If a hash value has been identified as new on the network 10, the metabase agent triggers the local agent to transfer the file corresponding with the new hash value from the local computing device 50 to the central infrastructure 100. The transferring of the file may be performed in a secured way, i.e. the file may be transferred such that it cannot be influenced by a virus present at a network connection or such that, it if contains a virus, this cannot be spread over the whole network 10. To obtain this, a known secure transmission route, a tunnel and/or known session ecryption/decryption techniques may be used. A conventional virus checker, installed and updated on the central infrastructure 100 then checks the file for viruses. This can be any conventional virus checker like e.g. Norton anti-virus by Symantec Corporation, McAfee by Network Associates Technologies Inc., PC-cillin by Trend Micro, Kapersky Anti-Virus by Kaspersky Lab, F-secure Anti-Virus by F-Secure Corporation, ...

[0051] A specific advantage of the above described embodiments in the current invention is that the virus scanning software does not need to be updated on every local agent but that this is restricted to updating of the virus scanning software of the central infrastructure 100. In this way the security level of the network 10 is increased significantly as the security does not depend on the punctuality of the different users of the network 10 to update their virus scanning software. If the scanned file has no virus it will be marked in the metabase 110 as being a virus free file. If there is a virus found in a file the file will be marked as dangerous. A query will happen to the metabase 110 to find all files over the network 10 having the same corrupted hashing key. The result is a list of files with path, and assetname where the file is located. This information can be used to do actions to eleminate the danger of found viruses on all local computing devices 50, i.e. all workstations, from the complete network 10. In this way proactive virus scanning can be performed on other local computing devices 50, based on a virus detection on a first local computing device 50. Depending the policy defined for virus checking, the virus engine will inform an agent installed on the affected system to remove the file and if possible replace with either a recovered version delivered by the virus engine located on the central infrastructure 100, or a previous version of the file which didn't have the virus yet. The latter can be done easily by searching the metabase for a previous version of that file, or it can be performed by searching an uninfected version on another local computing device 50. If an uninfected version cannot be retrieved from either another local computing device 50 or the metabase residing on 100, the virus scanner should have a feature which allows it to save a

new disinfected copy of the file on the central infrastructure 100. These advantages are also present for other content identifying packages.

[0052] In an alternative embodiment, if a file having a new hash value has been identified in the network 10, instead of transferring the file to the central infrastructure, the file may be automatically shared locally and a remote checker then may transfer a file-system which allows to check the file across the network 10 using the file sharing. The content tagging still is performed by the server. In order to improve security, the accessibility to the shared file is restricted to the server. Furthermore, a java applet could be transferred to the local agent to allow checking other files.

[0053] The previous embodiments are an improvement over a central virus checker which scans local computing devices 50 through the network 10. This is only possible if the local drives, e.g. C:\, D:\, ..., are shared. Besides the dangers of sharing with respect to security, the local user also easily can change the local sharing properties thereby preventing the remote checker from checking the files. This is at least partly avoided with the current invention as changing the network 10 sharing properties does not influence the operation of calculating the hash value of new files and sending it to the central infrastructure 100.

[0054] Another advantage is that it saves CPU time on the local computing device 50 as the CPU does not have to keep doing virus checking, it only has to calculate a one way function. It also saves network time : the administrating server does not have to update the virus checkers on the local computing devices 50 with virus updates, as a single central virus checker only is used and updated.

[0055] Fig. 3 shows a method 200 of the content identification process triggered by the local agent on the local computing device 50 according to the above mentioned embodiments. The different steps that occur during the process, both on a local computing device 50 and on the central infrastructure 100 are discussed.

[0056] The content identification process is based on continuously scanning for new data or applications on the local computing device 50 by the local agent. This scanning for data and applications is limited by the policy rule for determining when the local agent should be triggered, as described above. If a "new" file has been detected the method for securing and controlling the network 10 by content identification of new files is initiated. This is step 210. Method 200 then proceeds to step 212.

[0057] In step 212, a hash value of the "new" file is calculated using a hashing function like MD5 or SHA-1. This calculation is performed by using some CPU time of the local computing device. Nevertheless, the amount of CPU time used is drastically smaller than the CPU time that would be necessary if e.g. a conventional virus checker was used to check the file on the local computing device 50. Method 200 then proceeds to step 214.

[0058] In step 214, the hash value and the file source information is transferred from the local agent to the central infrastructure 100 of the network 10. If necessary, this transfer can be a secured transfer, whereby it is avoided that a virus which is positioned on a network connection changes both the file source information or the hashing key during transport of this data. Such a secured transmission can be made over a known secure transmission route, via a tunnel, or using known session ecryption/decryption techniques.

[0059] In step 216, the hash value is compared with the data already present in the metabase 110. As in the metabase 110, the hash values and file source information of all old files - i.e. every file that has been present on the network 10 and that is not "new" as described above - present in the network 10 are stored, it is possible to check whether the file already is present in the network 10. Therefore, if the hash value has been identified as new, this implies that the file is "new" for the whole network 10. If the file is new, method 200 proceeds to step 218. If the hash value is not new, this means that somewhere on a local computing device 50 in the network 10, the file does already exist. In this case, there already exists content attributes describing the content of the file. Method 200 then proceeds to step 224.

[0060] In step 218 the metabase agent triggers the local agent to transfer the file corresponding with the new hash value from the local computing device 50 to the central infrastructure 100. The transferring of the file may be performed in a secured way, i.e. the file may be transferred such that it cannot be influenced by a virus present at a network connection or such that, it if contains a virus, this cannot be spread over the whole network 10. To obtain this a known secure transmission route, a tunnel and/or known session ecryption/decryption techniques may be used. Method 200 further proceeds to step 220.

[0061] In step 220 the file is loaded in the content identification engine 120 and the file is processed. For this processing CPU time of the central infrastructure 100 is used. The content identification engine 120 can comprise, as described above, a conventional virus checker, a means for checking picture information, a means for checking SPAM, etc. This can be a repetitive action where multiple content identification engines are called in turn. Method 200 then proceeds to step 222.

[0062] In step 222 content attributes, which identify the content of the file, are determined for the file. These content attributes are then stored in the metabase 110, thus allowing to identify the status of the file if, in future operations, the file is found 'new' on another local computing device 50. Method 200 then proceeds to step 224. Depending on the embodiment used, a following step may include the storing of the file on the central infrastructure 100 and adding the path to this file to the metabase 110. This step is not shown in figure 3.

[0063] In step 224 the content attributes are sent to

the local agent. Based on this content attributes, the local agent performs an appropriate action in agreement with the policy rule set for these content attributes. This is performed in step 226. This can be e.g. deleting the file if it was infected, replacing the file with a previous version which was not infected, ... In a specific embodiment, the execution of appropriate actions based on the policy rules are triggered by the agent of the metabase 110, so that step 224 can be avoided.

[0064] The content policy is a policy that determines what should be done with a file depending on the content attributes determined by the content identification engine 120. The content policy can comprise actions such as deleting the file, deleting the file and replacing it with a previous version, copying the file onto another computing device while leaving a copy on the originating computing device, moving the file onto another computing device while deleting the original file on the originating computing device, logging the presence of the file, changing the attributes of the file like hiding it or making it read-only, making the file unreadable, making the file un-executable, etc. The content policy will be executed by the local agent, e.g. when the content attributes are received from the central infrastructure 100. The content policy for that agent will be downloaded to the local computing device 50 by the agent from a central policy infrastructure.

[0065] Fig. 4 shows a method 300 of the content identification process triggered by the content identification engine 120 according to the above mentioned embodiments . The different steps that occur during the process, both on a local computing device 50 and on the central infrastructure 100 are discussed.

[0066] This process typically is used in situations where new algorithms or tools are being used for content identification. Such new algorithms or tools can either be optimised algorithms and tools or previously uninstalled tools. As mentioned earlier this may be regulated by a policy : the triggering of the content identification process may be determined by the type of new algorithms and tools that are being used for content identification.

[0067] Method 300 is initiated by change of the content identification engine 120, e.g. by providing new algorithms or tools for the content identification engine 120.

[0068] In the first step 302 of method 300 upon triggering, the metabase 110 is scanned for hash values corresponding with hashing keys. Method 300 then proceeds to step 304.

[0069] In step 304, a file that corresponds with the hashing key is requested. This file can be either requested from the central storage on the central infrastructure 100 or it can be requested from a local computing device 50. The local computing device 50 then gives permission to the central infrastructure 100 to upload the corresponding file. The path to the file corresponding with the hash value is available from the record corresponding with each hash value. If the record stores different paths all corresponding with a copy of the corresponding file, the agent on the central infrastructure 100 retrieves one copy of the file, e.g. by scanning the paths listed in the record until a local computing device 50 has been found that is at that time connected to the network 10 and that allows uploading of the file. Method 300 then proceeds to step 306.

[0070] Once the file has been retrieved, the file is sent to the content identification engine 120. This is performed in step 306. The upgraded content identification engine 120 then scans the content of the file and produces content attributes corresponding with the file. Method 300 further proceeds to step 308.

[0071] In step 308, the content attributes are stored in the metabase 110, to allow in future security steps to immediately identify the content of the files. Method 300 further proceeds to step 310.

[0072] In step 310, the content attributes are sent to every local agent that resides on a local computing device 50 whereon the corresponding file is stored. The paths can be found in the record of the corresponding hashing key stored in the metabase 110. In this step, content attributes are send to every file for which a path is mentioned in the record of the corresponding hashing key. Method 300 proceeds to step 312.

[0073] In step 312, the local agent on the corresponding local computing devices 50 executes the policy according to the content attributes and the according to the local computing device 50.

[0074] One of the major advantages of the embodiments of the invention is that a file new to the entire network 10 only needs to be scanned once. If on another local computing device 50, an identical copy of this file is used, installed, opened or saved and closed, the file will be recognised by the central infrastructure 100 as being known to the network 10, in this way avoiding the need to re-check the content of the file. This especially is advantageous if the invention is used for networks 10 having a large number of local computing devices 50.

[0075] When operating, the method and system for identifying the content of new files optionally can comprise checking 'the heartbeat' of the local agent at regular times, i.e. it can be checked whether the local agent is still running on the local computing device 50. This can avoid that a user locally shuts down the agent, thus making the local computing device 50 vulnerable. If the local agent has been shut down, the network administrator can be warned. Furthermore a warning message could be send to the local computing device 50 thereby warning the user of the local computing device 50. The network administrator could also put the local computing device 50 in quarantine so that it can not damage other local computing devices 50 in the network 10. Furthermore, the central agent can also try to rerun the local agent.

[0076] In a similar way, the method and system for identifying the content of new files optionally can check

at regular times whether the local computing device 50 is still connected to the network 10. If the local computing device 50 is not connected to the network 10 anymore, the local agent may further operate, storing hashing keys of new files in a waiting list to be checked once the network connection is restored. In the mean time, the corresponding files may be put in quarantine or depending on the type of file e.g. may be prevented from being executed.

[0077] The above described embodiments may be used as a content firewall for the different computing devices connected to the external network. For every incoming/outgoing file, incoming/outgoing message or incoming/outgoing data frame, the content firewall calculates the hash, checks whether this is new, checks whether it is tagged for specific content and enforces the policy associated with the specific content.

[0078] In a further embodiment, another configuration for using the present invention as a content firewall is described. A schematic overview of a computer network wherein this method and system may be used, is shown in Fig. 5. Only one reconfigurable firewall electronic device 50, such as a local computing device which may be in the form of a dedicated reconfigurable firewall electronic device, is directly connected to an external network 400 such as e.g. the internet, and the remaining local computing devices 410, are not directly connected to the external network 400, but grouped in a network environment and only connected to the external network 400 by their connection to the electronic reconfigurable firewall device.. The external network may be any possible network available. It is a goal of the content firewall as represented by the reconfigurable electronic firewall device 50 to protect the network environment comprising the remaining local computing devices 410 from attacks originating from places and/or devices in the external network. The reconfigurable electronic firewall device 50 50 either contains a local copy of the metabase or it can use a high speed secured network to a central infrastructure 100 which is part of the internal network. This allows for fast queries through the metabase. During operation, the reconfigurable electronic firewall device 50 functioning as a content firewall performs the following actions : the hash value of incoming files or incoming messages or incoming data frames are calculated. Subsequently, the calculated hash values are compared with the metabase, which is either stored locally or by using a high speed secured network, and it is determined whether the incoming file, incoming message or incoming data frame is new. Furthermore, it is checked whether this file, this message or this data frame is tagged for specific content. Depending on the specific content, a policy is enforced which is associated with the specific content. This policy may be to let it pass through to its final destination, to drop it, to log it, to put it in quarantine, etc. This system requires sufficient CPU power, in order not to slow down the network speed noticeably.

[0079] In the case where none of the local computing devices connected to the network is equipped with a removable device, i.e. allowing for non-scanned content to be opened or executed on that device, this is a very secure and managable setup.

[0080] In another embodiment of the invention, a similar configuration for use of the present invention as a content firewall in promiscuous mode is provided. The content firewall thereby sees all traffic passing by, executes the hashing and comparing functions and contacts the agents to enforce a policy. The advantage of this approach is that there is no single point of failure and no bottleneck anymore and furthermore that still no resources are used on the local computing devices for calculating hashes. Furthermore, no bandwidth is used for contacting the central metabase. The disadvantage is that local agents need to be installed on all computing devices of the internal network.

In accordance with the above described embodiments, the present invention includes a computer program product which provides the functionality of any of the methods according to the present invention when executed on a computing device. Further, the present invention includes a data carrier such as a CD-ROM or a diskette which stores the computer product in a machine readable form and which executes at least one of the methods of the invention when executed on a computing device. Nowadays, such software is often offered on the Internet, hence the present invention includes transmitting the printing computer product according to the present invention over a local or wide area network.

Claims

1. A method for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part of the network environment including a central infrastructure and, the method comprising

   - calculating a reference value for a new file on one of said at least one local computing devices using a one-way-function,
   - transmitting said calculated reference value to said central infrastructure,
   - comparing said calculated reference value with reference values previously stored within the remaining part of the network environment,
   - after comparing,

     - deciding that the content of the new file is already identified if a match between said calculated reference value and a previously stored reference value is found and retrieving the corresponding content attributes; or

- deciding that the content of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure identifying the content of said new file and determining content attributes corresponding with the content of the new file and storing a copy of said content attributes,
- after deciding, triggering an action on said local computing device in accordance with said content attributes.

2. A method according to claim 1, wherein the reference value is a hash value.

3. A method according to claim 1 or 2, wherein reference values previously stored are stored within the central infrastructure.

4. A method according to the previous claim wherein said identifying the content of said new file comprises scanning the new file for viruses using a antivirus checker means on said central infrastructure.

5. A method according to any of the previous claims furthermore comprising transferring the new file from the local computing device to the central infrastructure before said identifying the content of said new file is performed.

6. A method according to claim 5, furthermore comprising storing a copy of the new file on the central infrastructure after said transferring the new file from the local computing device to the central infrastructure

7. A method according to any previous claim wherein said triggering an action on said local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with a copy of a previous version of said new file.

8. A method according to any previous claim, wherein said triggering an action on said local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with another version of said new file restored from the remaining part of the network environment.

9. A method according to claim 1, 2, 3 or 4 furthermore comprising sharing the new file on the local computing device to the central infrastructure before said identifying the content of said new file is performed and whereby said identifying the content of said new file is performed by remotely identifying the content over the network environment.

10. A method according to any of the previous claims, wherein said triggering an action on said local computing device in accordance with said content attributes is performed after transmitting the content attributes corresponding to the new file to the local computing device.

11. A method according to the previous claims wherein said identifying the content of said new file comprises one or more of the group of scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

12. A method according to any of claims 1 to 3 or 5, whereby one local computing device links said network environment with an external network, said one local computing device acting as a content firewall as to protect the network environment.

13. A computer program product for executing any of the methods as claimed in claims 1 to 12 when executed on a network.

14. A system for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part the network environment which includes a central infrastructure and, said remaining part including a stored database, whereby the system comprises :

- means for calculating a reference value for a new file on said local computing device using a one-way-function,
- means for transmitting said calculated reference value to said central infrastructure,
- means for comparing said calculated reference value with previously stored reference values from the database,

whereby the system further comprises:

- means for deciding whether the content of the new file is already identified based on comparison of said calculated reference value and reference values previously stored within the remaining part,
- means located on the central infrastructure, for identifying the content of the new file and as to assign content attributes if the new file has not been identified yet and means for storing said content attributes within the remaining part, and
- means for triggering an action on said local

computing device in accordance with content attributes for said new file.

15. A system according to claim 14 wherein said means for identifying the content of a file comprise an anti-virus checker means on said central infrastructure.

16. A system according to claim 14 or 15 furthermore comprising means for storing a copy of the new file within the remaining part.

17. A system according to any of claims 14 to 16 wherein said means for identifying the content of a file comprises one or more of the group of means for scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

18. A method for altering a system for identifying the content of a file in a network environment according to claims 14 to 17, said network environment comprising means for calculating a one-way function, at least one local computing device linked to a remaining part of the network environment including a central infrastructure and means for identifying the content and, the method comprising

- altering said means for identifying the content or said means for calculating a one-way function
- scanning the remaining part of the network environment for reference values calculated with a one-way function
- for each of said reference values,
    - requesting a file that corresponds with said reference value from said network environment
    - sending the file to means for identifying the content
    - identifying the content of said file and determining content attributes corresponding with the content of the file and storing a copy of said content attributes
    - sending the content attributes to every local computing device containing the file
    - after sending; triggering an action on said local computing device in accordance with said content attributes.

19. A computer program product for executing the method as claimed in claims 18 when executed on a network.

20. A machine readable data storage device storing the computer program product of any of claims 13 or 19.

21. Transmission of the computer program product of any of claims 13 or 19 over a local or wide area telecommunications network.

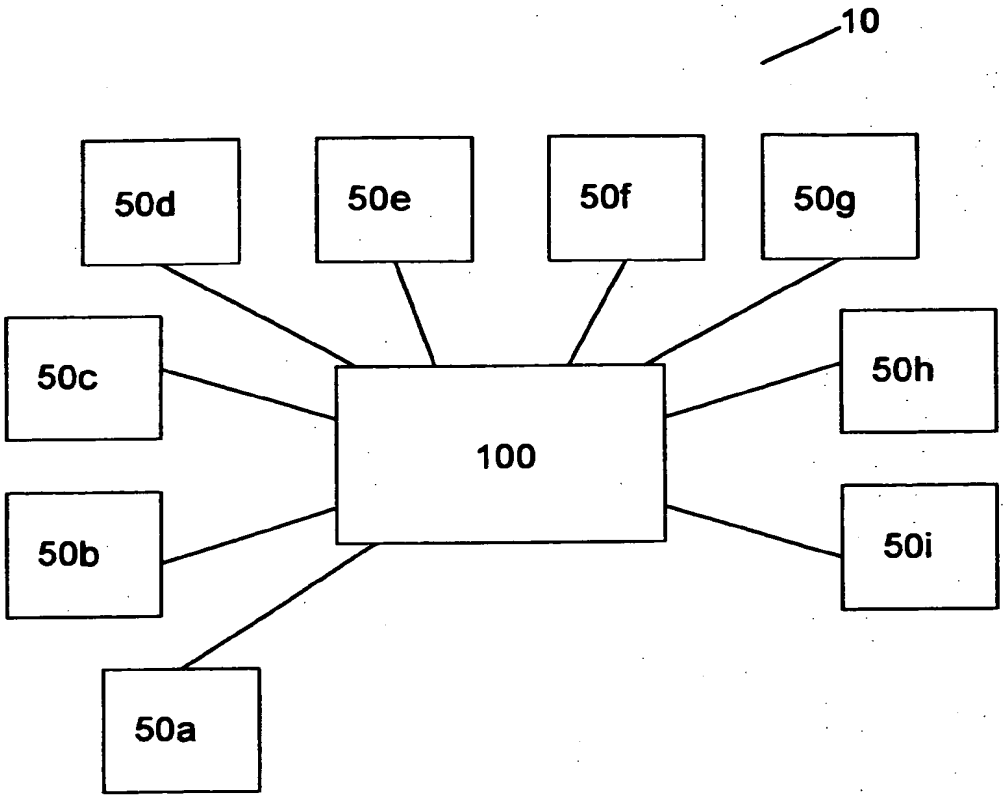

10
50d
50e
50f
50g
50c
50h
100
50b
50i
50a

Fig. 1

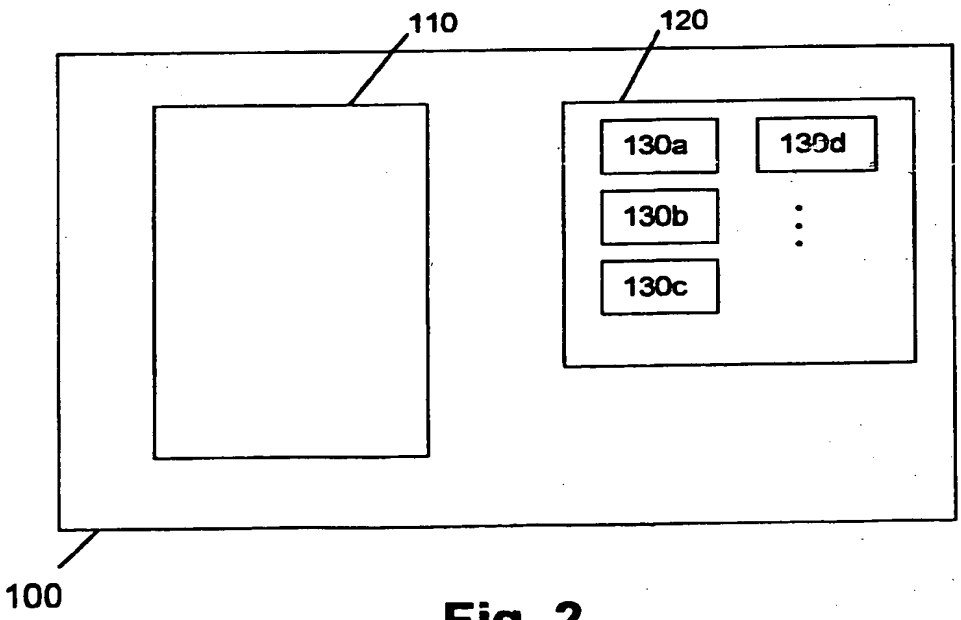

110
120
130a
130d
130b
130c
100

Fig. 2

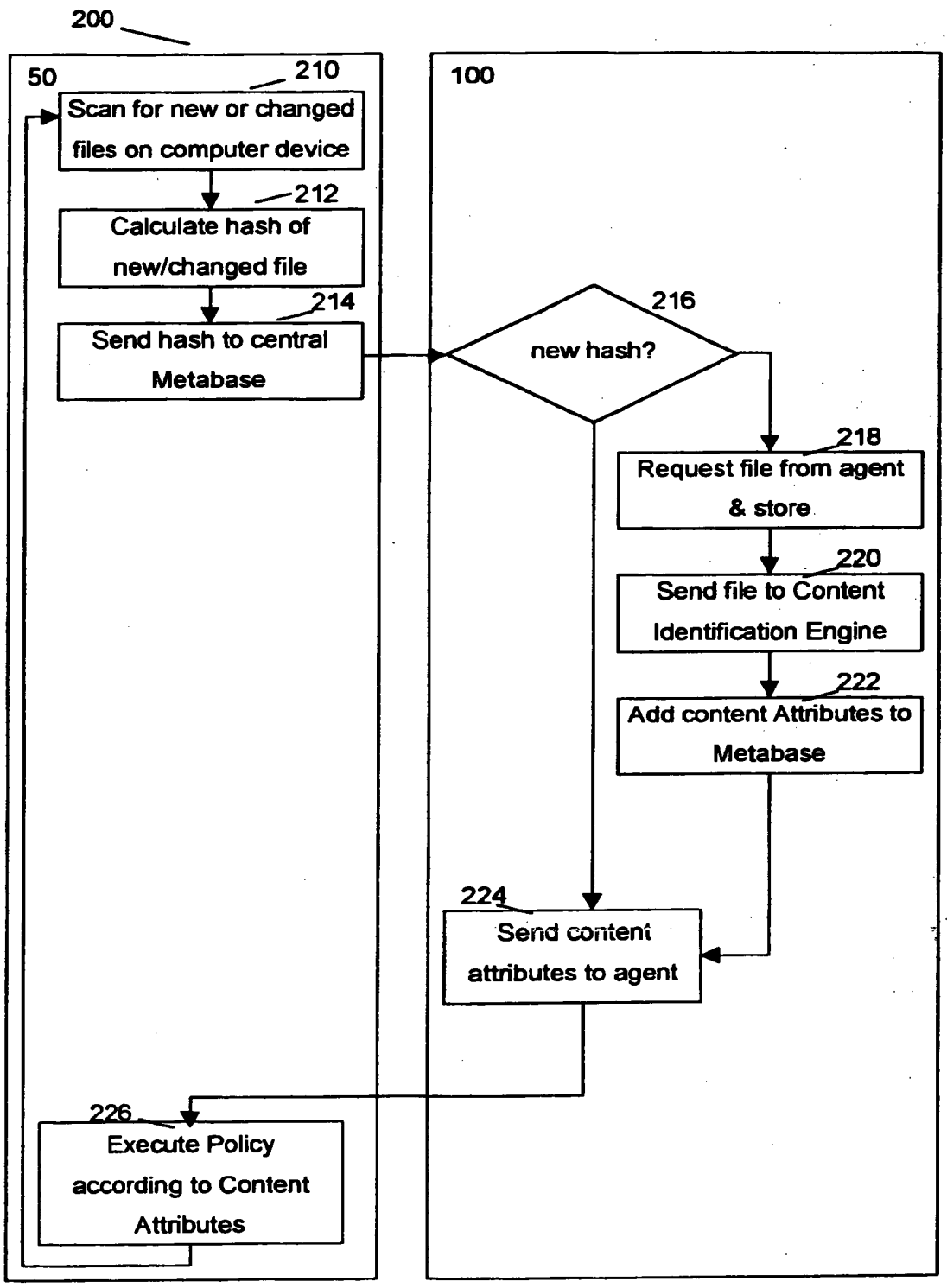

200
50
210
Scan for new or changed files on computer device
212
Calculate hash of new/changed file
214
Send hash to central Metabase
100
216
new hash?
218
Request file from agent & store
220
Send file to Content Identification Engine
222
Add content Attributes to Metabase
224
Send content attributes to agent
226
Execute Policy according to Content Attributes

Fig. 3

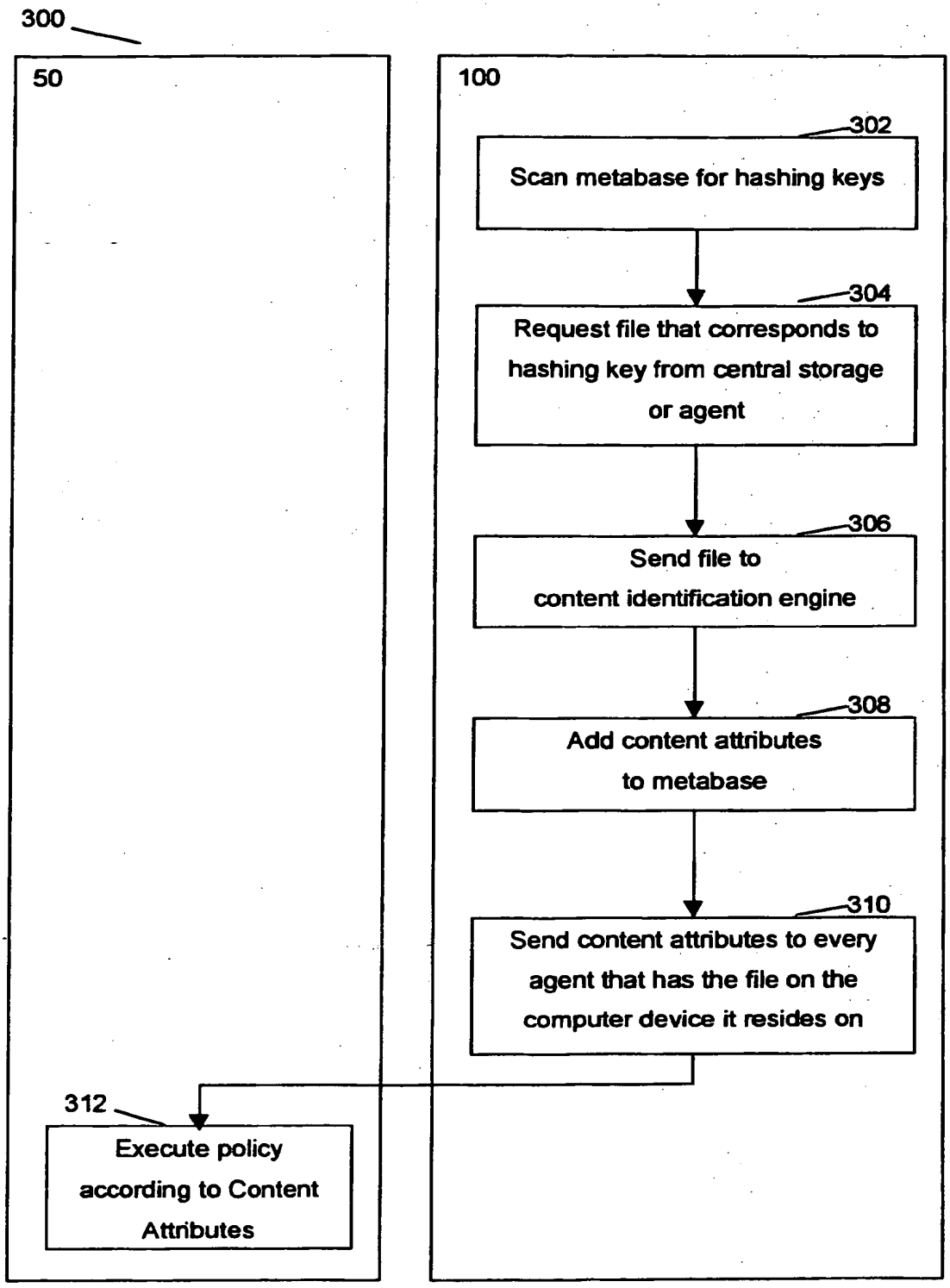

300
50
100
302
Scan metabase for hashing keys
304
Request file that corresponds to hashing key from central storage or agent
306
Send file to content identification engine
308
Add content attributes to metabase
310
Send content attributes to every agent that has the file on the computer device it resides on
312
Execute policy according to Content Attributes

Fig. 4

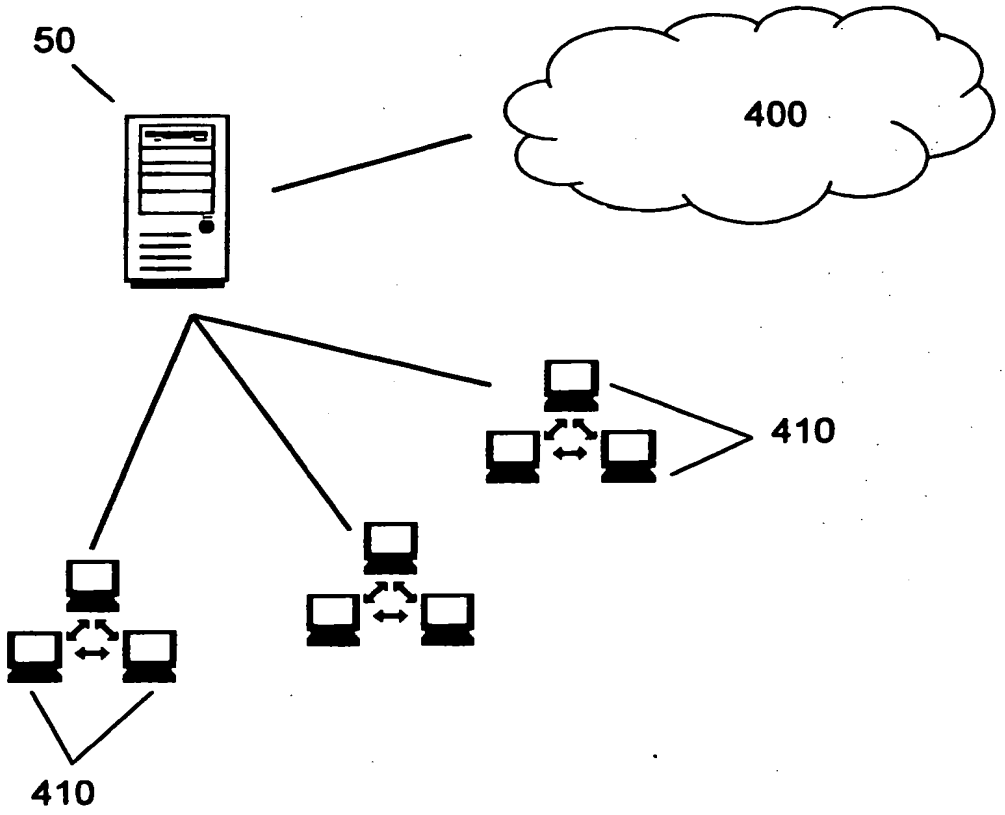


Fig. 5

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 03 44 7310

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	EP 1 280 040 A (NETWORKS ASSOCIATES TECHNOLOGY) 29 January 2003 (2003-01-29) * the whole document *	1-9, 11-17, 20,21	H04L29/06 G06F1/00
A		10,18,19	

A	US 6 195 587 B1 (HRUSKA JAN ET AL) 27 February 2001 (2001-02-27) * column 1, line 4-9 * * column 1, line 50 - line 65 * * column 3, line 55 - column 5, line 67 * * column 1, line 50 - column 3, line 3 * * column 3, line 55 - column 5, line 40 *	1-6, 9-11, 13-20	

A	US 2003/120953 A1 (RADATTI PETER V) 26 June 2003 (2003-06-26) * paragraphs [0013]-[0015] * * paragraph [0045] * * paragraph [0047] * * paragraph [0051] *	1-4,7,8, 11, 13-15, 17,20	

A	EP 1 111 507 A (SECURE OYJ F) 27 June 2001 (2001-06-27) * paragraphs [0002]-[0004] * * paragraphs [0010]-[0012] * * paragraph [0015] * * paragraphs [0025],[0026] * * paragraph [0031] *	1,4-8, 12-16,20	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)
H04L
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	7 June 2004	Ruiz Sanchez, J

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 03 44 7310

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

07-06-2004

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1280040 A	29-01-2003	US 2003023866 A1	30-01-2003
		EP 1280040 A2	29-01-2003
US 6195587 B1	27-02-2001	GB 2283341 A	03-05-1995
		AU 8000394 A	22-05-1995
		DE 69407812 D1	12-02-1998
		DE 69407812 T2	16-07-1998
		EP 0725951 A1	14-08-1996
		WO 9512162 A1	04-05-1995
		JP 9504395 T	28-04-1997
US 2003120953 A1	26-06-2003	NONE	
EP 1111507 A	27-06-2001	GB 2353372 A	21-02-2001
		EP 1111507 A2	27-06-2001
		US 2001005889 A1	28-06-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 November 2004 (11.11.2004)

PCT

(10) International Publication Number
WO 2004/097602 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number:
PCT/GB2004/001333

(22) International Filing Date: 29 March 2004 (29.03.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0309463.8 25 April 2003 (25.04.2003) GB

(71) Applicant *(for all designated States except US)*: MESSAGELABS LIMITED [GB/GB]; 1270 Lansdowne Court, Gloucester Business Park, Gloucester LG3 4AB (GB).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: SHIPP, Alexander [GB/GB]; Star Internet, Brighouse Court, Barmwood, Gloucester GL4 3RT (GB).

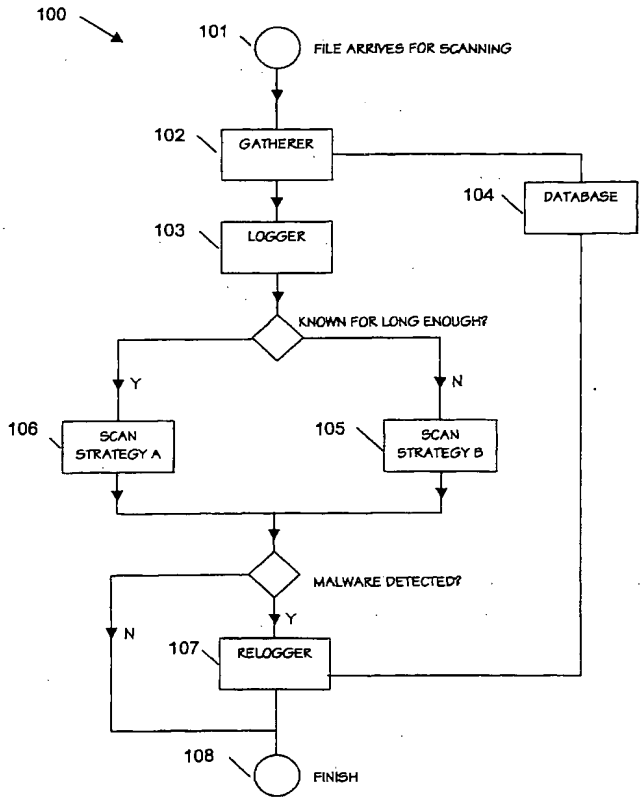
(74) Agents: AYERS, Martyn et al.; J.A. Kemp & Co., 14 South Square, Gray's Inn, London WC1R 5JJ (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),

[Continued on next page]

(54) Title: A METHOD OF, AND SYSTEM FOR, HEURISTICALLY DETERMINING THAT AN UNKNOWN FILE IS HARMLESS BY USING TRAFFIC HEURISTICS



(57) Abstract: A system for processing a computer file to determine whether it contains a virus or other maiware maintains a database of known files which it references to determine whether the file is an instance of a known file, and if so, whether it has been known about long enough that it can be regarded as safe. If it can be regarded as safe, the file is subject to less thorough processing for detecting malware, or no such processing at all.

BEST AVAILABLE COPY

WO 2004/097602 A2

Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A METHOD OF, AND SYSTEM FOR, HEURISTICALLY DETERMINING THAT AN UNKNOWN FILE IS HARMLESS BY USING TRAFFIC HEURISTICS

The present invention relates to a method of, and system for, heuristically determining that an unknown file is harmless by using traffic heuristics. This technique is especially applicable to situations where files enter a system, are checked, then leave, such as email gateways or web proxies. However, it is not intended to be limited to those situations.

Increasing use of the Internet, personal computers and local- and wide-area networks has made the problem of viruses and other malware (=malicious software) ever more acute.

There are numerous anti-virus packages available. These tend to be produced by specialist companies and are used by businesses and other organisations, home users, and by some internet service providers (ISPs) who scan e-mail and other network traffic on behalf of their customers as a value-added service. As new viruses and other malware arise, the package creators devise ways of detecting them and dealing with them and issue updates to their packages which customers can utilise. A common practice is to make the updates available for download over the internet, from the creator's website or ftp site.

Most anti-virus packages include a file-scanning engine and a database of characteristics of known viruses which are used by the scanning engine to determine whether a file being scanned is, or contains, a virus or other malware, or is likely to do so. The sort of update mentioned above typically includes an update to this database.

The scanning engine may implement a variety of heuristics to be applied, possibly selectively, to a file being scanned. Probably the most familiar kind of heuristic is signature detection, in which the file is examined for the occurrence of sequences or bytes, or patterns of such sequences, which are known to be characteristic of viruses in the package's virus database, though many other heuristics also exist, which can be used as well as or instead of signature detection.

The amount of malware in existence increases all the time, which makes the computational and storage resources necessary to detect it increasingly burdensome, particularly where the throughput of files is high, as is the case with ISPs.

According to the present invention, there is provided a system for processing a computer file to determine whether it contains a virus or other malware comprising:

a) means for generating data with regard to the file to characterise its identity and for thereby referencing a computer database to determine whether it is an instance of a known file;

b) means for selectively subjecting the file to a number of heuristic procedures to determine whether or not it contains, or is likely to contain, malware; and

c) means for determining, in dependence upon the record, if any, of the file in the database, whether the file can be regarded as safe and for controlling the means b) such that the file, if the file is to be regarded as safe, is either subject to less thorough processing than if it were not so regarded or not subject to processing by the means b) at all.

The invention also provides a method of processing a computer file to determine whether it contains a virus or other malware comprising:

a) generating data with regard to the file to characterize its identity and for thereby referencing a computer database to determine whether it is an instance of a known file;

b) selectively subjecting the file to a number of heuristic procedures to determine whether or not it contains, or is likely to contain, malware; and

c) determining, in dependence upon the record, if any, of the file in the database, whether the file can be regarded as safe and conducting the step b) such that the file, if the file is to be regarded as safe, is either subject to less thorough processing than if it were not so regarded or not subject to processing by the step b) at all.

The invention will be further described by way of non-limitative example with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of a system embodying the present invention.

Figure 1 illustrates one form of a system 100 according to the present invention, which might be used, for example by an ISP as part of a larger anti-virus scanning system which employs additional scanning methods on files which are not filtered out as "safe" by the system of Figure 1. Files considered safe can if desired be subject to further processing to check for malware, but less intensively so than files not considered safe.

The rationale of the system 100 is that if a particular file has been scanned by a virus scanner, and found to be harmless the two possibilities exist: The file could really be harmless, or the file could contain something nasty which the virus scanner is as yet unable to detect.

As time goes by, the file (or another instance of it) may be scanned again, and still found to be harmless.

This time the file is more likely to really be harmless, rather than to be malware which the virus scanner is as yet unable to detect. This is because virus scanners are continually updated to detect new malware as the new malware is discovered. The longer the time that passes, the more likely it is that a suspicious person will submit a file containing malware to the developers of the scanner, who will analyse the file, and update their scanner to detect it.

As more and more instances of the file are scanned coming from different sources, then if these are all flagged as harmless, it becomes less and less likely the file is malware. This is because the more copies of a piece of malware exist, the more likely it is that somebody will become suspicious and submit a copy to scanner developers.

It is therefore possible to create a feedback engine which logs copies of files scanned, together with the source they originated from. The log is updated and examined as each file is scanned, and if files are found which have come from a sufficient number of sources, in sufficient quantities, and over a long enough period of time, then that file can be flagged as 'known about long enough'. This might mean that future copies are then not scanned further, or are scanned using less rigorous scans with fewer heuristics enabled, or are only scanned if the scanner has been updated since the last scan.

The system 100 operates according to the following algorithm:

1) A file arrives at an input 101 for scanning, perhaps as an email attachment, or a web download.

2) A 'gatherer' module 102 gathers information about the file, such as a checksum of the file contents and the source of the file (eg the IP address). The source may be passed through a one way trapdoor function, generating a hash, in order to preserve confidentiality. The information gathered is for comparison with information stored in a database 104 about known files so that it can be determined whether the file under consideration is an instance of a file recorded in database 104.

3) Based on the checksum derived by gatherer 102, a 'logger' module 103 updates the database 104 to indicate that one more instance of the file has been detected.

The logger 103 saves the current 'last seen' date as the 'previously scanned date', and then updates the 'last seen' date of the file's entry in the database 104. If this is the first instance of the file, the logger 103 also updates a 'first seen' date. If this is a new source, the logger 103 adds the source to a list, stored in database 104, of sources the file has originated from.

4) From the information stored (number of copies of the file seen, length of time file has been known about, number of sources) the logger 103 calculates whether the file has been 'known about long enough'. For this purpose, the logger 103 may assign a weighted score to each of these factors individually and then calculate an overall score by combining the weighted scores, e.g. by adding them up.

5) If the file has not been known about long enough, scan strategy B is undertaken at 105. This will be the most complete scan available.

6) If the file has been known about long enough, scan strategy A is undertaken at 106. This will be a less thorough scan than strategy B. This will be site-dependent as to how less thorough a scan is desired. At the extreme it might involve no scanning at all. It might involve scanning with fewer scanners; with heuristics not fully enabled or turned off; or (assuming the file has been seen at least once before) only with scanners that have been updated since the 'previously scanned date'

The scanning techniques available to the scanning strategies A and B may include any suitable heuristics, such as signature-based scanning, generating checksums from the file or selected regions if it, etc.

7) Following the scan strategy A or B, then if no malware was detected, processing stops at 108.

8) If malware was detected, then a 'relogger' module 107 is invoked. This clears out all database entries in database 104 which are associated with the file so that it cannot become 'known about long enough' in the future.

9) Processing of the current file finishes at 108, whereupon the system can retrieve the next file from a queue of files waiting to be processed.

CLAIMS

1. A system for processing a computer file to determine whether it contains a virus or other malware comprising:

a) means for generating data with regard to the file to characterise its identity and for thereby referencing a computer database to determine whether it is an instance of a known file;

b) means for selectively subjecting the file to a number of heuristic procedures to determine whether or not it contains, or is likely to contain, malware; and

c) means for determining, in dependence upon the record, if any, of the file in the database, whether the file can be regarded as safe and for controlling the means b) such that the file, if the file is to be regarded as safe, is either subject to less thorough processing than if it were not so regarded or not subject to processing by the means b) at all.

2. A system according to claim 1 wherein the controlling means c) controls the means b) in dependence on factors including the length of time for which the database indicates that the file has been known without malware-containing instances of it being detected.

3. A system according to claim 1 or 2 wherein the controlling means c) controls the means b) in dependence on factors including sources, recorded in the database, from which instances of the file have originated.

4. A system according to claim 1, 2 or 3 wherein the controlling means c) controls the means b) in dependence on factors including the number of times, recorded in the database, of instances of the file have been processed.

5. A system according to any one of the preceding claims, and including means for updating the database in dependence upon the result of the processing of the file by the means b).

6. A system according to claim 5 wherein the updating of the database, in the event of the means b) determining that the file contains, or is likely to contain, malware is such that the record thereof in the database is deleted, or updated so that it is no longer taken be safe.

7. A method of processing a computer file to determine whether it contains a virus or other malware comprising:

a) generating data with regard to the file to characterise its identity and for thereby referencing a computer database to determine whether it is an instance of a known file;

b) selectively subjecting the file to a number of heuristic procedures to determine whether or not it contains, or is likely to contain, malware; and

c) determining, in dependence upon the record, if any, of the file in the database, whether the file can be regarded as safe and conducting the step b) such that the file, if the file is to be regarded as safe, is either subject to less thorough processing than if it were not so regarded or not subject to processing by the step b) at all.

8. A method according to claim 7 wherein the determining step c) controls the step b) in dependence on factors including the length of time for which the database indicates that the file has been known without malware-containing instances of it being detected.

9. A method according to claim 7 or 8 wherein the determining step c) controls the step b) in dependence on factors including sources, recorded in the database, from which instances of the file have originated.

10. A method according to claim 7, 8 or 9 wherein the determining step c) controls the step b) in dependence on factors including the number of times, recorded in the database, instances of the file have been processed.

11. A method according to any one claims 7 to 10, and including the step of updating the database in dependence upon the result of the processing of the file by the step b).

12. A method according to claim 11 wherein the updating of the database, in the event of the step b) determining that the file contains, or is likely to contain, malware is such that the record thereof in the database is deleted, or updated so that it is no longer taken be safe.

13. A system for processing a computer file to determine whether it contains a virus or other malware substantially as hereinbefore described and with reference to the accompanying drawings

14. A method of processing a computer file to determine whether it contains a virus or other malware substantially as hereinbefore described and with reference to the accompanying drawings

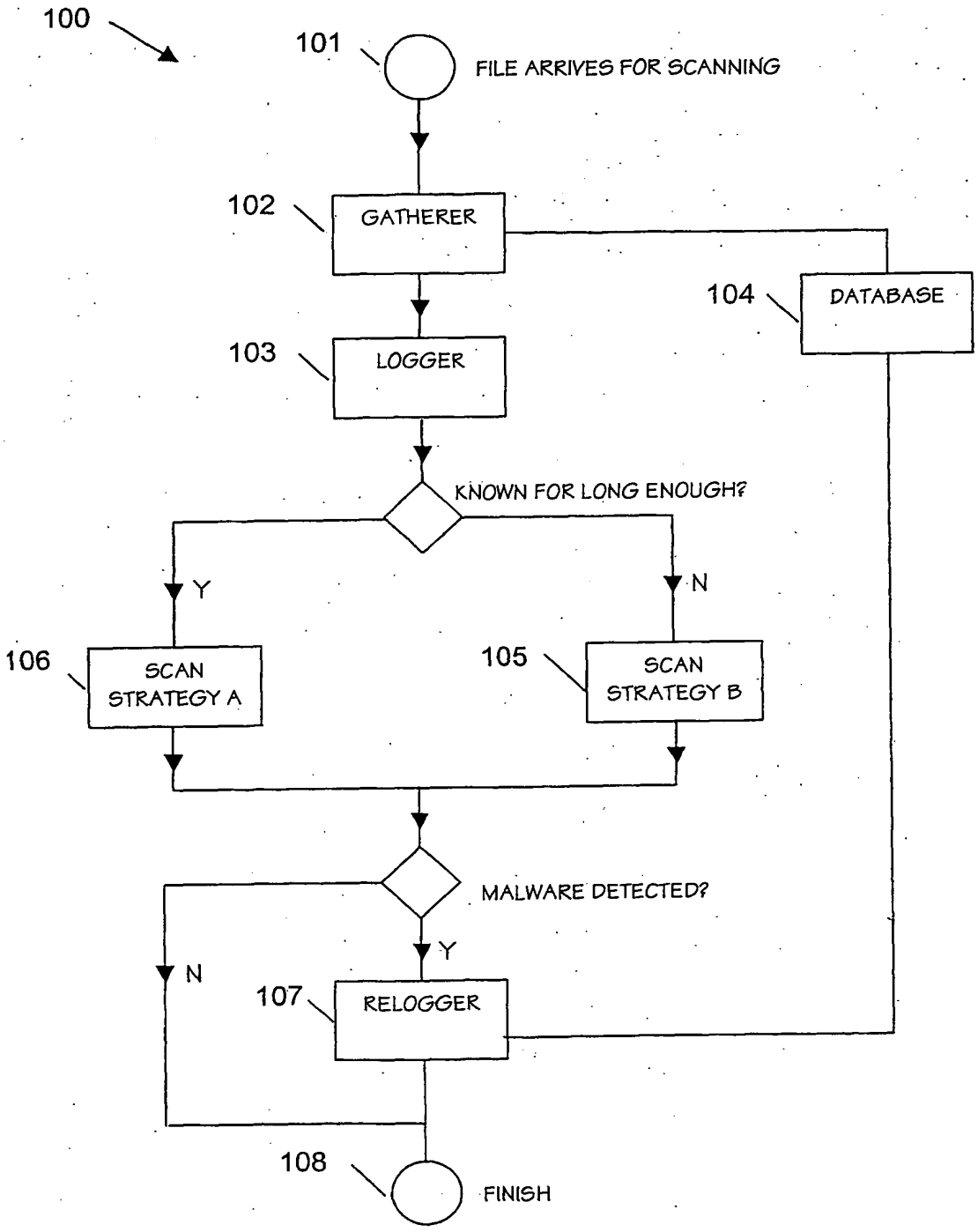

100
101
FILE ARRIVES FOR SCANNING
102
GATHERER
104
DATABASE
103
LOGGER
KNOWN FOR LONG ENOUGH?
Y
N
106
SCAN STRATEGY A
105
SCAN STRATEGY B
MALWARE DETECTED?
Y
N
107
RELOGGER
108
FINISH

Fig.1

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

OCT 31 2006
PATENT & TRADEMARK OFFICE
IAP44

Practitioner's Docket No.: 011765-0355986 **PATENT**
Client Reference No.: AF/JG/P10717US

OIPE
OCT 31 2006
PATENT & TRADEMARK OFFICE
IAP44

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: MELVYN MORRIS, ET AL. Confirmation No: 4785

Application No.: 11/477,807 Group No.: 2131

Filed: June 30, 2006 Examiner: Unassigned

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

SUBMISSION OF PRIORITY DOCUMENT

Attached please find the certified copy of the foreign application from which priority is claimed for this case:

Country	Application Number	Filing Date
Great Britain	0513375.6	06/30/2005

Date: October 31, 2006

P.O. Box 10500
McLean, VA 22102
Telephone: (703) 770-7900
Facsimile: (703) 770-7901
Customer Number: 00909

PILLSBURY WINTHROP SHAW PITTMAN LLP
Christine H. McCarthy
Registration No. 41844

THIS PAGE BLANK (USPTO)

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, [redacted] undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the [redacted] regulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Co[redacted]oller-General, hereby certify that annexed hereto is a true copy of the documents as orig[redacted]y filed in connection with the patent application identified therein.

I als[redacted]tify that the attached copy of the request for grant of a Patent (Form 1/77) bears an amen[redacted]nt, effected by this office, following a request by the applicant and agreed to by the Com[redacted]er-General.

In acc[redacted]ce with the Patents (Companies Re-registration) Rules 1982, if a company named in this ce[redacted]ate and any accompanying documents has re-registered under the Companies Act 1980 w[redacted]e same name as that with which it was registered immediately before re-registration save fo[redacted]ubstitution as, or inclusion as, the last part of the name of the words "public limited compan[redacted]their equivalents in Welsh, references to the name of the company in this certificate and any [redacted]panying documents shall be treated as references to the name with which it is so re-registe[redacted]

In accorda[redacted]ith the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or P[redacted]

Re-registrat[redacted]der the Companies Act does not constitute a new legal entity but merely subjects the [redacted]ny to certain additional company law rules.

CERTIFIED COPY OF
PRIORITY DOCUMENT

Signed [redacted]

Dated 12 July 2006

dti A DTI SERVICE

BEST AVAILABLE COPY

THIS PAGE BLANK (USPTO)

Patents Form 7/77

Patents Act 1977

(Rule 15)

The **Patent Office**

THE PATENT OFFICE
C
30 JUN 2005
NEWPORT

Statement of inventorship and of right to grant of a patent

The Patent Office
Cardiff Road
Newport
South Wales
NP9 1RH

1. Your reference	**13532:JK**
2. Patent application number *(if you know it)*	0513375.6 — 30 JUN 2005
3. Full name of the or each applicant	**Retento Limited**
4. Title of the invention	**Computer Security**
5. State how the applicant(s) derived the right from the inventor(s) to be granted a patent	**By virtue of an agreement between the inventors and applicant**
6. How many, if any, additional Patents Forms 7/77 are attached to this form? *(see note (c))*	**0**

7. I/We believe that the person(s) named over the page *(and on any extra copies of this form)* is/are the inventor(s) of the invention which the above patent application relates to.

Signature **Saunders & Dolleymore** — Date **29 June 2005**

8. Name and daytime telephone number of person to contact in the United Kingdom	**John Kensett** **01923 238311**

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there are more than three inventors, please write the names and addresses of the other inventors on the back of another Patents Form 7/77 and attach it to this form.

d) When an applicant does not declare any priority, or declares priority from an earlier UK application, you must provide enough copies of this form so that the Patent Office can send one to each inventor who is not an applicant.

e) Once you have filled in the form you must remember to sign and date it.

THIS PAGE BLANK (USPTO)

Patents Form 7/77

Enter the full names, addresses and postcodes of the inventors in the boxes and underline the surnames

Melvyn Morris
Redmire Gap
Intake Lane
Turnditch
Derbyshire DE56 2LU
United Kingdom

Patents ADP number *(if you know it)*: 91271840001

Paul Stubbs
7 Rookery Road
Wyboston
Bedfordshire MK44 3AX
United Kingdom

Patents ADP number *(if you know it)*: 91271920001

Markus Hartwig
Exchange House
494 Midsummer Boulevard
Milton Keynes
Buckinghamshire MK9 2EA
United Kingdom

Patents ADP number *(if you know it)*: 91272000001

Reminder

Have you signed the form?

Patents Form 7/77

THIS PAGE BLANK (USPTO)

Patents Form 7/77

Enter the full names, addresses and postcodes of the inventors in the boxes and underline the surnames

Darren Harter
178 Barnwood Road
Hucclecote
Gloucester
Gloucestershire GL4 3JZ
United Kingdom

Patents ADP number *(if you know it)*: 91272800 1

Patents ADP number *(if you know it)*:

Patents ADP number *(if you know it)*:

Reminder

Have you signed the form?

Patents Form 7/77

THIS PAGE BLANK (USPTO)

Patents Form 1/77
Patents Act 1977
(Rule 16)

The Patent Office

J114381/ 002 D02396 P01/77FEE.....
02JUL05 30.00 CHEQUE 0513375.6

Request for grant of a patent
(An explanatory leaflet on how to fill in this form is available from the Patent Office)

THE PATENT OFFICE
C
30 JUN 2005
NEWPORT

0513375.6

The Patent Office
Cardiff Road
Newport
South Wales
NP10 8QQ

30 JUN 2005

1. Your reference: *(optional)* **13532:JK**

2. Full name, address and postcode of the applicant or of each applicant *(underline all surnames)*:
Retento Limited
3 New Park Place
Pride Park
Derby DE24 8DZ
United Kingdom

Patents ADP number *(if you know it)*: 9127234001

If the applicant is a corporate body, give the country/state of its incorporation: **United Kingdom**

3. Title of the invention: **Computer Security**

4. Name of your agent *(if you have one)*: **Saunders & Dolleymore** BECK GREENER

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*:
9 Rickmansworth Road FULWOOD HOUSE
Watford 12 FULWOOD PLACE
Hertfordshire WD18 0JU LONDON WC1V 6HR

Patents ADP number *(if you know it)*: **1453001**

see sheet 30/6/05

5. Priority declaration: Are you claiming priority from one or more earlier-filed patent applications? If so, please give details of the application(s):

Country	Application number *(if you know it)*	Date *(day/month/year)*

6. Divisionals, etc: Is this application a divisional application, or being made following resolution of an entitlement dispute about an earlier application? If so, please give the application number and filing date of the earlier application:

Number of earlier UK application	Date of filing *(day/month/year)*

7. Inventorship: (Inventors must be individuals not companies) (Please tick the appropriate boxes)

Are all the applicants named above also inventors? YES ☐ NO ☑

If yes, are there any other inventors? YES ☐ NO ☐

8. Are you paying the application fee with this form? YES ☑ NO ☐

Patents Form 1/77

THIS PAGE BLANK (USPTO)

9. Accompanying documents: not counting duplicates, please enter the number of pages of each item accompanying this form:

Continuation sheets of this form:

Description: 26

Claim(s): 9

Abstract: 1

Drawing(s): 3 + 3 RM

If you are not filing a description, please give details of the previous application you are going to rely upon:

Country	Application number	Date of filing

10. If you are also filing any of the following, state how many against each item.

Priority documents:

Statement of inventorship and right to grant of a patent (Patents Form 7/77): 1

Request for search (Patens Form 9A/77): 1

Request for substantive examination (Patents Form 10/77):

Any other documents: (please specify)

11. I/We request the grant of a patent on the basis of this application.

Saunders & Dolleymores

Signature(s): Date: **29 June 2005**

12. Name, e-mail address, telephone, Fax and/or mobile number, if any of a contact point for the applicant:

John Kensett
01923 238311
info@dolleymores.com

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you are resident in the United Kingdom and your application contains information which relates to military technology, or would be prejudicial to national security or the safety of the public, section 23 of the Patents Act 1977 prohibits you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

A leaflet on how to fill in this form is available from the Patent Office. If you would like a copy of the leaflet it is available on our website at http://www.patent.gov.uk/patent/info/fact05.pdf or alternatively you could telephone 08459 500505 or send an email to enquiries@patent/go.uk to request a copy.

THIS PAGE BLANK (USPTO)

Computer Security

This invention relates to computer security. In particular, it relates to methods and systems for detecting whether a process operated on a computer is likely to be one which is safe to operate or is likely to be malware.

Background to the Invention

The term "malware" is intended to mean any process which might possibly be harmful to the computer, its software, firmware or data stored on the computer or its performance, reliability, control and utilisation. Non-limiting examples of malware are viruses, Trojans, worms, spyware, adware or similar.

Anti-malware systems have been in existence for nearly as long as malware processes have existed and several service providers produce anti-virus and other anti-malware software and services. Generally, the method of dealing with malware is that when new types of malware are released upon the Internet for example these are eventually detected and possibly infect user's computers or special monitoring computers specially connected to the Internet for this purposes. Once new items of malware have been detected, then the service providers in the field generate signatures which attempt to deal with these and these signatures are then released as updates to their anti-malware programs. Heuristic methods have also been employed.

These systems work well for protecting against known risks. However, since they rely on signature files being generated and/or updated, there is inevitably a delay between a

new piece of malware coming into existence or being released and the signatures for combating that malware being generated/updated and supplied to users. Thus, users are at risk from brand new malware for a certain period of time which might be up to a week or even more. During this time, many users can inadvertently become infected before proper signatures have been disseminated.

WO 2004/0976 A2 describes a system which analyses computer files received or generated by a local computer and compares these with a database to determine whether the file is known and if so whether it has been known about long enough that it can be regarded as safe. If it can be regarded as safe, then the file is subject to a less thorough processing for detecting malware then a program about which it is not known whether it is safe or not.

The system disclosed therein appears to only look at whole files rather than processes. A process may be included as part of a file or may be a separate item altogether. Furthermore, this previously proposed system assumes that once a file has been known about for "long enough" it is inherently safe. This is not necessarily the case in real life when, for example, the active payload of a virus or Trojan may only be programmed to happen at a particular date or event which may be many months or even years after the process has been first run or is released. Thus, just looking at the age of a file is an unsatisfactory way of determining whether it is properly safe and will remain so.

Furthermore malware may be programmed to become effective based on other criteria which may not be date or duration based, for example threshold values or other parameters.

An object of the invention is to be able to detect whether a process or component operated on a computer is likely to be one which is safe to operate or is likely to be malware.

A further object is to seek to eliminate or reduce risk in malware in the inter-regnum between its emergence and the generation of signatures. This might be known as zero-date risk.

Brief Summary of the Invention

According to the present invention in a first aspect, there is provided a computer network comprising at least one local terminal, at least one anti-malware service provider database terminal remote from the terminal, a local database on the local terminal which is populated as processes are run or downloaded onto the terminal, a server database on the or each server database terminal and means for determining, when a process is found at the local terminal, whether that process is known or unknown at the local terminal and is considered safe or unsafe and, if unknown, whether the process is known at the server database and is considered safe or unsafe, and means for dynamically altering the database contents and safe/unsafe status using predetermined conditions.

The invention further provides a method of protecting a local computer terminal against malware, comprising providing a local process signature database on the local computer and a remote database on a computer which is remote from the local terminal, searching the local database at the terminal when a process is run to determine whether the process is known or not at that terminal and, if known, is considered safe or unsafe; if the process is not known then searching the remote database to determine if the process is

known and, if so, is determined safe or unsafe; and if the process is known at the remote database submitting the safe/ unsafe result back to the local terminal.

According to a further aspect, there is provided a computer system including means for minimising the risk of malware infection, the system comprising at least one local terminal and a community database comprising a plurality of databases each on a different computer remotely distributed from each other and from the local terminal, whereby signature and/or mask files representative of known processes are stored and built up on the community database to obtain, over time, a community database storing signature and/or mask files relevant to or known processes and indicating whether each process is considered safe or unsafe.

According to a yet further aspect there is provided a method of determining the protection which a computer terminal has from malware, the method comprising providing a database remote from the terminal; transmitting from the terminal to a database details of all or selected security products operating at a point in time on that terminal; obtaining similar information from other terminals connected to the database; searching for the same combination of security products used by machine; and identifying any malware processes which were not identified by said other machine having that particular combination of security products and process.

In a further aspect, the invention comprises a database populated with information from a plurality of client terminals representative of the combination of security products and processes used by each client terminal at a point in time, and details of any known

malware processes that were not detected by said unique combination for each particular terminal combination.

Embodiments of the invention may respond to events, not to recognition of files themselves, and to time, both points in time when events occur and to elapsed time, and may also be responsive to volume and acceleration of apparently associated events captured by the database.

Description of the Drawings

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 shows a network;

Figure 2 is a flowchart showing the running of a process;

Figure 3 is a flowchart showing further processing steps; and

Figure 4 is a flowchart showing an information obtaining stage.

Detailed Description of Preferred Embodiments of the Invention

Referring to Figure 1, a computer network is generally shown as being based around a distributed network such as the Internet 1. A local computer terminal 2 is connected to the Internet by any suitable means as will be known in the art. The terminal 2 may be a computer, a PDA, mobile phone, an interactive television or any other device capable of loading and operating processes. A process may be a computer file, may be part of a file or may be a sub-program, macro, web page or any other piece of code to be operated by the terminal or any other event whether executed, emulated, simulated or interpreted. A

process 3 is shown in the figure and this may be applied directly to the terminal locally, as shown by line 4 or may be downloaded to the terminal via the Internet as shown by line 5.

Systems according to the invention utilise a database connected to the Internet and which can interact with the local terminal when that terminal runs a process to determine whether the process is safe or unsafe. The database may be provided on a single server connected to the Internet but is preferably provided over a distributed "community" database 5 as shown in Figure 1, which comprises a number of data servers 5a, 5b, ... 5n. The provision of a distributed community database of this type firstly enables each individual device to require less storage and more importantly makes the database much less vulnerable to denial of service or other attacks since if one server is attacked, then, particularly if entries in the database are duplicated, performance of the database will not be significantly affected.

The community database is populated, over time, with information relating to each process run on all of the local terminals within the system. A user of a terminal might subscribe to the services of the database provider or may alternatively be able to download software for free, which enables him to use such services. Data representative of each process may take the form of a so-called signature or key relating to the process and its affects and to a mask which sets out the parameters of the processes performance and operation. The mask will be described in more detail further below.

Figure 2 shows how the system may run in one embodiment.

At the start point 21, a computer process is run at the local terminal, either directly or being downloaded or run direct from a site over the Internet 1. The local terminal then searches its local database 6 at step 22 to search for a signature or key representing that particular process. If the local signature is present, it will indicate either that the process is considered to be safe or OK (in the manner described further below), or will indicate that that process is considered unsafe. An unsafe process might be one which has been found to be malware or to have unforeseen or known unsafe or malevolent results arising from its running. If the signature indicates that the process is OK, then that process is allowed to run at step 23. If the signature indicates that the process is not OK, then the process is stopped 24.

Alternatively, the user may be presented with a choice of whether to allow the process to run or not, but is informed that the process is not considered safe. Ultimately, it will be the user's choice whether to risk running the process or not.

In preferred embodiments there may be more than two 'states': GOOD-BAD-UNKNOWN AT LOCAL TERMINAL-UNKNOWN IN 'COMMUNITY DATABASE'. If the process is GOOD then it is, for example, always allowed to run. If BAD then it is NOT allowed to run - the user is given no choice in this. If UNKNOWN locally or in the Community then the user may be given informed choices of viewing or not. Other states and levels may of course be used.

If the process is unknown, then this means that the process has not been run before at that local terminal. In this case, details of the process are passed over the Internet or

other network to the community database and the community database is searched for a signature for that process at step 25. The community database is supplied with signatures representative of programs or processes run by each monitored local terminal (and in a typical version in the field there may be several millions of these terminals and so any processes which are newly released upon the Internet, etc will soon be found and signatures created). When the community database is searched, if the signature is found and indicates that that process is safe, then firstly a copy of the signature is sent to the local terminal database (step 26), to begin populating the local database so that the local terminal has this information immediately to hand then next time the process is run and a message is passed back to the local terminal allowing the process to run.

If the signature is found and this indicates for some reason that the process is unsafe, then again the signature is copied back to the local database and marked UNSAFE (step 27), and then the process is either stopped (or not allowed to run) or the user given an informed choice whether to run it or not.

If after the entire community database has been searched the process is still unknown, then it is assumed that this is a brand new process which has never been seen before in the field. A signature is therefore created representative of the process at step 28 and this signature is initially marked as bad or unsafe at step 29. The signature is copied to the local database of the terminal which first ran the process at step 30 and, in addition, a copy of the file or process itself may be requested at step 31 by the community database from the local terminal. A message may then be passed to the local terminal which

instructs the terminal not to run the process or alternatively the user may be given informed consent as to whether to allow the process to run or not.

If the user chooses to run a process which is considered unsafe because it is too new, then that process may be monitored by the local terminal and/or community database and, if no ill effect occurs or is exhibited, it may then be considered to be safe after a period of time of n days for example. Alternatively, the community database may keep a log of each instance of the process which is found by the many local terminals forming part of the network and after a particular number of instances have been recorded, possibly with another particular number of instances or the process being allowed to run and running safely, the signature may then be marked as safe rather than unsafe. Many other variations of monitoring safety may be done within this concept.

The signature of a process may comprise or include a mask which can be built up with use of that process and which indicates the particular types of behaviour to be expected from the process. If a process is allowed to run on a local terminal, even if the initial signature search 22 indicates that the process is safe, then operation of that process may be monitored within the parameters of the mask. The mask might indicate for example, the expected behaviour of the process; any external requests or Internet connections that that process might legitimately have to make or call upon the local terminal to make, including details of any ports or interfaces which might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, which may be expected to be interrogated by that

process and so on. Thus, the mask can give an overall picture of the expected "normal" behaviour of that signature of that process.

The behaviour of the process is continually monitored both at the local terminal and also by continually sending information to and from the community database relating to that process to determine whether the process is running within its expected mask. Any behaviour which extends beyond the mask is identified and can be used to continually assess whether the process continues to be safe or not. Thus, if the process, on a regular basis (say monthly or yearly) opens a new port to update itself for example or to obtain regular data, then this information is flagged. If it is found that the process has done this on other terminals and has had no ill effects, or this behaviour is known from other processes and known as being safe, then this behaviour might be considered as safe behaviour but the mask is then modified to allow for this. If it has been found previously that this new behaviour in fact causes unsafe or malevolent results, then the process can then be marked as unsafe even if previously it was considered safe. Similarly, if the process attempts to connect to a known unsafe website, database or to take action which is known as generally being action only taken by unsafe programs then again the process may be considered to be unsafe.

Thus, embodiments of the invention can continually modify a processes safe or unsafe rating with time.

This is shown in Figure 3. However, Figure 3 also introduces the concept that any process can be pre-authorised. Typically, it will be pre-authorised by a trusted partner, such

as a major software company, a verification authority, a Government department and so on. Pre-authorisation enables a supplier of a new process, which has not been released before, to get pre-authorisation for that process, and includes the provision by that supplier of a mask detailing the expected and allowable behaviour of that process. Thus, the mask defines the limits of that processes behaviour. Pre-authorisation may be done by the software provider themselves (if they are considered to be a Trusted Provider), or by the software provider providing details of that process to a trusted authority such as a Government or verification authority who in turn verifies the process and then enables the relevant electronic "documentation" to be provided to the database provider for its required pre-authorisation technique. So a new process or software item by a trusted software provider can be pre-authorised and have no risk of being prevented from running. Similarly, if a bespoke item of software is created for an organisation then that organisation may be able to give or obtain immediate pre-authorisation

Referring to Figure 3, when a process is run, the local and/or community database are searched as before at step 31. If the process is not a pre-authorised one, then the steps of Figure 2 are taken, for example and the process might be allowed to run or not at step 32. If the process is pre-authorised, as determined by step 33, then it is immediately allowed to run, step 34. This may be the end of the matter. However, in a preferred variation, the process is then monitored whilst running, and is monitored each time it is run in the future in a monitoring state step 35 to determine whether its behaviour falls within its pre-authorised mask 36. If the behaviour falls within the pre-authorised behaviour then the process is allowed to continue to run. If the behaviour extends beyond the allowed mask,

such as by trying to instigate further processes or connections which have not been pre-authorised, then this behaviour is flagged at an alert step 37.

Various actions could be taken at this stage. The process might simply not be allowed to run. Alternatively, the trusted authority which initially enabled pre-authorisation might be contacted, who may be able to confirm that this behaviour is acceptable or not. If it is acceptable, then the mask could be modified. If not acceptable, then the process might be marked as unsafe. Many other actions may be taken upon the noting of such an alert state.

If the process has been found not to be pre-authorised at step 33 and is allowed to run, then the process is monitored at step 38 in order to generate a mask 39 representative of the normal behaviour of that process. Data representative of this mask might be sent to the community database for scanning when other terminals run that process. By continually monitoring a process each time it is run or during running of the process, any behaviour which differs from previous behaviour of the process can be noted and the mask can modified, or this behaviour might be used to determine that a process which was once considered safe is now considered to be unsafe.

When a user of a local terminal subscribes to a service according to an aspect of the invention, or downloads software enabling this, his system may periodically provide information to the community database representative of all or selected ones of the software products loaded on or available to his system. In particular, this may be all the locally-loaded security products (such as anti-malware systems including anti-virus software, anti-

spyware, anti-adware and so on), firewall products, etc and includes version details, specific settings and details of which signature files are currently loaded, and also information such as which files are operating and the particular version and software settings at any time and similar types of software. Whilst these may be provided individually to the community database, it may prefer to generate a signature or key at the local terminal representative of the sum information (eg by a hashing or compression function) which can then be transmitted to the database for storage. Figure 4 shows how the details of all local security products, versions, signature files, firewall settings, etc 40 are used to create a key 41. The key is transmitted to the community database. Since the community database is provided with such information from many, possibly millions, of user's terminals, it can search for other terminals having the same combination of security products including the same setting, signature files loaded and so on.

The community database can also provided with details of processes run by every machine and knows whether a process has been detected by each local machine or not, hence the database can be used to obtain information as to whether a particular, specific, combination of security products, including settings and signature files existing at a point in time, render a particular terminal susceptible or vulnerable to any particular malware process. That is, if the database knows that a terminal in the past has version A of anti-virus product B, with downloaded signature update C and also has a firewall D with particular settings E and perhaps anti-spyware software F with signature updates G, but that a particular malware process P was not detected by this combination of programs at that point in time, then this information can be provided to a local terminal which is known as

having that combination of security programs/settings and can be used to indicate that that terminal is vulnerable in the short term to attack by that particular malware process.

This information can be presented to the user either by displaying a window 43 on the screen display of the local terminal or by directing the user to a particular website which explains the position in more detail. In other words, the community database, when provided with information relating to all the security products on a particular terminal at a particular time, searches for events for processes marked as "bad" that occurred on systems with that particular mix of security products and were not locally detected. It might indicate that process P1 was "bad" and process P2 was "bad" and so on. This information is then fed back to the terminal user either directly or by directing him to a website. A message might be provided on a window on his terminal which reads, for example;

"Your system currently comprises the following security products, versions and signature files:

Anti-virus system A, version A_1, signatures files A_{11} dated 01/06/05,

Anti-spyware system B, version B_1, signature files B_{11} dated 04/06/05

Firewall C, updated 30/05/05, etc ...

Be aware that your system may not protect against malware process P1, P2, etc"

The message will generally give these bad processes the names commonly associated with them (eg Sobig virus, and so on). The user may be directed to further guidance on a website for example where he may be given suggestions of how to deal with this situation.

While there may be a suggestion to download different software, it could alternatively be suggestions for downloading particular patches, closing particular ports, or so on. This information could be provided virtually in real-time and will generally only be applicable to a particular point in time.

It will be seen from the above that the community database can store a great deal of information about the security and other software products installed on a user's terminal and also about potential risks and malware processes and their effects. Up to now, anti-virus software providers generally provide databases on their website which a user can look up for details of particular viruses and possible suggestions as to preventing infection by them or eradicating any infection. A database generated in accordance with embodiments of the present invention, can provide a lot more detail including:

the name of malware and all the other names it is known by;

the way it propagates its prevalence; and

which folders and authorised processes it infects or affects, and so a more extensive database than before can be built up which users can interrogate. The database can also include details, as described above, of the interaction between these or various security products on their terminal and various malware products in order for the user to determine himself the best combination. Also, by continually obtaining details of the software set up on user's terminals and storing these as archive files, if a user's terminal does crash or is affected by malware, the contents of the archive of the community database may be used to provide a more effective backup to restore the user's terminal to a previous safe configuration.

Note that whereas much prior art Anti-Virus software attempts to identify by a scan of files newly introduced to the local computer or network that are known to be bad by virtue of a match against an index file of malware signatures and stops any part of said files from executing, embodiments of the present invention trap the execution activity of all code components on the subject computer and check the signature of each code component against a local database and/or remote database of previously benign activity. If any component is about to execute a not previously known type of activity, that execution is, by default, blocked. Accordingly, in embodiments of the invention all executing code is monitored and any code executing in a manner not previously known to be benign is blocked. Sometimes it is not possible to trap which particular component (of a known suite of executing code components) is producing suspect activity. In such a case composite signatures can be used to tie back or relate to a particular suspect event and thereby to a specific component that is suspect malware. For example, the well known browser "Internet Explorer" has different installations (for different operating systems, configurations etc.) which use many different types of subsidiary code components. At execution time it is not always possible to distinguish which particular code component is triggering a previously unknown event. The community database can collect data that allows a signature to be made of any user's configuration of composite sets of components, to achieve this.

In some embodiments of the invention meta data can be captured in the community database, this enables current and refined newly devised heuristics to be run on the community meta database. This in turn allows the ability to model heuristic techniques which themselves can be propagated to the agent level when the new heuristic has been

proved. "what if" modelling can also be done on the community metadata – e.g. what if we mark this code component as bad for the time being, or, what other applications among what number of users might be affected?

Amongst other types, the types of meta data captured might be:

"alias" – related to the frequency with which a code component appears in code that has a name different from other named code it has already been seen to be present in, and;

"event key" – being a signature that ties an event to a particular configuration of component code element present on the machine on which the event has occurred. This permits lowest common denominator/ higher-common factor type analysis on the community metadata to determine code components that are malware.

It should be noted that in contrast to previously proposed "signature" based methods, with methods according to the present invention there is intrinsic protection even for offline users since the system can be arranged, while out of contact with the community database (i.e. off-line or because of a problem connection therewith), to "lock down" (e.g. prevent execution of, or give the user informed consent) an unknown process. In effect, if a process is unknown at the local terminal then in such circumstances it can be considered, at least initially, automatically to be bad or suspect.

A further situation arises when users wish to install authorised software while offline.

By prompting the user when they attempt to install new software while offline, the execution of the installation can itself be "authorised" – the authorisation by the user is treated as a "Genesis" event and will be so termed hereinafter. However there are some processes commonly used in installation of software that communicate with other existing programs on the installation machine, e.g. Microsoft's MSIEXEC.

The Genesis approach involves a process that generates signatures as a result of the user's authorisation on the user's terminal.

Further, systems according to the invention may download signatures of so-called "licensed installer", being known code components, to users machines while they are online. This allows the operation of a method such that a "licensed installer" is permitted to execute while a Genesis event is "current", i.e. within a period of minutes after an authorisation from the user. There may be refinements to this method, such as to prevent licensed installers executing if a media change has occurred during the currency of a Genesis event.

The method may be such that a "licensed installer" together with any child processes of the licensed installer is permitted to execute while a GENESIS event is "current", i.e. within a period of minutes after an authorisation from the user. However, any unknown processor, which may be malware, may still be detected and processed.

There may be refinements to this method, such as to prevent licensed installers executing if a media change has occurred during the currency of a Genesis event and super-trusted installers such as for example 'Windows Update', where all new processes created by the super-trusted installer are marked immediately as safe.

Preferably signatures of licensed installers are ALWAYS downloaded, as and when added to the community database, to an agent machine while online. There probably are not that many of these.

When the terminal is connected to a network, the other aspect to this feature is that the system will, if the about to be installed program is not known to the community database, block the installation, or alert the user, i.e. "you are about to install some software, this software is not known to [community]. Are you sure you wish to proceed?". Having a limited number of licensed installers facilitates download, as minimal data is required, compared to downloading virus signature files.

Note that embodiments of the invention can be applied within servers (e.g. in a server-enterprise environment) and therefore can be executed on any computer or terminal within a network, including ones which may be connected to a network or not, from time to time. Even a server itself may be a local 'terminal' as the term is used in this specification.

The potentially high volume of active users (i.e. terminals) (which will in practice be many millions) gives a high probability of at least one of them being infected by new malware. The speed of propagation is detected and recorded in embodiments of the

invention so that the propagation of malware can be detected and malware designated as bad, and can be used to identify an incident emerging across the community as malware, or simple volume of infection can be used as a trigger.

In a further feature, difference of naming, of an otherwise identical piece of code combined with acceleration of first attempts to execute the code within the community allows pattern matching that will show up an identical signatured piece of code as bad.

This feature allows the statement in some embodiments, "nothing will propagate in our community faster than X without being locked down", so that if any process or event propagates quicker over a given duration, it is marked as bad.

This process can be automated by the identification of the vector of propagation in the community, from timestamp data held in the community database and the marking of a piece of code that has these attributes as bad. By comparison, it is believed that all other anti-malware providers rely on a simplistic known bad model and therefore are reliant primarily on malware infection actually occurring on terminals and being reported.

One method for plotting the acceleration of a suspect piece of code involves timestamping of each report of the same piece of code, showing how many terminals had reported the code within successive timeslots, within seconds or minutes initially but then perhaps against a more logarithmic set of lengthening time periods i.e. the method by which timestamping is done usually cannot be maintained at the same rate as during the earliest period of first detection. This logarithmic method avoids the database becoming swamped

with stamp data by avoiding a linear method of data capture, e.g. every few seconds of all incidents of the same code. Where appropriate, in some embodiments, suspect files can be uploaded automatically from individual terminals to the community database for more detailed examination.

Amongst other features and inventions, the following aspects form part of the present invention, any one or more of which, or any combination of which, or any one or more feature or combination hereinbefore described, are considered as possible claimable inventions in their own right.

a) The concept of a known and trusted parent process having an identifying component in its signature that uniquely defines the process (referred to here as a "gene"). By virtue of such gene being inherited in the signature of all of the parent's child processes and through them further child processes (referred to here as "issue") such signatures are all automatically recorded in the local database as good. The concept also allows the inherited gene to be removed should any issue abuse it, such as by creating a suspect child or one that becomes suspect. This concept, combined with Genesis, allows the extension of the trusted status of parent processes notwithstanding the involvement of surrogate programs (like Microsoft's MSIExed for example) in generating further issue. The marking of child process signatures as good might otherwise fail because of the involvement of surrogates like MSIExec, and other licensed installer programs. Alternatively, a parent process which is not necessarily known and trusted may have a unique identifier, which may possibly be considered inherently safe, unsafe or given other attributes. More than one identifier (gene) may be used. The attribute may be security related or may be related to other aspects such

as being subscription related so that a process can only be executed whilst a subscription is current, and so on.

b) Parental creation of a signature for a child or children including the ability for these to be automatically marked as either good or bad depending on the parent's behaviour and determination. Note that in some embodiments the product can 'watch' or monitor the birth of a child process and automatically create the signature upon arrival.

c) Genesis creation through a licensed or trusted installer or surrogate who may pass on the true parents (99% good) genes. The 'genesis' process allows the product to capture the birth and apply the appropriate genetic determination.

d) A feature may be included that allows for automatic forfeiture of a child's inherited ability to trigger the automatic creation of signatures on any further births because the child, as parent has already produced bad offspring. Preferably, a rule is that if a file has one bad offspring then the inherited ability can be automatically removed.

e) An ability to watch the clones or identical twins (i.e. the same process running on other systems in the community) to compare the pattern of their off-spring to ensure that Genesis events are definitely part of the same genetic line and remove any determination for an off spring which is not prevalent as a child of the clones or identical twins, or to make the determination unknown causing it to be watched intensely throughout the community before a determination can be made.

Some of the features a) to e) represent a solution to the issues of producing a product which can be used effectively without 100% reliance on being permanently connected to the internet, which is often impractical. Examples of this are Windows Update, and other processes used more and more by vendors who wish to be able to roll out other product updates automatically across the worldwide web.

Possible benefits of these types of features above conventional software are as follows. Antivirus software tends to have a cache of known bad signatures. The problem is keeping this up to date. Take the simple example of someone buying a new computer. The computer comes with an antivirus product preloaded with a signature cache. Between the time when the PC was built, shipped to the store and bought by the user several days or weeks will have passed. The user starts the PC and is exposed to any new virus or malware which was created after the PC was built. Full protection requires the user to connect to the internet and download updates. This cannot be guaranteed to occur ahead of other activities by the user on the internet (almost physically impossible to guarantee). With a local cache of known good processes, as in the embodiments of the present invention, it is possible to ship the computer/terminal preloaded with a pre-generated cache of signatures for all of the good (clean) software preloaded by the computer manufacturer. In this case the user can connect to the internet knowing that any new or updated programs will be immediately detected and verified. Also any auto-updating software can function forcing signatures to be automatically built for its children and more remote off-spring (i.e. grandchildren, great-grandchildren, etc).

f) The ability to monitor a process along with any optional sub-processes as an homogenous entity and then compare the activities of the top level process throughout the

community and deduce that certain, potentially malevolent practices only occur when one or more specific sub-processes are also loaded. This allows effective monitoring (without unnecessary blocking) of programs like Internet Explore (browsers) whose functionality may be easily altered by downloadable optional code that users acquire from the internet, the principal source of malevolent code today.

g) The ability, using the community database to make early diagnosis, or simply to take precautionary measures to stop potentially fast propagating worms and malware very, very early in their life cycle. Given that it is theoretically possible to create a worm which could infect every computer connected to the internet within a matter of a few minutes this feature is highly desirable.

h) The ability to make even faster determination by combining data defining the speed of propagation of a new piece of software with meta data collected by the Agent and fed to the database. This includes monitoring processes that attempt to conceal themselves from the user by randomly changing name and/or location on the local terminal, or every computer or network on which it installs. It also includes a process's attempt to create an identical copy (i.e. with identical code contents) on the computer but with a different name. This is a classic attribute of a worm.

i) The use of meta data derived from terminal agents defining the behaviour of a process across the community and held in a central database to model, test and create new automated rules for use in the central database and as rules that may be added to those held

and used in the local (terminal) database to identify and determine the agent's (system) response to new or unknown processes and process activity.

j) The use of two signature processes. One which defines the entire files and will change with almost any change to the files content. The other which attempts to define only the processing instructions of the process which changes much less. By tracking the occurrences of both signatures individually appearing with different counterparts it is possible to identify processes which have been changed or have been created from a common point but which have been edited to perform new, possibly malevolent functionality.

k) The use of an event mask which enables an agent to monitor a known process and detect if it performs abnormal events which could indicate the emergence of a new purpose from a Trojan. This enables the agent to take immediate protect action to block the abnormal activity and to report it to the central database.

l) A technique to enable an efficient and practical form of communication between the agent (i.e. local terminal) and the remote database which allows the local terminal and community to operate without unnecessary delay or impact on the users' enjoyment of the computer and processes he/she choose to use.

m) The technique of recording which security products, operating system, updates, settings and security patches are installed and active when an event occurs and to use this to help assess the likehood of a malevolent process bypassing specific security

products or combinations thereof. To use this information to warn users about the need to bolster their defences to prevent attack.

Claims

1. A computer network comprising at least one local terminal, at least one anti-malware service provider database terminal remote from the terminal, a local database on the local terminal which is populated as processes are run or downloaded onto the terminal, a server database on the or each server database terminal and means for determining, when a process is found at the local terminal, whether that process is known or unknown at the local terminal and is considered safe or unsafe and, if unknown, whether the process is known at the server database and is considered safe or unsafe, and means for dynamically altering the database contents and safe/unsafe status using predetermined conditions.

2. A system as claimed in Claim 1, wherein the server database is distributed over a plurality of server computers.

3. A method of protecting a local terminal against malware, comprising providing a local process signature database on the local and a remote database on a computer which is remote from the local terminal, searching the local database at the terminal when a process is about to be run to determine whether the process is known or not at that terminal and, if known, is considered safe or unsafe; if the process is not known then searching the remote database to determine if the process is known and, if so, is determined safe or unsafe; and if the process is known at the remote database submitting the safe/ unsafe result back to the local terminal.

4. A method as claimed in Claim 3, wherein when a process is first seen, a signature file is created at the local terminal and/or remote database representative of that process.

5. A method as claimed in Claim 3 or Claim 4, wherein if the process is not known at the remote database, a signature of the process is created at the remote database and initially marked as unsafe and a copy of the signature is sent to the local database.

6. A method as claimed in Claim 3, wherein if a process is marked as safe by the local or remote database it is allowed to run.

7. A process as claimed in Claim 3, wherein if a process is marked as unsafe by the local or remote database it is not allowed to run.

8. A process as claimed in Claim 3, wherein if a process is marked as unsafe by the local or remote database the user of the local terminal is provided with an informed choice as to whether to allow the process to run or not.

9. A method as claimed in Claim 3, wherein a mask is generated for each or a collection of processes, which mask defines the acceptable behaviour for that process or collection.

10. A method as claimed in Claim 3, wherein the mask includes one or more parameter selected from the group comprising;

the expected behaviour of the process or collection of processes; any external requests or Internet connections that that process or collection might legitimately have to make or call upon the local terminal to make, including details of any ports or interfaces which might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, which may be expected to be interrogated by that process or collection.

11. A method as claimed in Claim 9, wherein after the process is allowed to run, its behaviour is monitored and if its behaviour extends beyond that allowed by the mask an alert is raised.

12. A method as claimed in Claim 10, wherein after the alert is raised, a determination is made as to whether the modified behaviour renders the process safe or unsafe, even if the process was originally considered safe.

13. A method as claimed in Claim1 or Claim 3, wherein a process may be pre-authorised.

14. A method as claimed in Claim 13, wherein the process is pre-authorised by a software provider or by a trusted authority.

15. A method as claimed in Claim 13, wherein if a process is not pre-authorised and is allowed to run, its behaviour is monitored and is used to create a mask representative of its behaviour, the behaviour continuing to be monitored to determine whether it expands

beyond the mask, in which case the modified behaviour can be analysed to determine whether the mask itself should be modified to take the modified behaviour into account or whether the modified behaviour indicates that acceptable behaviour has been exceeded and the process should now be considered unsafe.

16. A computer system including means for minimising the risk of malware infection, the system comprising at least one local terminal and a community database comprising a plurality of databases each on a different computer remotely distributed from each other and from the local terminal, whereby signature and/or mask files representative of known processes are stored and built up on the community database to obtain, over time, a community database storing signature and/or mask files relevant to or known processes and indicating whether each process is considered safe or unsafe.

17. A system as claimed in Claim 16, wherein even after signatures and masks have been established, behaviour of a process is monitored to determine whether its status is still safe or unsafe.

18. A method of determining the protection which a computer terminal has from malware, the method comprising providing a database remote from the terminal; transmitting from the terminal to a database details of all or selected security products operating at a point in time on that terminal; obtaining similar information from other terminals connected to the database; searching for the same combination of security products used by machine; and identifying any malware processes which were not identified by said other machine having that particular combination of security products and

process.

19. A method as claimed in Claim 18, further comprising providing information to the user of said first local terminal that his system may be susceptible to attack by said malware processes.

20. A method as claimed in Claim 19, wherein the combination of processes includes security products, versions, signature files loaded and other security products.

21. A method as claimed in Claim 19, wherein information is provided to a user on a display associated with the local terminal.

22. A method as claimed in Claim 19, wherein the information is displayed on a website.

23. A database populated with information from a plurality of client terminals representative of the combination of security products and processes used by each client terminal at a point in time, and details of any known malware processes that were not detected by said unique combination for each particular terminal combination.

24. A method as claimed in claim 3 whereas files, or data representative of files can be uploaded from the local database to the remote database.

25. A method as claimed in claim 3 wherein composite sets of data are analysed to

determine their status.

26. A method as claimed in claim 3 wherein metadata is collected and stored in the local and/or remote database.

27. A method as claimed in claim 26 when the metadata includes at least one member of the set comprising "alias" and "Event Key".

28. A method as claimed in claim 3 whereas if a local terminal is not connected to the remote database and a process unknown to that terminal is attempted to be executed an alert is raised that the process might be unsafe.

29. A method as claimed in claim 28 when the possibly unsafe process is not allowed to execute.

30. A method as claimed in claim 3 wherein a process unknown to a terminal can be preauthorised for loading onto or executing on said terminal while the terminal is not connected to the remote database.

31. A method as claimed in claim 30 wherein one or more "licensed installer" signatures are pre-supplied to said terminal.

32. A method as claimed in claim 3 wherein volume of infection and/or speed of propagation is used to determine whether a process is safe or unsafe.

33. A method as claimed in claim 32 wherein a vector of propagation is used to determine whether a process is safe or unsafe.

34. A method as claimed in claim 32 or 33 wherein propagation is determined by timestamping each report of the same process, and providing an indication of the number of terminals reporting attempted invocation of the process within successive time slots.

35. A method as claimed in class 34 wherein an at least partially logarithmic timescale is used.

36. An anti-malware system, arranged to detect the spread of new processes across a network and to consider a process as bad if the volume or speed of propagation exceeds a predetermined threshold.

37. An anti-malware system as claim in claim 36 wherein each report of the same process is timestamped and comprising counting the number of terminals in the network reporting that process in successive time periods.

38. An anti-malware system as claimed in claim 37 wherein a logarithmic set of time periods is used.

39. A method as claimed in claim 3 wherein there are more than two possible states representing different levels of safe or unsafe, and the process is allowed to be run or not, or the user given informed choice, depending on the level.

40. A method as claimed in claim 39 wherein the states are GOOD-BAD-UNKNOWN LOCALLY and UNKNOWN IN REMOTE DATABASE.

41. A method as claimed in claim 3 wherein an application may be labelled as super trusted and any child processor generated, or instigated as a result thereof, be considered safe at least for a period.

42. A computer program product or method, wherein a signature relating to a parent process includes a component that uniquely identifies it and which is inherited in the signature of any child or other descendant of the parent, such unique identifier being used to ascribe one or more attributes to the descendants.

43. A product or method as claimed in Claim 42, wherein the parent process is known and trusted and whereby an attribute is that the descendant process is safe.

44. A product or method as claimed in Claim 42 or 43, wherein if the parent or a descendant exhibits behaviour which might be unsafe or suspicious the attribute is, or is changed to, the process being unsafe.

45. A product or method as claimed in any of Claims 42 to 44, wherein the attribute is security related.

46. A product or method as claimed in any of Claims 42 to 45, wherein the attribute is not security related.

Abstract

Computer Security

A computer network is disclosed with minimised risk of malware infection, comprising at least one local terminal, at least one malware service provider database remote from the terminal, a local database on the local terminal which is allowed to populate and processes are run or downloaded onto the terminal, a server database on the or each server database terminal and means for determining, when a process is found at the local terminal, whether that process is known or unknown at the local terminal and is considered safe or unsafe and, if unknown, whether the process is known at the server database and is considered safe or unsafe, and means for dynamically altering the database contents and safe/unsafe status using predetermined conditions.

OCT 31 2006
PATENT & TRADEMARK OFFICE

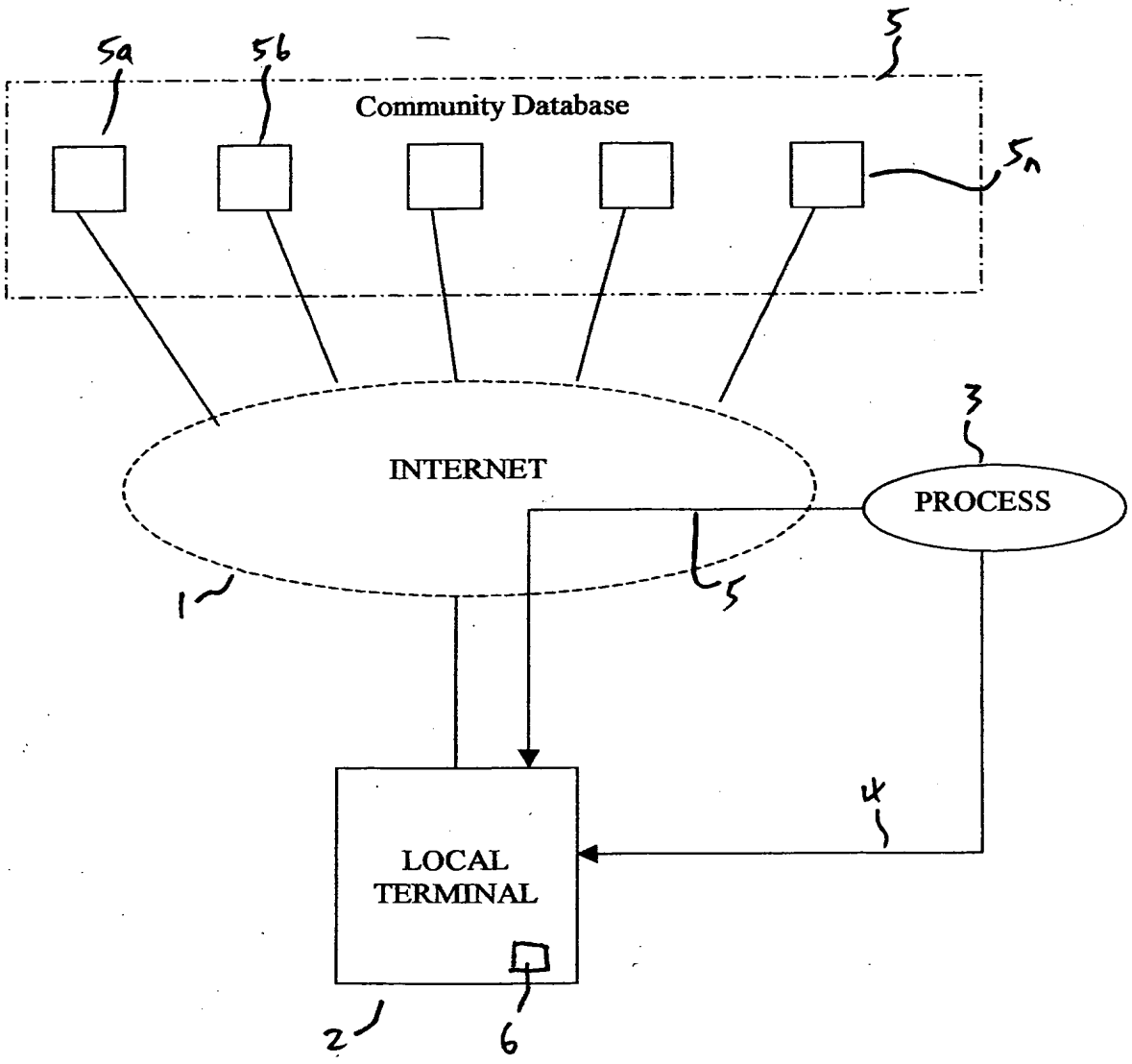


FIG 1

THIS PAGE BLANK (USPTO)

NOT TO BE AMENDED

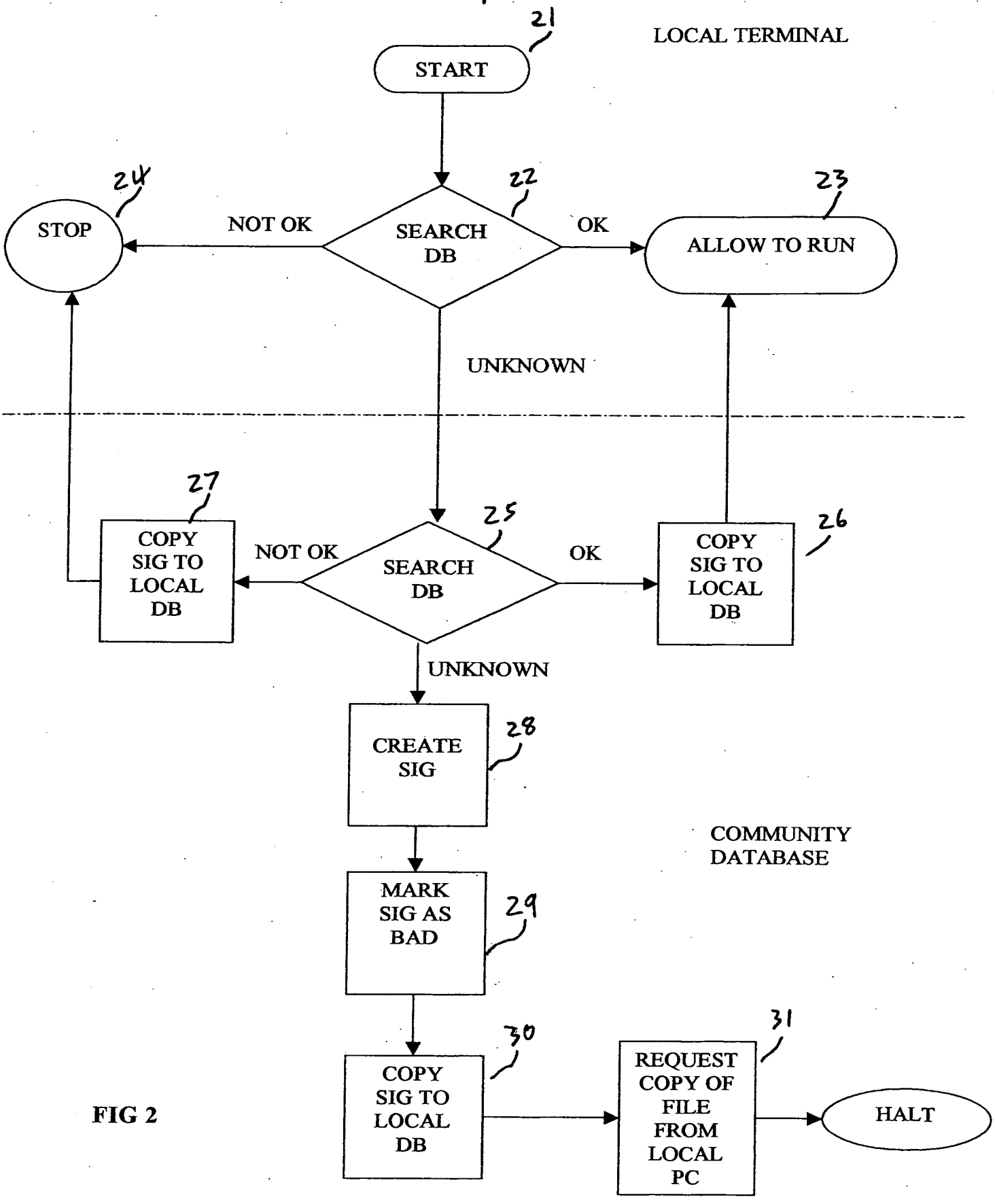


FIG 2

THIS PAGE BLANK (USPTO)

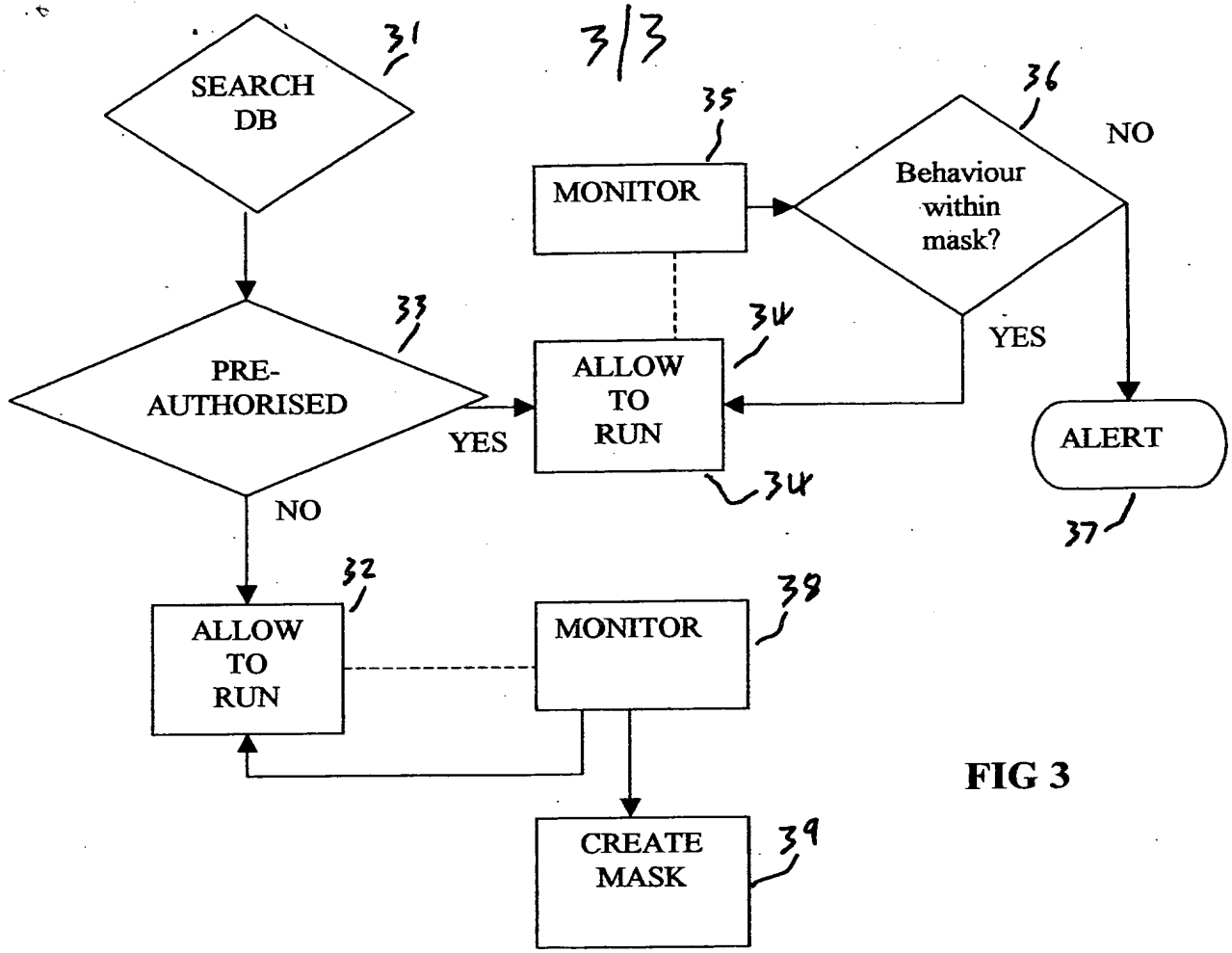

SEARCH DB
31
PRE-AUTHORISED
33
YES
NO
ALLOW TO RUN
32
MONITOR
35
Behaviour within mask?
36
NO
YES
ALLOW TO RUN
34
ALERT
37
MONITOR
38
CREATE MASK
39

FIG 3

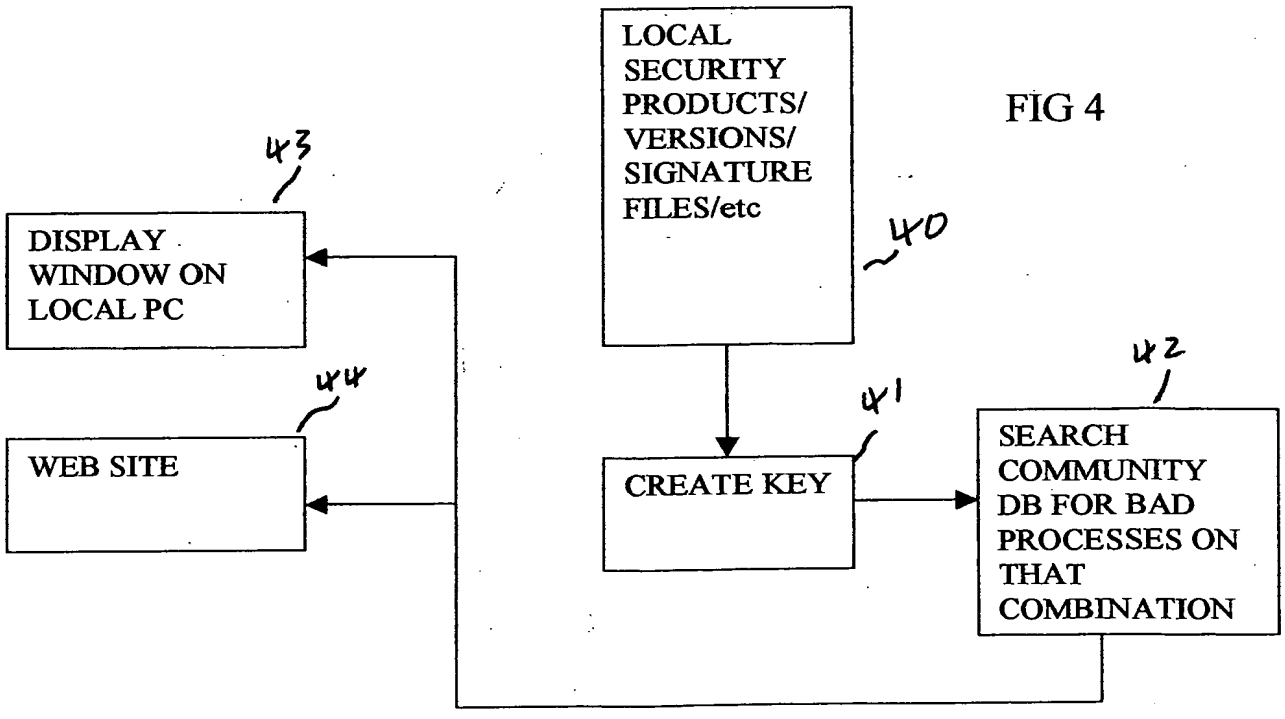

LOCAL SECURITY PRODUCTS/ VERSIONS/ SIGNATURE FILES/etc
40
DISPLAY WINDOW ON LOCAL PC
43
WEB SITE
44
CREATE KEY
41
SEARCH COMMUNITY DB FOR BAD PROCESSES ON THAT COMBINATION
42

FIG 4

THIS PAGE BLANK (USPTO)

PTO/SB/06 (12-04)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number: 11/477807

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))							300
SEARCH FEE (37 CFR 1.16(k), (l), or (m))							500
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))							200
TOTAL CLAIMS (37 CFR 1.16(i))	47 minus 20 =	* 27	x =		OR	x 50 =	1350
INDEPENDENT CLAIMS (37 CFR 1.16(h))	8 minus 3 =	* 5	x =		OR	x 200 =	1000
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).					250	
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))						360	
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	3350

APPLICATION AS AMENDED – PART II

10-31-06

AMENDMENT A	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 47	Minus	** 47	=	x =		OR	x 50 =	
Independent (37 CFR 1.16(h))	* 8	Minus	*** 8	=	x =		OR	x 200 =	
Application Size Fee (37 CFR 1.16(s))							OR	360	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

AMENDMENT B	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)	OR	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
Application Size Fee (37 CFR 1.16(s))							OR		
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2

OIPE TAP 36
JAN 12 2007
PATENT & TRADEMARK OFFICE

Docket Number: 011765-0355986 PATENT APPLICATION
Client Reference: AF/JG/P10717US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re the Application of

MELVYN MORRIS, et al. Group Art Unit: 2131

Application No.: 11/477,807 Examiner: Unassigned

Filed: June 30, 2006 Confirmation No.: 4785

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

Pursuant to 37 CFR 1.56, the attention of the Patent and Trademark Office is hereby directed to the references listed on the attached PTO-1449. Copies of the required references are provided herewith. It is respectfully requested that the information be expressly considered during the prosecution of this application, and that the references be made of record therein and appear among the "References Cited" on any patent to issue therefrom. Applicants respectfully request the Examiner return an initialed copy of the enclosed Form PTO-1449 to Applicants with the next Office communication to indicate that the references have been considered, per MPEP § 609.

This Information Disclosure Statement is being within three months of the mailing date of the International Search Report in the related international application. No fee is required.

Respectfully Submitted,
PILLSBURY WINTHROP SHAW PITTMAN LLP

Christine H. McCarthy
Registration Number 41844
Customer Number: 00909

Date: January 12, 2007
Telephone: (703) 770-7900
Facsimile: (703) 770-7901
P.O. Box 10500
McLean, VA 22102

400512805v1

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 315 066 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.05.2003 Bulletin 2003/22

(51) Int Cl.[7]: **G06F 1/00**, H04L 29/06

(21) Application number: **01309795.1**

(22) Date of filing: **21.11.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(71) Applicant: **BRITISH TELECOMMUNICATIONS public limited company**
London EC1A 7AJ (GB)

(72) Inventor: **The designation of the inventor has not yet been filed**

(74) Representative: **Nash, Roger William et al**
BT Group Legal Services,
Intellectual Property Department,
Holborn Centre, 8th Floor,
120 Holborn
London EC1N 2TE (GB)

(54) Computer security system

(57) A computer security system for use in a network environment comprising at least a plurality of user computers arranged to communicate over a network, the system comprising a warning message exchange system operable to allow the communication from the user computers of warning messages relating to suspect data identified as a possible security threat; a message counting system operable to maintain a count for every particular piece or set of suspect data based on the number of warning messages communicated relating thereto; and network security means operable to act against any particular piece or set of suspect data for which the count maintained therefor exceeds at least one threshold value

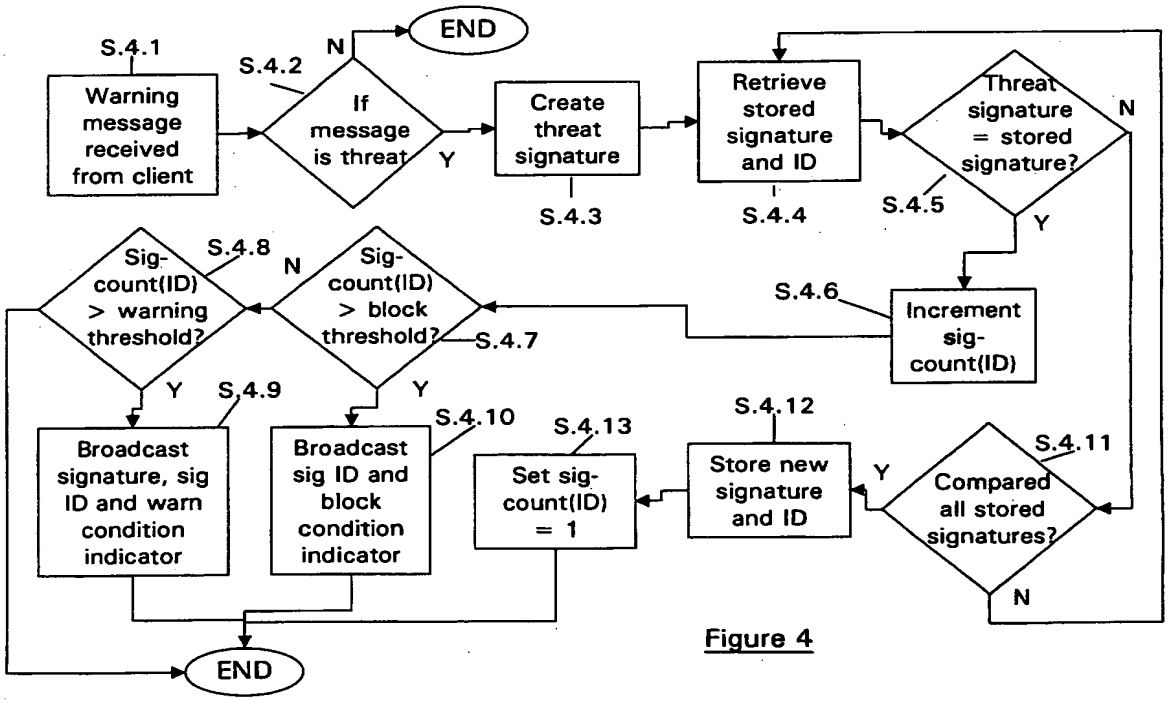


Figure 4

EP 1 315 066 A1

Printed by Jouve, 75001 PARIS (FR)

Description

Technical field

[0001] The present invention relates to a computer security system for use in a networked environment, the system providing a measure of security against suspect data such as computer viruses and the like.

Background to the Invention

[0002] Malicious attacks on computer systems have plagued individuals and organisations for many years. Initially these attacks mostly took the form of viruses in which a rogue program infected a machine via an infected floppy disk or network location for example. More recently so called worms and Trojan horses have caused much disruption in which the user is deceived into running a program via an attachment to an e-mail or a rogue program masquerading as something more innocuous. A large industry has grown up around protection from such attacks. These companies provide anti-virus software that typically resides on individual machines and monitors the system to check for the presence of known viruses. These viruses can take many forms - some more advanced forms are fully polymorphic in that the byte code "signature" is entirely different from one instance of the virus to the next. This is achieved through the use of encryption technology and/or the addition of spurious and random code to the virus. However, the majority of viruses that are in the "wild" and the cause of costly disruption to computer systems are relatively simple and can be detected by simple byte-code signature matching. Many of the current anti-virus programs use just such techniques and are successful if the signature of a discovered virus can be delivered to machines before the virus strikes.

[0003] The operation of such anti-virus programs and systems is well known in the art, and is usually as follows. A computer user will have installed on their computer an anti-virus program which is provided with a database of known computer virus byte-code signatures. The program will tend to run in the background continuously monitoring operations performed on the computer, and data received at and transmitted from the computer. If the byte-code signature of a known virus stored in the anti-virus program's database is detected by the program, the anti-virus program informs the user and takes appropriate action against the data, such as deleting it or storing it in a protected drive.

[0004] Such prior approaches tend to suffer from a big problem, however - delay. Even the simplest of viruses and worms may still cause great disruption at enormous cost to organisations. This is because the process from discovery of a new virus to delivering its signature to all protected machines takes too long, and requires an administrative authority such as the anti-virus program manufacturer (or in some cases an organisation's IT department) to recognise the problem, take action to identify the virus's signature, update the anti-virus database, and distribute the updated database to each user. By the time such a sequence of actions is complete the damage has already been done in many cases. What is required is a much more rapid approach - one that can operate on the same time scales as the spread of the virus and thus provide much more rapid and cost-saving protection.

Prior Art

[0005] There is much work exploring the use of agent technology to provide rapid reaction to malicious attack. In this work computer systems are inhabited by a number of agents whose job it is to detect intrusions. One interesting approach aims to draw metaphor from immune systems, which allow an organism to rapidly respond to previously unseen foreign objects through a "real-time" evolutionary system, as disclosed in Artificial Life IV, Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems, Rodney A. Brooks and Pattie Maes, eds., MIT Press, Cambridge, Massachusetts, 1994, pp. 130-139 These kinds of agent systems may prove useful in future security systems, but rely on the successful development and deployment of software agents which can detect intrusions. Many other potential future anti-virus systems are also discussed at http://www.research.ibm.com/antivirus/SciPapers.htm. In the meantime until the potential of these ideas is exploited, there is still a need for a system which allows for more rapid prevention of the spread of computer viruses.

[0006] One such prior art system which claims to increase response times is that of the Symantec® Digital Immune System (DIS) , which is described at http://securityresponse.symantec.com/avcenter/reference/dis.tech.brief.pdf. The DIS operates by providing an heuristic search engine which automatically identifies possible viruses at either the user's desktop, the server, or the network gateway. A secure channel is then provided direct to Symantec's security response centre over which the suspect data can be transmitted. At the security response centre the received data is subject to a number of automatic filters to detect, for example, known clean files, files which are known to return false positives from the heuristic search engine, and known virus files. If the suspect data is not filtered out by any of these filters then it is directed to an analysis program for further analysis and action.

[0007] The DIS system therefore allows for the automatic identification, secure submission, automatic prioritisation, and subsequent analysis of suspect data, but ultimately still involves a central authority in the loop to analyse suspect files which make it through the filters. Thus, whilst DIS may improve response times to virus infection through it's automatic filtering processes, it still relies on a central authority to analyse the suspect data

and decide on appropriate action, which must then be communicated outwards to each user. There is therefore still a need for a system which removes this centralised analysis step to speed the response.

Summary of the Invention

[0008] In order to address the above problems, the present invention provides a collaborative computer security system wherein the responsibility for detection of malicious data such as a computer virus is removed from that of any central authority, and is instead placed in the hands of each and every user of the network. More particularly, the present invention provides a distributed virus identification system which allows individual users or software agents running on a user's computer to identify malicious data when they receive it, and to pass a warning message concerning the received data to either their peers, or to a central server. A record is kept either at the server or at each peer computer as to the number of warning messages communicated concerning any particular piece or set of suspect data, and then appropriate security actions such as issuing warnings to users or blocking the transmission of the suspect data can be taken once the number of warning messages communicated from users has passed a certain threshold level. The advantages of the invention are that an organisation's response to a computer virus intrusion can be much more rapid than was previously the case where a central authority had to identify the problem and take action.

[0009] In view of the above, the present invention provides a computer security system for use in a network environment comprising at least a first group of user computers arranged to communicate over a network, the system comprising a warning message exchange system operable to allow the communication from the user computers of warning messages relating to suspect data identified as a possible security threat; a message counting system operable to maintain a count for every particular piece or set of suspect data based on the number of warning messages communicated relating thereto; and network security means operable to act against any particular piece or set of suspect data for which the count maintained therefor exceeds at least one threshold value

[0010] The primary advantage of the invention is that it allows the security system to work on the same time scales as the spread of the attack. The more the computer virus or worm spreads the more likely it is that enough positive identifications are made to allow its signature to be ascertained and protection to be installed. This overcomes the problem of traditional security systems where an authority has to learn of the virus, take action, and then distribute associated protective measures to the users.

[0011] A further advantage of the system is that it allows a lot of information to be rapidly collected about a computer virus. For example, current "worms" that are "in the wild" are mostly identical at the byte code level, however it is likely that in the future this will no longer be the case, and that there will be variations in different instances of the same worm. In such a case, the more instances of the worm that can be collected the more likely that it will be that suitable protection can be devised. By allowing for individual identification of a worm at each computer by the user, the present invention allows for multiple instances of the same virus to be identified and collected.

[0012] The one or more "threshold values" against which the respective counts for pieces or sets of suspect data are compared may be any value, and may be fixed or dynamically adaptive. Furthermore the actual values may be generated by any appropriate function, such as for example by a random number generator, or as a function of some metric of accuracy of users in identifying viruses. Furthermore, in other embodiments it is foreseen that the threshold values may not simply be actual numbers which the respective counts must be greater than, but could instead be probabilistic in nature. Furthermore, various logical functions or rules can be used for their creation, including for example, fuzzy logic.

[0013] Preferably, the judgement as to whether any piece or set of data present at any one or more of the user computers is a possible threat is performed by the respective human user or users of the or each user computer. Such an arrangement exploits the massive untapped resources and expertise of many computer users, and again allows for a rapid response to the spread of a computer virus.

[0014] Preferably, the network security means is further operable to compare the maintained count for a particular piece or set of suspect data against a plurality of different thresholds, and to take different action against the data dependent upon which threshold or thresholds are exceeded.

[0015] Such an approach allows the computer security systems to have a graded response to a computer virus intrusion, and to take different actions depending on the number of warnings received from users. Furthermore, it is also possible to weight warnings from different users, such that a warning from a user who is particularly expert in the field is weighted more heavily than a casual user of the system.

[0016] Preferably, the action taken by the network security means comprises warning each of the users as to the suspect nature of a particular piece or set of data. Furthermore, the action may also comprise preventing the transmission of the particular piece or set of suspect data across the network. Usually, the warning action will be taken when the number of messages communicated exceeds a lower threshold than a second threshold which must be crossed in order for the transmission blocking action to take place.

[0017] Preferably, the or each warning message com-

prises at least an identifier of the piece or set of suspect data to which it relates. The identifier may be the actual piece or set of suspect data itself, or a repeatably generatable signature substantially unique to the piece or set of data. By allowing for the collection of such identifiers, the security system also provides a means of collecting information on multiple computer viruses, and also on adaptive computer viruses whose byte code signature may change in each instantiation on a users machine.

[0018] In a preferred embodiment, the computer security system further comprises a network server arranged to receive the or each warning message communicated from the user computers, and further arranged to host the message counting system and at least part of the network security system, wherein that part of the network security system hosted by the server is operable to determine the action which should be taken against a particular piece or set of suspect data, and to communicate an action condition indicator indicative of the determined action to each of the plurality of users.

[0019] By using a central server to maintain the warning message counts and to receive the warning messages themselves, the network traffic generated by the computer security system is minimised, as users need only send a warning message direct to the server, which can then broadcast warnings about suspect data as appropriate to all the users.

[0020] In another embodiment, the warning messages generated by a user computer are broadcast to every other user computer, and each user computer is further arranged to host its own message counting system and network security system operable in accordance with any of the above. Such an approach represents a "peer-to-peer" system, and provides advantages that it is not dependent upon a central server for its operation, thereby presenting a more robust distributed system. Such a peer-to-peer system has a potential disadvantage, however, in that depending on the exact implementation it may be that the network traffic required to broadcast the warning messages from each user to every other user is unacceptably high.

[0021] To at least partially overcome this problem, in another embodiment there is provided an inter-group communications system operable to allow the communication of warning messages relating to suspect data identified as a possible security threat from the first group of user computers to one or more other groups of other user computers. By splitting the user computers intro groups, the number of warning messages which are required to be transmit is substantially reduced, thereby conserving network signalling bandwidth.

[0022] Preferably, the inter sub-set warning messages are transmitted by the inter-sub-set communications system if the number of warning messages communicated by the user computers in the sub-set concerning a particular piece or set of data exceeds at least one threshold value. Preferably the threshold value or values is/are the same as those at which the network security system takes action.

[0023] In further embodiments, the message counting system is further arranged to store one or more weighting coefficients relating to one or more particular user computers, and to increment the count maintained for a particular piece or set of suspect data by an amount based upon the weighting coefficient when a warning is received from the one or more particular user computers relating to the particular piece or set of suspect data

[0024] From a second aspect, the present invention also provides a computer readable storage medium storing one or more computer programs which when run on a computer cause the computer to operate in the manner of the computer security system as described above. Such a computer security system may provide all the additional features and advantages mentioned previously.

Brief Description of the Drawings

[0025] An overview of the operation of the invention, together with a description of a number of embodiments thereof, presented by way of example only, will now be made with reference to the accompanying drawings, wherein like reference numerals refer to like parts, and wherein:-

Figure 1 illustrates a typical network environment in which the computer security system of the present invention is intended to operate;

Figure 2 illustrates a computer storage medium located within each users computer, and depicts the programs and data stored thereon in a first embodiment of the invention;

Figure 3 depicts a computer storage medium located at a server used within the first embodiment of the inventions, and further depicts the programs and data stored thereon;

Figure 4 is a flow diagram showing the operation steps performed by the warning creation program stored at the server in the first embodiment of the present invention;

Figure 5 is a flow diagram showing the method steps performed by the warning receipt program stored at a users computer in the first embodiment;

Figure 6 is a flow diagram showing the steps performed by a filter program in either of the embodiments of the invention;

Figure 7 depicts a computer readable storage medium located at a users computer in a second embodiment of the present invention, and further depicts the programs and data stored thereon;

Figure 8 is a flow diagram of the steps performed by a warning creation program in the second embodiment of the present invention;

Figure 9 is a flow diagram of the steps performed by a warning receipt program in the second embod-

iment of the present invention;
Figure 10 is a flow diagram illustrating the steps performed by one of the functions in both Figures 8 and 9; and
Figure 11 illustrates an organisational architecture of user computers which may be used in another embodiment of the invention.

Overview of the operation of the invention

[0026] The present invention recognises and makes use of the fact that a computer system is already inhabited by a number of intelligent agents - the users themselves. In order to summarise the operation of the invention consider an attack by an e-mail "worm" (a type of computer virus such as the "Melissa" virus which was encountered in 2001). A worm is sent to an employee of a given organisation. With some probability the employee opens the e-mail attachment and unwittingly unleashes the worm that causes some damage to the individuals machine and propagates itself to a number of other machines through use of the individuals e-mail system and address book. In each of these cases there is some probability that the user will open the attachment and further propagate the virus. Thus, the statistics are in the worm's favour - not everyone will open the attachment but it only takes a small fraction to do so for the worm to propagate throughout an organisation's computer system. However, for every user who opens the attachment there are likely to be many more who correctly recognise the e-mail as a potential threat and discard it. Vital information is thus gained in this process - an intelligent agent has successfully detected an intrusion - and it is this information that the present invention exploits to its full effect. Doing so opens up the potential for the statistics to be turned against the worm and in favour of the protection system.

[0027] Within the invention the user informs the system of the recognition by forwarding the mail or passing the mail or other program to an anti-virus program that is running on that machine, which then passes it to the server (or in an alternative embodiment direct to the user's peers). The signature of the worm is then quickly ascertained and a count kept of the number of warnings received from users about the worm. Once the count reaches a certain threshold the signature of the worm is distributed to all the other individual machines, who then use the signature to filter any received data or mail. As an alternative the server itself could simply filter out all e-mails with such a signature. In either case the technique of allowing users (or user machines provided with appropriate software agents) to detect and provide warning of the worm aligns well with the trends in the industry towards peer to peer systems where the vast untapped resources at the edges of the network are beginning to be used to greatly enhance storage and computational power for example. The invention is therefore essentially a peer to peer security system that may tap the vast unused resource of the user's knowledge and experience, or the user computer's processing power.

Description of the embodiments

[0028] Two specific embodiments of the invention will now be described in detail, followed by description of various alterations that may be made to each to provide additional embodiments. Both main embodiments of the invention implement a computer security system for use in a network environment. A typical network environment in which the invention may be used is depicted in Figure 1.

[0029] Figure 1 illustrates a typical local area network (LAN) configuration. Here, a plurality of user computers 15 are connected to a network 10 which is another control of a network server 12. The network 10 provides typical network functionality, such as the ability to allow users 15 to communicate with each other and with the server over the network. The network 10 in the invention may be any particular network technology, may be wired or wireless, and may be scaled to any network size or architecture known in the art. Therefore, whilst the present invention is particularly useful for use within individual organisations provided with a LAN, it may also be used on a wider scale in a metropolitan area network (MAN) or a wide area network (WAN). Within the embodiments, each of the user computers 15, the network itself 10, and the server 12 are provided with appropriate equipment and software to permit the usual network communication functions as are known in the art.

[0030] Although the above makes reference to the user computers 15 being part of the same network, it should be noted that it is not essential that each user-computer be located in the same actual network (although this may be preferable). This is because all that the invention requires is the ability for any particular user computer to be able to send messages either to its peers or to a server, and hence all that is required is some message transport mechanism between such parties. Such a message transport mechanism may be a single network to which all the user computers and servers (where required) are connected, or may be a sequence of interconnected networks such as the Internet, with different user-computers actually being part of different networks.

[0031] Having described the context in which the invention is intended to operate, a first, preferred, embodiment which represents the best mode of the invention will now be described with reference to Figures 2 to 6.

[0032] The first embodiment of the invention provides a computer security system wherein identification of suspect data is performed by the users or suitable software agents installed and running on the user's computers at the user's computers themselves, and upon identification of a suspect piece or set of data a warning message is transmitted from the user's computer to the network server 12. At the server the number of warning

messages received about a particular piece or set of data is counted, and once the count passes a first warning threshold, a warning message is broadcast to all users, the message containing a signature of the suspect data, such that an anti-virus program located at each user's computer can filter incoming data for the suspect data. If further warnings are received from users by the server, and the count of warning messages exceeds a second threshold, then the server broadcasts a second message to all of the users on the network instructing the anti-virus programs located at each user computer to block the suspect data to prevent any onward transmission or infection.

[0033] Given the above overview, within the first embodiment each user computer 15 contains a computer readable storage medium 22, as shown in Figure 2. The computer readable storage medium will typically be a magnetic hard drive, or may equally be a DVD RAM, DVD-RW, DVD+RW, solid state memory, optical disc, magneto optical disc, or the like. The computer readable storage medium 22 at each user computer 15 is arranged to store a data filter program 24, and a warning receipt program 26. In addition, storage space is also provided for suspect data signatures and ID's, stored in the form of a data base 28.

[0034] The server 12 within the first embodiment also contains a computer readable storage medium 32 which will typically be a magnetic hard disc, but may also be any of the storage devices mentioned previously in respect of the or each user computer 15. Within the first embodiment the computer readable storage medium 32 stores a warning creation program 34, and a suspect data signature and ID database 36. Optionally, the computer readable storage medium 32 may also store a data filter program 38, to provide an alternative method of operation as will be described later.

[0035] Having described the main system hardware elements required by the first embodiment, the operation thereof will now be discussed with reference to the flow diagrams of Figures 4, 5, and 6.

[0036] Assume that a user computer 15 connected to the network 10 receives a message over the network which contains a computer virus or the like (referred to hereafter as suspect data). Within the first embodiment of the invention a user of the user computer 15 upon reading the message recognises that the message contains such suspect data, and hence within the first embodiment then forwards the suspect data as the payload in a warning message of predetermined format to the server 12. By forwarding the data as the payload encapsulated in a warning message of known format, the server knows that the payload is potentially suspect data, and hence will not become victim to the computer virus itself. At the server, upon receipt of such a warning message from one of the user computers 15, the warning creation program 34 is run, and the steps performed by this program are shown in Figure 4.

[0037] Firstly, at step 4.1 the server receives a warning message from one of its clients, being one of the user computers 15 (as described above). Then, at step 4.2 the server parses the warning message to retrieve the suspect data, and checks the message to see if it could be a computer virus. Such checking preferably takes the form of determining the data type to see if it could feasibly be a virus. Viruses are frequently executable files, such as .exe or .bat on a PC, or macros such as are found in Microsoft® Word® or Excel®. Such a check can conveniently be performed by looking at any file extension on the data. If such checking does not indicate that the data could be a virus then processing ends, as it is assumed that the user has simply made a mistake. Instead, if the warning message is in fact a true warning and the suspect data could possibly be a virus, then processing proceeds to step 4.3.

[0038] Note here that the checking step of step 4.2 is performed in the preferred embodiment, but in other embodiments may not be performed, and as such is optional. Where step 4.2 is not performed processing proceeds directly from step 4.1 to 4.3.

[0039] At step 4.3 the server reads the suspect data from the payload, and creates a "signature" for the suspect data, to permit the particular piece or set of suspect data to be identified in the future. Such a signature may be conveniently created by running the suspect data through a hash function. Hash functions are well known in the art, and commonly provide a 16 or 20 bit output word unique to any particular piece or set of data input thereto. The key feature of hash functions which may be of use in the present invention, however, is that the same output word is always output from the hash function for the same piece or set of input data. Thus the hash value for a certain piece or set of suspect data will always be the same, no matter where the suspect data has been received from.

[0040] It should be noted that other forms of signature creation other than the use of hash functions may also be of use, the only requirement being that an identifiable unique signature is generated for any particular piece or set of suspect data input into the signature creation function at any time.

[0041] Following the creation of the data signature, at step 4.4 the signature and ID database 36 is accessed to retrieve the first previously stored signature and accompanying signature ID therefrom. The signature and ID database 36 in the server 12 is used to store the generated signature ID's from pieces of suspect data received in previous warning messages. At step 4.4 the first signature and ID in the data base is retrieved the first time the controlling function for step 4.4 is called, and thereafter at succeeding calls the next signature and ID in the database list are successfully retrieved.

[0042] At step 4.5, the threat signature of the received suspect data generated at step 4.3 is compared with the stored signature retrieved from the database at step 4.4. If the threat signature is not the same as the retrieved stored signature then processing proceeds to step 4.11,

wherein a check of the signature and ID database 36 is made to see if all stored signatures have been compared with the new signature. If not then processing proceeds to step 4.4, which retrieves the next signature and ID from the database 36 for evaluation at step 4.5 with the generated data signature for the presently received.

[0043] If at step 4.11 it is determined that all of the stored signatures have been compared with the newly created threat signature then it must be the case that the warning message received from the user computer at step 4.1 is the first warning message from any of the user computers to bring to the server's attention the particular piece or set of suspect data contained in the warning message payload. In this case it is necessary for the server to store information on the suspect data, and this is achieved by storing the suspect data signature created at step 4.3 together with a new unique signature ID within the signature and ID database 76. This storing function is accomplished at step 4.12, and thereafter processing proceeds to step 4.13, wherein a count of warning messages received for the particular piece of suspect data is established, and initiated to a value of one. It will be appreciated that a separate count is maintained for every different piece or set of suspect data received in a warning message, and hence at any one time the server will be maintained in several different counts. Therefore, conveniently the various counts may be maintained in a one dimensional array structure, indexed by the particular suspect data's signature ID. Such storage of the count is shown in step 4.13.

[0044] Following the storage of the data signature, signature ID, and establishment of the message count for the suspect data, then in the case that the suspect data had never been received before processing then ends.

[0045] Returning to a consideration of step 4.5, if it turns out that the particular piece or set of suspect data included in the warning message has been received before in a previous warning message then at some point in the loop formed by steps 4.4, 4.5, and 4.11, the evaluation performed at step 4.5 will return a positive value, in that the threat signature generated at step 4.3 will match a previously stored signature. In this case processing then proceeds to step 4.6, wherein the ID of the stored signature is used at an index into the signature count array, and the particular count for the stored signature is incremented by one to reflect the receipt of an additional warning message relating to the particular piece or set of suspect data to which the signature relates. Following the incrementing of the signature count, processing proceeds to step 4.7.

[0046] Step 4.7 represents the first thresholding evaluation performed to determine whether or not a sufficient number of warnings have been received from the user computers concerning a particular piece or set of data such that action should then be taken against that data. Therefore, at step 4.7 an evaluation is performed to determine whether the signature count for the particular piece or set of suspect data is greater than a "block" threshold value. If so processing proceeds to step 4.10, wherein the server then broadcasts a message to all of the user computers 15 on the network, the message containing the particular data signature ID, and a message condition indicator field which in this case contains data instructing the user computers to block the suspect data if it is received thereat. Blocking of the suspect data is performed by the data filter program 24 provided at each user computer 15, the operation of which is described later.

[0047] It should be noted that at step 4.10 the message from the server to the clients contained only the signature ID and the condition indicator and does not contain the actual signature itself. The reason for this is that it is assumed that prior to any particular piece or set of suspect data becoming blocked a warning message will have been transmitted from the server to the user computers containing the suspect data signature, and hence there is no need to retransmit the signature at step 4.10.

[0048] If the evaluation at step 4.7 returns a negative, processing proceeds to the evaluation at step 4.8 wherein the signature count for the particular piece or set of suspect data is compared against a warning threshold level. It should be noted here that the warning threshold level is set less than the block threshold level such that the evaluation of step 4.8 is always found positive before the evaluation of step 4.7. If the evaluation of step 4.8 returns a negative such that the count is less than the warning threshold then this means that not enough warning messages have been received from user computers concerning the particular piece or set of suspect data such that no action should be taken as yet. In this case processing of the warning creation program 34 then ends.

[0049] In contrast, if the evaluation at step 4.8 is positive, the processing proceeds to step 4.9 wherein the server broadcasts a message over the network to all the user computers 15, the message containing the suspect data's signature as generated at step 4.3, a new data signature ID, being simply an identifier of the signature which can be used as a short hand means of identifying the particular piece or set of suspect data, as well as a "warn" flag in a data condition field of the message. The presence of the warn flag in the data condition field of the message causes the data filter program 24 at the user computer to warn the user whenever the suspect piece of data is received, as will be described later. Following the broadcast of the warning message at step 4.9, processing by the warning creation program 34 ends.

[0050] The output of the warning creation program 34 is therefore a warning message broadcast to all the user computers 15 instructing the user computers to upgrade their threat levels for a particular piece or set of suspect data provided that enough warning messages have been received at the server concerning the particular

suspect data. At each user computer upon receipt of a warning message from the server the warning receipt program 26 is run, and the steps performed thereby are shown in Figure 5.

[0051] With reference to Figure 5, at step 5.1 a warning message containing the information described earlier is received from the server. Following this, at step 5.2 the user computer 15 passes the warning message to determine whether or not the message contains a data signature.

[0052] If the message does not contain a data signature, then this means that a warning message must have been received at some point in the past from the server containing the data signature, and therefore the data signature will be already stored in the ID database 28 at the user computer 15. In this case processing proceeds to step 5.4, wherein the signature ID contained within the received warning message is used to index and retrieve the appropriate signature from the database 28, whereupon processing then proceeds to step 5.6.

[0053] At step 5.6 the data condition indicator which is contained in the warning message received from the receiver is read, and a condition field in the database 28 related to the particular suspect data signature is updated to match the condition indicated in the warning message from the server. In this case, as a warning message will have already been received concerning the particular suspect data and hence a "warn" condition already set, it is likely that the condition is being updated to a "block" condition. Following step 5.6 the processing by the warning receipt program ends.

[0054] If at step 5.2 it is determined that the warning message from the server does contain a data signature, then this will be the first warning message received from the server concerning the particular suspect data. In this case, processing proceeds to step 5.3 wherein the received signature is stored in the database 28 indexed by the ID which is also contained in the received message. Furthermore, a condition field in the database related to the signature is also set, at step 5.4. In this case, as this will be the first warning message received from the server about the particular suspect data the condition will be set to a "warn" condition. Following step 5.4 the processing by the warning receipt program 26 ends.

[0055] It has thus been described how users can generate warnings about suspect data which are then transmitted to the server, which then broadcasts warnings to all users if enough user's specific warnings have been received. Users can then record the particular security advice broadcast from the server concerning a piece or set of suspect data, and we now describe how user computers 15 may use the broadcast warnings within the data filter program 24.

[0056] More particular, as mentioned previously each user computer 15 is provided with a data filter program 24 which is arranged to run in the background of the computer all the time the computer is operating in a similar manner to anti-virus and other security programs of the prior art. The data filter program 24 acts to monitor all data received at the user computer 15 upon which it is stored to determine if any of the received data matches the signatures of any suspect data stored in the signature and ID database 28. The steps performed by the data filter program 24 in performing this function are shown in Figure 6.

[0057] At step 6.1 assume that the user computer 15 has received some data in the form of an email message or the like over the network 10. At step 6.2 the data filter program 24 acts to create a signature of the received data using the same signature generation function as is used in the server 12. That is, the signature of the received data may be created using a hash function or indeed any other signature generating function which provides a repeatably generatable unique signature for a given set of input data.

[0058] Following the creation of the data signature, at step 6.3 the signature and ID database 28 at the user computer 15 is accessed and the first signature and signature ID within the database is retrieved therefrom. Upon subsequent calls to the controlling function of step 6.3 the next signature and signature ID in the database list is retrieved successively.

[0059] Following step 6.3 processing proceeds to the evaluation of step 6.4, wherein the data signature created at step 6.2 is compared with the retrieved signature from the database 28. If these signatures do not match then processing proceeds to the evaluation of step 6.10, wherein the database is again accessed to see if all signatures have been accessed and compared. If at step 6.10 it turns out that all of the signatures stored in the database have been compared against the signature of the received data and no match has been found, then it can be concluded that the received data is not suspect data, and hence the data filter program ends. In such a case the user is then free to use the received data in whatever manner she chooses.

[0060] In contrast, if at step 6.10 it is determined that not all of the stored signatures have been compared against the received data signature, processing returns to step 6.3 wherein the next stored signature and ID in the database are retrieved, and the evaluation of 6.4 is performed once again.

[0061] If at step 6.4 it is determined that the received data signature is the equivalent of a previously stored signature then processing proceeds to step 6.5. Here, the database is again accessed to retrieve the signature condition associated with the stored signature to determine what security condition was attached to the signature by the server in the last warning message received from the server concerning the particular signature. Following step 6.5 the retrieved condition is checked at step 6.6 to see if it is a "warn" condition, and if so processing proceeds to step 6.7, wherein a warning is displayed on the user computer's screen to warn the user about the received data. Following the display of the warning the particular process performed by the data filter program

24 ends, although it should be noted that the data filter program will continue to run continuously in the background to check whenever data is received.

[0062] If the retrieved signatory condition is not a "warn" condition then processing proceeds to the evaluation of step 6.8 wherein the condition is checked to see if it is a "block" condition. If so then at step 6.9 the data filtering program 24 acts to block the receipt of the suspect data either by preventing the computer from downloading it from the server or from merely deleting it from the user computer's local storage, and again the user is preferably informed of this action. Following this action the processing performed by the data filter program 24 ends, although again the program will continue to run in the background to detect the receipt of new data.

[0063] If the evaluation of step 6.8 does not indicate that the retrieved signature is a "block condition", then the data filter program 24 processing ends, and the program returns to running in the background, as described previously.

[0064] The first embodiment of the invention therefore presents a computer security system whereby computer viruses and the like can be detected by individual users, who transmit warnings to a server which then broadcasts warnings as appropriate to all users if the number of individual warnings received from individual users exceeds certain thresholds. The use of thresholding in the server instils in the invention a degree of order, in that it is only once a particular threshold level of warnings have been received that action is taken by the server and user computers against the suspect data. This prevents, for example, inexperienced users who may mistake acceptable data for suspect data from initiating a security alert on the acceptable data which may prevent other users from receiving that data, for example.

[0065] Furthermore, the threshold levels for both the "warn" and "block" conditions may of course be set at any level required, and may even be simply one. The assumption is made, however, that in order for the embodiment to operate in a meaningful manner the warn threshold is less than or equal to the block threshold.

[0066] Within the preferred embodiment we have described the data filter program 24 as being located at each of the user computers 15 and running thereat. However, in a slight variation of the first embodiment it would of course be possible for the data filter program 24 to also run at the server to screen messages and data routed therethrough. In this case the operation of the data filter program would be exactly the same as previously described with respect to Figure 6, however it would mean that the user computers 15 would neither need the data filter program 24, the warning receipt program 26, or the signature and ID database 28, as all blocking operations could be performed centrally. However, in the case that it is desirable for users to be warned about data to be received, then it will still be necessary to have the data filter program 24, warning receipt program 26, and signature and ID database 28 at the user computer 15 as described previously in order to allow such warnings.

[0067] In another variation of the first embodiment, in another embodiment a step equivalent to step 4.2 can be performed at the user computers before a warning message is generated. Here, after a user has identified a piece of data as suspect and instructed her computer to send a warning message to the server, prior to compiling and sending the message the warning creation program 34 controls the computer to check if the data identified as suspect by the user could possibly be a virus. Such checking is substantially identical to the check performed at step 4.2, i.e. takes the form of determining the data type to see if it could feasibly be a virus. Viruses are frequently executable files, such as .exe or .bat on a PC, or macros such as are found in Microsoft® Word® or Excel®. Such a check can conveniently be performed by looking at any file extension on the data. If such checking does not indicate that the data could be a virus then no warning message is sent, as it is assumed that the user has simply made a mistake. Instead, if the data is of a data type which could be a virus (e.g. an executable or the like) then the warning message is transmitted.

[0068] The primary advantage of the first embodiment which makes use of the server to keep track of the number of warning messages received about a particular piece of suspect data is that the network traffic overhead introduced by the security system is minimised, as users need only send warning messages to the server, which can then broadcast warning condition messages as appropriate.

[0069] A second embodiment of the invention will now be described with reference to Figures 7 to 10.

[0070] The second embodiment of the invention presents a pure "peer to peer" system which does without the server 12 of the first embodiment. Instead, within the second embodiment each user computer 15 itself individually keeps count of the number of warning messages received from its peers concerning suspect data, and in turn when a user computer detects suspect data it generates and broadcasts a warning message to all of its peers. This arrangement has the advantage that it is not dependent upon a central server for operation, and is therefore much more robust and immune to the loss of one or more of its elements than the client-server architecture of the first embodiment.

[0071] Figure 7 illustrates the software elements present at a user computer 15 in the second embodiment of the invention. More particularly, as in the first embodiment the user computer 15 is provided with a computer readable storage medium 22 on which is stored a data filter program 24, a warning creation program 72, a warning receipt program 74, and a signature and ID database 76.

[0072] Within the second embodiment the data filter program 24 and the warning creation program 72 both

continuously run in the background on the users computer at all times. The data filter program 24 operates in exactly the same manner as previously described in the first embodiment with respect to Figure 6, that is it acts to check the signature of received data against signatures of suspect data stored in the signature database 76, and takes action against any received suspect data as necessary. In this respect, the operation of the second embodiment is exactly the same as that of the first embodiment. The difference between the second embodiment and the first embodiment lies, however, in the fact that each user computer 15 directly broadcasts warnings to each of its peer computers (i.e. the other user computers 15), and in addition each user computer 15 also keeps its own track of the count of warning messages received against each piece of suspect data. These functions are performed by the warning creation program 72 and the warning receipt program 74, as described next.

[0073] The operation of the warning creation program 72 is shown in Figure 8. Here, at step 8.1 data is received at the user computer 15 over the network 10. At step 8.2 an evaluation is performed to determine whether the received data is a security threat or not. As with the first embodiment, this evaluation is preferably performed by the actual human user of the computer 15 who employs her expertise to determine whether or not the received data is a computer virus or the like. In alternative embodiments, however, it may be that this evaluation could be performed by a software agent installed on the users computer, the software agent acting to monitor the functions of the computer to determine if it is under attack from a computer virus. Within the first and second embodiments, however, we prefer the use of a human operator to perform the evaluation.

[0074] If the evaluation at step 8.2 concludes that the received data is not a threat then the processing performed by the warning creation program on the received data ends and the program returns to running in the background until further data is received in the future.

[0075] If it is determined at step 8.2 that the received data is a threat, then at step 8.3 a data signature for the received data is created. As in the first embodiment, the signature generation function may be a hash function or the like or any other suitable signature generation function which is capable of repeatably generating the same signature for a given set of input data. Furthermore, it should be noted here that each user computer 15 must use the same signature generation function, such that when the same set or piece of suspect data arrives at any of the user computers, each of them creates the same signature therefor.

[0076] At step 8.4 a check is made to see whether the generated data signature has already been stored in the signature and ID database 76, and if not then at step 8.8 the signature is stored in the database, and is allocated a globally unique ID. The ID for the signature is generated by the user computer, preferably from a list of ID's which have been specifically allocated to that user computer for allocation to signatures thereby.

[0077] Following step 8.8, at step 8.9 a message count is instantiated for the particular signature generated by the received data, and is set to the value one. As in the first embodiment, the various count values for each piece of suspect data may be stored in a one dimensional array indexed by signature ID.

[0078] Following step 8.9 at step 8.10 the warning creation program causes the user computer to broadcast the data signature and the signature ID to all of its peers as a warning message. As in the first embodiment, the warning message preferably takes a known format, with the signature and signature ID as particular fields within the warning message. Following the broadcast of the warning message, at step 8.10 processing of the warning creation program ends, and it returns to run in the background.

[0079] Returning to the evaluation at step 8.4, if it is determined that the data signature for the received data has already been stored (i.e. the data has already been identified as suspect by either the present user, or one of the other users who have then broadcast a warning message which has been received at each user computer) then processing proceeds to step 8.5, wherein the message count maintained for the particular signature is incremented to reflect the fact that the data has in fact been identified once again as a threat.

[0080] Next, at step 8.6 the signature ID is broadcast to every other user in the system as part of a warning message, and then following that at step 8.7 the appropriate warning or block conditions are set for the signature based on the warning and block threshold values. The specific operations required by step 8.7 are shown in more detail in Figure 10.

[0081] With reference to Figure 10, the steps required for setting the warning or block conditions for a particular data signature are as follows. At step 10.1 the message count for the particular data signature is retrieved, by indexing into the counter array using the signature ID. Then, at step 10.2 the retrieved message count is compared against the block threshold, and if larger than the threshold at step 10.3 the security condition for the data signature is set to "block". As in the first embodiment, this security condition is stored in the signature and ID database 76, and is used by the data filter program 24 to filter received data.

[0082] If the warning message count for the particular data signature is not greater than the block threshold then the message count is compared against the warning threshold. If it is not greater than the warning threshold then no warning or block condition is set for the particular data signature. In contrast, if it is greater than the warning threshold then the security condition for the particular data signature is set to "warn", which is again used by the filter data program 24 as described before in respect of the first embodiment.

[0083] Returning to Figure 8, once the warning or

block conditions have been set for the data signature, the specific processing performed by the warning creation program 72 ends, and it returns to running in the background until further data is received.

[0084] Turning now to the operation of the warning receipt program 74, this program operates whenever a warning message generated by a warning creation program stored on another user's computer is received. The operation of the warning receipt program 74 is shown in Figure 9.

[0085] At step 9.1, a warning message is received from one of the other user computers. The warning message is of course in a predetermined format which identifies it as such. Then, at step 9.2 the warning receipt program 74 acts to parse the message to see if it contains a data signature relating to suspect data. Recall here that the data will have been determined as suspect by the user of the user computer from which the message was received. If it is determined that the message does not contain a signature, then this means that the data will have already been determined as suspect previously, and hence should already be stored in the user computer's signature and ID database 76. Therefore, at step 9.3 the warning receipt program acts to retrieve the message count for the particular suspect data to which the warning message relates, using the signature ID which will have been included in the warning message (recall here that at step 8.6 in the warning creation program where the suspect data has already been identified as such a warning message is still sent out including the suspect data signature ID). Then, at step 9.4 the retrieved message count for the suspect data is incremented to reflect the additional warning, and at step 9.5 the warning or block conditions for the data signature are also set, following the procedure of Figure 10, as described previously.

[0086] If it is determined at step 9.2 that the message does contain a signature, then processing proceeds to the evaluation of step 9.6 wherein it is determined whether or not the received signature has already been stored. If not processing proceeds to step 9.7 wherein the signature is stored in the signature and ID database 76 indexed by the signature ID included in the received warning message. At step 9.8 a message count is instantiated for the newly stored data signature, and is set to one. As described previously, the count values for each particular signature can be stored in a one dimensional array, indexed by each signature ID.

[0087] Following step 9.8 processing proceeds to step 9.5, wherein the warning or block conditions are set for the particular signature, using the procedure of Figure 10.

[0088] Returning to step 9.6, if it is determined that the signature received in the warning message has already been stored then processing proceeds to step 9.9, wherein the received signature ID is stored for use as another key to the signature. In such a case a count value should be instantiated related to the received signature ID, with its value set to the same value as the count indexed by the ID of the already stored signature which matches the received signature. Then, at step 9.10 both of the count values stored for each ID (the received ID, and the ID of the previously stored ID which matches the received signature) are incremented to reflect the receipt of the additional warning. Finally, following the increment of the message counts for the signature the warning and block conditions are set for the signature once more, using the procedure of Figure 10.

[0089] Therefore, as described the second embodiment provides a pure peer to peer system which incorporates the distributed warning capabilities of the security system of the present invention with the ability to threshold the warnings so as to be able to provide a graded reaction locally at each user computer. The provision of such a peer-to-peer operation provides for a more robust system, but as mentioned earlier might have detrimental network overheads due to the need to broadcast every warning message to every user.

[0090] As a modification to the operation of the second embodiment, an additional checking step analogous to step 4.2 of the first embodiment may be performed between steps 8.2 and 8.3 of Figure 8 depicting the operation of the warning creation program 72. Here, after the human user has identified data as suspect at step 8.2, an additional check is made to see if the data could possibly be a computer virus. Such checking preferably takes the form of determining the data type to see if it could feasibly be a virus. Viruses are frequently executable files, such as .exe or .bat on a PC, or macros such as are found in Microsoft® Word® or Excel®. Such a check can conveniently be performed by looking at any file extension on the data. If such checking does not indicate that the data could be a virus then processing ends, as it is assumed that the user has simply made a mistake. Instead, if the warning message is in fact a true warning and the suspect data could possibly be a virus, then processing proceeds to step 8.3, and onwards as previously described. The operation of the other programs in the second embodiment remains unchanged.

[0091] It should be noted here that as the second embodiment does not require a server to act centrally, but rather is a true peer-to-peer system, it is particularly suitable for implementation on networks such as ETHERNET and the like.

[0092] Various further modifications may be made to either of the first or second embodiments to produce further embodiments, as described next.

[0093] In further embodiments it is possible to add more sophistication to increase the power of the system. For example, users could be given the ability to respond to a warning to inform the server of whether they agree with the assessment or not - this would give the potential of reducing the count number if appropriate. The decision as to whether a particular program was a threat or not would then be a collective decision from the whole community.

[0094] Furthermore it is also possible to build in trust models for each user such that some users have greater influence than others. Such trust models could be predefined according to the users position within the community/organisation, such that a warning received from a user with a high trust rating increments the signature count for a particular piece or set of suspect data more than another user with a lower trust rating. Each trust model may then also be adaptively varied by judging the accuracy of a prediction that a user has made against a final collective decision, and any further warnings registered by that user could be weighted according to the trust in that users demonstrated decision-making ability.

[0095] In other embodiments a community of organisations could be formed that would extend the collaboration beyond the limits of a single organisation through sharing of knowledge. Any discoveries in one organisation could be passed on to all other organisations in the community and vice versa. This would greatly extend the power of the system and in many cases allow rapid response to intrusions experienced in other organisations without any internal hits being experienced to the remaining organisations using the system.

[0096] Moreover, in another embodiment the actual number of warnings received concerning a particular piece or set of suspect data could be displayed to the user if the suspect data is received at the user's computer, and the user could then decide what action to take himself. Such a scheme is akin to the "warn" action of the primary embodiments, but provides the user with the additional information of how many other users have determined the data to be threat, thus providing the user with an assessment analogous to a community "confidence-level" in the data, such a level being of greater resolution than the simple "warn" or "block" conditions. Furthermore, such a scheme would not require the system to actually perform automatic blocking of suspect data, as this could be left to the user to do himself. This mode of operation provides the additional potential advantage that a highly skilled or confident user may not wish to act on the community warning if he is sure that it may be wrong, thus increasing the flexibility of the system.

[0097] Furthermore, in another embodiment based on the peer-to-peer architecture of the second embodiment, rather than broadcast to all peers e.g. all users in BT - sub-communities of peers are formed so that the initial broadcasting of warning messages concerning a particular is limited to a smaller number of peers. An example of such an architecture is shown in Figure 11, whence it will be seen that a plurality of sub-communities 111, 112, 113, and 114 may be provided, each connected to each other. Only when the count passes the warning threshold in one of the sub-communities is the suspect data signature distributed further. This limits the overheads incurred by false positives in that any particular sub-community would need to agree that a particular piece or set of data was a threat before it was broadcast to the wider community. "Crossover points" are provided between the sub-communities so that inter-group (or alternatively "inter-community") warnings may be passed on to other sub-communities by the provision of one of the peers 115 in a particular sub-community acting as the bridge via a communications link to another peer in a neighbouring sub-community, and only communicating with that other peer when the warning (or block) threshold has been passed in its own parent sub-community i.e. the sub-community has decided that the particular piece or set of suspect data really could be a threat.

[0098] The processing steps involved in such an embodiment are almost identical to those required in the second embodiment, with the addition that the processing steps depicted in Figure 10 are altered to include an additional step after step 10.3 or 10.5 wherein once the filter condition has changed (as a consequence of either step 10.3 or 10.5) if the computer is a bridging peer to another sub-community then an inter-group warning message is sent to the connected sub-community informing the other peer that a change in the filter condition has occurred for a particular piece or set of suspect data. Such an inter-group warning message should preferably include at least the data signature, as well as a warning condition indicator to communicate the threat level (e.g. warn or block) decided by the sending sub-community. By sending such a warning message to other sub-communities, the bridging peer is effectively declaring to the other sub-communities to which it is connected that its peers in its own sub-community have found the suspect data identified in the warning message to be a security threat.

[0099] Conversely, when an inter-group warning message is received at a bridging peer from another sub-community, the bridging peer should broadcast the inter-group warning message onwards to all its own peers as an intra-group message, to inform them of the threat assessment made by the neighbouring community. Each peer can then store the suspect data signature in its signature and ID database 76, for future reference by the data filter program 24 as needed (and in the same manner as already described in relation to the second embodiment).

[0100] Whether or not a user computer in a sub-community uses a forwarded inter-group warning message directly to initiate action (such as "warn" or "block" for the indicated suspect data) is optional, as it could be that it may wish to wait until it's own community has also agreed on action in respect of the same suspect data before actually initiating a "warn" or "block" condition for the data. Conversely, where a particular user-computer's own sub-community has already come to agreement on a particular piece or set of suspect data (i.e. either the "warn" or "block" threshold has been passed), then it may be that the user-computer may wish to wait until one or more connected sub-communities have also made the same finding before setting the appropriate

condition in its database for the data. It will be apparent that there are a multitude of possibilities as to when any particular user computer may wish to actually set an action condition for a particular piece or set of suspect data. For example, an action condition may be set for a particular piece or set of suspect data only if a user computer's own sub-community has agreed on the action in addition to a threshold number of other sub-communities, or alternatively if merely a threshold number out of the total number of sub-communities has agreed on the action regardless of the particular user computer's own sub-community's finding. Other such alternatives representing possible combinations of the logical conditions required in order to set an action condition will be apparent to the intended reader.

[0101] It is clear that such an architecture of sub-communities is analogous to a plurality of inter-connected local area networks, although it is not essential that each peer in a particular sub-community need actually be located in the same actual LAN (although this may be preferable). This is because all that defines any particular sub-community is the common cause that upon detecting suspect data a user broadcasts a message to all of its peers in the same sub-community, and hence all that is required is some message transport mechanism between peers. Such a message transport mechanism may be a single LAN to which all the peers are connected, or may be a sequence of interconnected networks such as the Internet, with different peers in a sub-community actually being part of different networks.

[0102] The modification to the second embodiment to split the user computers into sub-communities provides the additional advantage over the basic second embodiment that it, at least partially, overcomes the primary disadvantage of the high signalling bandwidth which would be required for the transmission of the warning messages to every other peer, and hence allows for scalability to add more users. As an example, consider the case where a community consists of n peers, such that when a warning message is broadcast from one of them to each of the others $(n-1)$ individual intra-group warning messages are sent. Now, assume that in order to agree on action about a particular piece or set of data a proportion of $1/a$ peers have to agree i.e. the action threshold is n/a. In such a case a total of $n(n-1)/a$ intra-group warning messages have to be transmitted in total for any one particular piece or set of data. Thus, where there are 40 peers, and half of them have to agree before action can be taken, a total of 780 warning messages would have to be transmitted (without error) before the action is agreed upon.

[0103] Now consider that the community of n peers is split equally into x sub-communities, as shown in Figure 11. In this case, within a sub-community for any single warning from one peer a total of $(n/x - 1)$ intra-group warning messages are broadcast. Assuming that in order to agree on action about a particular piece or set of data a proportion of $1/a$ peers have to agree i.e. the action threshold is n/ax, then a total of $n(n-x)/ax^2$ intra-group warning messages have to be broadcast within a single sub-community for action to be agreed on a single piece of suspect data. Assuming that each sub-community has to reach agreement within itself and communicate to each other group that it has so agreed before action can be taken by any peer in any sub-community, there will still only be a total of $n(n-x)/ax$ intra-group warning messages broadcast in total (the sum of the number of warning messages in total broadcast in each sub-community), plus $x(x-1)$ inter-group messages between sub-communities, plus $(n/x - 1)$ further intra-group messages from the bridging peer in each sub-community to each other peer in its respective community informing them of the findings of other sub-communities (i.e. forwarding received inter-group messages onwards). Note that with respect to this latter number of intra-group messages, $(n/x - 1)$ messages are sent from a bridging peer in a sub-community for each inter-group warning message received from another sub-community, such that where full knowledge of every other sub-community's findings about a particular piece or set of suspect data is required to be disseminated throughout a particular sub-community, the total number of intra-group messages transmitted by a single bridging node forwarding on the findings of other sub-communities would be $(x-1)(n/x - 1)$. This leads to the total number of intra-group messages forwarding inter-group messages across the entire community of users of $x(x-1)(n/x-1)$.

[0104] For example, if there are 40 peers split into 2 sub-communities of 20, then in each sub-community 190 messages are sent before action is agreed (i.e. the action threshold is reached) in each sub-community, 2 messages are exchanged between the respective bridging peers of each sub-community, one in each direction, and then the respective bridging peers each send 19 messages to inform their own peers of the finding of the other sub-community, to give a total of 420 messages. This is substantially less than the 780 messages which were required when the peers were not split into sub-communities.

[0105] It should be apparent that the above numeric examples are dependent solely on the logical conditions which were ascribed to the group operation, and in particular in the example above the need for every sub-community to agree on action about a particular piece or set of data and each issue its own inter-group warning message before any action is taken. Where slightly lower thresholds are set for action (e.g. if only half the number of sub-communities are required to agree), then the number of both inter- and intra-group warning messages which would be required to be broadcast before action is agreed would be further reduced.

[0106] In any of the embodiments of the invention described above and in other embodiments not described it should be noted that the one or more "threshold values" against which the respective counts for pieces or sets of suspect data are compared may be any value,

and may be fixed or dynamically adaptive. Furthermore the actual values may be generated by any appropriate function, such as for example by a random number generator, or as a function of some metric of accuracy of users in identifying viruses. Furthermore, it is foreseen that the threshold values may not simply be actual numbers which the respective counts must be greater than, but could instead be probabilistic in nature. Furthermore, various logical functions or rules can be used for their creation, including for example, fuzzy logic. The threshold values may therefore be any values, derived by any function or method, and are not limited to the particular examples of threshold values given in the specific embodiments described herein.

[0107] As described, the present invention therefore provides a collaborative computer security system wherein a distributed decision is taken involving all or substantially all of the users of the system, to effectively allow the users to "vote" on whether or not a particular piece or set of suspect data is a threat. By providing such a distributed system the response time to a computer virus attack can be improved over the prior art case where a central authority is required to perform analysis and take action, thereby reducing the potential damage which may be caused by such an attack.

Claims

1. A computer security system for use in a network environment comprising at least a first group of user computers arranged to communicate over a network, the system comprising a warning message exchange system operable to allow the communication from the user computers of warning messages relating to suspect data identified as a possible security threat; a message counting system operable to maintain a count for every particular piece or set of suspect data based on the number of warning messages communicated relating thereto; and network security means operable to act in respect of any particular piece or set of suspect data for which the count maintained therefor is substantially equal to or greater than at least one threshold value.

2. A computer security system according to claim 1, wherein the judgement as to whether any piece or set of data present at any one or more of the user computers is a possible threat is performed by the respective human user or users of the or each user computer.

3. A computer security system according to any of the preceding claims, wherein the network security means is further operable to compare the maintained count for a particular piece or set of suspect data against a plurality of different thresholds, and to take different action in respect of the data dependent upon which threshold or thresholds are met.

4. A computer security system according to any of the preceding claims, wherein the action taken by the network security means comprises warning each of the users as to the suspect nature of a particular piece or set of data.

5. A computer security system according to any of the preceding claims, wherein the action taken by the network security means comprises preventing the transmission of the particular piece or set of suspect data across the network.

6. A computer security system according to any of the preceding claims, wherein the or each warning message comprises at least an identifier of the piece of or set of suspect data to which it relates.

7. A computer security system according to claim 6, wherein the identifier is the piece or set of suspect data itself.

8. A computer security system according to claim 6, wherein the identifier is a repeatably generatable signature substantially unique to the piece or set of data.

9. A computer security system according to any of the preceding claims, and further comprising a network server arranged to receive the or each warning message communicated from the user computers, and further arranged to host the message counting system and at least part of the network security system, wherein that part of the network security system hosted by the server is operable to determine the action which should be taken against a particular piece or set of suspect data, and to communicate an action condition indicator indicative of the determined action to each of the plurality of users.

10. A computer security system according to any of the preceding claims, wherein a warning message generated by a user computer is broadcast to every other user computer, and each user computer is further arranged to host its own message counting system and network security system operable in accordance with any of the preceding claims.

11. A computer security system according to any of the preceding claims, and further comprising an inter-group communications system operable to allow the communication of warning messages relating to suspect data identified as a possible security threat from the first group of user computers to one or more other groups of other user computers.

12. A computer security system according to claim 11, wherein the inter group warning messages are transmitted by the inter-group communications system if the number of warning messages communicated by the user computers in the first group concerning a particular piece or set of data is substantially equal to or greater than at least one threshold value.

13. A computer security system according to claim 12, wherein the threshold value or values is/are the same as those at which the network security system takes action.

14. A computer security system according to any one of claims 11, 12, or 13, wherein the inter-group communications system is further arranged to receive inter-group warning messages from the one or more other groups of user computers.

15. A computer security system according to claim 14, and further comprising a second message counting system operable to maintain a count for every particular piece or set of suspect data based on the number of inter-group warning messages communicated relating thereto; and wherein the network security means is further operable to act against any particular piece or set of suspect data for which the count maintained therefor by the second message counting system is substantially equal to or greater than at least one inter-group threshold value.

16. A computer security system according to claim 15, wherein the network security means is further arranged to act against a particular piece or set of suspect data only if both the respective counts maintained by the message counting system and the second message counting system are substantially equal to or greater than at least the threshold value and the inter-group threshold value respectively.

17. A computer security system according to any of the preceding claims, wherein the message counting system is further arranged to store one or more weighting coefficients relating to one or more particular user computers, and to increment the count maintained for a particular piece or set of suspect data by an amount based upon the weighting coefficient when a warning is received from the one or more particular user computers relating to the particular piece or set of suspect data.

18. A computer readable storage medium storing one or more computer programs which when run on a computer causes the computer to operate in the manner of a system according to any of the preceding claims.

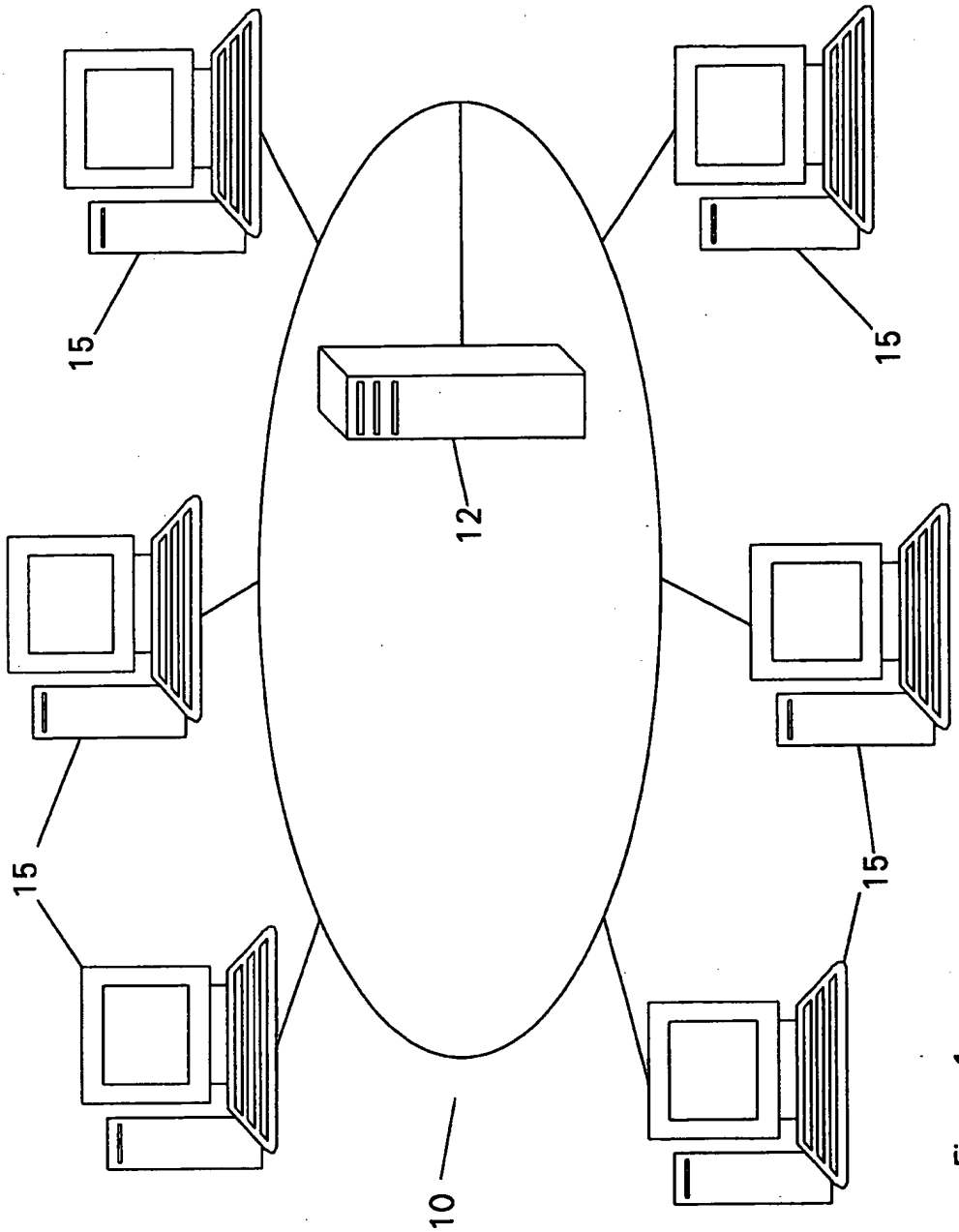


Figure 1

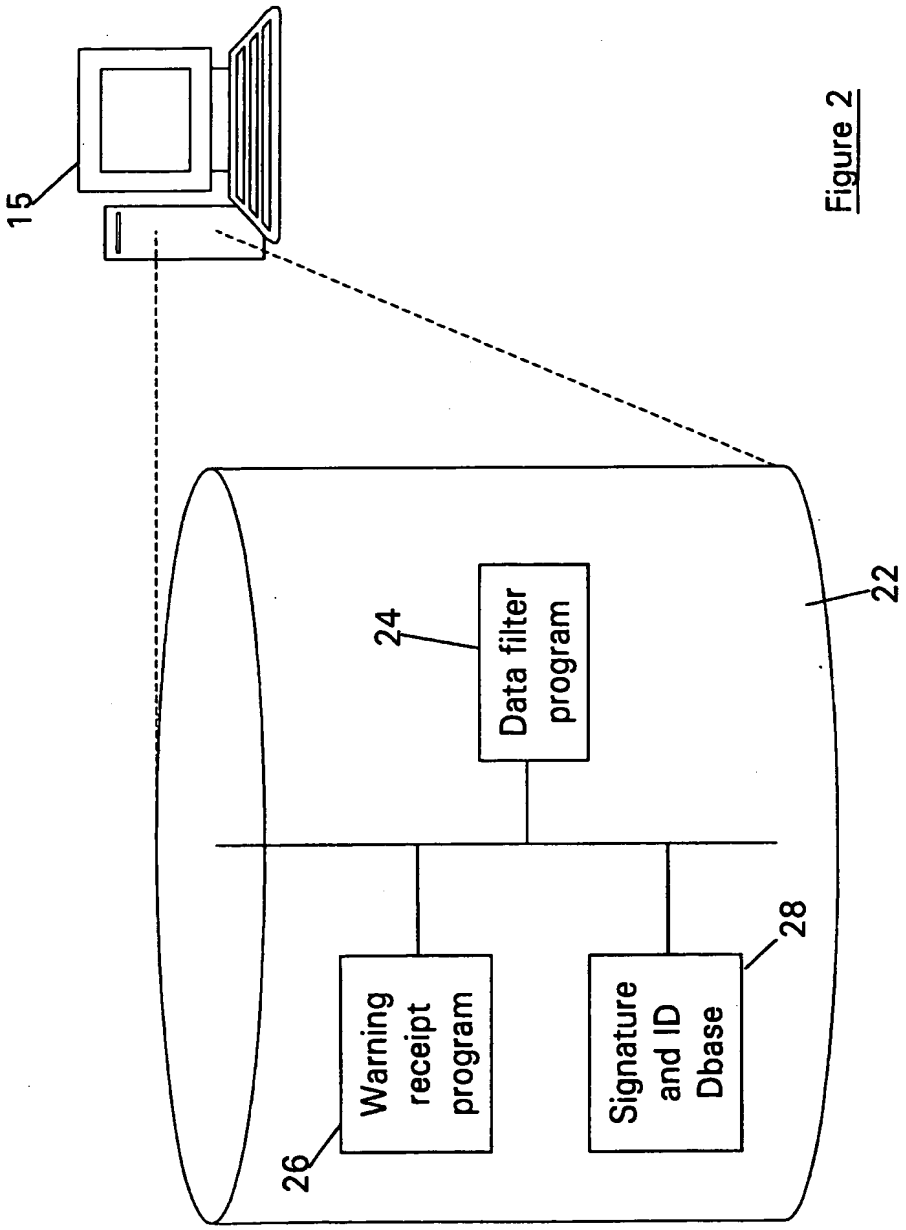

15
22
24
Data filter program
26
Warning receipt program
28
Signature and ID Dbase

Figure 2

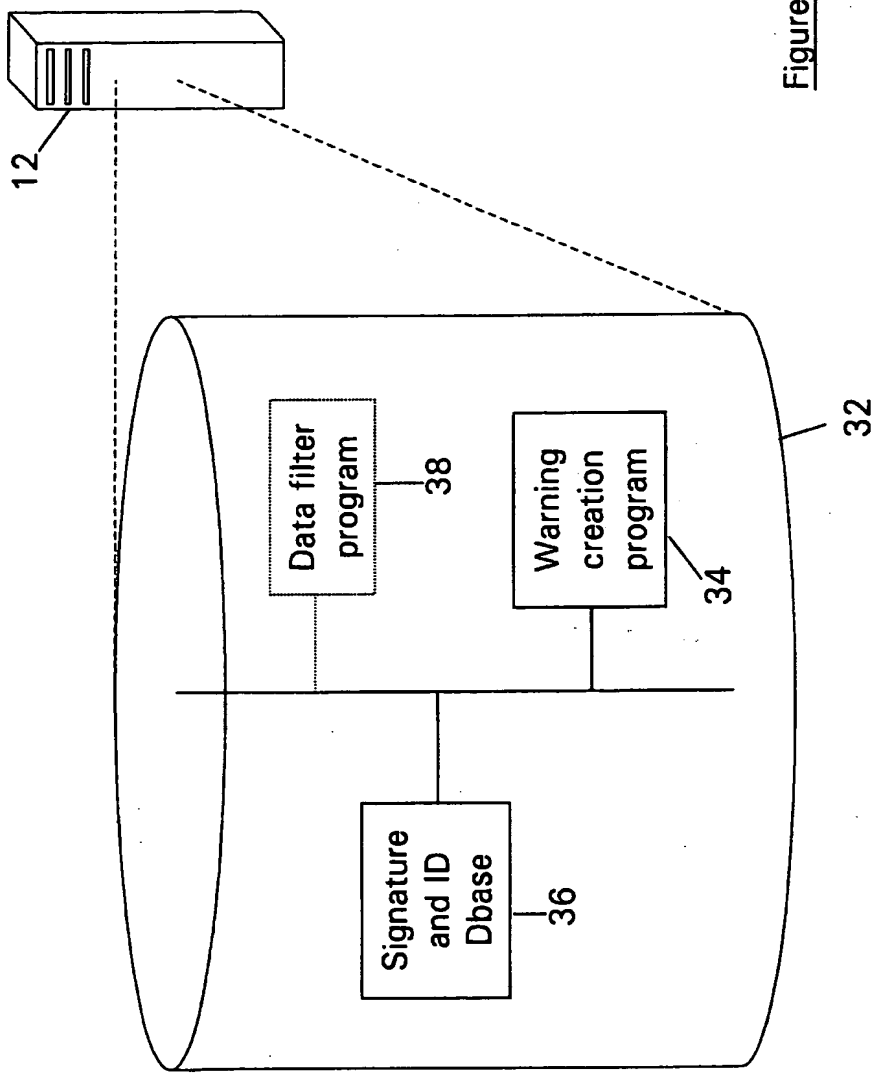

12
32
Data filter program
38
Warning creation program
34
Signature and ID Dbase
36

Figure 3

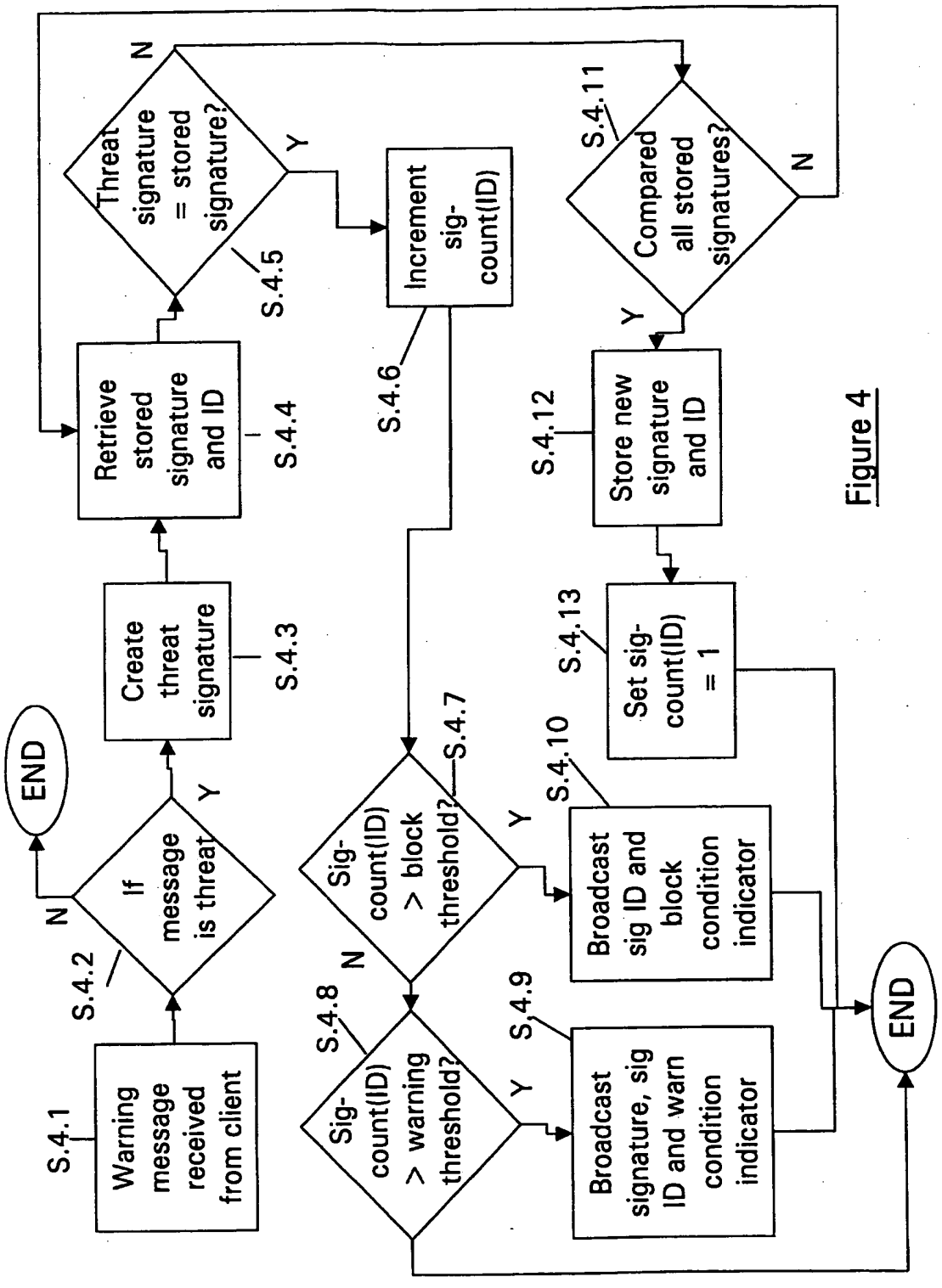

Warning message received from client
S.4.1
If message is threat
S.4.2
N
Y
END
Create threat signature
S.4.3
Retrieve stored signature and ID
S.4.4
Threat signature = stored signature?
S.4.5
N
Y
Increment sig-count(ID)
S.4.6
Sig-count(ID) > warning threshold?
S.4.8
N
Y
Broadcast signature, sig ID and warn condition indicator
S.4.9
Sig-count(ID) > block threshold?
S.4.7
Y
Broadcast sig ID and block condition indicator
S.4.10
Set sig-count(ID) = 1
S.4.13
Compared all stored signatures?
S.4.11
N
Y
Store new signature and ID
S.4.12
END

Figure 4

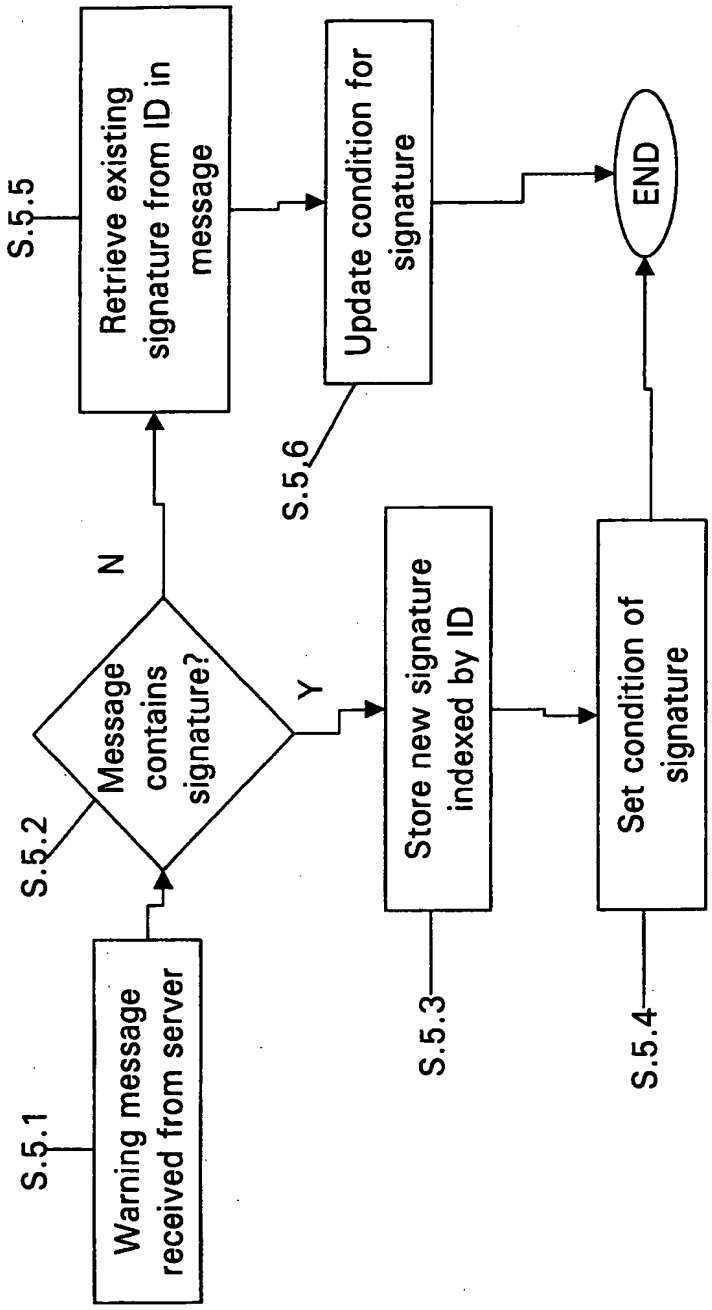

Warning message received from server
S.5.1
Message contains signature?
S.5.2
N
Y
Retrieve existing signature from ID in message
S.5.5
Update condition for signature
S.5.6
Store new signature indexed by ID
S.5.3
Set condition of signature
S.5.4
END

Figure 5

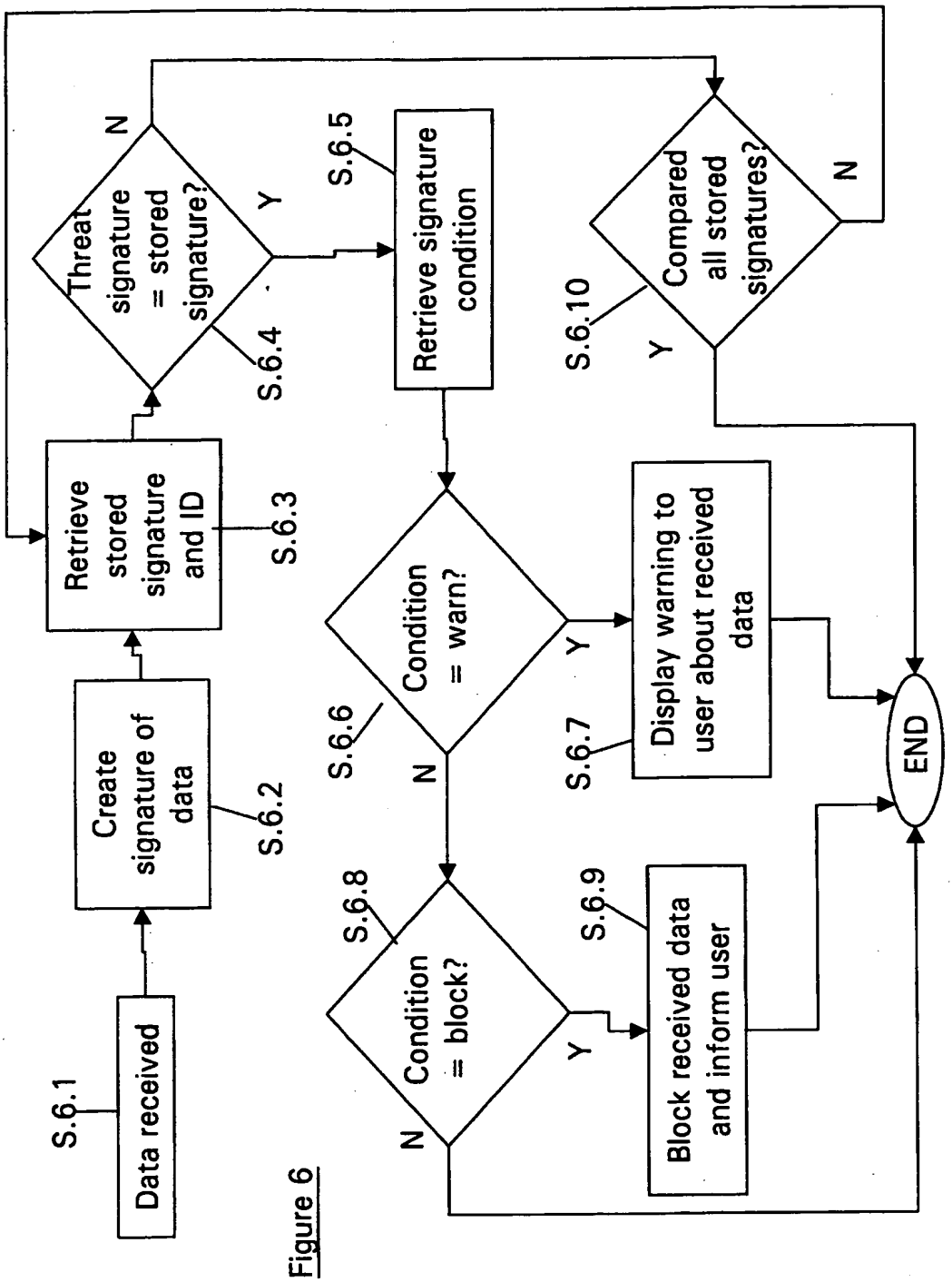

S.6.1
Data received
S.6.2
Create signature of data
S.6.3
Retrieve stored signature and ID
S.6.4
Threat signature = stored signature?
N
Y
S.6.5
Retrieve signature condition
S.6.6
Condition = warn?
N
Y
S.6.7
Display warning to user about received data
S.6.8
Condition = block?
N
Y
S.6.9
Block received data and inform user
S.6.10
Compared all stored signatures?
N
Y
END

Figure 6

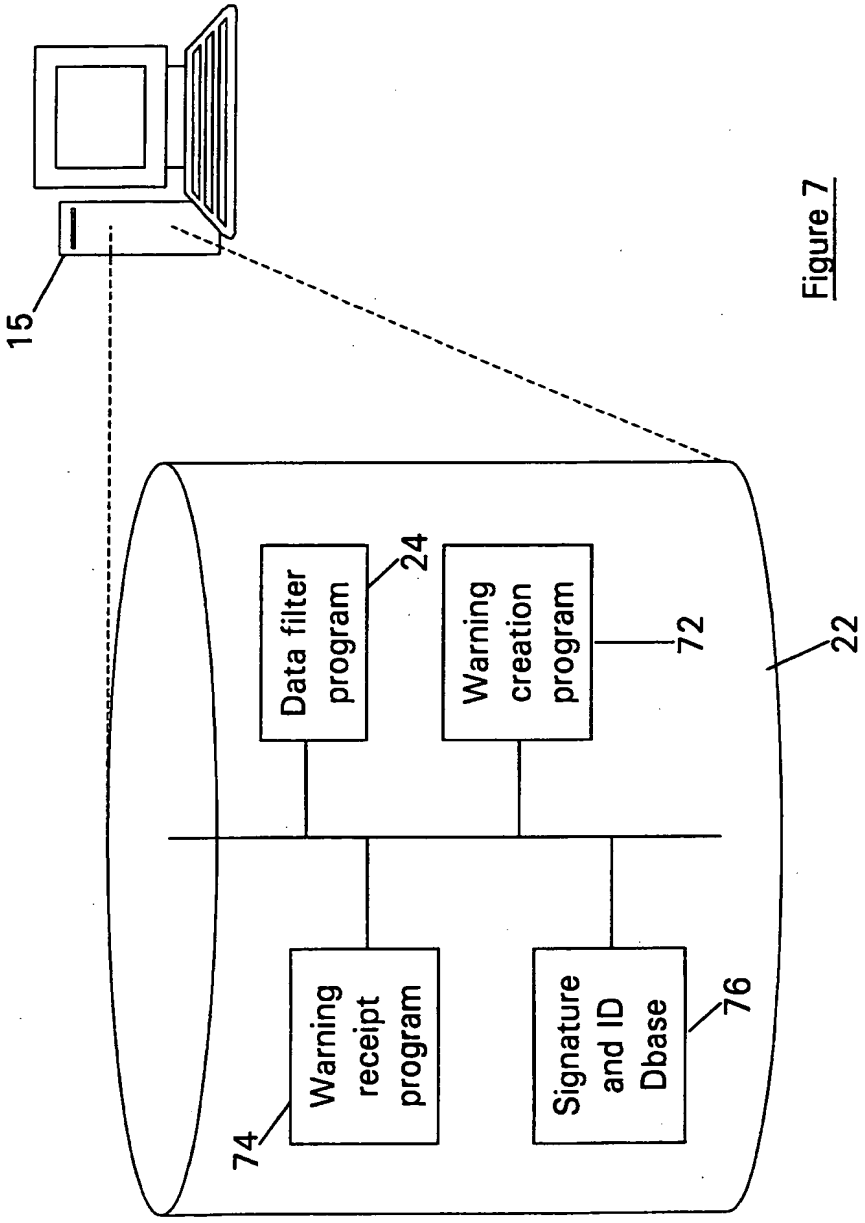

15
Data filter program
24
Warning creation program
72
22
Warning receipt program
74
Signature and ID Dbase
76

Figure 7

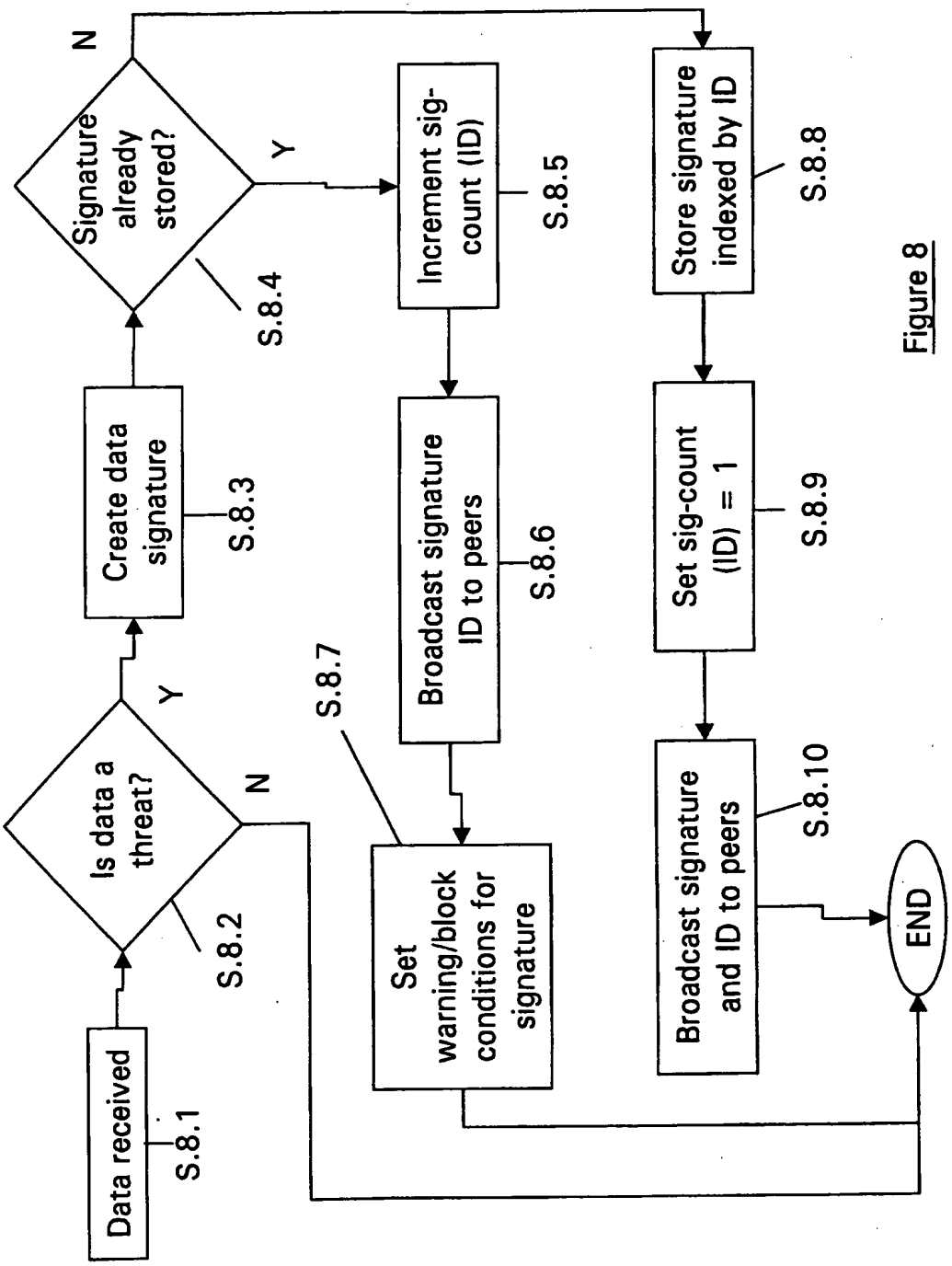

Data received
S.8.1
Is data a threat?
S.8.2
Y
N
Create data signature
S.8.3
Signature already stored?
S.8.4
Y
N
Increment sig-count (ID)
S.8.5
Broadcast signature ID to peers
S.8.6
S.8.7
Set warning/block conditions for signature
Store signature indexed by ID
S.8.8
Set sig-count (ID) = 1
S.8.9
Broadcast signature and ID to peers
S.8.10
END

Figure 8

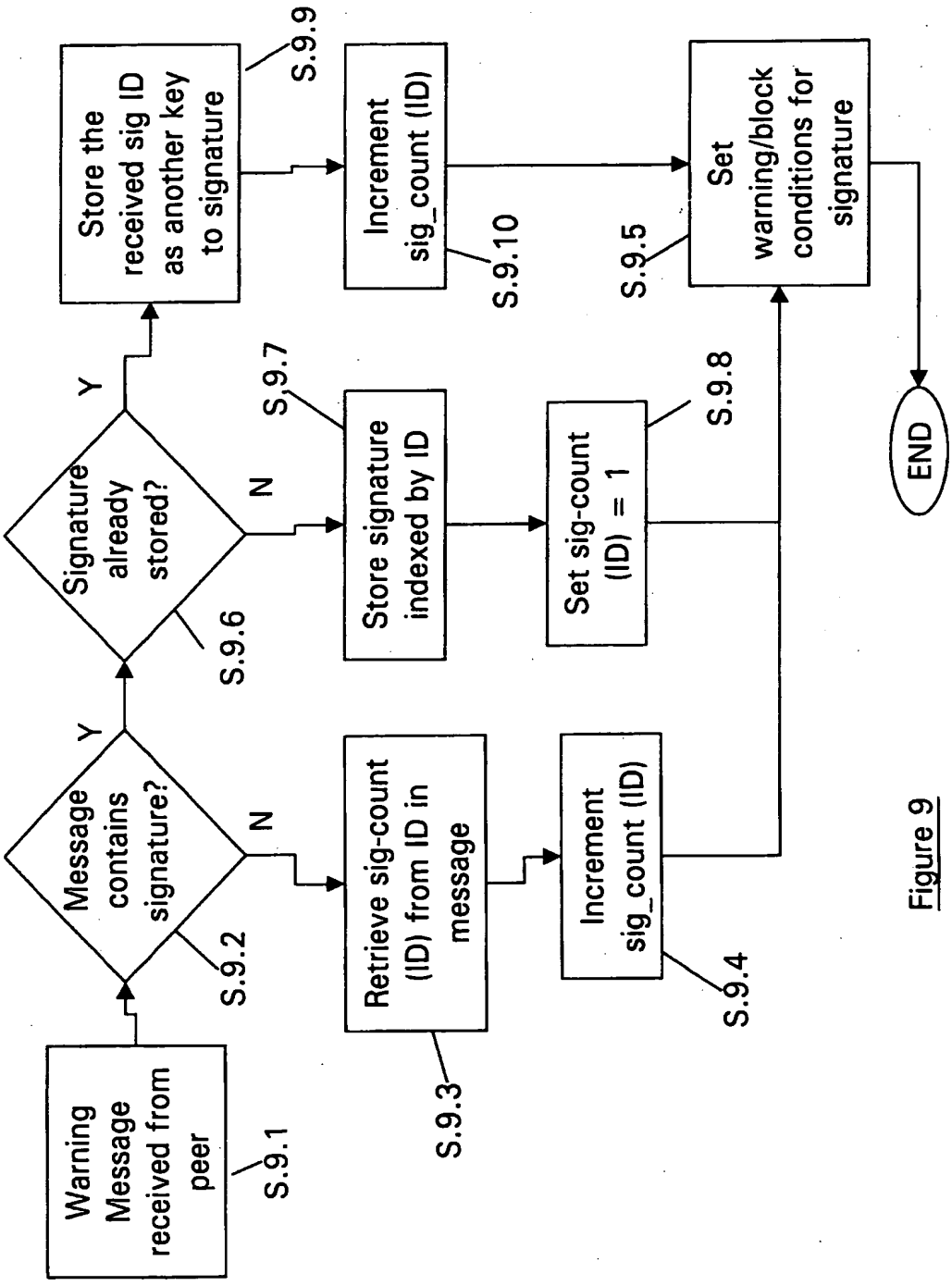

Warning Message received from peer
S.9.1
Message contains signature?
S.9.2
Y
N
Retrieve sig-count (ID) from ID in message
S.9.3
Increment sig_count (ID)
S.9.4
Signature already stored?
S.9.6
Y
N
Store signature indexed by ID
S.9.7
Set sig-count (ID) = 1
S.9.8
Store the received sig ID as another key to signature
S.9.9
Increment sig_count (ID)
S.9.10
Set warning/block conditions for signature
S.9.5
END

Figure 9

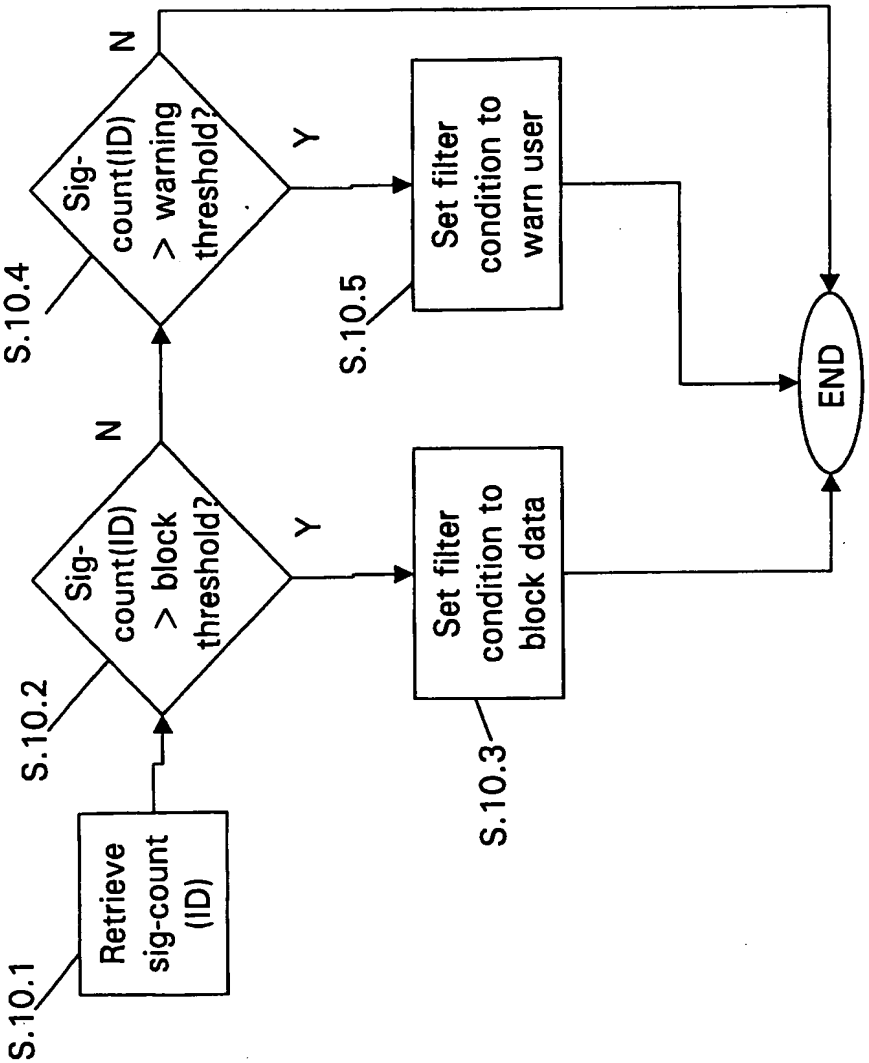

S.10.1
Retrieve sig-count (ID)
S.10.2
Sig-count(ID) > block threshold?
Y
N
S.10.3
Set filter condition to block data
S.10.4
Sig-count(ID) > warning threshold?
Y
N
S.10.5
Set filter condition to warn user
END

Figure 10

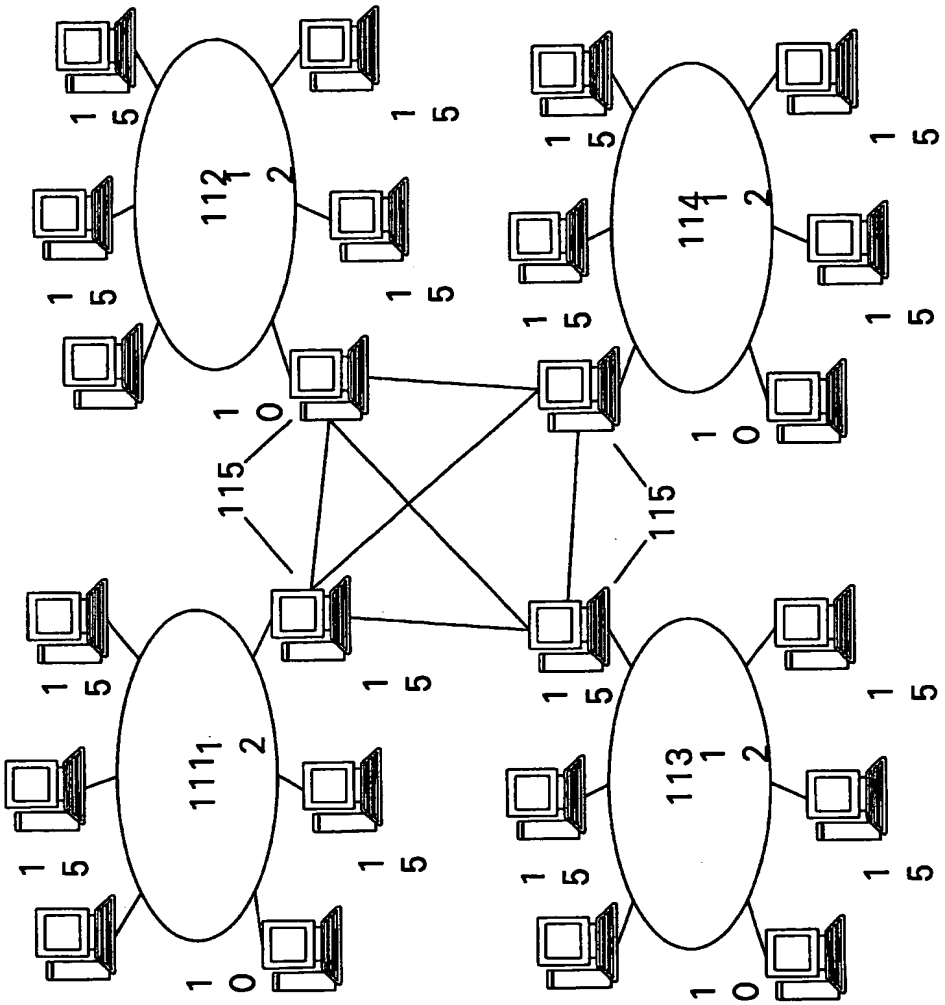

111
112
113
114
115
115
10
10
10
10
15
15
15
15
15
15
15
15
15
15
15
15
15
15
15
15
15
15
15
15

Figure 11

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 30 9795

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	US 5 440 723 A (KEPHART JEFFREY O ET AL) 8 August 1995 (1995-08-08) * abstract; figures 2,3,7,8 * * column 4, line 29 - column 5, last line * * column 19, line 45 - column 20, last line *	1	G06F1/00 H04L29/06
A		2,3,5-8, 10-14	
A	US 5 621 889 A (EMERY THIERRY ET AL) 15 April 1997 (1997-04-15) * abstract; figure 2 * * column 2, line 9 - line 55 * * column 3, line 30 - column 4, line 43 *	1-3,17	
A	WO 01 71499 A (COUNTERPANE INTERNET SECURITY) 27 September 2001 (2001-09-27)		
A	ZENKIN D: "Fighting Against the Invisible Enemy - Methods for detecting an unknown virus" COMPUTERS & SECURITY. INTERNATIONAL JOURNAL DEVOTED TO THE STUDY OF TECHNICAL AND FINANCIAL ASPECTS OF COMPUTER SECURITY, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, vol. 20, no. 4, 31 July 2001 (2001-07-31), pages 316-321, XP004254268 ISSN: 0167-4048		TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	31 July 2002	Powell, D

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 30 9795

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

31-07-2002

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5440723 A	08-08-1995	NONE	
US 5621889 A	15-04-1997	FR 2706652 A1	23-12-1994
		CA 2125445 A1	10-12-1994
		DE 69428631 D1	22-11-2001
		EP 0629940 A1	21-12-1994
		ES 2164091 T3	16-02-2002
		NO 942108 A	12-12-1994
WO 0171499 A	27-09-2001	US 2002087882 A1	04-07-2002
		AU 4735101 A	03-10-2001
		WO 0171499 A1	27-09-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

Annex to Form PCT/ISA/206
COMMUNICATION RELATING TO THE RESULTS OF THE PARTIAL INTERNATIONAL SEARCH

International Application No
PCT/GB2006/002439

1. The present communication is an Annex to the invitation to pay additional fees (Form PCT/ISA/206). It shows the results of the international search established on the parts of the international application which relate to the invention first mentioned in claims Nos.:
see 'Invitation to pay additional fees'
2. This communication is not the international search report which will be established according to Article 18 and Rule 43.

3. If the applicant does not pay any additional search fees, the information appearing in this communication will be considered as the result of the international search and will be included as such in the international search report.

4. If the applicant pays additional fees, the international search report will contain both the information appearing in this communication and the results of the international search on other parts of the international application for which such fees will have been paid.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 549 012 A (DATACENTERTECHNOLOGIES N.V) 29 June 2005 (2005-06-29) cited in the application page 2, paragraphs 6,7	1-11, 13-23, 25-30, 37,39,40
Y	page 3, paragraphs 17,18 page 6, paragraph 40 - paragraph 43 page 8, paragraph 50 - paragraph 51 page 10, paragraph 68 - paragraph 72	12,24, 38,41
X	US 2004/153644 A1 (MCCORKENDALE BRUCE ET AL) 5 August 2004 (2004-08-05) cited in the application page 2, paragraphs 21,25 page 4, paragraph 43 - page 5, paragraph 53	1-11, 13-23, 25-30, 37,39,40
X	EP 1 315 066 A (BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY) 28 May 2003 (2003-05-28) page 3, paragraph 8 page 6, paragraph 36 - page 7, paragraph 46 abstract	1,13,25, 30,37, 39,40

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

3

(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/039921 A1 (CHUANG SHYNE-SONG) 26 February 2004 (2004-02-26) cited in the application page 2, paragraph 33 - page 4, paragraph 51	1,13,25, 30,37, 39,40
Y	US 2005/086500 A1 (ALBORNOZ JORDI A) 21 April 2005 (2005-04-21) page 1, paragraph 8	12,24, 38,41
A	ZENKIN D: "Fighting Against the Invisible Enemy - Methods for detecting an unknown virus" COMPUTERS & SECURITY, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, vol. 20, no. 4, 31 July 2001 (2001-07-31), pages 316-321, XP004254268 ISSN: 0167-4048	

3

Patent Family Annex

Information on patent family members

International Application No
PCT/GB2006/002439

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1549012 A	29-06-2005	NONE	
US 2004153644 A1	05-08-2004	NONE	
EP 1315066 A	28-05-2003	CA 2467186 A1	05-06-2003
		WO 03046699 A1	05-06-2003
		US 2005022014 A1	27-01-2005
US 2004039921 A1	26-02-2004	AU 9620501 A	29-04-2002
		WO 0233525 A2	25-04-2002
US 2005086500 A1	21-04-2005	NONE	

Whole issue.qxd 6/22/01 1:34 PM Page 316

Computers & Security, 20 (2001) 316-321

Fighting Against the Invisible Enemy

Methods for detecting an unknown virus

Dr Denis Zenkin

Kaspersky Lab, 10, Geroyev Panfilovtcev Str., Moscow, 123363, Russia. Denis@kaspersky.com

Protection from unknown viruses is, indisputably, the issue of the day in computer virology. As is well known, the main deficiency of commonly used anti-virus scanners is that these scanners are able to detect and delete only those malicious programs described (by the so-called virus signatures) in their anti-virus databases. And for the new virus signature to be included in the anti-virus database, first, the scanner manufacturer must receive this virus from one of its customers, then the company experts must analyse it and develop a corresponding update. Then they must test this update and deliver it to the end users. Sometimes this procedure takes several days, and for the most complicated viruses, it can take up to a week. To our regret, during this time, end users of these scanners remain unprotected from this new virus threatening their computer security.

To shorten the period between when a new virus appears and when the corresponding update is delivered to end-users – and to speed-up the development of those updates - anti-virus manufacturers have created and implemented certain technology allowing for automation of the analysis and development and testing of updates for anti-virus databases. E-mail and advantages of other contemporary data-transmission technologies (for example, the so-called "Push" technology) allow delivery of ready-made anti-virus database updates to end-users within seconds. But all these efforts still haven't solved the key problem: protection from unknown (new) viruses before they actually appear.

So, let us discuss the so-called unknown virus in terms of conventional anti-virus scanners. The approach that is used in these programs allows them to check files on a computer when searching for virus-code fragments within their contents. The scanner acts just like a police officer checking every suspicious person (in this case, it is a suspicious program code) against the police database of criminals. The main disadvantage to this approach is that the scanner cannot detect a virus if the database doesn't contain its description (its code fragment). At the same time, to detect a malicious program (as well as a criminal) one does not need its detailed picture; it is enough to have just its rough description and to understand the features that are shared by all computer viruses. This advanced approach underlies the most up-to-date software tools allowing search for unknown viruses: a heuristic analyzer of the program code, redundant scanning, and behavior blocker and integrity checker.

Heuristic Analyzer

The main feature of computer viruses is their ability to embed in other programs while partially or completely retaining those programs' operability. During

0167-4048/01$20.00 © 2001 Elsevier Science Ltd

BNSDOCID: <XP_____4254268A__I_>

Whole issue.qxd 6/22/01 1:34 PM Page 317

the process of infection (when a virus embeds itself into a program), viruses modify the parent program so that when the system calls this program, the malicious code is executed. In addition, while trying to infect files, to mask their presence etc., all viruses use the same algorithms.

This feature was detected by anti-virus experts, and at the beginning of the 1990s, they developed and implemented in anti-virus scanners a tool allowing for the detection of unknown viruses – a heuristic analyzer. This tool checked objects by analyzing the instructions sequence in the objects' contents, accumulating statistics and making a decision about whether the object was infected with an unknown computer virus or not.

This generation of heuristic tools used a fairly primitive approach when detecting unknown viruses – if the tool detected a certain instruction sequence that is appropriate for computer viruses, the program, where this sequence had been detected, was automatically identified as infected. As a result, this technology produced a large number of false alarms and was ineffective. Very soon, virus-writers responded to this challenge by implementing various techniques, like encryption or polymorphism, allowing viruses to deceive heuristic analyzers. Then anti-virus experts developed the second-generation heuristic tool. Unlike its predecessor, this analyzer reconstructed the behavior of the program it was checking, and used it as a basis for conclusions about the potential danger from this program. This approach proved to be much more effective than that of the first generation and is still used by most heuristic tools today.

However, the heuristic technology is far from being a cure-all for a computer infection. This approach still has too many weak points to be the absolute defense: false alarms, deficient reliability, slowdown in system operation and the inability to delete the viruses detected are problematic of heuristic analyzers. Furthermore, to be able to detect different virus types, you must use different approaches and, as a result, different heuristic algorithms.

Redundant Scanning

Unlike the heuristic analyzer that is currently used in every competent anti-virus program, the redundant scanning technique is not so common. At the same time, while having a lot of disadvantages (the main one is that it tremendously slows down computer operation – up to several times), this technique allows for the detection of viruses that use extraordinary methods when infecting files. One of these methods is used by viruses "not having an entry point" (Entry Point Obscuring viruses (EPO-viruses)). The EPO family includes viruses that don't write the transfer-of-control instructions in the header of COM files (JMP) and do not change the starting point address in the header of EXE files. These viruses write the transfer-to-virus instruction somewhere in the file contents and obtain control when the procedure containing a code transferring control to the virus body is executed rather than directly after the infected file has been started. And this procedure might well be one of those that are rarely executed (for example, the procedure displaying a message about some program error). As a result, this virus may well be a "sleeper" inside of a file for years, breaking loose under certain circumstances. Before writing the transfer-to-virus instruction into a file, the virus must choose a "correct" address in this file; otherwise this file might well be corrupted. There are several methods that viruses use when defining these addresses:

The first method is to search for a sequence of standard C/Pascal code in the file ("Lucretia" and "Zhengxi" viruses). These viruses search for standard C/Pascal header procedures and overwrite these with their code.

The second method is to trace and disassemble the file code ("CNTV," "MidInfector" and "NexivDer" viruses). These viruses load the file into the computer memory, then trace or disassemble it, and depending on various conditions, choose an instruction (or instructions) to be overwritten with the transfer-to-virus code.

The third method is used by memory-resident viruses only. These viruses control various system

BNSDOCID: <XP_____4254268A_1_>

Whole issue.qxd 6/22/01 1:34 PM Page 318

interrupts (INT 21h most often). When the file to be infected summons this interrupt, the virus overwrites this call-interruption instruction with its code ("Avatar.Positron" and "Markiz" viruses).

Redundant scanning - thorough scanning - allows for the scanning of not only the system processing entry points, but the entire contents of any given object. In fact, this method is not absolutely an unknown-virus search technology. This function allows an anti-virus program to look far inside a file in order to see its entire contents are inaccessible for others.

Integrity Checker

An integrity checker is yet another anti-virus technology that allows for unknown-virus detection. It is based on the fact that viruses can be considered average programs that have the ability for the unauthorized creation of new and modifying or self-planting into existing objects (such as files, boot sectors or system registry). In other words, they make changes in the file system that can be detected, and the presence of the malicious code is revealed.

An integrity checker's operating mode is based on a collection of original "prints" (CRC-values) of files, boot sectors and a system registry. These "prints" are stored in the database. When started up anew, the integrity checker compares information from its database with current "prints" and informs a user of changes that have occurred, marking virus-like changes and other unsuspicious changes. Some integrity checkers offer improved file analysis for the fastest and the most effective checking of a computer requiring minimum system resources. Thanks to the support of the most popular executable-file formats—such as PE EXE (Win32), NE EXE (Win16), MZ EXE (DOS), SYS, COM, OLE2 (Word, Excel and Access), and ELF (Linux)—an integrity checker can create Fast-CRC sums for the file system. This increases productivity up to five times while keeping the highest reliability of data control.

In 1990, the first stealth viruses, "Frodo" and "Whale," caused discredit to this anti-virus technology, because the operating mode of a stealth virus is based on hiding its activity within an infected system by substituting the original (not infected) files and boot sectors while the integrity checker tries to scan them. To do so, these viruses intercept disk-access interrupts and, in the case that an attempt to read, write or execute an infected object is detected, the virus passes off the clean copy of the object. Fortunately, this difficulty with detecting stealth-viruses was solved soon, and integrity checkers began accessing disks directly through an IOS disk driver (Input/Output Supervisor), leaving the system interrupts. This allowed integrity checkers to effectively protect computers against even stealth viruses.

Considered amongst the advantages of the most improved integrity checkers can be: the extremely high operating speed; low hardware and software requirements; and the high rate of file- and boot-sector restoration that previously have been infected with computer viruses, including unknown ones. An integrity checker's approach to infected-object restoration is based on the knowledge of what should be inside a clean file or boot sector rather than on the knowledge of a virus' appearance: everything that violates this rule is considered as a change and is subject to being reported to the user and having the original content restored. This is the reason why integrity checkers don't require a multi-megabyte, anti-virus database; instead being content with the virus-infecting method descriptions that usually need no more than 500 Kbytes. Knowing these methods, an integrity checker can quickly and effectively delete a virus and restore a file or a boot sector regardless in which part of an infected object the virus code is found - whether it is in the beginning, middle or end of the file, or the virus code is scattered all around it. It doesn't matter whether a virus is known and presented in the anti-virus database of the traditional virus scanner or not.

Integrity checkers are used not only in basic standalone home computers, but also on the server level of computing systems. For example, they are very useful for control over a Web server's content in order to prevent any unauthorized alternations. In real-time mode, an integrity checker performs the continuous checking of the Web server's integrity,

BNSDOCID: <XP_____4254268A__1_>

Whole issue.qxd 6/22/01 1:34 PM Page 319

tracking all changes. In case an unauthorized alternation is detected, the program can immediately send an SMS or pager message or an e-mail notification to the system administrator reporting the incident, and allows for full recovery of the original Web server's contents from back up. Therefore, integrity checkers can be considered powerful tools not only against known and unknown computer viruses, but also as strong alternative intrusion detection and anti-hacker utilities.

However, this type of anti-virus program has some disadvantages. Firstly, integrity checkers cannot detect a virus upon entry into the system and can only do so after a period of time. By then, the virus has already spread into the computer. Secondly, they cannot detect a virus in new files (e-mail, files on disks, files restored from backup, and files extracted from archive), since there is not any information in their databases about these objects. Some viruses even use this feature and infect only newly developed files, thereby staying "invisible" to the integrity checkers. Finally, in order to provide anti-virus protection, these anti-virus programs must be run on a regular basis. The more often they are run, the more reliable the virus-activity control will be.

Behaviour Blocker

The common disadvantage of all of the aforementioned techniques for the detection of unknown viruses is lack of guaranteed protection. None of them can provide 100% defense. It seems that the absolute solution for this problem could be found only after the discovery of an artificial intellect that would be able to analyze computer data as today's best anti-virus experts do. But nowadays, we can predict the most likely trends for anti-virus software development. In our opinion, these trends will include the so-called behaviour blocker (sandbox) anti-virus. Such programs can guarantee 100% protection against attacks from new viruses.

What is a behaviour blocker? It is a memory-resident program intercepting various computer operations. If a virus-suspected action (actions that can be performed by a virus or some other harmful program) is detected, a behaviour blocker thwarts it or asks the user for instructions. In other words, the behavior blocker does not search for virus signatures (i.e., a virus code as do traditional anti-virus scanners), and does not compare files with their original content, but rather monitors any program's activity and prevents harmful actions that could be done by malicious code.

The blocker idea is not new. These anti-virus programs were developed long ago, but did not achieve widespread use in the majority of desktop applications, because they were very intrusive, difficult to customize and required an in-depth knowledge of computer technologies on the part of the user. Nevertheless, the behavior blocker has found its niche in other areas of information technology. For instance, the widely used Java standard, developed by the Sun company, provides each single Java application with a strictly limited virtual space (a set of allowed activities) that prevents any attempts by any Java-based program to execute functions (such as deleting files) that a user may be suspicious of and could pose a threat to his or her computer security.

Let us look at the pros and cons of the behaviour blocker in detail. In theory, the behaviour blocker may prevent the distribution of any known and/or unknown (written after the blocker was developed) virus, warning the user about the virus before it is able to infect other files or damage his or her computer. However, the operating system itself or some useful utilities can also perform virus-like actions (such as creating files or modifying the system registry). Here we need to define two types of blockers: behaviour blockers for file systems and behaviour blockers for certain applications.

The behaviour blocker for file systems (a standard blocker used to combat file viruses) cannot independently determine the source of a virus-suspected action (virus, operating system or utility), and asks the user for instructions. This means that ultimately, it is the user who has to make the decision. And in order to make the correct decision, the user must possess sufficient knowledge and experience, otherwise the operating system or utility will be disabled

BNSDOCID: <XP_____4254268A__1_>

Whole issue.qxd 6/22/01 1:34 PM Page 320

from performing the action required, or a virus will penetrate the system.

This is the main reason why blockers have yet to become popular. Their strength is their weakness - they annoy users with their intrusive questions and have been uninstalled. This situation can only be improved with the invention of an artificial intelligence able to independently define the reasons for any virus-suspected action.

The behaviour blockers for applications have much more of a chance of becoming as popular as the traditional anti-virus scanners today, because their area of operating is strictly limited by the structure of a given application. Therefore, the blocker should only keep its eye on the defined number of activities that can be performed by programs developed especially for this application.

The most illustrative example of how efficient and easy-to-operate a behaviour blocker could be is Microsoft Office and the problem of fighting against unknown macro-viruses. For macro-programs written in the most widespread programming language VBA (Visual Basic for Application), good actions can be automatically differentiated from harmful with a fairly high success rate.

The behavioural analysis of macro-viruses reveals the most frequent sequences of operations. It allows embedding in the blocker a highly intelligent system of macro-instructions filtering that is able to detect the macro-instructions and, with a high probability rate, to define those that can be considered threatening. Due to this, behaviour blockers for applications are not so "intrusive" as its file-brothers. Although asking fewer questions, this blocker still provides reliable anti-virus protection and is able to protect a user from virtually all known and unknown macro-viruses.

Furthermore, a behaviour blocker intercepts and blocks the execution of even multi-platform macro-virus instructions; i.e., viruses that are able to operate in various Office applications. The program is equally reliable in Microsoft Word, Excel, Access and PowerPoint and even applications that are not included in the Microsoft Office package, but still use VBA (such as Visio and AutoCAD)

The blocker also monitors macros working with external applications, including e-mail programs. In so doing, it eliminates the possibility of macro-virus distribution through e-mail, as was the case in May 2000 when the ILOVEYOU virus infected hundreds of thousands of computers worldwide. Unlike traditional anti-virus technologies, a behaviour blocker solves this problem by blocking macro access to e-mail. In case the blocker detects an attempt by a macro-program to access an e-mail program, it can block the function or even terminate the whole macro (according to the pre-defined user preferences).

While employing a behaviour blocker for Microsoft Office, a user avoids the boring need to regularly download and install updates for the anti-virus database in order to protect his or her computer from new macro-viruses, since the program intercepts any new macro-virus by default. This solves the problem of the most dangerous period between the appearance of a new virus and the release of a corresponding update. Once installed, a behaviour blocker reliably protects a computer from macro-viruses until the next version of VBA (with expanded capabilities that can be used by macro-viruses) is released.

The behaviour blocker solves the problems of macro-virus detection and distribution prevention, but is not designed to delete macro-viruses and restore the infected files. This is why it should be used in conjunction with an anti-virus scanner that is able to delete viruses. The blocker allows a user to run Microsoft Office applications during the period between the appearance of a new macro-virus and the release of a corresponding scanner-database update without fear of losing vital data or serious damage to the hardware.

In our opinion, with the development of computer technologies - especially in the field of artificial intelligence - the magnitude, efficiency and simplicity-of-use of behaviour blockers (even those designed for a file system) will rapidly increase. In the near future, this type of anti-virus program preventing new and,

BNSDOCID: <XP_____4254268A__1_>

Whole issue.qxd 6/22/01 1:34 PM Page 321

thus unknown viruses from penetrating and distributing will become the major tool for anti-virus protection.

Conclusion

This overview of the modern technologies for protection against unknown viruses would not be complete if we did not mention the best way of using them. Which method is the best and most effective for use with home computers and enterprise-wide networks? We recommend something pretty simple: the best choice for protection against unknown viruses is a well-organized combination of all of the aforementioned techniques. Following the well-known proverb - don't put all your eggs in one basket - we also recommend not relying on integrity checkers or heuristic analyzers only. All of them have advantages and disadvantages. When used together, they ideally compensate for each other's deficiencies, and many years of their successful combined usage has proved this to be the best way to protect computing systems of any scale and complexity, whether it is a stand-alone home computer or huge corporate network.

BNSDOCID: <XP_____4254268A_|_>

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/477,807	06/30/2006	Melvyn Morris	011765-0355986

CONFIRMATION NO. 4785

909
PILLSBURY WINTHROP SHAW PITTMAN, LLP
P.O. BOX 10500
MCLEAN, VA22102

Title: Methods and apparatus for dealing with malware

Publication No. US-2007-0016953-A1
Publication Date: 01/18/2007

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 703-305-3028.

Pre-Grant Publication Division, 703-605-4283

JAN 18 2007
PATENT & TRADEMARK OFFICE

Docket Number: 011765-0355986 PATENT APPLICATION
Client Reference: AF/JG/P10717US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re the Application of

MELVYN MORRIS, et al. Group Art Unit: 2131

Application No.: 11/477,807 Examiner: Unassigned

Filed: June 30, 2006 Confirmation No.: 4785

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

Pursuant to 37 CFR 1.56, the attention of the Patent and Trademark Office is hereby directed to the references listed on the attached PTO-1449. Copies of the required references are provided herewith. It is respectfully requested that the information be expressly considered during the prosecution of this application, and that the references be made of record therein and appear among the "References Cited" on any patent to issue therefrom. Applicants respectfully request the Examiner return an initialed copy of the enclosed Form PTO-1449 to Applicants with the next Office communication to indicate that the references have been considered, per MPEP § 609.

This Information Disclosure Statement is being filed before the mailing of the first Office Action on the merits, and therefore no fee is required herewith.

Respectfully Submitted,
PILLSBURY WINTHROP SHAW PITTMAN LLP

Christine H. McCarthy
Registration Number 41844
Customer Number: 00909

Date: January 18, 2007
Telephone: (703) 770-7900
Facsimile: (703) 770-7901
P.O. Box 10500
McLean, VA 22102

OIPE IAP97
JAN 18 2007
PATENT & TRADEMARK OFFICE

FORM PTO-1449 (modified)
To: U.S. Department of Commerce
(PW FORM PAT-1449) Patent and Trademark Office

Atty. Dkt. No.	M#	Client Ref.
	11765-0355986	

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Applicant: MORRIS et al.

Appln. No.: 11/477,807

Filing Date: June 30, 2006

Date: January 18, 2007 Page 1 of 1

Examiner: Unassigned | Group Art Unit: 2131

U.S. PATENT DOCUMENTS

Examiner's Initials*		Document Number	Date MM/YYYY	Name (Family Name of First Inventor)	Class	Sub Class	Filing Date (if appropriate)
	AR	6,772,346	08/2004	CHESS			
	BR	6,944,772	09/2005	DOZORTSEV			
	CR	2004/0006704	01/2004	DAHLSTROM			
	DR	2004/0073810	05/2004	RACINE			
	ER	2005/0021994	01/2005	BARTON			
	FR	2004/0153644	08/2004	McCORKENDALE			
	GR						
	HR						
	IR						
	JR						
	KR						
	LR						
	MR						
	NR						

FOREIGN PATENT DOCUMENTS

		Document Number	Date MM/YYYY	Country	Inventor Name		English Abstract		Translation Readily Available	
							Enclosed	No	Enclose	No
	OR	1 280 040	01/2003	EP						
	PR	1 549 012	12/2003	EP						
	QR	03/021402	03/2003	WO						
	RR	2004/097602	11/2004	WO						
	SR	02/33525	04/2002	WO						

OTHER (Including in this order Author, Title, Periodical Name, Date, Pertinent Pages, etc.)

	TR				
	UR				
	VR				
	WR				

Examiner | Date Considered:

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP § 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to Applicant.

PAT-1449 12/98

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 280 040 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.01.2003 Bulletin 2003/05

(51) Int Cl.[7]: **G06F 1/00**

(21) Application number: **02253235.2**

(22) Date of filing: **09.05.2002**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **26.07.2001 US 912392**

(71) Applicant: **Networks Associates Technology Inc.**
Santa Clara, CA 95054 (US)

(72) Inventors:
- **Hinchliffe, Alexander James**
  Milton Keynes, Buckinghamshire MK7 7LL (GB)
- **Howard, Fraser Peter**
  Chipping Norton, Oxon OX7 3BT (GB)
- **Kemp, Andrew**
  Oxford, Oxon OX4 2QT (GB)
- **Rai, Bobby**
  Kettering, Northants NN15 5BP (GB)

(74) Representative:
Robinson, Nigel Alexander Julian
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Centrally managed virus scanning system

(57) A malware detecting system is described in which a client computer 8 sends a file access clearance request to an assessment computer 24 to determine whether or not access is permitted to that file. In this way, the different client computers on a network 2 may share their scan results and centralised control may be provided over access permissions to individual files or groups of files.

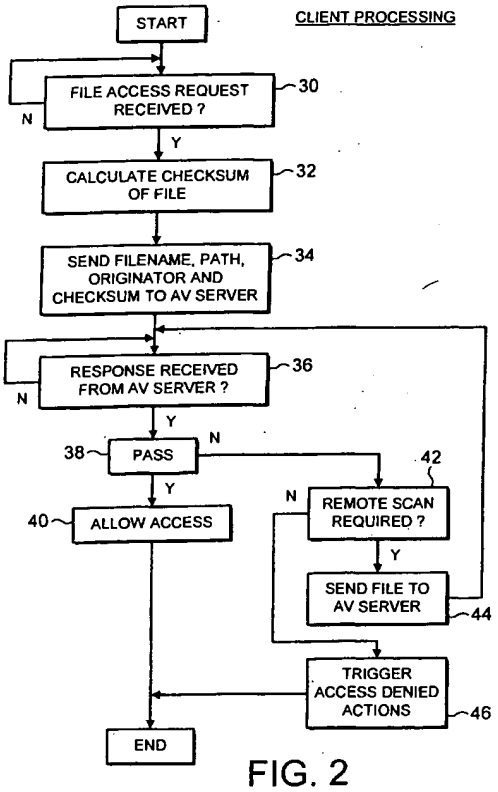


FIG. 2

EP 1 280 040 A2

Printed by Jouve, 75001 PARIS (FR)

Description

[0001] This invention relates to the field of data processing systems. More particularly, this invention relates to the field of malware scanning, such as, for example, scanning for computer viruses, Trojans, worms, banned computer files or E-mails containing banned words or content.

[0002] It is known to provide malware scanning systems and mechanisms for identifying malware within a computer file to be accessed. As the number of computer viruses and the like that are present in the wild increases the processing associated with scanning a computer file to identify the presence of any of those viruses similarly increases. This increase in the processing required is disadvantageous. Furthermore, with the increasing levels of network connectivity between computer systems and the use of E-mail and other fast messaging systems, the spread of computer viruses has become increasingly rapid. In a number of recent cases mass mailing viruses have spread at such a speed that considerable damage has been caused before appropriate countermeasures have been able to be put in place. The delay in deploying such countermeasures is further increased by the need in many cases to update virus definition data on every individual node computer to be protected. The speed of spread of recent viruses is such that the download and installation delays associated with installing such new countermeasures, even when they have been developed and are available, is a significant disadvantage.

[0003] An anti-virus system produced by Sophos uses a locally held record of previously conducted on-access scans on the individual computer in question to determine whether or not a scan should be conducted for a computer file or a previously determined result used instead when this is available.

[0004] Viewed from one aspect the present invention provides a computer program product for controlling a computer to detect malware, said computer program product comprising:

detecting logic operable to detect a file access request to a computer file by a requesting computer;
file access clearance request generating logic operable to generate a file access clearance request including data identifying said computer file;
file access clearance request transmitting logic operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
file access clearance response receiving logic operable to receive at said requesting computer a file access clearance response from said assessment computer; and
file access permitting logic operable if said file access clearance response indicates said computer file does not contain malware, to permit said file access request by said requesting computer.

[0005] The invention recognises that an individual computer requiring a file access may pass the task of determining whether or not that access should be allowed to a different assessment computer. Whilst at first sight this may seem that it would slow down the file access, in practice there are considerable advantages. For example, the assessment computer to which the task is passed may be more rapidly updated with new virus definition data as this is released than would be possible for the requesting computer. Accordingly, the requesting computer can benefit from the most up-to-date virus definition data more rapidly. Furthermore, the invention is particularly well suited to systems in which a plurality of requesting computers share an assessment computer since in many cases the individual requesting computers will show a high degree of correlation in the computer files to which they are requesting access and for which a malware scan is needed. Thus, rather than each requesting computer individually scanning the computer files that are also being scanned by a large number of other computers, (e.g. the computer files associated with the operating system shared by all the requesting computers), these computer files can instead be scanned once by the assessment computer and then each requesting computer can check that the computer file has been scanned and cleared by sending an appropriate request to the assessment computer rather than having to scan the file itself. Thus, in exchange for the use of a small amount of bandwidth on the connections between the requesting computers and the assessment computer, a considerable processing burden may be lifted from the requesting computers. The additional processing burden on the assessment computer does not increase disproportionately since the high degree of correlation between the computer files accessed by the different requesting computers means that in many cases the assessment computer will be able to respond to a clearance request from a requesting computer on the basis that it has already scanned that file and without the need to rescan that file.

[0006] In order to identify the computer file to be scanned in a secure manner to the assessment computer, the data identifying the computer file preferably includes a checksum value. Checksum values may be made sufficiently specific for a computer file from which they have been calculated so as to be difficult to bypass as a way of uniquely identifying a computer file.

[0007] Additional data that is highly useful to the assessment computer in managing the requests it receives includes the filename of the computer file, data identifying the requesting computer and the storage location of the computer file. This provides useful audit information as well as providing for the possibility of files having the same name and same storage location on different computers in fact being different files with dif-

ferent checksums, both of which should be treated as separate entities when determining whether or not a request from another requesting computer relates to a file that has already scanned.

[0008] In circumstances when a requesting computer wishes to access a file that has not already been scanned, the assessment computer may send a scan request message back to the requesting computer such that the requesting computer may send a copy of the file to the assessment computer for scanning. It will be appreciated that this may be slower than scanning the file locally, but the benefit will be that should a different requesting computer later wish to access that same file, then the scanning result produced by this action can be shared with that other computer, so speeding the operation of that other computer.

[0009] If the assessment computer determines that access should be denied to a computer file, then this may be used to trigger denied access actions within the requesting computer, the assessment computer, or elsewhere. Such denied access actions may include deletion of the computer file, repair of the computer file, quarantining of the computer file, generation of user warning messages, generation of administrator warning messages and the like.

[0010] Viewed from another aspect the invention also provides a computer program product for controlling a computer to detect malware, said computer program product comprising:

file access request receiving logic operable to receive at an assessment computer a file access clearance request from a requesting computer, said file access clearance request including data identifying a computer file to be accessed by said requesting computer;

file access clearance response generating logic operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response; and

file access clearance response transmitting logic operable to transmit said file access clearance response to said requesting computer.

[0011] The assessment computer may advantageously store the results of its previous scans within a database. This database may include a variety of fields relating to each computer file that has been scanned. These fields may include an access flag indicating whether access is to be denied to that computer file and a persistence flag indicating whether or not the entry for that computer file should be purged during purge operations.

[0012] The access flag is particularly useful as in addition to allowing the recording that access should be denied due to malware being detected, it also allows the central management of whether or not a particular individual file or class or type of file should be permitted to be accessed by all of those requesting computers that seek their access permissions from that assessment computer. This centralised control is a powerful tool that may be used to implement techniques such as the triggering of a lock down mode of operation in which higher level security provisions are put in place by denying access to certain files or types of files. As an example, if a message was received indicating that the higher security mode should be entered, then the assessment computer may use its mechanisms to deny access to any newly encountered computer file that had not previously been scanned and cleared for use. This would typically allow the large majority of computer activity to continue whilst providing protection against newly released malware threats until the appropriate countermeasures could be put in place.

[0013] The persistence flag allows control of the flushing of entries from the assessment computer. Whilst one of the advantages of the invention is storing the results of previously conducted scans such that they need not be repeated, this has to be tempered by allowing the results to be refreshed at a later time for at least some files. It is possible to envisage that a particular computer file carrying a newly released virus may not be detected as carrying that virus when it is first scanned, but later when a new virus driver is available, that computer file would be detected and blocked. Accordingly, as an example, it may be that all previous scan results could be purged from the system whenever the virus definition data was updated.

[0014] Viewed from a further aspect the invention provides a computer program product for controlling a computer to detect malware, said computer program product comprising:

file access request detecting logic operable to detect a file access request to a computer file by a requesting computer;

file access clearance request generating logic operable to generate a file access clearance request including data identifying said computer file;

file access clearance request transmitting logic operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;

file access clearance request receiving logic operable to receive at said assessment computer said file access clearance request from a requesting computer;

file access clearance response generating logic operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response;

file access clearance response transmitting logic operable to transmit said file access clearance response to said requesting computer;
file access clearance response receiving logic operable to receive at said requesting computer said file access clearance response from said assessment computer; and
file access permitting logic operable if said file access clearance response indicates said computer file does not contain malware to permit said file access request by said requesting computer.

[0015] As well as the complementary aspects of the invention embodied in the form of the client software, the server software and the combination of the client and server software, the invention may also take the form of corresponding methods of malware detection and apparatus for malware detection.

[0016] Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 schematically illustrates a computer network to which the present technique may be applied;

Figure 2 is a flow diagram illustrating an example of the processing that may be performed on a client computer;

Figure 3 is a flow diagram illustrating an example of the processing that may be performed on a server computer;

Figure 4 is a flow diagram illustrating an example process that may be run on a server computer waiting for a lock down signal;

Figure 5 schematically illustrates a database entry relating to a computer file that has previously been scanned; and

Figure 6 is a diagram schematically illustrating the form of a general purpose computer that may be used to implement the above described techniques.

[0017] Figure 1 schematically illustrates a computer network 2 comprising a plurality of client computers 4, 6, 8, 10, 12, 14, 16, 18 connected to respective departmental servers 20, 22 and to a corporate anti-virus (assessment) server 24. The computer network 2 is also connected via a network link to an anti-virus providers FTP server 26 from which virus definition data may be downloaded and from which messages, such as lock down messages or messages relating to the availability of new virus definition data may be received.

[0018] In operation a client computer 8 conducts processing activity during which it seeks to make a file access to a particular computer file (in this example, winword.exe). Rather than scanning this file itself for malware such as computer viruses, Trojans, worms, banned content etc, the client computer 8 instead passes an access clearance request relating to the computer file through to the assessment computer 24 via the departmental server 20. The access clearance request includes the file name of the computer file being accessed, a checksum derived from that file in an effort to uniquely identify it (e.g. an MD5 checksum), data identifying the name of the client computer 8, and the path used by that client computer 8 to access the computer file. The assessment computer 24 receives the access clearance request and uses the data identifying the computer file to look up within an associated database 28 as to whether or not a malware scan has already been conducted for that particular computer file. The computer file is uniquely identified by its filename and checksum value. If the computer file in question has already been scanned, then the result of that scan may be reused rather than conducting the scan again. The assessment computer 24 can accordingly pass back to the client computer 8 a clearance request response indicating whether or not access to that file is permitted or some other further action should be taken. The client computer 8 can then use this access clearance response to either permit access to the file or take the further action specified.

[0019] The further action specified could be that the computer file has already been identified as containing malware and accordingly an appropriate anti-virus or other anti-malware response should be initiated on the client computer 8. A corresponding response could already have been initiated on the assessment computer 24. The denied access responses could take the form of deleting the computer file concerned, repairing the computer file concerned, quarantining the computer file concerned, issuing a user alerting message, issuing an administrator alerting message, issuing a message to an anti-virus provider or some other action.

[0020] A different type of further action that could be required by the client computer 8 in response to the access clearance response arises when the assessment computer 24 has not previously scanned that file and accordingly the client computer 8 should send a copy of that file through to the assessment computer 24 to be scanned. When such a copy of the computer file has been sent through to the assessment computer 24 and scanned, an appropriate pass or denied access action triggering response can be sent back from the assessment computer 24 to the client computer 8.

[0021] It will be appreciated that the client computers 4, 6, 8, 10, 12, 14, 16, 18 operate to share the results of the malware scanning that is performed on their behalf by the assessment computer 24. Since there is likely to be a high degree of correlation between the files being accessed by the different client computers, in many cases the scan that a client computer is request-

ing will already have been performed on behalf of another computer and that scan need not be run again by the assessment computer but instead merely the result of the previous scan returned. The use of a checksum allows the computer files to be reliably uniquely identified.

[0022] The co-ordination of file access permissions by the assessment computer 24 also allows central management of which computer files may be accessed upon the network 2 in a manner that allows the rapid implementation of any higher security level mode should this be desired. As an example, a lock down trigger message may be received from the anti-virus provider FTP server 26 that will trigger the assessment computer 24 to enter a higher security level mode compared to its normal mode of operation. In the higher security level mode, it may be that the access to whole variety of different types of files may be temporarily banned. As an example, the access to VBS files which are often a source of viruses may be banned across the network 2, as could access to e-mail attachments which are another potential source of virus propagation. Thus, when a new threat is identified by the anti-virus provider, lock down messages may be used to trigger predetermined, in accordance with user configuration and preferences, higher security level modes within connected assessment computers 24 to provide a degree of protection for the networks concerned whilst allowing the majority of their normal operations to continue.

[0023] Figure 2 is a flow diagram illustrating the processing performed by a client computer. At step 30 the computer waits to receive a file access request from a computer program executing on that client computer. At step 32 a checksum value is calculated in accordance with one of several different possible checksum algorithms, such as the MD5 algorithm. At step 34 the filename, path, originating computer name and checksum value are sent through to the assessment computer. At step 36 the client computer waits to receive a response from the assessment computer. When that response is received, step 38 serves to determine whether or not the response indicates that the computer file had passed its scanning. If the computer file had passed its scanning and access is to be permitted, then processing proceeds to step 40 at which the access is allowed.

[0024] If the response is not that the computer file had passed its scan, then processing proceeds to step 42 at which a test is made as to whether or not the response indicated that a remote scan was required. If a remote scan was required, then the computer file in question is sent to the assessment computer at step 44 and processing is returned to step 36.

[0025] If the test at step 42 does not identify a remote scan request, then since the response is neither a pass, or a remote scan request, then the computer file must be one to which access is denied and accordingly step 46 serves to trigger the denied access actions. These may include deletion, repair, quarantining or other actions upon the computer file in question as well as the generation of appropriate warning messages to a user or an administrator.

[0026] Figure 3 illustrates the processing performed by the assessment computer. At step 48 the assessment computer waits to receive a request from a client computer. When a request is received, step 50 serves to identify whether a copy of a computer file is being returned to the assessment computer for scanning following a remote scan request that had earlier been issued by the assessment computer. If a copy of a computer file is being returned for scanning, then processing proceeds to step 52 at which the necessary malware scanning is conducted. Step 54 determines whether or not the computer file passed this malware scanning. If the computer file did pass the malware scanning, then step 56 serves to add details of that computer file to the database of scanned files held at the assessment computer relating to scans already performed. Step 58 then returns the pass result to the client computer.

[0027] If the scanning at step 52 was detected as not being passed at step 54, then processing proceeds to step 60 at which malware detected actions are triggered within the assessment computer. These malware detected actions may be similar to those previously described in relation to the client computer file. In addition, an entry specifying the malware scan fail of that computer file may also be added to the database. At step 62 an access denied result is returned to the client computer.

[0028] If the test at step 50 did not indicate that a computer file was being returned for scan, then step 64 serves to compare the file details being passed to the server with the database entries of computer files that have already been scanned. The computer file may be uniquely identified by its filename and its checksum. If at step 66 a match within the database is detected, then step 68 determines whether or not this entry indicates that access should be allowed to that computer file. If access is not to be allowed, then processing proceeds to step 60. If access is allowed, then processing proceeds to step 70 at which a pass result is returned to the client computer.

[0029] If the test at step 66 indicated that a match was not found within the database, then step 72 serves to return a remote scan required result to the client computer in order to trigger the client computer to return a copy of the computer file to the assessment computer for scanning at step 52.

[0030] Figure 4 illustrates a process that may run on the assessment computer as a background task. At step 74 the assessment computer waits to receive a lock down trigger message from the anti-virus provider. If such a message is received, then step 76 serves to activate a lock down mode in the assessment computer. The lock down mode can switch on a user predetermined set of measures intended to provide a higher degree of security, normally at the cost of at least some

functionality. As an example, access to e-mail attachments or VBS files as a class may be denied. These are known to be particular vulnerabilities.

[0031] At step 78 the assessment computer waits for a user command to cancel the lock down mode. When such a command is received, then step 80 serves to cancel the lock down mode.

[0032] Figure 5 schematically illustrates the data that may be stored for a particular computer file within the database of previously conducted scan results held by the assessment computer 24. For an individual computer file, its filename, the originating computer for which the scan of that file was first conducted, the path to the file, the checksum value for the file, an allowed access flag and a persistence flag are all stored. The allowed access flag may be used to indicate whether or not that file passed its scan result. The allowed access flag may also be used as a powerful tool for switching off or on access to individual files or classes of files by an administrator. The persistence flag controls how the entry is flushed from the database on a regular interval, such as when new virus definition data is received. It will be appreciated that other fields could be added to the database relating to the particular file as required.

[0033] Figure 6 schematically illustrates a general purpose computer 200 of the type that may be used to implement the above techniques. The general purpose computer 200 includes a central processing unit 202, a random access memory 204, a read only memory 206, a hard disk drive 208, a display driver 210 and display 212, a user input/output circuit 214 and keyboard 216 and mouse 218 and a network interface unit 220 all connected via a common bus 222. In operation the central processing unit 202 executes program instructions stored within the random access memory 204, the read only memory 206 or the hard disK drive 208. The working memory is provided by the random access memory 204. The program instructions could take a variety of forms depending on the precise nature of the computer 200 and the programming language being used. The results of the processing are displayed to a user upon the display 212 driven by the display driver 210. User inputs for controlling the general purpose computer 200 are received from the keyboard 216 and the mouse 218 via the user input/output circuit 214. Communication with other computers, such as exchanging e-mails, downloading files or providing internet or other network access, is achieved via the network interface unit 220.

[0034] It will be appreciated that the general purpose computer 200 operating under control of a suitable computer program may perform the above described techniques and provide apparatus for performing the various tasks described. The general purpose computer 200 also executes the method described previously. The computer program product could take the form of a recordable medium bearing the computer program, such as a floppy disk, a compact disk or other recordable medium. Alternatively, the computer program could be dynamically downloaded via the network interface unit 220.

[0035] It will be appreciated that the general purpose computer 200 is only one example of the type of computer architecture that may be employed to carry out the above described techniques. Alternative architectures are envisaged and are capable of use with the above described techniques.

Claims

1. A computer program product for controlling a computer to detect malware, said computer program product comprising:

detecting logic operable to detect a file access request to a computer file by a requesting computer;
file access clearance request generating logic operable to generate a file access clearance request including data identifying said computer file;
file access clearance request transmitting logic operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
file access clearance response receiving logic operable to receive at said requesting computer a file access clearance response from said assessment computer; and
file access permitting logic operable if said file access clearance response indicates said computer file does not contain malware, to permit said file access request by said requesting computer.

2. A computer program product as claimed in claim 1, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

3. A computer program product as claimed in claims 1 and 2, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

4. A computer program product as claimed in claims 1, 2 and 3, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then computer file transmitting logic is operable to transmit said computer file from said requesting computer to said assessment computer.

5. A computer program product as claimed in any

one of the preceding claims, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

6. A computer program product as claimed in any one of the preceding claims, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

7. A computer program product for controlling a computer to detect malware, said computer program product comprising:

file access request receiving logic operable to receive at an assessment computer a file access clearance request from a requesting computer, said file access clearance request including data identifying a computer file to be accessed by said requesting computer;
file access clearance response generating logic operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response; and
file access clearance response transmitting logic operable to transmit said file access clearance response to said requesting computer.

8. A computer program product as claimed in claim 7, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

9. A computer program product as claimed in any one of claims 7 and 8, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

10. A computer program product as claimed in any one of claim 7, 8 and 9, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then computer file receiving logic is operable to receive at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

11. A computer program product as claimed in any one of claims 7 to 10, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

12. A computer program product as claimed in any one of claims 7 to 11, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

13. A computer program product as claimed in claim 12, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

14. A computer program product as claimed in any one of claims 7 to 13, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

15. A computer program product as claimed in claim 14, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

16. A computer program product as claimed in any one of claims 7 to 15, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

17. A computer program product for controlling a computer to detect malware, said computer program product comprising:

file access request detecting logic operable to detect a file access request to a computer file by a requesting computer;
file access clearance request generating logic operable to generate a file access clearance request including data identifying said computer file;
file access clearance request transmitting logic operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
file access clearance request receiving logic

operable to receive at said assessment computer said file access clearance request from a requesting computer;
file access clearance response generating logic operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response;
file access clearance response transmitting logic operable to transmit said file access clearance response to said requesting computer;
file access clearance response receiving logic operable to receive at said requesting computer said file access clearance response from said assessment computer; and
file access permitting logic operable if said file access clearance response indicates said computer file does not contain malware to permit said file access request by said requesting computer.

18. A computer program product as claimed in claim 17, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

19. A computer program product as claimed in claims 17 and 18, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

20. A computer program product as claimed in any one of claims 17 to 19, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then computer file transmitting logic is operable to transmit said computer file from said requesting computer to said assessment computer, receiving at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

21. A computer program product as claimed in any one of claims 17 to 20, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

22. A computer program product as claimed in any one of claims 17 to 21, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

23. A computer program product as claimed in any one of claims 17 to 22, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

24. A computer program product as claimed in claim 23, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

25. A computer program product as claimed in any one of claims 17 to 25, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

26. A computer program product as claimed in claim 25, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

28. A computer program product as claimed in any one of claims 17 to 27, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

29. A method of detecting malware, said method comprising the steps of:

detecting a file access request to a computer file by a requesting computer;
generating a file access clearance request including data identifying said computer file;
transmitting said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
receiving at said requesting computer a file access clearance response from said assessment computer; and
if said file access clearance response indicates said computer file does not contain malware, then permitting said file access request by said requesting computer.

30. A method as claimed in claim 29, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

31. A method as claimed in claims 29 and 30, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

32. A method as claimed in any one of claims 29 to 31, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then transmitting said computer file from said requesting computer to said assessment computer.

33. A method as claimed in any one of claims 29 to 31, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

34. A method as claimed in any one of claims 29 to 33, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

35. A method of detecting malware, said method comprising the steps of:

receiving at an assessment computer a file access clearance request from a requesting computer, said file access clearance request including data identifying a computer file to be accessed by said requesting computer;
in dependence upon said data identifying said computer file determining if said computer file has previously been assessed as not containing malware and generating a file access clearance response; and
transmitting said file access clearance response to said requesting computer.

36. A method as claimed in claim 35, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

37. A method as claimed in claims 35 and 36, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

38. A method as claimed in any one of claims 35 to 37, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then receiving at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

39. A method as claimed in any one of claims 35 to 38, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

40. A method as claimed in any one of claims 35 to 39, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

41. A method as claimed in claim 40, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

42. A method as claimed in any one of claims 35 to 41, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

43. A method as claimed in any one of claims 35 to 42, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

44. A method as claimed in any one of claims 35 to 43, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

45. A method of detecting malware, said method comprising the steps of:

detecting a file access request to a computer file by a requesting computer;
generating a file access clearance request including data identifying said computer file;
transmitting said file access clearance request

from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
receiving at said assessment computer said file access clearance request from a requesting computer;
in dependence upon said data identifying said computer file determining if said computer file has previously been assessed as not containing malware and generating a file access clearance response;
transmitting said file access clearance response to said requesting computer;
receiving at said requesting computer said file access clearance response from said assessment computer; and
if said file access clearance response indicates said computer file does not contain malware, then permitting said file access request by said requesting computer.

46. A method as claimed in claim 45, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

47. A method as claimed in claims 45 and 46, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

48. A method as claimed in any one of claims 45 to 47, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then transmitting said computer file from said requesting computer to said assessment computer, receiving at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

49. A method as claimed in any one of claims 45 to 48, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

50. A method as claimed in any one of claims 45 to 49, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

51. A method as claimed in any one of claims 45 to 50, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

52. A method as claimed in claim 51, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

53. A method as claimed in any one of claims 45 to 52, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

54. A method as claimed in claim 53, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

55. A method as claimed in any one of claims 45 to 54, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

56. Apparatus for detecting malware, said apparatus comprising:

a detector operable to detect a file access request to a computer file by a requesting computer;
a file access clearance request generator operable to generate a file access clearance request including data identifying said computer file;
a file access clearance request transmitter operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
a file access clearance response receiver operable to receive at said requesting computer a file access clearance response from said assessment computer; and
a file access permission unit operable if said file access clearance response indicates said computer file does not contain malware, to permit said file access request by said requesting

computer.

57. Apparatus as claimed in claim 56, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

58. Apparatus as claimed in claims 56 and 57, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

59. Apparatus as claimed in any one of claims 56 to 58, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then a computer file transmitter is operable to transmit said computer file from said requesting computer to said assessment computer.

60. Apparatus as claimed in any one of claims 56 to 59, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

61. Apparatus as claimed in any one of claims 56 to 60, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

62. Apparatus for controlling a computer to detect malware, said apparatus comprising:

a file access request receiver operable to receive at an assessment computer a file access clearance request from a requesting computer, said file access clearance request including data identifying a computer file to be accessed by said requesting computer;
a file access clearance response generator operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response; and
a file access clearance response transmitter operable to transmit said file access clearance response to said requesting computer.

63. Apparatus as claimed in claim 62, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

64. Apparatus as claimed in claims 62 and 63, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

65. Apparatus as claimed in any one of claims 62 to 64, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then a computer file receiver is operable to receive at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

66. Apparatus as claimed in any one of claims 62 to 65, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

67. Apparatus as claimed in any one of claims 62 to 66, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

68. Apparatus as claimed in claim 67, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

69. Apparatus as claimed in any one of claims 62 to 68, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

70. Apparatus as claimed in claim 69, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

71. Apparatus as claimed in any one of claims 62 to 70, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied.

72. Apparatus for controlling a computer to detect

malware, said apparatus comprising:

a file access request detector operable to detect a file access request to a computer file by a requesting computer;
a file access clearance request generator operable to generate a file access clearance request including data identifying said computer file;
a file access clearance request transmitter operable to transmit said file access clearance request from said requesting computer to an assessment computer responsible for assessment of whether said computer file contains malware;
a file access clearance request receiver operable to receive at said assessment computer said file access clearance request from a requesting computer;
a file access clearance response generator operable in dependence upon said data identifying said computer file to determine if said computer file has previously been assessed as not containing malware and to generate a file access clearance response;
a file access clearance response transmitter operable to transmit said file access clearance response to said requesting computer;
a file access clearance response receiver operable to receive at said requesting computer said file access clearance response from said assessment computer; and
a file access permission unit operable if said file access clearance response indicates said computer file does not contain malware to permit said file access request by said requesting computer.

73. Apparatus as claimed in claim 72, wherein said data identifying said computer file includes a checksum value calculated from said computer file.

74. Apparatus as claimed in claims 72 and 73, wherein said data identifying said computer file includes one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file.

75. Apparatus as claimed in any one of claims 72 to 74, wherein if said file access clearance response indicates a scan of said computer file is required by said assessment computer, then a computer file transmitter is operable to transmit said computer file from said requesting computer to said assessment computer, receiving at said assessment computer said computer file from said requesting computer and performing a malware scan of said computer file.

76. Apparatus as claimed in any one of claims 72 to 75, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said assessment computer.

77. Apparatus as claimed in any one of claims 72 to 76, wherein if said file access clearance response indicates access to said computer file is denied, then triggering a denied access response in said requesting computer.

78. Apparatus as claimed in any one of claims 72 to 77, wherein said assessment computer stores a database of computer files previously assessed as to whether they contain malware.

79. Apparatus as claimed in claim 78, wherein said database includes for each computer file fields specifying one or more of a filename of said computer file, data identifying said requesting computer and a storage location of said computer file, a checksum value calculated from said computer file, an access flag indicating whether access to said computer file is denied and a persistence flag indicating whether entries relating to said computer file should be purged from said database during purge operations.

80. Apparatus as claimed in any one of claims 72 to 79, wherein said assessment computer is operable in at least a higher level security mode and a lower level security mode, said assessment computer serving to deny access to greater range of computer files when operating in said higher level security mode compared with said lower level security mode.

81. Apparatus as claimed in claim 80, wherein said assessment computer is triggered to change from said lower level security mode to said higher level security mode by a lock down trigger message received at said assessment computer from a remote computer.

82. Apparatus as claimed in any one of claims 72 to 81, wherein a plurality of requesting computers share access to an assessment computer for determining whether file access requests by those requesting computers should be denied

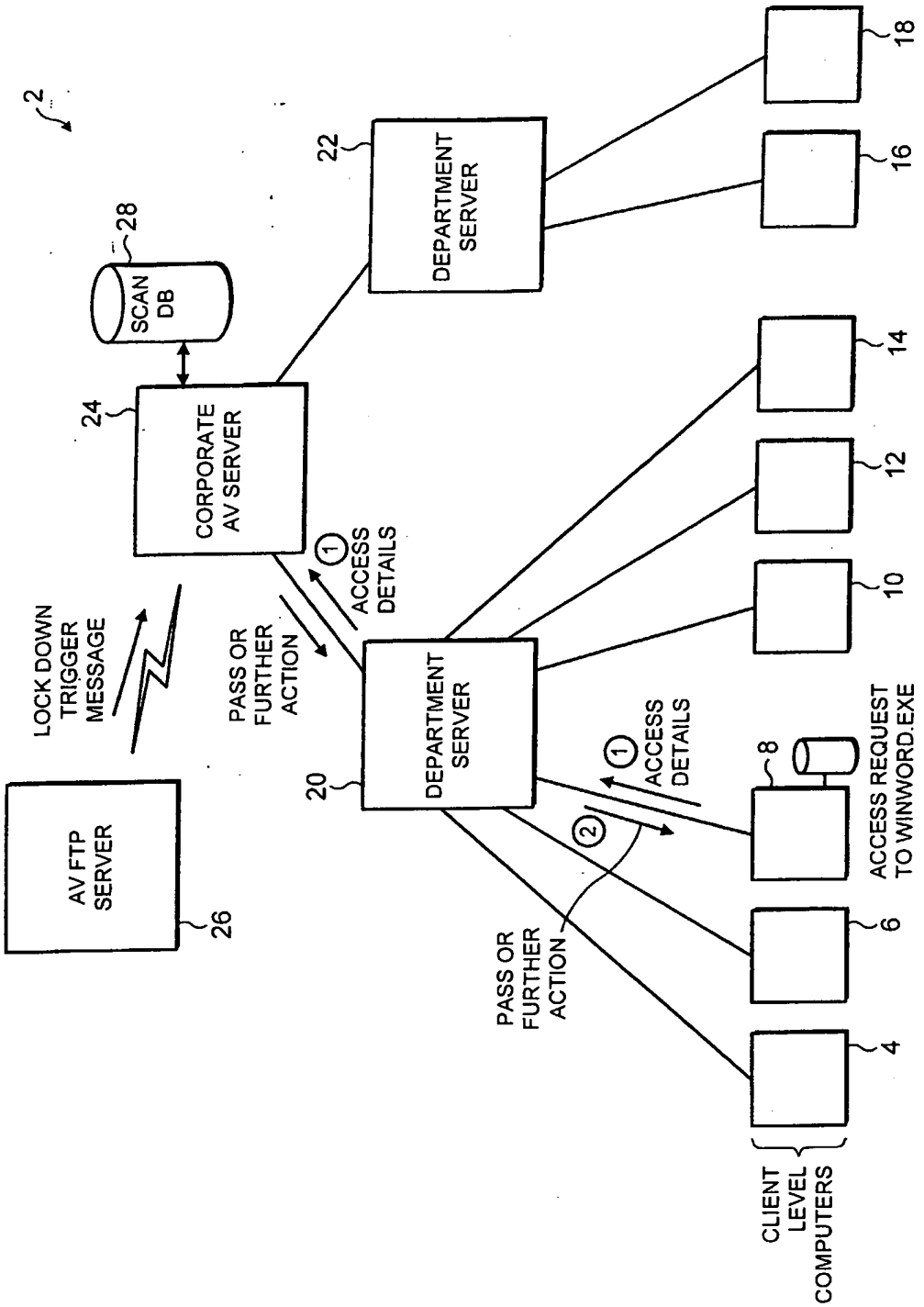

2
22
DEPARTMENT SERVER
18
16
28
SCAN DB
24
CORPORATE AV SERVER
14
12
10
1
ACCESS DETAILS
LOCK DOWN TRIGGER MESSAGE
PASS OR FURTHER ACTION
20
DEPARTMENT SERVER
1
ACCESS DETAILS
8
ACCESS REQUEST TO WINWORD.EXE
2
AV FTP SERVER
26
PASS OR FURTHER ACTION
6
4
CLIENT LEVEL COMPUTERS

FIG. 1

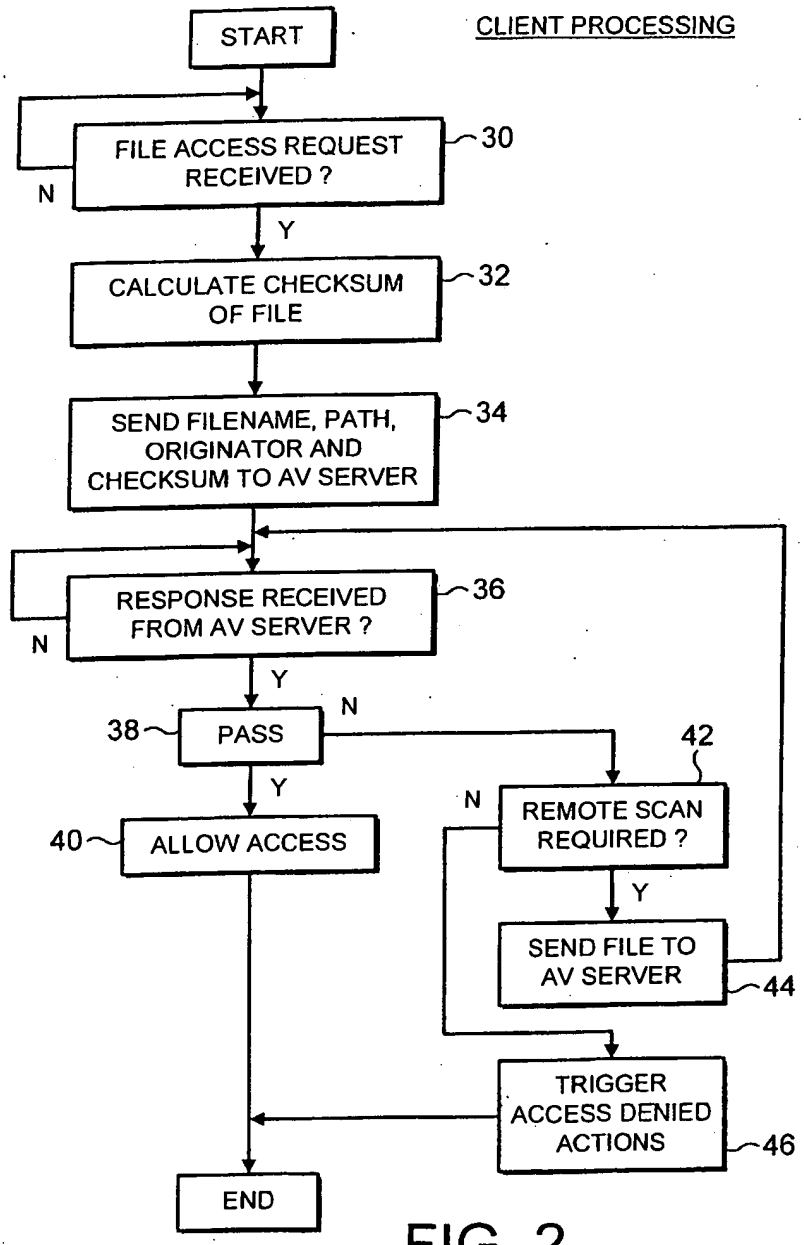

START
CLIENT PROCESSING
FILE ACCESS REQUEST RECEIVED ?
30
N
Y
CALCULATE CHECKSUM OF FILE
32
SEND FILENAME, PATH, ORIGINATOR AND CHECKSUM TO AV SERVER
34
RESPONSE RECEIVED FROM AV SERVER ?
36
N
Y
38
PASS
N
42
Y
N
REMOTE SCAN REQUIRED ?
40
ALLOW ACCESS
Y
SEND FILE TO AV SERVER
44
TRIGGER ACCESS DENIED ACTIONS
46
END

FIG. 2

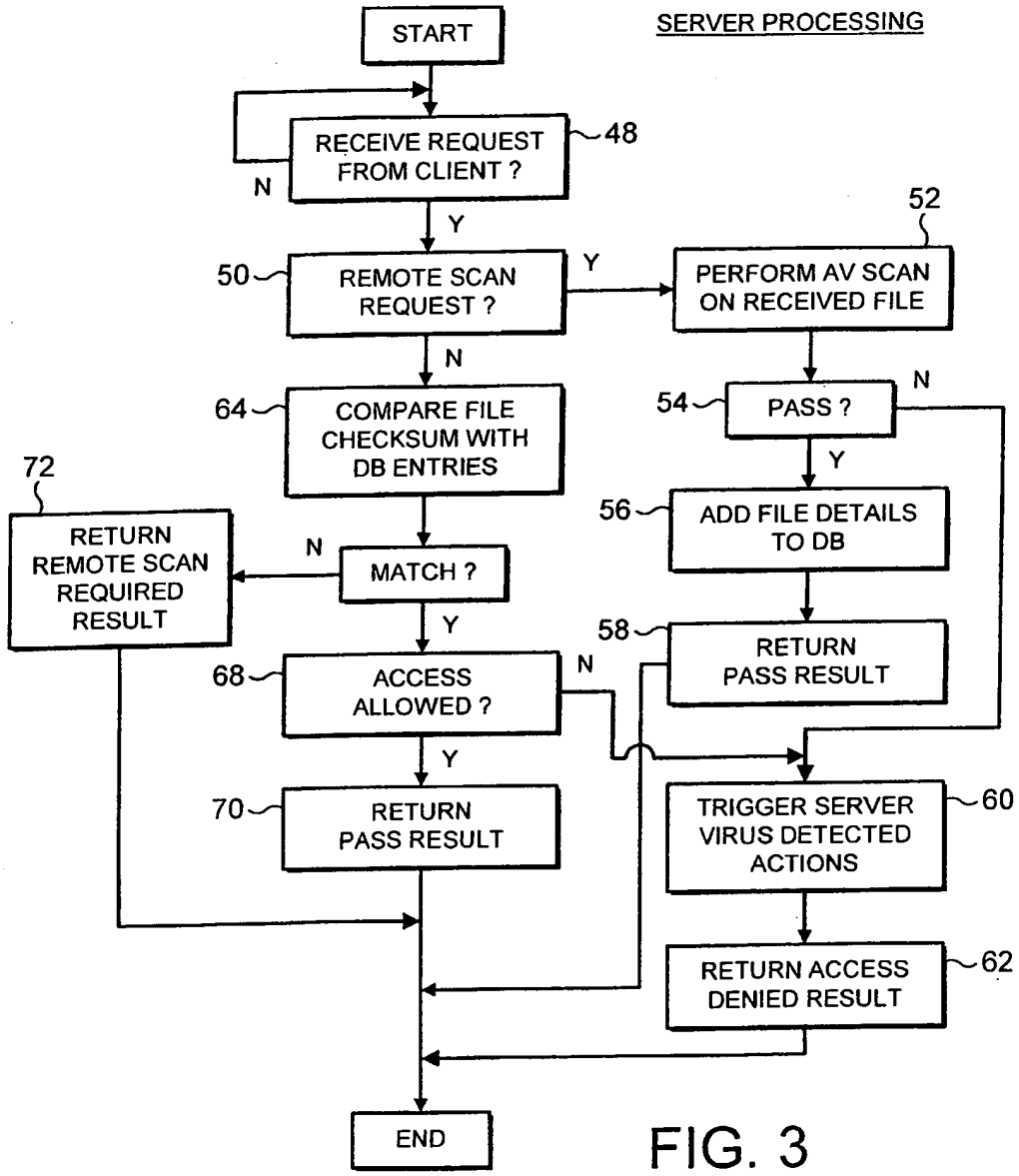

SERVER PROCESSING
START
RECEIVE REQUEST FROM CLIENT ?
48
N
Y
50
REMOTE SCAN REQUEST ?
Y
52
PERFORM AV SCAN ON RECEIVED FILE
N
64
COMPARE FILE CHECKSUM WITH DB ENTRIES
54
PASS ?
N
Y
72
RETURN REMOTE SCAN REQUIRED RESULT
N
MATCH ?
56
ADD FILE DETAILS TO DB
Y
58
RETURN PASS RESULT
68
ACCESS ALLOWED ?
N
Y
70
RETURN PASS RESULT
TRIGGER SERVER VIRUS DETECTED ACTIONS
60
RETURN ACCESS DENIED RESULT
62
END

FIG. 3

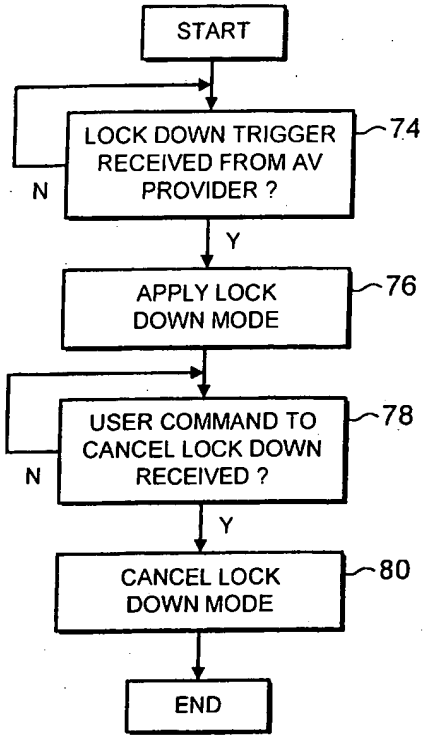

START
LOCK DOWN TRIGGER RECEIVED FROM AV PROVIDER ?
74
N
Y
APPLY LOCK DOWN MODE
76
USER COMMAND TO CANCEL LOCK DOWN RECEIVED ?
78
N
Y
CANCEL LOCK DOWN MODE
80
END

FIG. 4

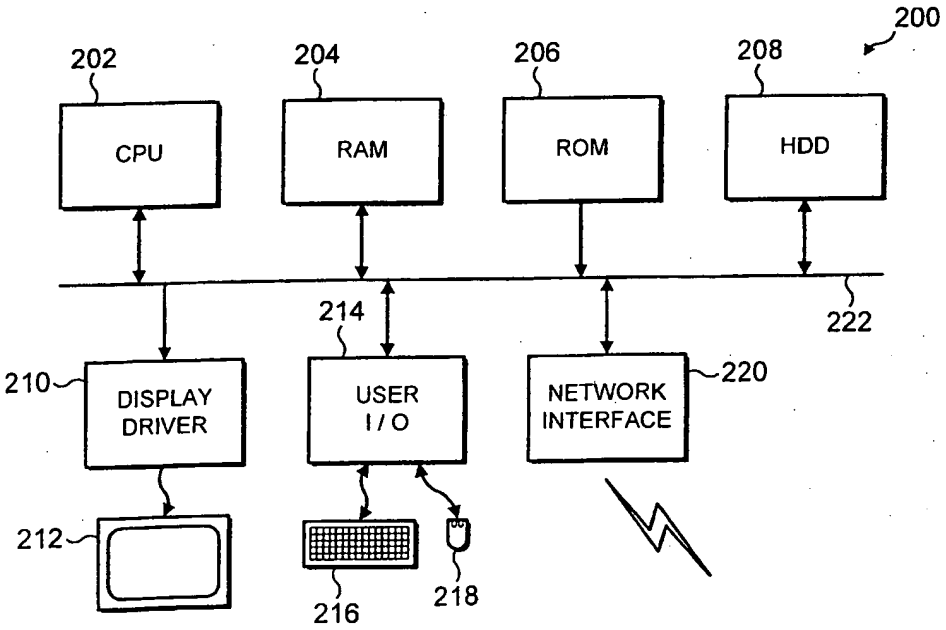

200
202
CPU
204
RAM
206
ROM
208
HDD
222
214
210
DISPLAY DRIVER
USER I / O
NETWORK INTERFACE
220
212
216
218

FIG. 6

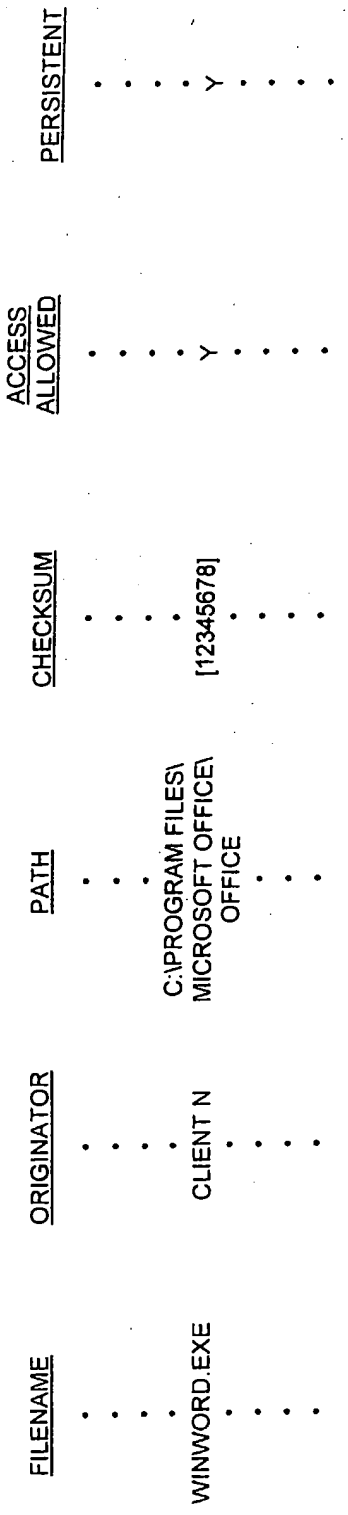

FILENAME
ORIGINATOR
PATH
CHECKSUM
ACCESS ALLOWED
PERSISTENT
WINWORD.EXE
CLIENT N
C:\PROGRAM FILES\ MICROSOFT OFFICE\ OFFICE
[12345678]
Y
Y

FIG. 5

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 280 040 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
24.03.2004 Bulletin 2004/13

(43) Date of publication A2:
29.01.2003 Bulletin 2003/05

(21) Application number: **02253235.2**

(22) Date of filing: **09.05.2002**

(51) Int Cl.[7]: **G06F 1/00**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **26.07.2001 US 912392**

(71) Applicant: **Networks Associates Technology, Inc.**
Santa Clara, CA 95054 (US)

(72) Inventors:
- **Hinchliffe, Alexander James**
  Milton Keynes, Buckinghamshire MK7 7LL (GB)
- **Howard, Fraser Peter**
  Chipping Norton, Oxon OX7 3BT (GB)
- **Kemp, Andrew**
  Oxford, Oxon OX4 2QT (GB)
- **Rai, Bobby**
  Kettering, Northants NN15 5BP (GB)

(74) Representative:
Robinson, Nigel Alexander Julian
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) Centrally managed virus scanning system

(57) A malware detecting system is described in which a client computer 8 sends a file access clearance request to an assessment computer 24 to determine whether or not access is permitted to that file. In this way, the different client computers on a network 2 may share their scan results and centralised control may be provided over access permissions to individual files or groups of files.

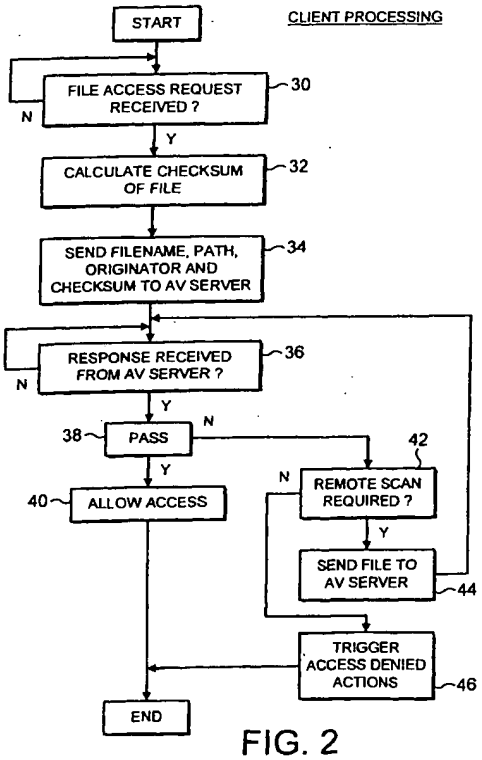


FIG. 2

EP 1 280 040 A3

Printed by Jouve, 75001 PARIS (FR)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 25 3235

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	US 6 195 587 B1 (HRUSKA JAN ET AL) 27 February 2001 (2001-02-27) * abstract * * column 1, line 1-9; figure 3 * * column 2, line 15-65; figures 1,2 * * column 3, line 5-25,40-67 * * column 5, line 1-60 * * column 4 * ---	1-13, 16-24, 28-41, 44-52, 55-68, 71-79,82	G06F1/00 G06F21/00
X	US 5 960 170 A (LAU STEVEN YUEN-LAM ET AL) 28 September 1999 (1999-09-28) * abstract * * column 1-10; figures 1-8 * * column 11, line 1-55 * * column 12-27 * ---	1,7,17, 29,35, 45,56, 62,72	
X	EP 1 111 507 A (SECURE OYJ F) 27 June 2001 (2001-06-27) * the whole document * --- -/--	1,7,17, 29,35, 45,56, 62,72	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	26 January 2004	Nazzaro, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 25 3235

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	MARMELSTEIN R E ET AL: "A distributed architecture for an adaptive computer virus immune system" SYSTEMS, MAN, AND CYBERNETICS, 1998. 1998 IEEE INTERNATIONAL CONFERENCE ON SAN DIEGO, CA, USA 11-14 OCT. 1998, NEW YORK, NY, USA,IEEE, US, 11 October 1998 (1998-10-11), pages 3838-3843, XP010310923 ISBN: 0-7803-4778-1 * abstract; figure 1 * * page 3841 * * page 3842 * -----	14,15, 25,26, 42,43, 53,54, 69,70, 80,81	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	26 January 2004	Nazzaro, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

European Patent Office

Application Number
EP 02 25 3235

CLAIMS INCURRING FEES

The present European patent application comprised at the time of filing more than ten claims.

☐ Only part of the claims have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims and for those claims for which claims fees have been paid, namely claim(s):

☐ No claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims.

LACK OF UNITY OF INVENTION

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

see sheet B

☒ All further search fees have been paid within the fixed time limit. The present European search report has been drawn up for all claims.

☐ As all searchable claims could be searched without effort justifying an additional fee, the Search Division did not invite payment of any additional fee.

☐ Only part of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the inventions in respect of which search fees have been paid, namely claims:

☐ None of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the invention first mentioned in the claims, namely claims:

European Patent Office

LACK OF UNITY OF INVENTION
SHEET B

Application Number
EP 02 25 3235

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

1. Claims: 1-13,16-24,28-41,44-52,55-68,71-79,82

Computer program , method and apparatus for detecting malware, comprising: detecting logic, file access clearance request generating and transmitting logic, file access clearance receiving logic at an assessment computer, file access clearance response generating and transmitting logic, file access clearance response receiving logic and file access permitting logic if file access clearance response indicates said file does not contain malware.

2. Claims: 14-15, 25-26, 42-43, 53-54, 69-70, 80-81

Computer program method and apparatus for detecting malware comprising detecting logic, file access clearance request generating and transmitting logic, file access clearance receiving logic at an assessment computer, file access clearance response generating and transmitting logic, file access clearance response receiving logic and file access permitting logic if file access clearance response indicates said file does not contain malware. The assessment computer is operable in a higher level security mode serving to deny access to greater range of computer files. The change from lower to higher level security mode is triggered from a remote computer.

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 02 25 3235

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

26-01-2004

Patent document cited in search report		Publication date		Patent family member(s)	Publication date
US 6195587	B1	27-02-2001	GB	2283341 A	03-05-1995
			AU	8000394 A	22-05-1995
			DE	69407812 D1	12-02-1998
			DE	69407812 T2	16-07-1998
			EP	0725951 A1	14-08-1996
			WO	9512162 A1	04-05-1995
			JP	9504395 T	28-04-1997
US 5960170	A	28-09-1999	AU	6537698 A	12-10-1998
			EP	0968466 A1	05-01-2000
			JP	2001515625 T	18-09-2001
			WO	9841919 A1	24-09-1998
EP 1111507	A	27-06-2001	GB	2353372 A	21-02-2001
			EP	1111507 A2	27-06-2001
			US	2001005889 A1	28-06-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 549 012 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.06.2005 Bulletin 2005/26

(51) Int Cl.[7]: **H04L 29/06**, G06F 1/00

(21) Application number: **03447310.8**

(22) Date of filing: **24.12.2003**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PT RO SE SI SK TR
Designated Extension States:
AL LT LV MK

(71) Applicant: **DataCenterTechnologies N.V.**
9080 Lochristi (BE)

(72) Inventor: **De Spiegeleer, Kristof**
8301 Knokke-Heist (BE)

(74) Representative: **Bird, William Edward et al**
Bird Goen & Co.,
Klein Dalenstraat 42A
3020 Winksele (BE)

(54) Method and system for identifying the content of files in a network

(57) A method and system for performing securing and controlling of a network using content identification of files in a network having a central infrastructure and local computing devices is presented. The method comprises calculating a hash value of a new file created or received on a local computing device, transmitting the hash value to the central infrastructure, comparing the hash value with a previously determined hash value stored in a database on the central infrastructure to determine whether the file is new to the network and if the file is new to the network, checking the file content with a content identifying engine, installed and updated on the central infrastructure. Content attributes are determined for the files which allow to perform appropriate actions on the local computing devices according to policy rules.

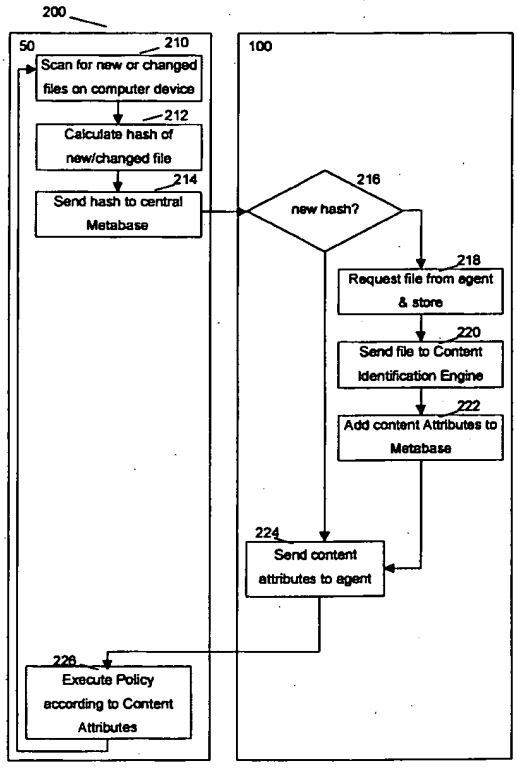


Fig. 3

EP 1 549 012 A1

Printed by Jouve, 75001 PARIS (FR)

Description

Technical field of the invention

[0001] The invention relates to a method and system to control the content of computer files, e.g. containing text or graphical data. More specifically, a method and system is described for checking and managing the security status and the content of computer files on a local computing device in a network environment.

Background of the invention

[0002] In today's world, computers are widely spread. Very often, especially in business environment, they are interconnected in small or larger networks. As software and data often are an important part of the investment goods of both private persons and firms, it is important to protect single computing devices and complete networks and their workstations against attacks from viruses, trojan horses, worms and malicious software. Another problem is related to the amount of files containing undesirable content such as explicit adult content. These files are often received on local computing devices uninvited and unwanted.

[0003] To solve the security problems associated with viruses, virus protection systems, also called virus checkers, have been developed. Some examples of conventional virus checkers are Norton AntiVirus, McAfee VirusScan, PC-cillin, Kaspersky Anti-Virus. Most of these conventional virus protection software packages can be configured so that they are continuously running in the background of the computing device and providing continuous protection. These virus protection systems compare codes of new or amended software with fingerprints (e.g. parts of code introduced in files by the viruses) of well known viruses. Other virus protection systems compare codes of all data available on the computing device. This leads to the use of a significant amount of central processing unit (CPU) time, which limits the capacity of the computing device for performing other tasks. Furthermore, the working principle of these virus checkers makes these software packages work rather reactive than proactive, as the fingerprint of the virus needs to be known in order for a virus scanning program to recognise it. This implies that the database of fingerprints needs to be updated very regularly in order to be secured against relatively new viruses. Consequently, the secure state of the computer is not only depending on external factors like the accurateness with which fingerprints of new viruses are made available by the suppliers of virus protection software packages, but also on the sense of duty of the user regarding performing updates regularly. If updates are provided centrally from a server automatically, then network capacity is reduced as these virus updates must be sent to each workstation.

[0004] In a network environment, the problem of updating such a database of fingerprints becomes significantly more important, as it implies that the responsibility is put to all users, who all have to update their virus checker database. Alternatively, the virus scanning could be performed by a central server, thus limiting the updating for new fingerprints to the central server. Nevertheless this implies that a large amount of data needs to be transferred over the network on a regular basis thereby utilising large amounts of expensive network bandwidth and possibly (depending on the number of clients for the server) overloading the network or server capacity for other activities.

[0005] In order to limit the amount of CPU time used, additional techniques have been developed to speed up the virus scanning process. These very often include hashing of the content of files. Hashing is one example of application of a "one-way-function". A one-way-function is an algorithm which when applied in one direction makes the reverse direction almost impossible to perform. A one-way-function generates a value such as a hash value by a calculation on the content of a file and can uniquely fingerprint this file if the one-way-function is complex enough to avoid duplicate values from different files. The uniqueness of a hashing function depends on the type of hashing function that is used, i.e. the size of the digest that is formed and the quality of function. Good hashing functions have the fewest collisions in a table, i.e. the chance of providing the same hash value for different files is the smallest. As mentioned, this is also determined by the size of the digest, i.e. hash value, that is calculated. If e.g. a 128-bit digest is used, the number of possible different values that can be obtained is 2^{128}.

[0006] It is known to use hashing for virus checking, possibly in a network environment. Typically, a hash of an application selected to run on a local computer is calculated, a stored hash from a database on a secured computer is retrieved on the local computer, whereby the secured computer can be a secured part of the local computer or a network server, and both values are compared. If there is a match, the application is executed, if there is no match, a security action is performed. This security action comprises loading a virus scanner on the local computer. It may also comprise alerting the network administrator. Furthermore it is also known to use this for differentiating accessibility to software from different workstations and as a way of checking whether software is licensed.

[0007] It is also known to use hashing in a method of identifying rogue software on a computer system or device. The method typically is applicable in a network environment. A hash value of a software application to be executed is calculated, this hash value is transferred to a server and compared with previously stored values. One of the essential features is that the method uses a database on a server, the server being a server with a large number of clients. The database on the server thereby is built up by adding information by different cli-

ents so that most software applications and their corresponding fingerprints are already stored in the database. The database is built up by checking software applications on authenticity with the owners of the application. If this is not possible, the system is also able to give a heuristic result, evaluating the occurrence of this application on local computers from other clients.

[0008] Methods for sending an electronic file by electronic mail, i.e. e-mail, including a file content and message content identifier are known. Depending on the message content identification, the message is delivered to a customer or not. The method may be used to organise e-mail delivery, but it has the disadvantage of being focussed on e-mail delivery and it does not allow to secure all files in a network.

[0009] Monitoring of electronic mail messages, to protect a computer system for protection against virus attacks and unsolicited commercial e-mail (UCE) is also known. Such a system is preferably installed on a mail server or an Internet Service Provider and checks specific parts of e-mails by calculating a digest and comparing it with stored digest values of e-mails previously received. In this way it is determined whether the e-mail has an approved digest or whether the e-mail is UCE or contains a e-mail worm. The system has the disadvantage that it is focussed on e-mail viruses and SPAM and that it does not allow to check all data files or executable files which are possibly infected, e.g. by files copied from external memory storage means like floppy disks or CD-ROMs or by e.g. Trojan horses.

[0010] Controlling the execution of software on different workstations according to certain policy rules by a network server is known, whereby an improved computer security system is obtained, by classifying software. It is suggested that this classification can be based on several forms of data one of which is e.g. the hash values of software data. This typically is performed by the calculation of hash values of a program if it is selected for loading and execution, and comparison of the hash value with a trusted value to determine the rule of execution. The classification also may be based on a hash of the content, a digital signature, the file system or network path or the URL zone.

[0011] The above mentioned methods and systems describe the use of hashing functions to check whether software applications are authentic or to regulate the execution of software applications. Nevertheless the problem of virus scanning all new files in a network using a conventional virus scanner whereby the necessity of updating the database of fingerprints of a conventional virus scanner on every local computer is limited is not discussed.

Summary of the invention

[0012] It is an object of the present invention to provide a system and a method for identifying the content of files new on local computing devices in a network while limiting the necessity of updating the information needed by a content identifier on every local computing device.

[0013] It is an advantage of the present invention that, if the invention is used as a virus checker, the security level is further increased as the database of fingerprints of a conventional virus scanner does not have to be updated on every local computing device.

[0014] It is a specific advantage of the present invention that the content of a file new to a network is only identified once for the whole network.

[0015] It is furthermore a specific advantage of the present invention that the total processor (CPU) processing time in the network and the amount of network traffic is reduced.

[0016] The above described object and advantages are obtained with a method and system of content identification in a network according to the present invention.

[0017] The method for identifying the content of a data file in a network environment is used for a network having at least one local computing device linked to a remaining part of the network environment including a central infrastructure. The method and system comprises calculating a reference value for a new file on one of said at least one local computing devices using a one-way-function, transmitting said calculated reference value to said central infrastructure, comparing said calculated reference value with reference values previously stored within the remaining part of the network environment.

[0018] The method further comprises, after comparing, deciding that the content of the new file is already identified if a match between said calculated reference value and a previously stored reference value is found and retrieving the corresponding content attributes; or deciding that the content of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure identifying the content of said new file and determining content attributes corresponding with the content of the new file and storing a copy of said content attributes, after deciding, triggering an action on said local computing device in accordance with said content attributes.

[0019] In the method for identifying the content of a data file in a network environment, the reference value may be a hash value. The reference values previously stored may be stored within the central infrastructure. In the method and system for identifying the content of a data file in a network environment, identifying the content of the new file may comprise scanning the new file for viruses using an anti-virus checker means on a central infrastructure.

[0020] The method may furthermore comprise transferring the new file from the local computing device to the central infrastructure before said identifying the content of said new file is performed. Furthermore it may comprise storing a copy of the new file on the central

infrastructure after transferring the new file from the local computing device to the central infrastructure. An address of where the file is stored may be stored together with the hash value, as to be able to quickly track copies of the files stored on the central infrastructure.

[0021] In the method of the present invention, triggering an action on the local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with a copy of a previous version of said new file. Furthermore, triggering an action on the local computing device in accordance with said content attributes may also comprise replacement of the new file on the local computing device with another version of said new file restored from the remaining part of the network environment.

[0022] The method of the present invention furthermore may comprise sharing the new file on the local computing device to the central infrastructure before said identifying the content of said new file is performed and whereby said identifying the content of said new file is performed by remotely identifying the content over the network environment. The method may comprise checking the functioning of the local agent on the local computing device.

[0023] Furthermore, triggering an action on the local computing device may be performed after transmitting the content attributes corresponding to the new file to the local computing device.

[0024] In the method for identifying the content of a data file in a network environment, identifying the content of the new file may comprise one or more of the group of scanning for adult content, scanning for Self Promotional Advertising Messages or Unsolicited Commercial E-mail (UCE) and scanning for copyrighted information. The method may further relate to a method and system for providing a content firewall, whereby one local computing device is connected to the external network, which may e.g. be the internet, and the one local computing device is also connected to the network environment formed by the remaining local computing devices. The one local computing device thus links the network environment with an external network and is the only computing device that is directly connected to sources external from the network environment. The local computing device thus acts as a content firewall as to protect the network environment from attacks originating from places in the external network. The local computing device may act as a content firewall working in a promiscuous way, i.e. whereby the local computing device acts as a content firewall that sees all traffic passing by, executes the hashing and comparing functions and contacts the agents to enforce a policy.

[0025] The method may be specifically related to a method for checking the security status of a network and its components. In this embodiment, a method for determining the security status of a data file in a network environment is used in a network having at least one local computing device linked to a remaining part of the network environment including a central infrastructure. The method comprises calculating a reference value for a new file on one of the at least one local computing devices using a one-way-function, transmitting said calculated reference value to said central infrastructure, comparing said calculated reference value with reference values previously stored within the remaining part of the network environment and after comparing, deciding that the security status of the file has already been checked if a match between the calculated reference value and a previously stored reference value is found and retrieving the corresponding security status; or deciding that the security status of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure checking the security status of the new file and determining the security status corresponding with the new file and storing a copy of the security status, followed by
after deciding, triggering an action on said local computing device in accordance with the security status of the new file. This action may be e.g. making the file inaccessible for the user of the local computing device and for other users in the network or restoring the infected file.

[0026] The methods described above may be triggered by an action performed on the local agent. The triggering by an action performed on the local agent may be e.g. running an application or opening a file.

[0027] The invention also relates to a method for altering a system for identifying the content of a file in a network environment according to the systems described above, the network environment comprising means for calculating a one-way function, at least one local computing device linked to a remaining part of the network environment including a central infrastructure and means for identifying the content and, the method comprising altering said means for identifying the content or said means for calculating a one-way function, scanning the remaining part of the network environment for reference values calculated with a one-way function and for each of the reference values, requesting a file that corresponds with said reference value from said network environment, sending the file to means for identifying the content, identifying the content of said file and determining content attributes corresponding with the content of the file and storing a copy of said content attributes, sending the content attributes to every local computing device containing the file and after sending; triggering an action on said local computing device in accordance with said content attributes. Altering of a system for identifying the content of a file in a network environment may be triggered by the introduction of a new one-way function to calculate reference values or may be also triggered by the updating of the means for identifying the content of the files. The method may be used to scan only part of the hashing keys in the remain-

ing part of the network environment, e.g. hashing keys of files of which the content is identified after a certain date, as to minimise the actions to be performed. The date of the previous content identification may be retrieved from the content attributes.

[0028] The reference values may be hashing values. The means for identifying the content may be an anti-virus checker means, a means for scanning for adult content, a means for scanning for Self Promotional Advertising Messages or a means for scanning for copyrighted information. Triggering an action on the local computing device in accordance with said content attributes may comprise replacement of the file on the local computing device with another version of the file restored from the remaining part of the network environment or may comprise replacement of a file with a copy of a previous version of the file or may comprise putting the file in quarantine or removing the file.

[0029] The invention is also related to a computer program product for executing any of the above described methods, when executed on a network. The invention furthermore relates to a system for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part the network environment which includes a central infrastructure and, said remaining part including a stored database, whereby the system comprises means for calculating a reference value for a new file on said local computing device using a one-way-function, means for transmitting said calculated reference value to said central infrastructure and means for comparing said calculated reference value with previously stored reference values from the database. The system furthermore comprises means for deciding whether the content of the new file is already identified based on comparison of said calculated reference value and reference values previously stored within the remaining part, means located on the central infrastructure, for identifying the content of the new file and as to assign content attributes if the new file has not been identified yet and means for storing said content attributes within the remaining part, and means for triggering an action on said local computing device in accordance with content attributes for said new file.

[0030] In the system according to the present invention, the means for identifying the content of a file may comprise an anti-virus checker means on said central infrastructure. Furthermore, means for storing a copy of the new file within the remaining part. The means for identifying the content of a file may comprise one or more of the group of means for scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

[0031] The invention may also relate to a machine readable data storage device, storing the computer program product for executing any of the above described methods, when executed on a network. Furthermore, the invention may also relate to the transmission of the computer program product for executing any of the above described methods.

[0032] Although there has been constant improvement, change and evolution of methods of virus scanning and content identification of data files, the present concepts are believed to represent substantial new and novel improvements, including departures from prior practices, resulting in the provision of more efficient, stable and reliable methods of this nature.

[0033] These and other characteristics, features and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, which illustrate, by way of example, the principles of the invention. This description is given for the sake of example only, without limiting the scope of the invention. The reference figures quoted below refer to the attached drawings.

Brief description of the drawings

[0034]

Fig. 1 is a schematic representation of a computer network

Fig. 2 is a schematic representation of a central infrastructure and its basic software components

Fig. 3 is a schematic representation of a local agent-driven content identification process.

Fig. 4 is a schematic representation of a metabase-driven content identification process.

Fig. 5 is a schematic representation of a computer network to which the content firewall system and method can be applied.

[0035] In the different figures, the same reference figures refer to the same or analogous elements.

Description of illustrative embodiments

[0036] The present invention will be described with respect to particular embodiments and with reference to certain drawings but the invention is not limited thereto but only by the claims. The drawings described are only schematic and are non-limiting. In the drawings, the size of some of the elements may be exaggerated and not drawn on scale for illustrative purposes. Where the term "comprising" is used in the present description and claims, it does not exclude other elements or steps.

[0037] Furthermore, the terms first, second, third and the like in the description and in the claims, are used for distinguishing between similar elements and not necessarily for describing a sequential or chronological order. It is to be understood that the terms so used are interchangeable under appropriate circumstances and that the embodiments of the invention described herein are capable of operation in other sequences than described or illustrated herein.

[0038] In this description, the terms "file", "program",

"computer file", "computer program", "data file" and "data" are used interchangeably, and any one use may imply the other terms, according to the context used. The terms "hash" and "hashing" will be used as examples of the application of one-way-functions but the present invention is not limited to a particular form of one-way-function.

[0039] The term "computing device" should be interpreted widely to include any device capable of carrying out computations and/or executing algorithms. A computing device may be any of a laptop, workstation, personal computer, PDA, smart phone, router, network printer or any other device which has a processor and can be connected to a network such as e.g. faxing devices or copiers or any dedicated electronic device such as a so-called "hardware firewall" or a modem.

[0040] The method and system to secure and control a network by identifying the content of each new file in the network can be used on any type of network. This may be a private network which may be a virtual private network, a local area network (LAN) or a wide area network (WAN). This may also be within a part of a public wide area network such as the internet. If a part of a public wide area network is used, this may be performed by remotely providing the method and system for identifying the content of each file by a service provider using an ASP or XSP business model, wherein the central infrastructure is provided to a paying client operating a local computing device. An exemplary network 10 is shown in figure 1, showing several local computing devices 50a, 50b, ... , 50i and a central infrastructure 100, also called a server. The number of local computing devices 50 connected to the network 10 is not limiting for the method of securing and controlling a network 10 according to the current invention. In business environment this number of local computing devices 50 typically ranges from a few to a few thousands. The method and system for identifying the content of each new file present in the network 10 may be used with many different operating systems such as Microsoft DOS, Apple Macintosh OS, OS/2, Unix, DataCenter-Technologies' Operating Systems, ...

[0041] In order to provide a quick method of securing and determining content identification of files, the method and system according to the present invention will determine hash values of new files present on the local computing devices 50, compare them with previously stored hash values and file information on a central server and determine the content of files new to the network 10 using a content identifying engine on the central infrastructure 100. The content attributes describing the content of a new file are then send to the local computing device 50 where an appropriate action is performed. It is also possible that the content attributes are not sent to the local computing device 50 but that the appropriate action is triggered from the central infrastructure 100. New files typically are files wherein new content has been generated on a local computing device 50 or when an external file has been received. The wording "file" may refer to data as well as to software applications, also called software.

[0042] The central infrastructure 100 contains a database, also called metabase 110, which contains a record for every hash value that is calculated for a file that already exists on one of the local computing devices 50. Besides the hash value, this record also contains a number of other fields. In these fields, file source information is stored. The file source information corresponding with a specific hash value includes the file name, a list of local computing devices 50 where the files that correspond to this hash value are residing on, including the path to the file on the file system of the local computing devices 50 and the date of last modification. An example of file source information for a specific file is given in Table 1.

Table 1

Filename	Myexampleword.doc
Path	c:\data\
Assetname	PCmarketing001
ModDate	23/4/2002

[0043] In a further field, a list of content attributes that identifies the type of content that is enclosed by the file is stored. The content attributes can e.g. refer to a file that contains a virus, a file that is a copyrighted MP3 audio file, a file that is a copyrighted video file, a file that is a picture, a file that is a picture that might contain adult content, a file that is a Self Promotional Advertising Message (SPAM), a file that is a HOAX, a file containing explicit lyrics or a file containing pieces of executable code. This list is not limiting.

[0044] The central infrastructure 100 furthermore contains a content identification engine 120. This can be a software application 130 or a set of software applications 130a, 130b, 130c, 130d, ... that use the content of a file to determine which type of content the file contains. These software applications may be various :

- a virus scanner : this is a piece of software that scans the content of the presented file and compares it with a database of known fingerprints of viruses. This can be any conventional virus scanning software like e.g. Norton anti-virus by Symantec Corporation, McAfee by Network Associates Technologies Inc., PC-cillin by Trend Micro, Kapersky Anti-Virus by Kaspersky Lab, F-secure Anti-Virus by F-Secure Corporation, ...
- an adult content in pictures scanner : this is a piece of software that scans the content of the presented file for the presence of shading, colors, textures that might represent adult content. Scanning pictures for adult content is already known. Content adult can e.g. be determined by the amount of nude that is

shown. Skin tones have hue-saturation values that are in a specific range. Therefore, if an image is scanned, it is possible to determine the amount of pixels having a skin tone character and to compare it with the total number of pixels. The ratio of skin tone pixels to the total number of pixels allows to determine a ratio of possible adult content in an image. Thresholds often are introduced so that images can be classified according to their possible adult content. In similar way, video images can be categorised, whereby the video is split into its different frames and wherein the images are categorised according to the above method.

- A scanner for internet content ratings : A piece of software that scans objects for adult content based on the PICS, i.e. the Platform for Internet Content Selection, label system. On voluntary basis, internet content providers can provide internet objects with a PICS rating determining the adult content in the internet object. This PICS rating is stored in the meta data of the object. This data normally is not visible to the viewer of an internet object. The rating systems is well known and an example of a scanner for internet content ratings is provided in the Netscape web browser for scanning the content of web pages.
- A scanner for scanning an object for explicit lyrics which may indicate adult content. This is known for both text files and audio files. Audio files are first transferred to text files. Subsequently, the text files are scanned and compared with databases which contain explicit lyrics.
- a SPAM-engine: A piece of software that scans the content of e-mail messages for the presence of unalleged SPAM. Algorithms to recognize SPAM area already known. These are typically based on decomposing the text in an electronic mail message, associating statistics with the text using a statistical analyzer and coupling a neural network engine to the statistical analyzer to recognize unwanted messages based on statistical indicators.

Other examples of software applications that could be used in the content identification engine 120 are e.g. engines that scan for copyrighted content and that compares the content of the file to a database of copyrighted information, etc. In some adoptions, a human operator can pursue the role of content identification engine 120, where he manually tags a file with a content identification attribute. When the content identification engine 120 is activated, it will take a file from the local agent as input and produce a set of attributes that represent the detected content.

[0045] The content Identification engine 120 will thus be constructed as a piece of software aggregating the functionality of a set of third party engines.

[0046] In a further embodiment of the invention, a system and method in accordance with the above embodiment is described whereby the record corresponding with a specific hash value stored in the metabase 110 also comprises a field wherein the location of the file on the central infrastructure 100 corresponding with the hash value is stored. In this embodiment, a copy of all different files present on the local computing devices 50 in the network 10 is stored on the central infrastructure 100. So, the central infrastructure 100 of this embodiment also comprises a large amount of storing space. This preferably is a secured part of the central infrastructure 100, not directly connected to the network 10 so that these identical copies of the files present on the local computing devices 50 can be used in case the files on the local computing devices 50 are corrupted e.g. by a virus.

[0047] The hash value of the files are calculated using a hashing function. A hashing function typically is a one way function, i.e. given the digest, it is at least computationaly prohibitive to reconstruct the original data. Different types of hashing functions could be used : MD5, SHA-1 or ripemd all available from RSA Data Security Inc., haval which is designed at the University of Wollongong, snefru which is a Xerox secure hash function, etc.. The hashing functions most often used are MD5 and SHA-1. The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be 'compressed' in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem. The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. An alternative hashing function SHA-1, i.e. Secure Hashing Algorithm - 1, is a hashing algorithm generating a 160-bit hash. Newer versions of this algorithm also provide bit lengths of 256 and 512.

[0048] In the above mentioned embodiments describing a method and system to secure and/or control the network 10, a local agent is installed on the local computing device 50. The local agent is a piece of software that is running on a local computing device 50 and that performs certain algorithms and procedures. The local agent on the local computing device 50 is triggered typically in situations where new content is being generated on local computing devices 50. In order to avoid unnecessary hash value calculations and data transfer, a policy is setup to determine which actions will trigger the local agent and which actions do not trigger it. If e.g. a text document is being created, it is not necessary to check the file every time the document is saved. The policy for such a type of documents would preferably be that the document is checked e.g. if the file is both saved

and closed. Some examples of actions which could trigger the local agent and thus start the content identification process are opening or receiving e-mail messages, opening or receiving e-mail attachments, running executable files, running files with dll or piff extension ... Applying this policy thus allows to prevent from continuously checking and scanning of documents leading to a limitation of the number of unnecessary hash calculations and content identification operations and thus limiting the unnecessary use of CPU time and load on the network traffic. The method and system of content identification is not limited due to the type of application in which the file is made.

[0049] The content identification process can be either triggered by the local agent on the local computing device 50 or it can be triggered by the central infrastructure 100. The latter process typically occurs in situations wherein new algorithms or tools are being used for content identification. Such new algorithms or tools can either be optimised algorithms and tools or previously uninstalled tools. Some examples of these tools, without restricting to these functions, could be virus checking, checking whether a file is a copyrighted MP3 Audio File, checking whether a file is a copyrighted Video File, checking whether a file is a picture that might contain adult content, checking whether a file is tagged as being SPAM or HOAX, checking whether a file contains explicit lyrics or checking whether a file contains copyrighted pieces of executable code. Updating of these tools may influence the status of the files and thus may in principle influence the corresponding records in the metabase 110. Therefore, depending on the type of update of the content identification means 120, it may be interesting to update the corresponding records.

[0050] In a specific embodiment, the method relates to a virus checker for a network environment. The networks 10 on which this method can be applied are the same as those described for the previous embodiments. The local agent calculates the hash value of a new file on the local computing device 50. This new file may comprise new content generated on the local computing device 50 or an external file which is received on the local computing device 50. The hash value of the new file and the corresponding file information then is sent to a central infrastructure 100, also called server, where it is compared to previously stored hash values corresponding with files that are already present on the different local computing devices 50 of the network 10. This comparison allows to check whether the file is new or not in the entire network 10. Alternatively, the hash value may also be first compared with a local database of hash values and file information corresponding with the files present on that particular local computing device 50 and subsequently, if the file has been found not yet present on the local computing device 50, the hash value and the corresponding file information may be interchanged with the central infrastructure 100 so it can be checked whether the file is new or not in the entire network 10. Although transferring the file information and the hash value of every new file only corresponds with a very small fraction of the network traffic for a conventional central virus checker, this alternative could reduce the network traffic used for virus checking even further. If a hash value has been identified as new on the network 10, the metabase agent triggers the local agent to transfer the file corresponding with the new hash value from the local computing device 50 to the central infrastructure 100. The transferring of the file may be performed in a secured way, i.e. the file may be transferred such that it cannot be influenced by a virus present at a network connection or such that, it if contains a virus, this cannot be spread over the whole network 10. To obtain this, a known secure transmission route, a tunnel and/or known session ecryption/decryption techniques may be used. A conventional virus checker, installed and updated on the central infrastructure 100 then checks the file for viruses. This can be any conventional virus checker like e.g. Norton anti-virus by Symantec Corporation, McAfee by Network Associates Technologies Inc., PC-cillin by Trend Micro, Kapersky Anti-Virus by Kaspersky Lab, F-secure Anti-Virus by F-Secure Corporation, ...

[0051] A specific advantage of the above described embodiments in the current invention is that the virus scanning software does not need to be updated on every local agent but that this is restricted to updating of the virus scanning software of the central infrastructure 100. In this way the security level of the network 10 is increased significantly as the security does not depend on the punctuality of the different users of the network 10 to update their virus scanning software. If the scanned file has no virus it will be marked in the metabase 110 as being a virus free file. If there is a virus found in a file the file will be marked as dangerous. A query will happen to the metabase 110 to find all files over the network 10 having the same corrupted hashing key. The result is a list of files with path, and assetname where the file is located. This information can be used to do actions to eleminate the danger of found viruses on all local computing devices 50, i.e. all workstations, from the complete network 10. In this way proactive virus scanning can be performed on other local computing devices 50, based on a virus detection on a first local computing device 50. Depending the policy defined for virus checking, the virus engine will inform an agent installed on the affected system to remove the file and if possible replace with either a recovered version delivered by the virus engine located on the central infrastructure 100, or a previous version of the file which didn't have the virus yet. The latter can be done easily by searching the metabase for a previous version of that file, or it can be performed by searching an uninfected version on another local computing device 50. If an uninfected version cannot be retrieved from either another local computing device 50 or the metabase residing on 100, the virus scanner should have a feature which allows it to save a

new disinfected copy of the file on the central infrastructure 100. These advantages are also present for other content identifying packages.

[0052] In an alternative embodiment, if a file having a new hash value has been identified in the network 10, instead of transferring the file to the central infrastructure, the file may be automatically shared locally and a remote checker then may transfer a file-system which allows to check the file across the network 10 using the file sharing. The content tagging still is performed by the server. In order to improve security, the accessibility to the shared file is restricted to the server. Furthermore, a java applet could be transferred to the local agent to allow checking other files.

[0053] The previous embodiments are an improvement over a central virus checker which scans local computing devices 50 through the network 10. This is only possible if the local drives, e.g. C:\, D:\, ..., are shared. Besides the dangers of sharing with respect to security, the local user also easily can change the local sharing properties thereby preventing the remote checker from checking the files. This is at least partly avoided with the current invention as changing the network 10 sharing properties does not influence the operation of calculating the hash value of new files and sending it to the central infrastructure 100.

[0054] Another advantage is that it saves CPU time on the local computing device 50 as the CPU does not have to keep doing virus checking, it only has to calculate a one way function. It also saves network time : the administrating server does not have to update the virus checkers on the local computing devices 50 with virus updates, as a single central virus checker only is used and updated.

[0055] Fig. 3 shows a method 200 of the content identification process triggered by the local agent on the local computing device 50 according to the above mentioned embodiments. The different steps that occur during the process, both on a local computing device 50 and on the central infrastructure 100 are discussed.

[0056] The content identification process is based on continuously scanning for new data or applications on the local computing device 50 by the local agent. This scanning for data and applications is limited by the policy rule for determining when the local agent should be triggered, as described above. If a "new" file has been detected the method for securing and controlling the network 10 by content identification of new files is initiated. This is step 210. Method 200 then proceeds to step 212.

[0057] In step 212, a hash value of the "new" file is calculated using a hashing function like MD5 or SHA-1. This calculation is performed by using some CPU time of the local computing device. Nevertheless, the amount of CPU time used is drastically smaller than the CPU time that would be necessary if e.g. a conventional virus checker was used to check the file on the local computing device 50. Method 200 then proceeds to step 214.

[0058] In step 214, the hash value and the file source information is transferred from the local agent to the central infrastructure 100 of the network 10. If necessary, this transfer can be a secured transfer, whereby it is avoided that a virus which is positioned on a network connection changes both the file source information or the hashing key during transport of this data. Such a secured transmission can be made over a known secure transmission route, via a tunnel, or using known session ecryption/decryption techniques.

[0059] In step 216, the hash value is compared with the data already present in the metabase 110. As in the metabase 110, the hash values and file source information of all old files - i.e. every file that has been present on the network 10 and that is not "new" as described above - present in the network 10 are stored, it is possible to check whether the file already is present in the network 10. Therefore, if the hash value has been identified as new, this implies that the file is "new" for the whole network 10. If the file is new, method 200 proceeds to step 218. If the hash value is not new, this means that somewhere on a local computing device 50 in the network 10, the file does already exist. In this case, there already exists content attributes describing the content of the file. Method 200 then proceeds to step 224.

[0060] In step 218 the metabase agent triggers the local agent to transfer the file corresponding with the new hash value from the local computing device 50 to the central infrastructure 100. The transferring of the file may be performed in a secured way, i.e. the file may be transferred such that it cannot be influenced by a virus present at a network connection or such that, it if contains a virus, this cannot be spread over the whole network 10. To obtain this a known secure transmission route, a tunnel and/or known session ecryption/decryption techniques may be used. Method 200 further proceeds to step 220.

[0061] In step 220 the file is loaded in the content identification engine 120 and the file is processed. For this processing CPU time of the central infrastructure 100 is used. The content identification engine 120 can comprise, as described above, a conventional virus checker, a means for checking picture information, a means for checking SPAM, etc. This can be a repetitive action where multiple content identification engines are called in turn. Method 200 then proceeds to step 222.

[0062] In step 222 content attributes, which identify the content of the file, are determined for the file. These content attributes are then stored in the metabase 110, thus allowing to identify the status of the file if, in future operations, the file is found 'new' on another local computing device 50. Method 200 then proceeds to step 224. Depending on the embodiment used, a following step may include the storing of the file on the central infrastructure 100 and adding the path to this file to the metabase 110. This step is not shown in figure 3.

[0063] In step 224 the content attributes are sent to

the local agent. Based on this content attributes, the local agent performs an appropriate action in agreement with the policy rule set for these content attributes. This is performed in step 226. This can be e.g. deleting the file if it was infected, replacing the file with a previous version which was not infected, ... In a specific embodiment, the execution of appropriate actions based on the policy rules are triggered by the agent of the metabase 110, so that step 224 can be avoided.

[0064] The content policy is a policy that determines what should be done with a file depending on the content attributes determined by the content identification engine 120. The content policy can comprise actions such as deleting the file, deleting the file and replacing it with a previous version, copying the file onto another computing device while leaving a copy on the originating computing device, moving the file onto another computing device while deleting the original file on the originating computing device, logging the presence of the file, changing the attributes of the file like hiding it or making it read-only, making the file unreadable, making the file un-executable, etc. The content policy will be executed by the local agent, e.g. when the content attributes are received from the central infrastructure 100. The content policy for that agent will be downloaded to the local computing device 50 by the agent from a central policy infrastructure.

[0065] Fig. 4 shows a method 300 of the content identification process triggered by the content identification engine 120 according to the above mentioned embodiments . The different steps that occur during the process, both on a local computing device 50 and on the central infrastructure 100 are discussed.

[0066] This process typically is used in situations where new algorithms or tools are being used for content identification. Such new algorithms or tools can either be optimised algorithms and tools or previously uninstalled tools. As mentioned earlier this may be regulated by a policy : the triggering of the content identification process may be determined by the type of new algorithms and tools that are being used for content identification.

[0067] Method 300 is initiated by change of the content identification engine 120, e.g. by providing new algorithms or tools for the content identification engine 120.

[0068] In the first step 302 of method 300 upon triggering, the metabase 110 is scanned for hash values corresponding with hashing keys. Method 300 then proceeds to step 304.

[0069] In step 304, a file that corresponds with the hashing key is requested. This file can be either requested from the central storage on the central infrastructure 100 or it can be requested from a local computing device 50. The local computing device 50 then gives permission to the central infrastructure 100 to upload the corresponding file. The path to the file corresponding with the hash value is available from the record corresponding with each hash value. If the record stores different paths all corresponding with a copy of the corresponding file, the agent on the central infrastructure 100 retrieves one copy of the file, e.g. by scanning the paths listed in the record until a local computing device 50 has been found that is at that time connected to the network 10 and that allows uploading of the file. Method 300 then proceeds to step 306.

[0070] Once the file has been retrieved, the file is sent to the content identification engine 120. This is performed in step 306. The upgraded content identification engine 120 then scans the content of the file and produces content attributes corresponding with the file. Method 300 further proceeds to step 308.

[0071] In step 308, the content attributes are stored in the metabase 110, to allow in future security steps to immediately identify the content of the files. Method 300 further proceeds to step 310.

[0072] In step 310, the content attributes are sent to every local agent that resides on a local computing device 50 whereon the corresponding file is stored. The paths can be found in the record of the corresponding hashing key stored in the metabase 110. In this step, content attributes are send to every file for which a path is mentioned in the record of the corresponding hashing key. Method 300 proceeds to step 312.

[0073] In step 312, the local agent on the corresponding local computing devices 50 executes the policy according to the content attributes and the according to the local computing device 50.

[0074] One of the major advantages of the embodiments of the invention is that a file new to the entire network 10 only needs to be scanned once. If on another local computing device 50, an identical copy of this file is used, installed, opened or saved and closed, the file will be recognised by the central infrastructure 100 as being known to the network 10, in this way avoiding the need to re-check the content of the file. This especially is advantageous if the invention is used for networks 10 having a large number of local computing devices 50.

[0075] When operating, the method and system for identifying the content of new files optionally can comprise checking 'the heartbeat' of the local agent at regular times, i.e. it can be checked whether the local agent is still running on the local computing device 50. This can avoid that a user locally shuts down the agent, thus making the local computing device 50 vulnerable. If the local agent has been shut down, the network administrator can be warned. Furthermore a warning message could be send to the local computing device 50 thereby warning the user of the local computing device 50. The network administrator could also put the local computing device 50 in quarantine so that it can not damage other local computing devices 50 in the network 10. Furthermore, the central agent can also try to rerun the local agent.

[0076] In a similar way, the method and system for identifying the content of new files optionally can check

at regular times whether the local computing device 50 is still connected to the network 10. If the local computing device 50 is not connected to the network 10 anymore, the local agent may further operate, storing hashing keys of new files in a waiting list to be checked once the network connection is restored. In the mean time, the corresponding files may be put in quarantine or depending on the type of file e.g. may be prevented from being executed.

[0077] The above described embodiments may be used as a content firewall for the different computing devices connected to the external network. For every incoming/outgoing file, incoming/outgoing message or incoming/outgoing data frame, the content firewall calculates the hash, checks whether this is new, checks whether it is tagged for specific content and enforces the policy associated with the specific content.

[0078] In a further embodiment, another configuration for using the present invention as a content firewall is described. A schematic overview of a computer network wherein this method and system may be used, is shown in Fig. 5. Only one reconfigurable firewall electronic device 50, such as a local computing device which may be in the form of a dedicated reconfigurable firewall electronic device, is directly connected to an external network 400 such as e.g. the internet, and the remaining local computing devices 410, are not directly connected to the external network 400, but grouped in a network environment and only connected to the external network 400 by their connection to the electronic reconfigurable firewall device.. The external network may be any possible network available. It is a goal of the content firewall as represented by the reconfigurable electronic firewall device 50 to protect the network environment comprising the remaining local computing devices 410 from attacks originating from places and/or devices in the external network. The reconfigurable electronic firewall device 50 50 either contains a local copy of the metabase or it can use a high speed secured network to a central infrastructure 100 which is part of the internal network. This allows for fast queries through the metabase. During operation, the reconfigurable electronic firewall device 50 functioning as a content firewall performs the following actions : the hash value of incoming files or incoming messages or incoming data frames are calculated. Subsequently, the calculated hash values are compared with the metabase, which is either stored locally or by using a high speed secured network, and it is determined whether the incoming file, incoming message or incoming data frame is new. Furthermore, it is checked whether this file, this message or this data frame is tagged for specific content. Depending on the specific content, a policy is enforced which is associated with the specific content. This policy may be to let it pass through to its final destination, to drop it, to log it, to put it in quarantine, etc. This system requires sufficient CPU power, in order not to slow down the network speed noticeably.

[0079] In the case where none of the local computing devices connected to the network is equipped with a removable device, i.e. allowing for non-scanned content to be opened or executed on that device, this is a very secure and managable setup.

[0080] In another embodiment of the invention, a similar configuration for use of the present invention as a content firewall in promiscuous mode is provided. The content firewall thereby sees all traffic passing by, executes the hashing and comparing functions and contacts the agents to enforce a policy. The advantage of this approach is that there is no single point of failure and no bottleneck anymore and furthermore that still no resources are used on the local computing devices for calculating hashes. Furthermore, no bandwidth is used for contacting the central metabase. The disadvantage is that local agents need to be installed on all computing devices of the internal network.

In accordance with the above described embodiments, the present invention includes a computer program product which provides the functionality of any of the methods according to the present invention when executed on a computing device. Further, the present invention includes a data carrier such as a CD-ROM or a diskette which stores the computer product in a machine readable form and which executes at least one of the methods of the invention when executed on a computing device. Nowadays, such software is often offered on the Internet, hence the present invention includes transmitting the printing computer product according to the present invention over a local or wide area network.

Claims

1. A method for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part of the network environment including a central infrastructure and, the method comprising

   - calculating a reference value for a new file on one of said at least one local computing devices using a one-way-function,
   - transmitting said calculated reference value to said central infrastructure,
   - comparing said calculated reference value with reference values previously stored within the remaining part of the network environment,
   - after comparing,

     - deciding that the content of the new file is already identified if a match between said calculated reference value and a previously stored reference value is found and retrieving the corresponding content attributes; or

- deciding that the content of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure identifying the content of said new file and determining content attributes corresponding with the content of the new file and storing a copy of said content attributes,
- after deciding, triggering an action on said local computing device in accordance with said content attributes.

2. A method according to claim 1, wherein the reference value is a hash value.

3. A method according to claim 1 or 2, wherein reference values previously stored are stored within the central infrastructure.

4. A method according to the previous claim wherein said identifying the content of said new file comprises scanning the new file for viruses using a anti-virus checker means on said central infrastructure.

5. A method according to any of the previous claims furthermore comprising transferring the new file from the local computing device to the central infrastructure before said identifying the content of said new file is performed.

6. A method according to claim 5, furthermore comprising storing a copy of the new file on the central infrastructure after said transferring the new file from the local computing device to the central infrastructure

7. A method according to any previous claim wherein said triggering an action on said local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with a copy of a previous version of said new file.

8. A method according to any previous claim, wherein said triggering an action on said local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with another version of said new file restored from the remaining part of the network environment.

9. A method according to claim 1, 2, 3 or 4 furthermore comprising sharing the new file on the local computing device to the central infrastructure before said identifying the content of said new file is performed and whereby said identifying the content of said new file is performed by remotely identifying the content over the network environment.

10. A method according to any of the previous claims, wherein said triggering an action on said local computing device in accordance with said content attributes is performed after transmitting the content attributes corresponding to the new file to the local computing device.

11. A method according to the previous claims wherein said identifying the content of said new file comprises one or more of the group of scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

12. A method according to any of claims 1 to 3 or 5, whereby one local computing device links said network environment with an external network, said one local computing device acting as a content firewall as to protect the network environment.

13. A computer program product for executing any of the methods as claimed in claims 1 to 12 when executed on a network.

14. A system for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part the network environment which includes a central infrastructure and, said remaining part including a stored database, whereby the system comprises :

- means for calculating a reference value for a new file on said local computing device using a one-way-function,
- means for transmitting said calculated reference value to said central infrastructure,
- means for comparing said calculated reference value with previously stored reference values from the database,

whereby the system further comprises:

- means for deciding whether the content of the new file is already identified based on comparison of said calculated reference value and reference values previously stored within the remaining part,
- means located on the central infrastructure, for identifying the content of the new file and as to assign content attributes if the new file has not been identified yet and means for storing said content attributes within the remaining part, and
- means for triggering an action on said local

computing device in accordance with content attributes for said new file.

15. A system according to claim 14 wherein said means for identifying the content of a file comprise an anti-virus checker means on said central infrastructure.

16. A system according to claim 14 or 15 furthermore comprising means for storing a copy of the new file within the remaining part.

17. A system according to any of claims 14 to 16 wherein said means for identifying the content of a file comprises one or more of the group of means for scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

18. A method for altering a system for identifying the content of a file in a network environment according to claims 14 to 17, said network environment comprising means for calculating a one-way function, at least one local computing device linked to a remaining part of the network environment including a central infrastructure and means for identifying the content and, the method comprising

- altering said means for identifying the content or said means for calculating a one-way function
- scanning the remaining part of the network environment for reference values calculated with a one-way function
- for each of said reference values,
    - requesting a file that corresponds with said reference value from said network environment
    - sending the file to means for identifying the content
    - identifying the content of said file and determining content attributes corresponding with the content of the file and storing a copy of said content attributes
    - sending the content attributes to every local computing device containing the file
    - after sending; triggering an action on said local computing device in accordance with said content attributes.

19. A computer program product for executing the method as claimed in claims 18 when executed on a network.

20. A machine readable data storage device storing the computer program product of any of claims 13 or 19.

21. Transmission of the computer program product of any of claims 13 or 19 over a local or wide area telecommunications network.

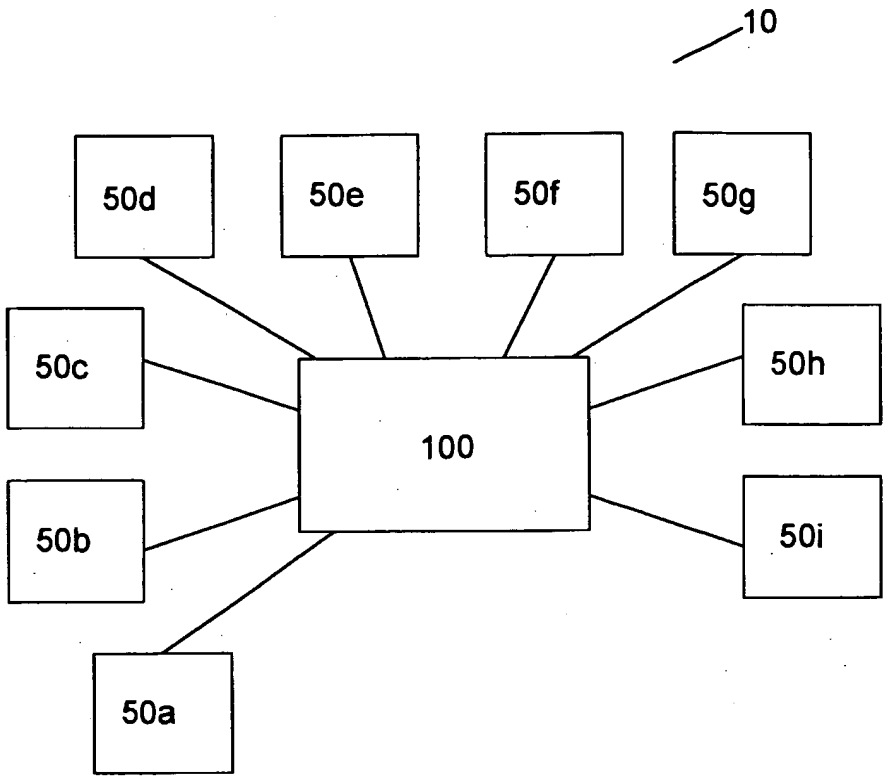

10
50d
50e
50f
50g
50c
50h
100
50b
50i
50a

Fig. 1

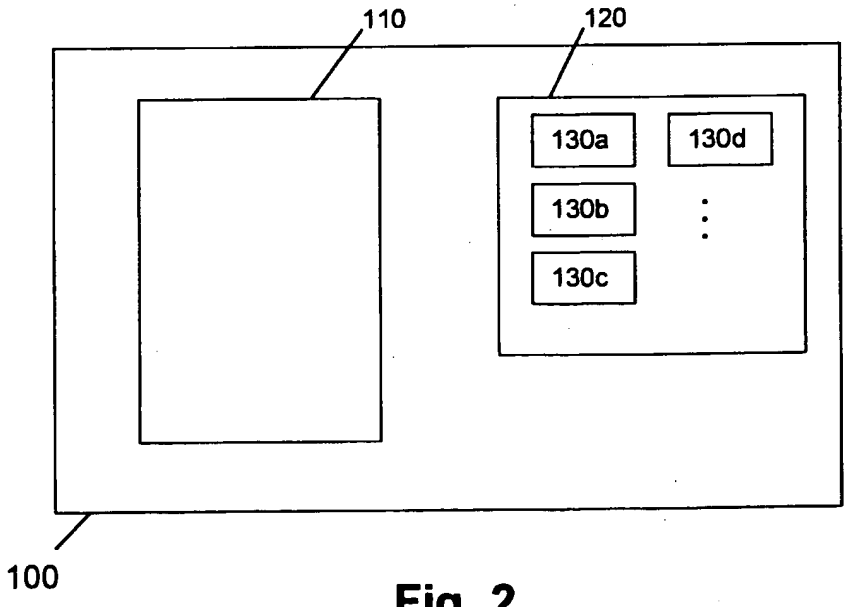

110
120
130a
130d
130b
130c
100

Fig. 2

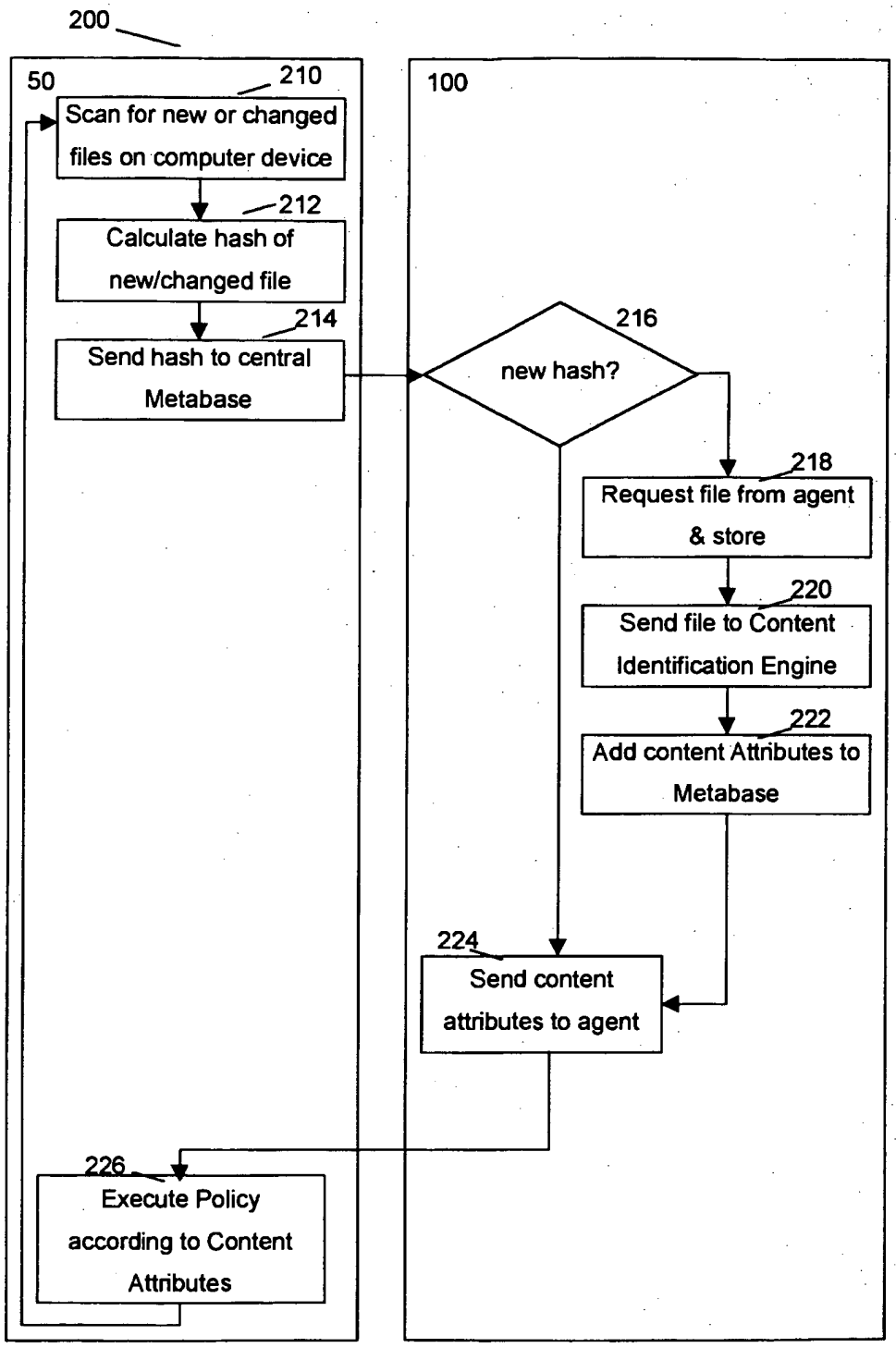

200
50
210
Scan for new or changed files on computer device
212
Calculate hash of new/changed file
214
Send hash to central Metabase
100
216
new hash?
218
Request file from agent & store
220
Send file to Content Identification Engine
222
Add content Attributes to Metabase
224
Send content attributes to agent
226
Execute Policy according to Content Attributes

Fig. 3

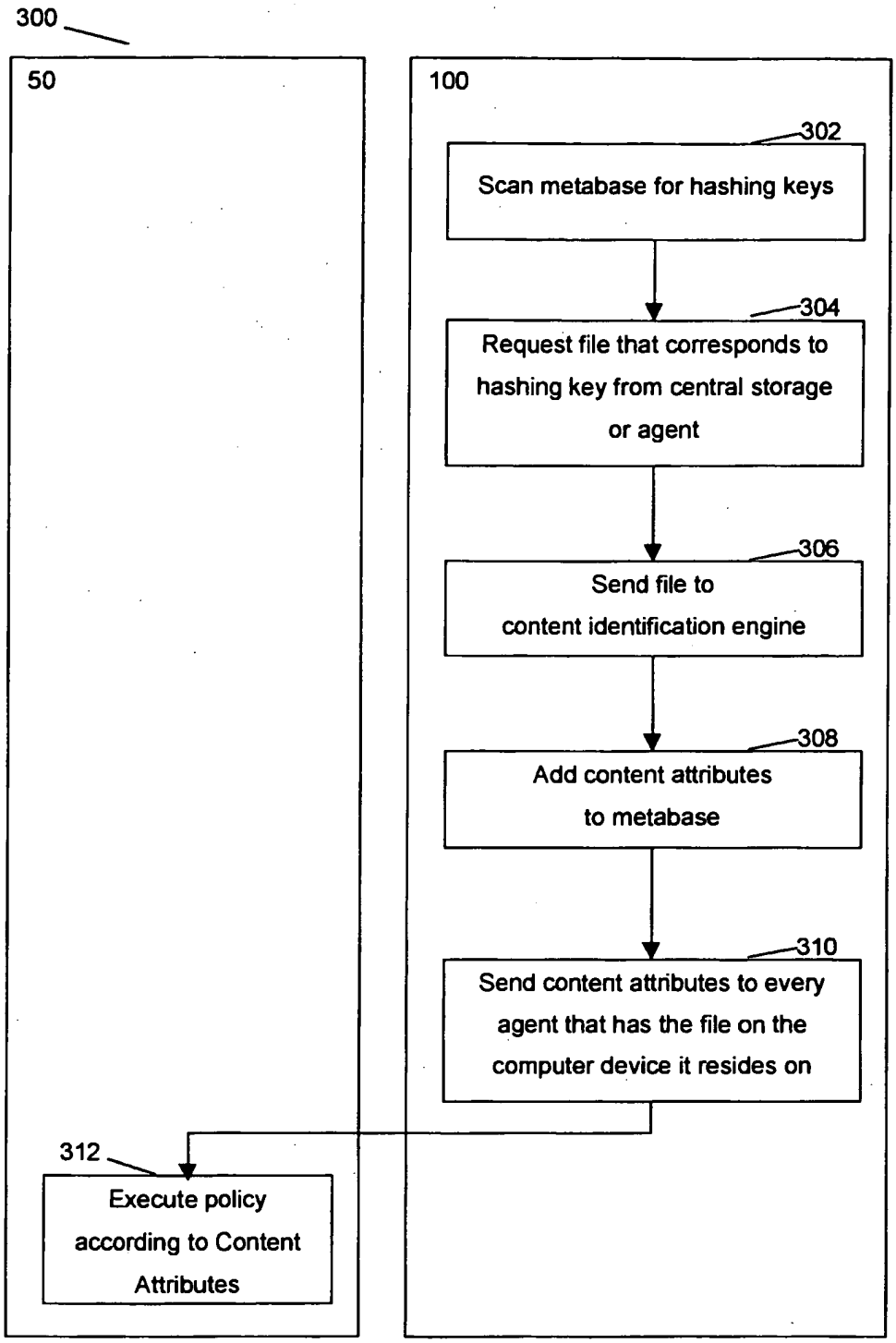

300
50
100
302
Scan metabase for hashing keys
304
Request file that corresponds to hashing key from central storage or agent
306
Send file to content identification engine
308
Add content attributes to metabase
310
Send content attributes to every agent that has the file on the computer device it resides on
312
Execute policy according to Content Attributes

Fig. 4

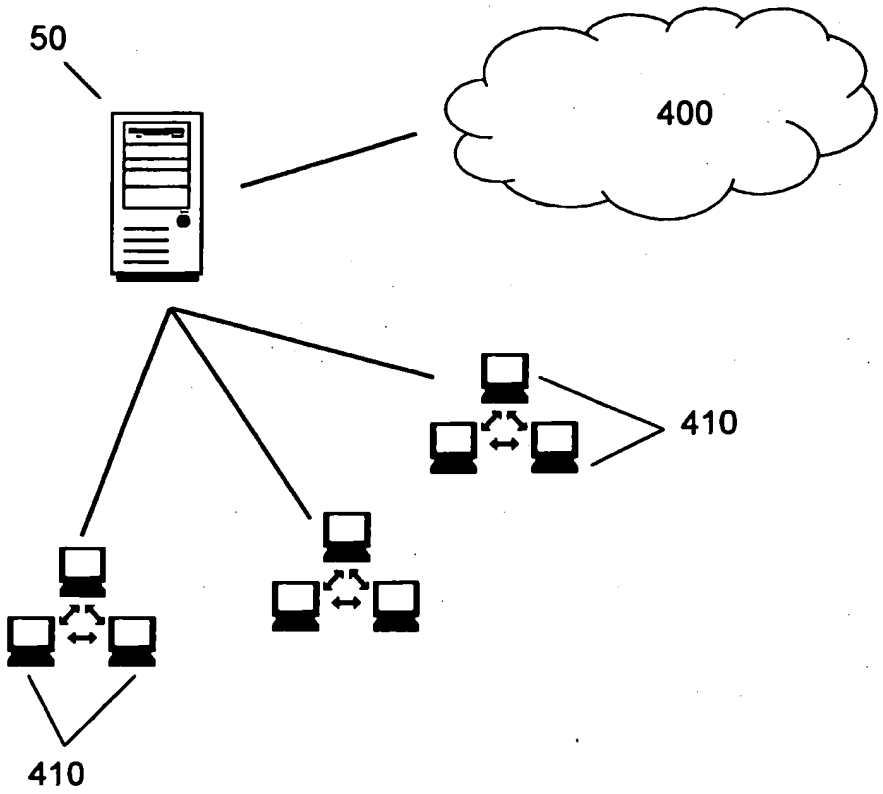


Fig. 5

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 03 44 7310

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	EP 1 280 040 A (NETWORKS ASSOCIATES TECHNOLOGY) 29 January 2003 (2003-01-29) * the whole document *	1-9, 11-17, 20,21	H04L29/06 G06F1/00
A		10,18,19	

A	US 6 195 587 B1 (HRUSKA JAN ET AL) 27 February 2001 (2001-02-27) * column 1, line 4-9 * * column 1, line 50 - line 65 * * column 3, line 55 - column 5, line 67 * * column 1, line 50 - column 3, line 3 * * column 3, line 55 - column 5, line 40 *	1-6, 9-11, 13-20	

A	US 2003/120953 A1 (RADATTI PETER V) 26 June 2003 (2003-06-26) * paragraphs [0013]-[0015] * * paragraph [0045] * * paragraph [0047] * * paragraph [0051] *	1-4,7,8, 11, 13-15, 17,20	

A	EP 1 111 507 A (SECURE OYJ F) 27 June 2001 (2001-06-27) * paragraphs [0002]-[0004] * * paragraphs [0010]-[0012] * * paragraph [0015] * * paragraphs [0025],[0026] * * paragraph [0031] *	1,4-8, 12-16,20	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)
H04L
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	7 June 2004	Ruiz Sanchez, J

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 03 44 7310

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

07-06-2004

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1280040 A	29-01-2003	US 2003023866 A1	30-01-2003
		EP 1280040 A2	29-01-2003
US 6195587 B1	27-02-2001	GB 2283341 A	03-05-1995
		AU 8000394 A	22-05-1995
		DE 69407812 D1	12-02-1998
		DE 69407812 T2	16-07-1998
		EP 0725951 A1	14-08-1996
		WO 9512162 A1	04-05-1995
		JP 9504395 T	28-04-1997
US 2003120953 A1	26-06-2003	NONE	
EP 1111507 A	27-06-2001	GB 2353372 A	21-02-2001
		EP 1111507 A2	27-06-2001
		US 2001005889 A1	28-06-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 March 2003 (13.03.2003)

PCT

(10) International Publication Number
WO 03/021402 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/GB02/04059

(22) International Filing Date:
5 September 2002 (05.09.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0121497.2 5 September 2001 (05.09.2001) GB

(71) Applicant *(for all designated States except US)*: **CRYPTIC SOFTWARE LIMITED** [GB/GB]; The Business Exchange, Rockingham Road, Kettering, Northants NN16 8JX (GB).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **DUKE, David, John** [GB/GB]; 3 The Saplings, Kettering, Northants NN16 9UL (GB).

(74) Agent: **CRAWFORD, Andrew, B.**; AA Thornton & Co., 235 High Holborn, London WC1V 7LE (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 03/021402 A2

(54) Title: NETWORK SECURITY

(57) Abstract: A computer network comprises a server computer and a plurality of local computers. Each local computer is provided with an engine program which monitors all files and peripherals and detects charges in the files or configuration. The results of the monitoring are used to create compressed information relating to the files and peripherals which is communicated to the server computer which stores previous information relating to the files and peripherals of the local computers. The server computer compares the received information with the stored information in order to detect the presence of altered and unwanted files or changes in configuration. In this way both known and unknown viruses can be detected as can un-authorised use of the local computers.

NETWORK SECURITY

The present invention relates to an arrangement for monitoring the security of a computer especially computers on a network and particularly individual devices and information contained therein.

It is well known to provide individual computers with virus checking software but it is also well known that such software consumes resources when it is run and also takes time. Additionally, updating such software is somewhat cumbersome as each individual computer has to be updated separately.

Further, it is known to provide firewalls to protect computers and computer systems which are connected to a network or to the outside world but again these firewalls have limitations and they do not provide a completely effective defence to potential hackers.

Both of these methodologies are reactive and only detect threats to a network which have been seen and indexed previously.

It is an object of the present invention to provide an arrangement whereby protection from attack by viruses is improved as well as providing a more effective firewall in the event that the computer or computer system is provided with such. It is a further object that the invention is deployed in such a way that it is itself not vulnerable to attack.

From one aspect, the present invention is based on a concept whereby a main memory holds a record of all files used by a local computer, is provided with update information on all existing files as well as all new files and determines whether the updated files or new files represent a security threat.

In a preferred embodiment, as data is created, modified or deleted on the local computer, events are monitored and collected by an engine program and forwarded to the main memory. Preferably, the engine program itself consists of a

single file which is arranged to run a program in parallel with the normal operating system of the local computer thus making the file easier to hide so that the presence of the engine program will not be noticed either by the user or by a hacker. Further, as far as the user of the local computer is concerned, the engine program has no noticeable effect on the system. Preferably, the main memory is provided by a central computer (hereafter referred to as the console) and the local computer communicates with the console over a network or in some other convenient manner.

The engine program monitors each and every file and by inspecting preselected parts of each file can create a highly compressed accurate image of the file which can then be transmitted to the central database. It will be appreciated that all files can be handled in this way. Consequently, when the term "file" is used we mean all files associated with the operating system, data, registries, directories, hardware, software and such like. In this way, a complete virtual image of a local computer can be stored within the central database and the console can have management software for monitoring changes in the data files, programs or hardware of the local computer and thus warn of tampering with such files, programs and/or hardware.

From another aspect, the present invention provides a method of monitoring the security of a computer system comprising monitoring a file as it is created or updated by inspecting preselected portions of the file selected from the whole file, storing information derived from the preselected portions and transmitting the stored information to a main memory location. The main memory location is preferably a central database of a network.

From a further aspect the present invention provides a method of improving the security of computer apparatus by providing two copies of a security program and interconnecting them such that as one is switched off, either deliberately or

inadvertently, the other is automatically switched on and vice versa. It is to be understood that this aspect is not limited to any particular type of security program and is of general application. However, it has particular application with the present invention where the security program is relatively small and easy to hide.

In order that the present invention be more readily understood, an embodiment thereof will now be described with reference to the accompanying drawings in which: -

Fig 1 shows a block diagram of a network according to the present invention;

Fig 2 shows a block diagram representing the main functions of an engine program arranged to be installed on each computer to be protected; and

Fig 3 is a flow chart for explaining the operation of the engine program of Fig 1.

The present invention will be described in relation to its use for network security but it will be appreciated that the invention can be implemented in a number of different ways independently of any connection technology. In fact, the totality of the invention could be located within a single computer or a local computer could be in communication with a central server over conventional telephone lines rather than utilising any network protocol.

As shown in Fig 1, a central server 1 communicates via transmit/receive ports 2 with a number of local computers 3, each of which is provided with a monitoring engine program 4. The central server is provided with a plurality of databases, namely an archive of threats database 5, a virtual image of each computer database 6 and an audit database 7.

The basis of the present embodiment is that a virtual copy of a local computer and all its files will be kept in memory at the database to the central server computer with each file having its own individual characteristics stored at

the central computer. The virtual copies are created by the small engine programs 4 loaded on the local computers 3 which monitor all files by inspecting preselected portions of the files in order to create a so-called "fingerprint" of each file which is in fact a highly compressed version of the file. The fingerprint can then be very speedily forwarded to the central location where the current fingerprint can be compared with a previous fingerprint and any changes detected. The changes are then evaluated by the central computer 1 in order to determine the level of threat, if any, to the security of the local computer system and an appropriate signal sent back from the central server to the local computer 3 in the event that a particular file should not be opened or a particular program should not be run. Simultaneously, an event is displayed at the central computer and/or forwarded to other programs.

It will be appreciated that by the use of a virtual copy of the local computer and all its files, the central server can then monitor and log all changes to files, programs and/or hardware in order to provide evidence of breaches of security at the local computer. As an example of this, one could consider a computer fitted with several pieces of memory. Removal of all or part of this memory is an event which can be detected by the engine program and thus signalled to the central computer which can log the event.

In further explanation of the evidential facilities of this invention on the triggering of specified events, biometric information can be gathered in real time and forwarded to the central database. For example, if an event is detected, a screen shot of the local computer is gathered for evidence and at the same time the engine program can gather biometric information such as a photograph of the user of the local computer.

The above description represents a simple brief overall view of the arrangement in accordance with the present invention but for a more detailed

description we will now separate the operation of the engine program at the local computer and firstly describe this before we move onto a description of the operation of the monitoring program at the central server.

The engine program is shown diagrammatically in Fig 2 where all file data is monitored at monitoring section 11 under the control of a program section 12 which determines which parts of the data will be sampled. The selected parts of the data are then stored as indicated at 14 in order to create a virtual compressed copy of the file data for transmission as indicated at 16.

The engine program is arranged to commence running as soon as the computer on which it is loaded is powered up. The program then monitors all files as they are opened and in particular it monitors all changes to a file as indicated by the flow chart shown in Fig 3. Each and every change is inspected in order to determine whether the change is the creation of a new file. It also determines whether the change is a valid change. Each file is also checked by inspecting the file from a plurality of points of view by sampling predetermined portions of the file. This in turn creates a fingerprint or virtual copy of the file which is then assembled for transmission to a central database either over the network or via some other communications link. Among the predetermined portions of the file which the engine program inspects is the tag indicating whether or not the file is an executable. It also determines whether the file is a manipulated file, eg a zipped or encrypted file. It looks for the presence of capabilities such as keystroke logging, FTP server capability, IP notification, joystick controls, game libraries, etc

The engine program has a memory capability 12 so that it can store instructions received from the central processor as described below and then carry out those instructions in the event that the fingerprint it has assembled of a file indicates that action is necessary. As an example, if a known threat is present in a

file as indicated by the fingerprint, the engine program can immediately kill the threat in view of the fact that it has the necessary instruction previously received from the central server. In certain circumstances, the file in question can be automatically dealt with.

A particular feature of the engine program is that it does not wait to be polled by a central computer. Rather, it itself generates a message for transmission to the central computer. This has the advantage that the engine program is immune to attack from a hacker because there is no "listening" port waiting for an incoming transmission. It is not until the engine program has communicated with the central server that a two way communication is possible and only after the engine program's communication with the central server can the central server transmit modified information to the engine program. However, the monitoring and modification of the files on the local computer occur in real time while the local computer is in operation either by being started up or while the local computer is running programs. Further, the engine program can itself run other software in order to transmit and/or record data as a result of the detection of a particular event or change to a file. As an example of this, if the local computer is fitted with a web cam, should particular files be modified, the engine program can take a picture of the user of the machine at the appropriate time and correlate the event with the actual user as evidence for subsequent use.

In an especially secure system, it is possible to provide two identical engine programs which are interlinked with each other so that in the event that one is disabled for any reason, eg being switched off, the other is activated before the first is terminated and vice versa. This ensures security in the event that one of the engine programs is detected by a hacker or other person seeking to tamper with the local computer.

Turning now to the central server 1, this contains a reception port 2 for receiving data transmitted from the local computer. As fingerprints are stored, the central computer builds up a virtual picture of each local computer, its hardware, programs and files generally in its database 6. In addition, the central server maintains a database 5 of all known security threats and viruses. As fingerprints of files are received these are compared with the information in the threats database 5 so that any threat can be detected instantaneously. Additionally, because the central server maintains an up to date virtual copy of each local computer with which it is in communication, in the event that it identifies a security threat in relation to one computer 3, it can then inspect all other local computers for similar configurations and then be ready to communicate the necessary instruction to another local computer 3 as soon as the other local computer communicates with the central server for any reason. In this way, security threats and the mechanism of deleting files or repairing infected files can be undertaken much more quickly and in fact even before a local computer might have been subjected to attack or infection. In other words, when a new threat is identified, it is added to the server database so that the central processor can check its data relating to the local computers to which it is connected ot identify other infected systems without the need to communicate with them. Therefore even if a laptop is disconnected or machines are powered down, threats can still be identified and remedial action taken once such machines are back in communication with the central server.

In the same way, the central computer can monitor changes to hardware and software using the information in the database 7 and consequently know when a security breach might have taken place. In this case, the central computer can log the event in question or signal the local computer to take the necessary action to

provide evidence of the possible security breach, eg by taking a web cam picture as well as recording the user name and password.

Another sophisticated ability of the server is to construct a dynamic accurate focused detection formula which can be custom built to detect dynamic changing threats which avoid typical security fingerprinting techniques.

The central server can communicate back to the local computer in the event of a security breach or potential security breach or it can communicate with some other communications device such as a mobile telephone or personal digital assistant by sending an e-mail SMS or fax to indicate the existence of the security breach or potential security breach. In this way, users of portable computers could be warned of security breaches relating to their machine without the need to actually switch it on and establish communications with the central server.

CLAIMS:

1. A computer arrangement comprising a central processor, first memory means for storing an operating system program for the central processor, further memory means for storing data relating to all files used by the computer arrangement, storage means separate from the first storage memory means for storing an engine program for monitoring files used or created by the computer arrangement, and means for communicating the results of the monitoring to the further memory means.

2. A computer arrangement according to claim 1, wherein the engine program is arranged to inspect preselected parts of each file and create a compressed version of the file.

3. A computer arrangement according to claim 1, wherein the further memory means is located remote from the central processor and first memory means.

4. A computer arrangement according to claim 3, wherein the means for communicating results to the further memory means comprises a network connection.

5. A computer arrangement according to claim 1, wherein the storage means is arranged to store two engine programs each of which is capable of being switched off and which are interconnected such that as one is switched off, the other is switched on.

6. A computer arrangement according to claim 1, wherein the engine program is arranged to monitor peripheral devices connected to the arrangement in order to detect the presence of new devices and the absence of previously connected devices.

7. A computer arrangement according to claim 6, wherein the engine program is arranged to record data relating to any change in peripheral devices or files.

8. A computer arrangement according to claim 3, wherein the further memory means is located in a further computer which is arranged to analyse the results of the monitoring.

9. A computer arrangement according to claim 8, wherein the further computer is arranged to transmit instructions to the central processor depending on the results of the analysis.

1/2

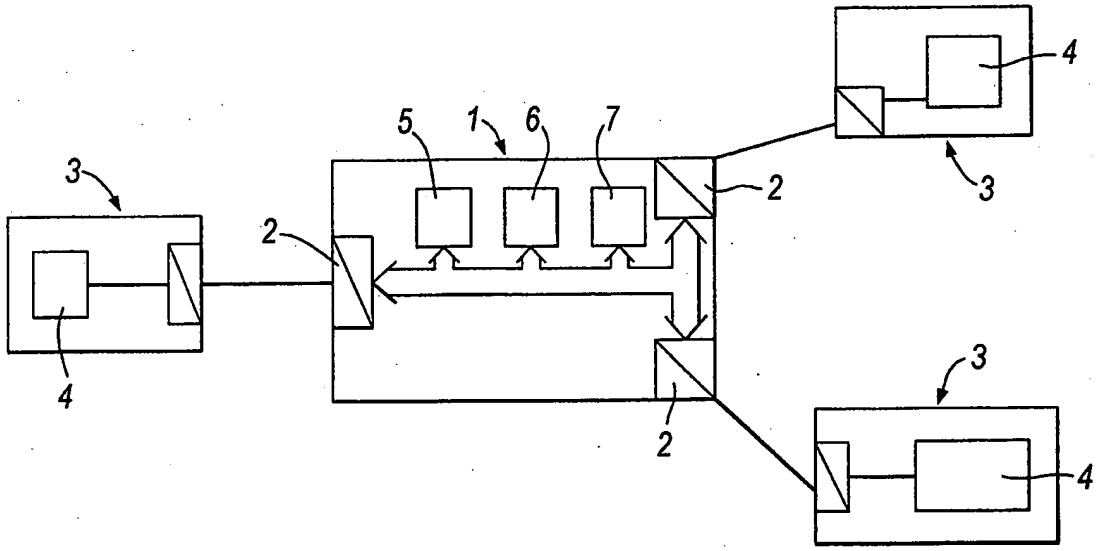


Fig.1

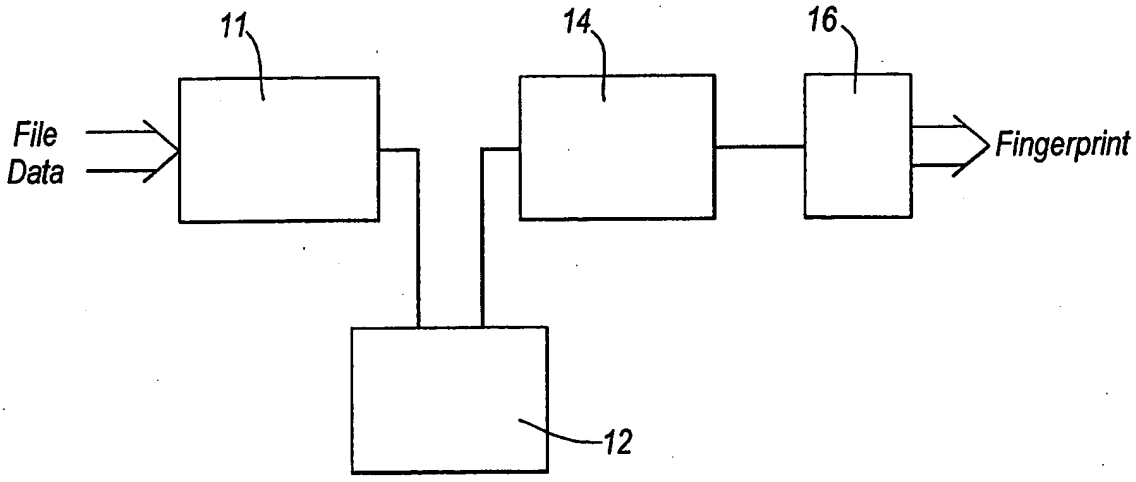


Fig.2

2/2

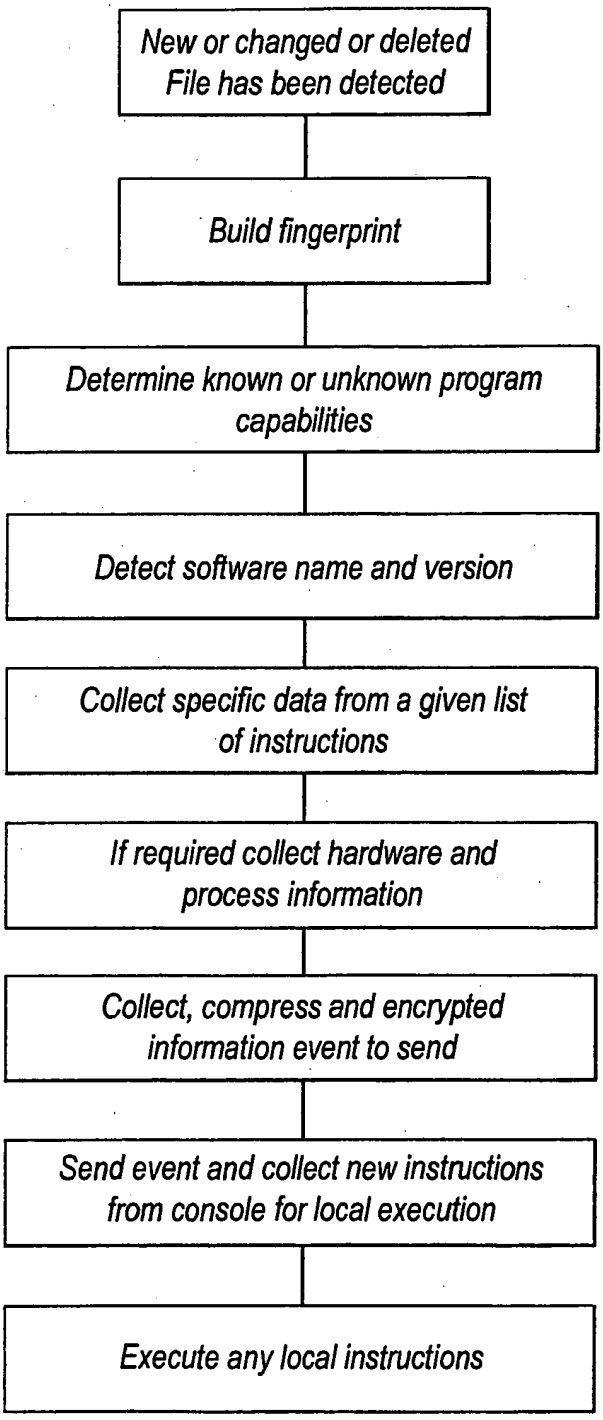

New or changed or deleted File has been detected
Build fingerprint
Determine known or unknown program capabilities
Detect software name and version
Collect specific data from a given list of instructions
If required collect hardware and process information
Collect, compress and encrypted information event to send
Send event and collect new instructions from console for local execution
Execute any local instructions

Fig.3

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 March 2003 (13.03.2003)

PCT

(10) International Publication Number
WO 2003/021402 A3

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number:
PCT/GB2002/004059

(22) International Filing Date:
5 September 2002 (05.09.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0121497.2 5 September 2001 (05.09.2001) GB

(71) Applicant *(for all designated States except US)*: **CRYPTIC SOFTWARE LIMITED** [GB/GB]; The Business Exchange, Rockingham Road, Kettering, Northants NN16 8JX (GB).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **DUKE, David, John** [GB/GB]; 3 The Saplings, Kettering, Northants NN16 9UL (GB).

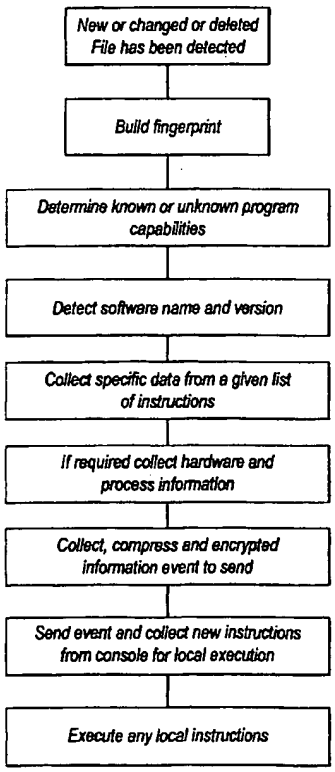
(74) Agent: **CRAWFORD, Andrew, B.**; AA Thornton & Co., 235 High Holborn, London WC1V 7LE (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK,

[Continued on next page]

(54) Title: NETWORK SECURITY



(57) Abstract: A computer network comprises a server computer and a plurality of local computer. Each local computer is provided with an engine program which monitors all file and peripherals and detects changes in the files or configuration. The results f the monitoring are used to create compressed information relating to the files and peripherals which is communicated to the server computer which stores previous s information relating to the files and peripherals of the local computers. The server computer compares the received information with the stored information in order to detect the presence of altered and unwanted files or changes in configuration. In this way both known and unknown viruses can be detected as can un-authorised use of the local computers.

WO 2003/021402 A3

TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
19 August 2004

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

In ional Application No
PCT/GB 02/04059

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
WPI Data, EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 952 521 A (HEWLETT PACKARD CO) 27 October 1999 (1999-10-27) column 2, paragraphs 13,14 column 3, paragraphs 16,18,19 column 4, paragraph 22 -column 6, paragraph 29 column 10, paragraph 44 column 12, paragraphs 55,56 figure 1 ---	1-9
X	EP 0 899 662 A (HEWLETT PACKARD CO) 3 March 1999 (1999-03-03) column 4, paragraph 23 - paragraph 25 column 6, paragraph 31 -column 9, paragraph 43 figures 2,3 --- -/--	1-5,8,9

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 November 2003	04/12/2003

Name and mailing eddress of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Arbutina, L

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 475 625 A (GLASCHICK RAINER) 12 December 1995 (1995-12-12) column 2, line 10 - line 67 column 3, line 34 - line 46 column 5, line 12 - line 30 column 6, line 10 - line 18 column 8, line 17 - line 20 ---	1-4,8,9
X	GB 2 350 704 A (CARTER NICHOLAS PETER) 6 December 2000 (2000-12-06) page 2, line 3 - line 19 page 4, line 8 - line 10 page 7, line 1 - line 11 page 9, line 1 -page 10, line 2 page 25, line 12 -page 26, line 25 ---	1,3,4, 6-8
X,P	WO 02 33525 A (CHUANG SHYNE SONG) 25 April 2002 (2002-04-25) page 6, line 5 - line 25 page 7, line 15 - line 25 page 8, line 6 - line 25 figure 1 ---	1-4,8,9
X	WILLIAMS R N: "DATA INTEGRITY WITH VERACITY" INTERNET, 12 September 1994 (1994-09-12), XP002096828 Retrieved from the Internet: <URL:ftp://ftp.rocksoft.com/clients/rocksoft/papers/vercty10.ps> 'retrieved on 1999-03-16! abstract page 3, right-hand column, paragraph 3.1 page 8, left-hand column, paragraph 5.1 page 10, left-hand column, paragraph 5.8 ---	1
X	MCKOSKY R A ET AL: "A FILE INTEGRITY CHECKING SYSTEM TO DETECT AND RECOVER FROM PROGRAMMODIFICATION ATTACKS IN MULTI-USER COMPUTER SYSTEMS" COMPUTERS & SECURITY. INTERNATIONAL JOURNAL DEVOTED TO THE STUDY OF TECHNICAL AND FINANCIAL ASPECTS OF COMPUTER SECURITY, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, vol. 9, no. 5, 1 August 1990 (1990-08-01), pages 431-446, XP000147838 ISSN: 0167-4048 page 431, left-hand column, paragraph 1 -page 437, left-hand column, paragraph 4 figures 2,5 --- -/--	1

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 094 731 A (WALDIN RAY ET AL) 25 July 2000 (2000-07-25) column 1, line 59 - line 61	2

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0952521 A	27-10-1999	US 6282175 B1	28-08-2001
		DE 69911681 D1	06-11-2003
		EP 0952521 A2	27-10-1999
		JP 2000003347 A	07-01-2000
EP 0899662 A	03-03-1999	EP 0899662 A1	03-03-1999
		DE 69802294 D1	06-12-2001
		DE 69802294 T2	16-05-2002
		EP 1008048 A1	14-06-2000
		WO 9912098 A1	11-03-1999
US 5475625 A	12-12-1995	DE 4101141 C1	02-07-1992
		AT 112078 T	15-10-1994
		DK 567492 T3	09-01-1995
		WO 9213307 A1	06-08-1992
		EP 0567492 A1	03-11-1993
		ES 2059211 T3	01-11-1994
GB 2350704 A	06-12-2000	AU 4941300 A	28-12-2000
		WO 0075782 A1	14-12-2000
WO 0233525 A	25-04-2002	AU 9620501 A	29-04-2002
		WO 0233525 A2	25-04-2002
US 6094731 A	25-07-2000	US 6021510 A	01-02-2000
		CA 2351197 A1	18-05-2000
		EP 1129406 A1	05-09-2001
		WO 0028420 A1	18-05-2000
		AT 215714 T	15-04-2002
		CA 2311658 A1	03-06-1999
		DE 69804658 D1	08-05-2002
		DE 69804658 T2	14-08-2002
		EP 1032877 A1	06-09-2000
		WO 9927450 A1	03-06-1999

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 November 2004 (11.11.2004)

PCT

(10) International Publication Number
WO 2004/097602 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number:
PCT/GB2004/001333

(22) International Filing Date: 29 March 2004 (29.03.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0309463.8 25 April 2003 (25.04.2003) GB

(71) Applicant *(for all designated States except US)*: MESSAGELABS LIMITED [GB/GB]; 1270 Lansdowne Court, Gloucester Business Park, Gloucester LG3 4AB (GB).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: SHIPP, Alexander [GB/GB]; Star Internet, Brighouse Court, Barmwood, Gloucester GL4 3RT (GB).

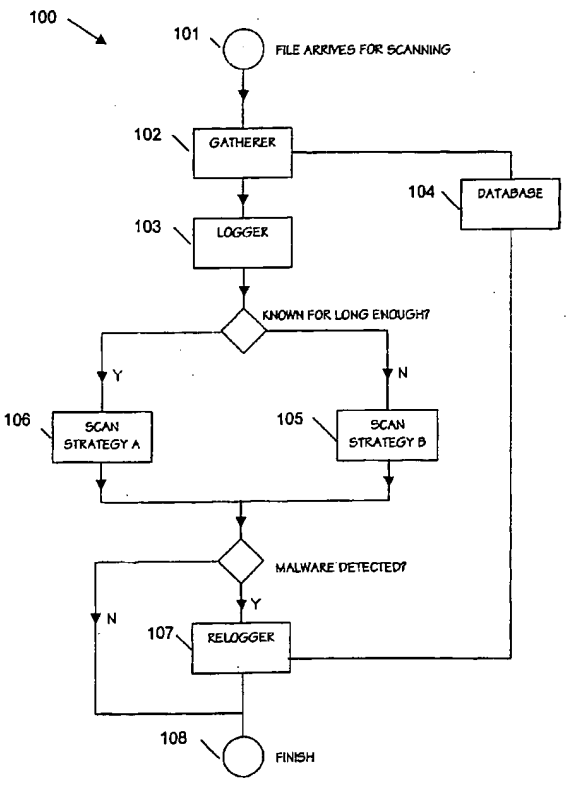
(74) Agents: AYERS, Martyn et al.; J.A. Kemp & Co., 14 South Square, Gray's Inn, London WC1R 5JJ (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),

[Continued on next page]

(54) Title: A METHOD OF, AND SYSTEM FOR, HEURISTICALLY DETERMINING THAT AN UNKNOWN FILE IS HARMLESS BY USING TRAFFIC HEURISTICS



(57) Abstract: A system for processing a computer file to determine whether it contains a virus or other maiware maintains a database of known files which it references to determine whether the file is an instance of a known file, and if so, whether it has been known about long enough that it can be regarded as safe. If it can be regarded as safe, the file is subject to less thorough processing for detecting malware, or no such processing at all.

WO 2004/097602 A2

Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A METHOD OF, AND SYSTEM FOR, HEURISTICALLY DETERMINING THAT AN UNKNOWN FILE IS HARMLESS BY USING TRAFFIC HEURISTICS

The present invention relates to a method of, and system for, heuristically determining that an unknown file is harmless by using traffic heuristics. This technique is especially applicable to situations where files enter a system, are checked, then leave, such as email gateways or web proxies. However, it is not intended to be limited to those situations.

Increasing use of the Internet, personal computers and local- and wide-area networks has made the problem of viruses and other malware (=malicious software) ever more acute.

There are numerous anti-virus packages available. These tend to be produced by specialist companies and are used by businesses and other organisations, home users, and by some internet service providers (ISPs) who scan e-mail and other network traffic on behalf of their customers as a value-added service. As new viruses and other malware arise, the package creators devise ways of detecting them and dealing with them and issue updates to their packages which customers can utilise. A common practice is to make the updates available for download over the internet, from the creator's website or ftp site.

Most anti-virus packages include a file-scanning engine and a database of characteristics of known viruses which are used by the scanning engine to determine whether a file being scanned is, or contains, a virus or other malware, or is likely to do so. The sort of update mentioned above typically includes an update to this database.

The scanning engine may implement a variety of heuristics to be applied, possibly selectively, to a file being scanned. Probably the most familiar kind of heuristic is signature detection, in which the file is examined for the occurrence of sequences or bytes, or patterns of such sequences, which are known to be characteristic of viruses in the package's virus database, though many other heuristics also exist, which can be used as well as or instead of signature detection.

The amount of malware in existence increases all the time, which makes the computational and storage resources necessary to detect it increasingly burdensome, particularly where the throughput of files is high, as is the case with ISPs.

According to the present invention, there is provided a system for processing a computer file to determine whether it contains a virus or other malware comprising:

a) means for generating data with regard to the file to characterise its identity and for thereby referencing a computer database to determine whether it is an instance of a known file;

b) means for selectively subjecting the file to a number of heuristic procedures to determine whether or not it contains, or is likely to contain, malware; and

c) means for determining, in dependence upon the record, if any, of the file in the database, whether the file can be regarded as safe and for controlling the means b) such that the file, if the file is to be regarded as safe, is either subject to less thorough processing than if it were not so regarded or not subject to processing by the means b) at all.

The invention also provides a method of processing a computer file to determine whether it contains a virus or other malware comprising:

a) generating data with regard to the file to characterize its identity and for thereby referencing a computer database to determine whether it is an instance of a known file;

b) selectively subjecting the file to a number of heuristic procedures to determine whether or not it contains, or is likely to contain, malware; and

c) determining, in dependence upon the record, if any, of the file in the database, whether the file can be regarded as safe and conducting the step b) such that the file, if the file is to be regarded as safe, is either subject to less thorough processing than if it were not so regarded or not subject to processing by the step b) at all.

The invention will be further described by way of non-limitative example with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of a system embodying the present invention.

Figure 1 illustrates one form of a system 100 according to the present invention, which might be used, for example by an ISP as part of a larger anti-virus scanning system which employs additional scanning methods on files which are not filtered out as "safe" by the system of Figure 1. Files considered safe can if desired be subject to further processing to check for malware, but less intensively so than files not considered safe.

The rationale of the system 100 is that if a particular file has been scanned by a virus scanner, and found to be harmless the two possibilities exist: The file could really be harmless, or the file could contain something nasty which the virus scanner is as yet unable to detect.

As time goes by, the file (or another instance of it) may be scanned again, and still found to be harmless.

This time the file is more likely to really be harmless, rather than to be malware which the virus scanner is as yet unable to detect. This is because virus scanners are continually updated to detect new malware as the new malware is discovered. The longer the time that passes, the more likely it is that a suspicious person will submit a file containing malware to the developers of the scanner, who will analyse the file, and update their scanner to detect it.

As more and more instances of the file are scanned coming from different sources, then if these are all flagged as harmless, it becomes less and less likely the file is malware. This is because the more copies of a piece of malware exist, the more likely it is that somebody will become suspicious and submit a copy to scanner developers.

It is therefore possible to create a feedback engine which logs copies of files scanned, together with the source they originated from. The log is updated and examined as each file is scanned, and if files are found which have come from a sufficient number of sources, in sufficient quantities, and over a long enough period of time, then that file can be flagged as 'known about long enough'. This might mean that future copies are then not scanned further, or are scanned using less rigorous scans with fewer heuristics enabled, or are only scanned if the scanner has been updated since the last scan.

The system 100 operates according to the following algorithm:

1) A file arrives at an input 101 for scanning, perhaps as an email attachment, or a web download.

2) A 'gatherer' module 102 gathers information about the file, such as a checksum of the file contents and the source of the file (eg the IP address). The source may be passed through a one way trapdoor function, generating a hash, in order to preserve confidentiality. The information gathered is for comparison with information stored in a database 104 about known files so that it can be determined whether the file under consideration is an instance of a file recorded in database 104.

3) Based on the checksum derived by gatherer 102, a 'logger' module 103 updates the database 104 to indicate that one more instance of the file has been detected.

The logger 103 saves the current 'last seen' date as the 'previously scanned date', and then updates the 'last seen' date of the file's entry in the database 104. If this is the first instance of the file, the logger 103 also updates a 'first seen' date. If this is a new source, the logger 103 adds the source to a list, stored in database 104, of sources the file has originated from.

4) From the information stored (number of copies of the file seen, length of time file has been known about, number of sources) the logger 103 calculates whether the file has been 'known about long enough'. For this purpose, the logger 103 may assign a weighted score to each of these factors individually and then calculate an overall score by combining the weighted scores, e.g. by adding them up.

5) If the file has not been known about long enough, scan strategy B is undertaken at 105. This will be the most complete scan available.

6) If the file has been known about long enough, scan strategy A is undertaken at 106. This will be a less thorough scan than strategy B. This will be site-dependent as to how less thorough a scan is desired. At the extreme it might involve no scanning at all. It might involve scanning with fewer scanners; with heuristics not fully enabled or turned off; or (assuming the file has been seen at least once before) only with scanners that have been updated since the 'previously scanned date'

The scanning techniques available to the scanning strategies A and B may include any suitable heuristics, such as signature-based scanning, generating checksums from the file or selected regions if it, etc.

7) Following the scan strategy A or B, then if no malware was detected, processing stops at 108.

8) If malware was detected, then a 'relogger' module 107 is invoked. This clears out all database entries in database 104 which are associated with the file so that it cannot become 'known about long enough' in the future.

9) Processing of the current file finishes at 108, whereupon the system can retrieve the next file from a queue of files waiting to be processed.

CLAIMS

1. A system for processing a computer file to determine whether it contains a virus or other malware comprising:

a) means for generating data with regard to the file to characterise its identity and for thereby referencing a computer database to determine whether it is an instance of a known file;

b) means for selectively subjecting the file to a number of heuristic procedures to determine whether or not it contains, or is likely to contain, malware; and

c) means for determining, in dependence upon the record, if any, of the file in the database, whether the file can be regarded as safe and for controlling the means b) such that the file, if the file is to be regarded as safe, is either subject to less thorough processing than if it were not so regarded or not subject to processing by the means b) at all.

2. A system according to claim 1 wherein the controlling means c) controls the means b) in dependence on factors including the length of time for which the database indicates that the file has been known without malware-containing instances of it being detected.

3. A system according to claim 1 or 2 wherein the controlling means c) controls the means b) in dependence on factors including sources, recorded in the database, from which instances of the file have originated.

4. A system according to claim 1, 2 or 3 wherein the controlling means c) controls the means b) in dependence on factors including the number of times, recorded in the database, of instances of the file have been processed.

5. A system according to any one of the preceding claims, and including means for updating the database in dependence upon the result of the processing of the file by the means b).

6. A system according to claim 5 wherein the updating of the database, in the event of the means b) determining that the file contains, or is likely to contain, malware is such that the record thereof in the database is deleted, or updated so that it is no longer taken be safe.

7. A method of processing a computer file to determine whether it contains a virus or other malware comprising:

a) generating data with regard to the file to characterise its identity and for thereby referencing a computer database to determine whether it is an instance of a known file;

b) selectively subjecting the file to a number of heuristic procedures to determine whether or not it contains, or is likely to contain, malware; and

c) determining, in dependence upon the record, if any, of the file in the database, whether the file can be regarded as safe and conducting the step b) such that the file, if the file is to be regarded as safe, is either subject to less thorough processing than if it were not so regarded or not subject to processing by the step b) at all.

8. A method according to claim 7 wherein the determining step c) controls the step b) in dependence on factors including the length of time for which the database indicates that the file has been known without malware-containing instances of it being detected.

9. A method according to claim 7 or 8 wherein the determining step c) controls the step b) in dependence on factors including sources, recorded in the database, from which instances of the file have originated.

10. A method according to claim 7, 8 or 9 wherein the determining step c) controls the step b) in dependence on factors including the number of times, recorded in the database, instances of the file have been processed.

11. A method according to any one claims 7 to 10, and including the step of updating the database in dependence upon the result of the processing of the file by the step b).

12. A method according to claim 11 wherein the updating of the database, in the event of the step b) determining that the file contains, or is likely to contain, malware is such that the record thereof in the database is deleted, or updated so that it is no longer taken be safe.

13. A system for processing a computer file to determine whether it contains a virus or other malware substantially as hereinbefore described and with reference to the accompanying drawings

14. A method of processing a computer file to determine whether it contains a virus or other malware substantially as hereinbefore described and with reference to the accompanying drawings

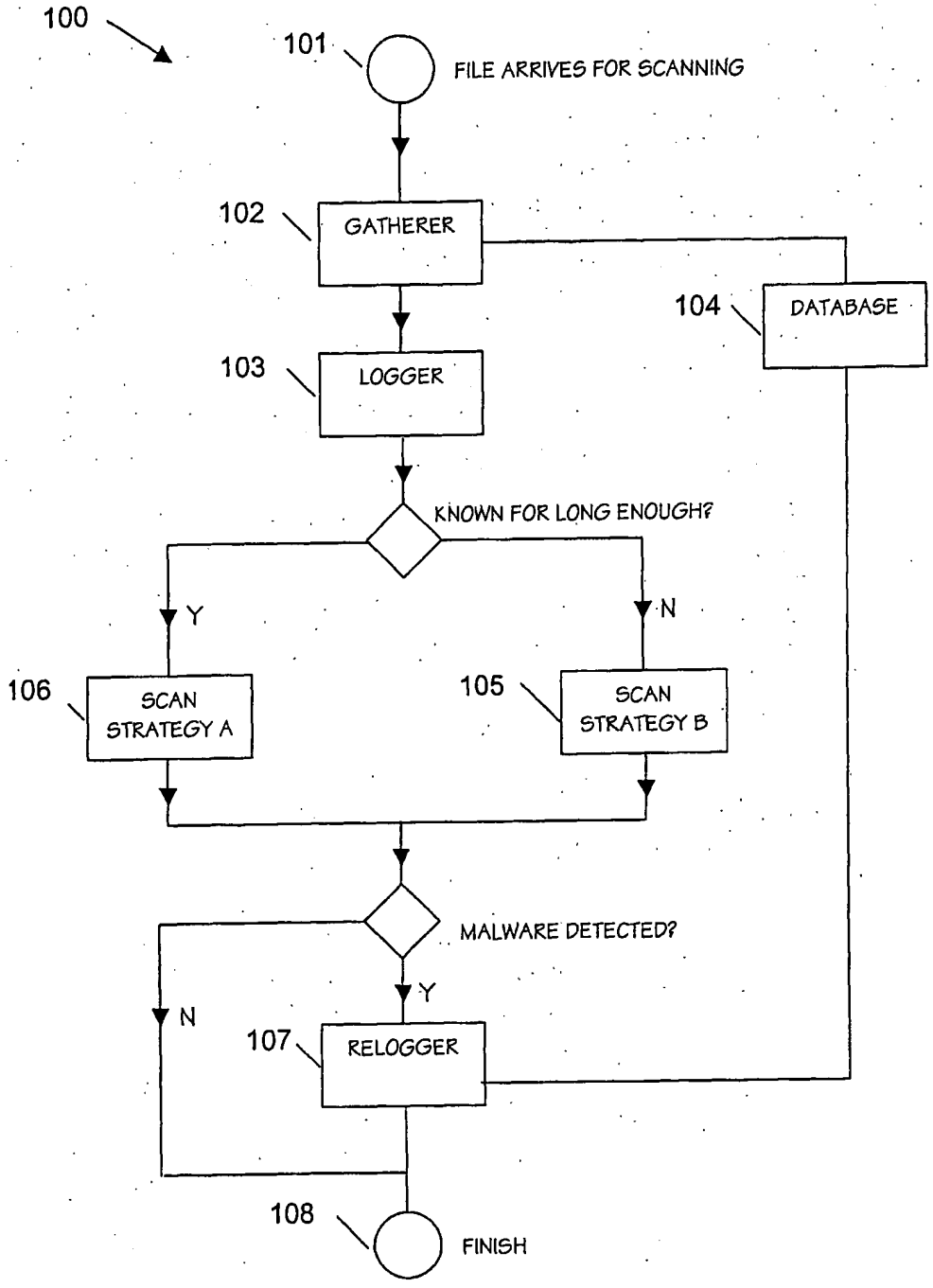

100
101
FILE ARRIVES FOR SCANNING
102
GATHERER
104
DATABASE
103
LOGGER
KNOWN FOR LONG ENOUGH?
Y
N
106
SCAN STRATEGY A
105
SCAN STRATEGY B
MALWARE DETECTED?
Y
N
107
RELOGGER
108
FINISH

Fig.1

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 April 2002 (25.04.2002)

PCT

(10) International Publication Number
WO 02/33525 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/SG01/00213

(22) International Filing Date: 17 October 2001 (17.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
200005973-3 17 October 2000 (17.10.2000) SG

(71) Applicant and
(72) Inventor: CHUANG, Shyue-Song [SG/SG]; Singapore Post Centre, P.O. Box 493, Singapore 914017 (SG).

(74) Agent: DREW & NAPIER LLC; 20 Raffles Place, #17-00 Ocean Towers, Singapore 048620 (SG).

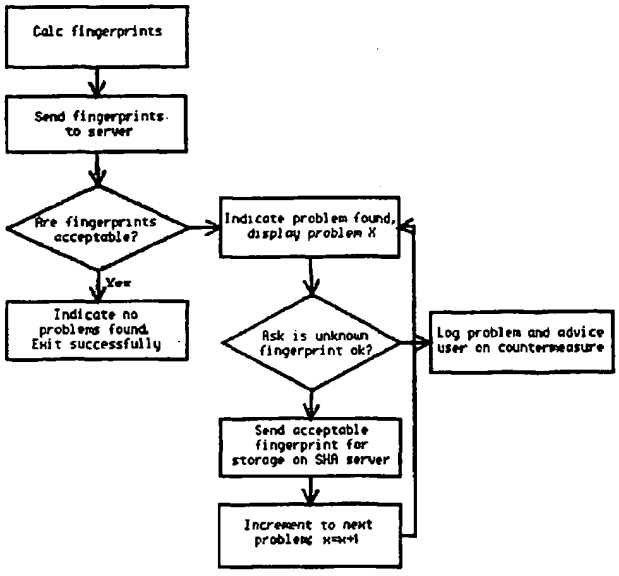
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A METHOD AND SYSTEM FOR DETECTING ROGUE SOFTWARE



(57) Abstract: A method of detecting rogue software includes the step of creating a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, wherein the pre-calculated fingerprints are calculated using one or more cryptographic formulae. The one or more cryptographic formulae are then used to calculate fingerprints of files on a computer system which is to be scanned for rogue software. The fingerprints calculated for the files on the computer system are compared with the fingerprints which are contained in the first database of pre-calculated fingerprints. Files on the computer system which may contain rogue software are identified by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

WO 02/33525 A2

A METHOD AND SYSTEM FOR DETECTING ROGUE SOFTWARE

Field of the invention

The present invention relates to a method and system for detecting rogue software such as trojan horses, root-kits, viruses and other unauthorized software which masquerades as valid software) on a computer system or data processing device such as a personal digital assistant. It relates particularly but not exclusively to a method and system for calculating and comparing fingerprints for files which are used either on a stand-alone computer system or on a computer system which is part of a computer network.

Back-ground to the Invention

Undesired rogue software is a nuisance and security threat. As computer systems and other information devices become even more interconnected with modern day networking technology and the Internet, the danger from rogue software has magnified considerably. Instead of being programmed to do damage once, today's rogue software can continue to receive commands and do the bidding of an unauthorized intruder for an extended period of time, effectively giving the creator of the rogue software continuous illegal access to a computer system.

One example of rogue software is the so-called trojan horse. Such software may be installed by innocent users unknowingly (whether via social engineering or otherwise) or it may be installed by an attacker when a system has been broken into. These trojan horses are back doors which allow an attacker to reconnect back into the compromised system and illegally access files and make unauthorized changes.

A trojan horse typically consists of new software and has new functionality. It is installed on a compromised system and disguised to look like original system software whenever necessary, so as to avoid detection. Sometimes, the trojan

horse is a modified piece of original system software and is almost identical to the one it replaced. However, other techniques are also used to obfuscate its existence.

Once the trojan horse is installed, a user continues operating his/her system without knowing that an intruder is now able to access his/her data illegally whilst remaining hidden. These are highly relevant problems which are encountered on a day to day basis. Trojan horses such as "Back Orifice" or "Netbus" hit PC systems in the late 1990s, and "root-kits" are a concern for various UNIX systems.

Normally, according to current practice, little is done to prevent or detect such rogue software. Anti-virus vendors maintain databases of rogue software signatures, and their software searches for files on a system associated with all known rogue software. Unfortunately, this technique has inherent scaling problems - the more signatures there are, the slower the scan process for each file. More importantly, the only rogue software types that can be detected are the ones that the anti-virus vendors know about. If the rogue software is, as an example, a custom trojan horse built by an expert professional hacker for penetrating a specific target, none of the anti-virus vendors will know about it. Therefore none of the anti-virus tools will be looking for it and the attacker and their trojan horse will exist completely undetected.

A more recent incremental innovation with this technology involves smarter scanning engines. Aside from looking for signatures of known rogue software, they are also able to look for software code that appears to be doing unusual things. This allows the scanning engine to detect additional rogue software that may not be known and whose signatures may not be in the database. However, this approach also has limitations. Trojan horses can be encrypted or compressed using special proprietary algorithms or encryption keying material. The rogue software is shipped in an encrypted and/or compressed format where it appears to be gibberish to a scanner. This rogue software is then decompressed or decrypted upon execution on the victim's computer system. A single trojan horse can thus be encrypted or compressed into thousands of

possibilities, each with its own unique signature. Traditional scanning technology will fail miserably when attempting to detect this type of rogue software, since there is no way that anti-virus engineers can keep track of thousands of mutations of the same piece of rogue software.

To summarise, today's scanning techniques for detecting rogue software fail for two main reasons:

1. They cannot detect unknown rogue software that has not already been identified. This is a serious problem because it is this kind of rogue software that may involve professional hackers and therefore warrant serious attention.
2. They cannot efficiently detect rogue software which has mutated, using new methods of compression and/or encryption. This problem exists to a large extent even for rogue software that is already known.

Another approach for detecting rogue software is to ensure that a system's files have not been altered, rather than looking for signatures of rogue software. If a system has no added files and all files remain unchanged from their original, unaltered state, it is clear that no rogue software is present on the system.

Academic work at Purdue University by Gene Kim and Gene Spafford resulted in a product called Tripwire which is now commercially sold. The product requires users to generate a database containing fingerprints of files on a system when the system is freshly loaded and in a pristine state. Subsequently, fingerprints can be recalculated and compared with the database of original pristine fingerprints and detect changes which have been made to the computer system.

This technology requires users to generate a database of fingerprints of a system's files while it is still pristine and free from alteration. This is not always feasible because many systems would already have been placed on public networks and exposed to risk for some time (often years). Since changes can be detected only by calculating new fingerprints and comparing them with the

database of original fingerprints, any rogue software which already exists when the original fingerprint database was generated will not be detected.

Existing products do not use a central database of fingerprints which are acceptable for a broad collection of system and application software. Therefore users need to make the following tedious and expensive steps when installing a software upgrade:

1. Schedule downtime in which to create the new database of fingerprints;
2. Re-calculate fingerprints to ensure that no rogue software has been added;
3. Install the software upgrade; and
4. Generate a new fingerprint database.

This is often a time-consuming and costly exercise.

Summary Of The Invention

Therefore it is an object of the present invention to provide a more reliable and effective method of identifying rogue software on a computer system or device, especially rogue software with unknown signatures or characteristics. The invention is preferably usable on systems or devices that have already been exposed to risk of intrusion by rogue software, and in cases where no fingerprints for the files on the system were calculated or archived when the system or device was known to be in a pristine state.

According to a first aspect of the invention, there is provided a method for detecting rogue software including the steps of:

(a) creating a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, wherein the pre-calculated fingerprints are calculated using one or more cryptographic formulae;

(b) using the one or more cryptographic formulae to calculate fingerprints of files on a computer system which is to be scanned for rogue software;

(c) comparing the fingerprints calculated for the files on the computer system with the fingerprints which are contained in the first database of pre-calculated fingerprints; and

(d) identifying files on the computer system which may contain rogue software by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

According to a second aspect of the invention, there is provided a system for detecting rogue software including:

(a) a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, the fingerprints having been calculated using one or more cryptographic formulae;

(b) a software component which uses one or more cryptographic formulae to calculate fingerprints for files on a computer system; and

(c) a software component which compares the calculated fingerprints for the files on the computer system with corresponding pre-calculated fingerprints stored in the first database, such that files on the computer system which may contain rogue software are identified by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

Brief Description Of The Drawings

The invention will now be described in greater detail by reference to the drawings which show an example form of the invention. It is to be understood that the particularity of the drawings does not supersede the generality of the foregoing description of the invention.

Figure 1 is a schematic representation of a client portion and server portion of a security system on a Redhat Linux platform connected via a network according to a preferred embodiment of the present invention.

Figure 2 illustrates a more detailed data flow diagram relating to the schematic representation of Figure 1.

Detailed Description Of A Preferred Embodiment Of The Invention

Figure 1 is a schematic representation of a client portion and server portion of a security system on a Redhat Linux platform connected via a network 10 according to a preferred embodiment of the present invention. The system includes a client 12, a server 14 and a database of acceptable file fingerprints 16. Communication between the client 12 and server 14 may be via the Internet 18, using the TCP/IP protocol. The system is first set up by calculating and archiving fingerprints for all files relating to operating system or application software used in a typical Redhat Linux system, perhaps from original Redhat CDs or other secure software distribution methods. This software can be installed on test systems (not shown in Figure 1) so that the new files added or replaced can be fingerprinted and profiled. These new fingerprints, the file location of each file added or replaced, and other information, can then be stored in the database of acceptable file fingerprints 16. An alternative method that eradicates the need for software installation on a test system involves the use of a custom developed program that understands the RPM (Redhat Package Manager) format of software packages on the Redhat Linux CD. By examining each RPM software package's installation instructions, the program determines the file location and calculates the fingerprints of each file to be installed. This information and other information is then stored in the database of acceptable file fingerprints 16.

The fingerprints are preferably calculated using one or more cryptographic formulae. In the preferred embodiment, such cryptographic formulae may include hash functions to generate hash values for each file, or asymmetric cryptographic functions to generate digital signatures for each file. The original version of the files as well as patches, updates/upgrades of all types of operating system or application software should be fingerprinted. System performance and reliability will improve as more operating system and application software is fingerprinted and archived.

Hashing is a contraction of the file contents created by a cryptographic hash function. A hash value (or simply hash) is the output when an arbitrary input is passed into a hash function. The hash is substantially smaller than the input itself, and is generated by a formula in such a way that it is extremely unlikely that slight modifications of the input will result in the same hash. Hashes conventionally play a role in security systems where they are used to ensure that transmitted messages have not been tampered with. As an illustration, a sender generates a hash of the message and sends it with the message itself. The recipient then calculates another hash from the received message, and compares the two hashes. If they are the same, there is a very high probability that the message was transmitted intact. There may be other equivalent methods for calculating fingerprints that may be implemented as the relevant technology develops.

The system's client component is installed on the client 12 that requires file integrity protection. During the first time the client component is executed, the client software recurses through the file system and calculates and stores the cryptographic hash of every single file on the system. When the file system has been completely traversed, the client software makes a secure TCP/IP connection via the Internet 18 to the server component on the server, which usually resides on premises remote from the client component. However, the client component need not be physically located remote from the server component.

For security purposes, it is preferable that bi-directional authentication takes place between the client component and the server component before any further communication and this can be done with SSL (Secure Socket Layer) or TLS (Transport Layer Security). In a nutshell, the server presents its digital certificate to the client software and the client uses its hardwired CA (Certificate Authority) public credentials to verify the CA signature on the server's certificate. If the signature is authentic and the server's address matches the machine which the certificate was issued to, the client can be certain that the server is who it claims to be. Subsequently, the same thing happens in the reverse

direction. The client presents the server with its digital certificate and the server goes through the same process to verify that the client is who it claims to be. This practice is very common today and is an industry standard method of mutually authenticating two nodes communicating with one another. Other authentication methods may also be used.

The calculated hash results and gathered basic client system information from the client 12 are then transferred to the server 14 for validation. On the server 14, each hash result for each file on the client system is compared against what are the expected hash values given certain parameters such as the client system's operating system version and software patch/update level. This expected hash information is fetched from the database of acceptable file fingerprints 16 which houses all the pre-calculated hash values for all files in various operating systems and applications.

A report is then generated on the fly and returned to the client 12. This report lists the files on the client which are possibly unsafe since they do not represent authentic software from the vendor. There are 3 possible results for each file:

(a) the hash result matches so the given file on the client is definitely authentic;

(b) the database of acceptable fingerprints 16 has no information on such a file in the database and it is uncertain if the file is authentic;

(c) the hash result does not match the fingerprint in the database of acceptable fingerprints 16 and the file is suspicious.

Armed with this report, the systems administrator for the client server 12 can then verify each of the files in categories (b) and (c). Outcomes in categories (b) and (c) are typically from files that are part of an internal customer specific application that the database 16 will not contain. If the administrator verifies the hash with the owners of the application, the authenticity of the file can be determined. This should be done for all questionable files in the report so that a client system can be certified as 100% authentic. If some of the questionable files cannot be resolved via these means, it is likely that they have been

augmented by rogue software and should be replaced or the system should be reinstalled.

Using additional management software, the administrator can then check off all remaining questionable files as acceptable and the security system will take the additional hashes into account in all subsequent runs. These additional hashes can then be stored in a second database (not shown in Figure 1) so that they can be considered when checking other systems from the same customer - this is a configurable feature.

Figure 2 illustrates a more detailed data flow diagram relating to the schematic representation of Figure 1. Using the database of pre-calculated acceptable fingerprints 16, the system will be able to determine if any given file on a client's system is authentic, i.e. not invaded by rogue software. When comparisons are done, file location, time stamps, platform information, user preferences and other parameters can also be taken into consideration.

The system should be continuously updated with new fingerprint information in the database of acceptable file fingerprints 16 as new software and updates become available. The system thus provides pristine fingerprint information that is made available to the file integrity checking software installed on a client's computer system. Instead of identifying bad files, the system therefore ensures that the data is good. Instead of requiring users to have generated a fingerprint database some time back, the system provides pre-calculated fingerprints and greatly reduces the barriers to adoption of this important file integrity technology.

In addition to the above, the system may also store fingerprints of various customers' files in a separate database (not shown in Figure 1) so that the system can provide heuristic, statistically based best effort guesses on whether a certain fingerprint is acceptable for a given file.

In the above example, in the event of a questionable outcome (categories (b) and (c)) where the system does not have pre-calculated hash information on a

certain file on the client, the system may also render a heuristic result on whether a file is safe. This result can be provided by accessing the second database (not shown in Figure 1) which contains hashes that the customer's administrators have confirmed to be acceptable. For example, if the system does not know about whether a file such as "/usr/bin/myspecialprogram" should have a hash result of "xyz", it can inform the administrator, and also point out one of the following:

1) no other systems in the client's class have such a file so authenticity is uncertain;
2) other systems in the client's class which have such a file do not have a hash of "xyz" so the file is suspicious;
3) other systems in the client's class which have such a file have the same hash of "xyz" so the file is probably safe.

This information can also be provided with a percentage figure so that administrators have a best guess of where they stand before engaging in manual verification as described above.

As the client base grows, this information will allow the system to make increasingly improved guesses. The system can render an opinion along the lines of "10,000 other customers have this file and 9,985 of them have the same fingerprint, so your file is probably safe" - perhaps a common application whose fingerprint that does not already exist in the first database 16. Such information, while not substantive, allows users to zoom into more critical anomalies on their systems sooner. For example, consider this other response: "10,000 other customers have this file and no one has the same fingerprint as you. Worse yet, all these 10,000 customers have the same fingerprint so your file is most probably unsafe." The system can thus provide a percentage or quantifiable risk rating in either a numeric fashion or with the use of colours.

From the client's point of view, the advantage is that even systems currently deployed in risky public network environments can be easily reliably scanned and put onto a file integrity protection regime without re-installation to assure a pristine state and with significantly reduced downtime. As customers apply upgrades to their systems, the system will similarly be able to verify that new

software being installed is authentic since the fingerprints of the new software should be in the system's database 16. Conversely, the system can be programmed to warn the user if the update contains software the system does not believe is authentic.

While a particular embodiment of the invention has been shown and described, it will be obvious to those skilled in the art that changes and modifications of the present invention may be made without departing from the invention in its broader aspects. As such, the scope of the invention should not be limited by the particular embodiment and specific construction described herein but should be defined by the appended claims and equivalents thereof. Accordingly, the aim in the appended claims is to cover all such changes and modifications as fall within the spirit and scope of the invention.

Claims:

1. A method of detecting rogue software including the steps of:

(a) creating a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, wherein the pre-calculated fingerprints are calculated using one or more cryptographic formulae;

(b) using the one or more cryptographic formulae to calculate fingerprints of files on a computer system which is to be scanned for rogue software;

(c) comparing the fingerprints calculated for the files on the computer system with the fingerprints which are contained in the first database of pre-calculated fingerprints;

(d) identifying files on the computer system which may contain rogue software by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

2. A method according to claim 1 including the further step of generating a list of questionable files on the computer system for which the calculated fingerprints do not correspond to pre-calculated fingerprints stored in the first database.

3. A method according to claim 1, wherein the cryptographic formulae used in the pre-calculation of fingerprints which are stored in the first database and in the calculation of fingerprints for files on the computer system, use one or more hash functions to generate hash values for each file.

4. A method according to claim 1, wherein cryptographic formulae used in the pre-calculation of fingerprints which are stored in the first database and in the calculation of fingerprints for files on the computer system, use one or more asymmetric cryptographic functions to generate digital signatures for each file.

5. A method according to claim 2 wherein questionable files are considered by a system administrator and may be marked by the system administrator as acceptable.

6. A method according to claim 5 wherein the fingerprints for questionable files which are accepted by the system administrator are stored in a second database.

7. A method according to claim 6 which includes the step of calculating the probability that a questionable file has been corrupted by rogue software, by comparing its fingerprint with fingerprints for similar files that have previously been accepted by the system administrator and stored in the second database.

8. A method according to claim 7 wherein the system provides statistical information regarding:

(a) the number of fingerprints in the second database which represent files with the same characteristics as the questionable file;

(b) the number of fingerprints in the second database which are identical to the fingerprint of the questionable file;

(c) the number of fingerprints in the second database which are different to the fingerprint of the questionable file.

9. A method according to claim 5 wherein files that are not acceptable are replaced or reinstalled.

10. A method according to claim 6 wherein the step of calculating fingerprints for files on the computer system and the step of comparing fingerprints on the computer system with corresponding pre-calculated fingerprints stored on the first database are both implemented by the computer system and wherein verification of questionable files takes place before fingerprints from the computer system are added to the second database.

11. A method according to claim 10 wherein the computer system is physically remote from the first database and communication between the

computer system and the first database takes place over a network such as the Internet.

12. A method according to any one of claims 6 to 8 wherein the step of calculating fingerprints for the files on the computer system is implemented by the computer system and the step of comparing the fingerprints which represent files on the client system with corresponding pre-calculated fingerprints stored in the database is implemented by a server, and wherein verification of questionable files takes place between the computer system and the server before the corresponding fingerprints are transferred from the computer system to the second database.

13. A method according to claim 12 wherein the computer system is physically remote from the server and communication between them takes place over a communications network such as the Internet.

14. A system for detecting rogue software including:

(a) a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, the fingerprints having been calculated using one or more cryptographic formulae;

(b) a software component which uses one or more cryptographic formulae to calculate fingerprints for files on a computer system; and

(c) a software component which compares the calculated fingerprints for the files on the computer system with corresponding pre-calculated fingerprints stored in the first database, such that files on the computer system which may contain rogue software are identified by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

15. A system according to claim 14 including a software component which generates a list of questionable files for which the calculated fingerprints do not correspond to the pre-calculated fingerprints which are stored in the first database.

16. A system according to claim 14 or 15 wherein the software components are installed on the computer system.

17. A system according to claim 14 wherein the pre-calculation of fingerprints which are stored in the first database and calculation of fingerprints for files on the computer system use one or more hash functions to generate hash values for each file.

18. A system according to claim 14 wherein the pre-calculation of fingerprints which are stored in the database and calculation of fingerprints for files on the computer system use one or more asymmetric cryptographic functions to generate digital signatures for each file.

19. A system according to claim 15 further including a second database in which fingerprints of questionable files which are found to be acceptable by a system administrator are stored.

20. A system according to claim 15 wherein the system calculates the probability that a questionable file is a file that has been corrupted by rogue software, by comparing its fingerprint with fingerprints for similar files that have been verified and stored in the second database.

21. A system according to claim 20 wherein the system produces statistical information regarding:

(a) the number of fingerprints in the second database which represent files with the same characteristics as the questionable file;

(b) the number of fingerprints in the second database which are identical to the fingerprint of the questionable file;

(c) the number of fingerprints in the second database which are different to the fingerprint of the questionable file.

22. A system according to claim 19 wherein files that are not acceptable are replaced or reinstalled.

23. A system according to any one of claims 14 to 22 wherein the step of calculating fingerprints of files on the computer system and the step of comparing the fingerprints on the computer system with corresponding pre-calculated fingerprints stored on the database are both implemented by the computer system and wherein verification of questionable files takes place before fingerprints from the computer system are added to the second database.

24. A system according to claim 23 wherein the computer system is physically remote from the first database and communication between the computer system and the first database takes place over a network such as the Internet.

25. A system according to any one of claims 14 to 20 wherein the step of calculating fingerprints for the files on the computer system is implemented by the computer system and the step of comparing the fingerprints which represent files on the computer system with corresponding pre-calculated fingerprints stored in the first database is implemented by a server and wherein verification of questionable files takes place between the computer system and the server before the corresponding fingerprints are transferred between them.

26. A system according to claim 25 wherein the computer system is physically remote from the server and communication between them takes place over a communications network such as the Internet.

1 / 1

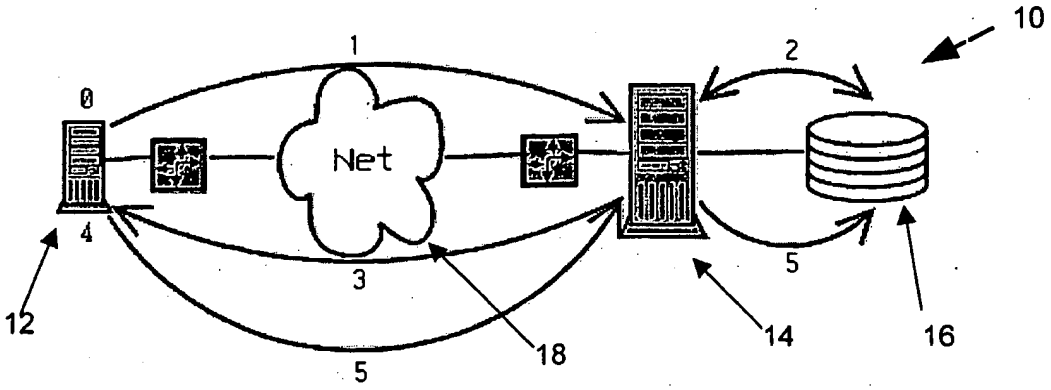


FIGURE 1

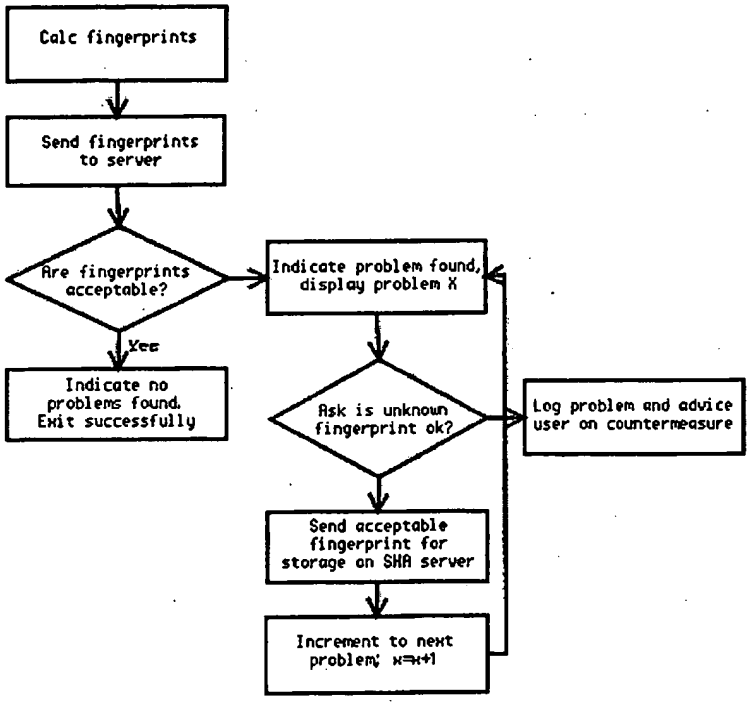


FIGURE 2

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 April 2002 (25.04.2002)

PCT

(10) International Publication Number
WO 02/033525 A3

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/SG01/00213

(22) International Filing Date: 17 October 2001 (17.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
200005973-3 17 October 2000 (17.10.2000) SG

(71) Applicant and
(72) Inventor: CHUANG, Shyne-Song [SG/SG]; Singapore Post Centre, P.O. Box 493, Singapore 914017 (SG).

(74) Agent: DREW & NAPIER LLC; 20 Raffles Place, #17-00 Ocean Towers, Singapore 048620 (SG).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

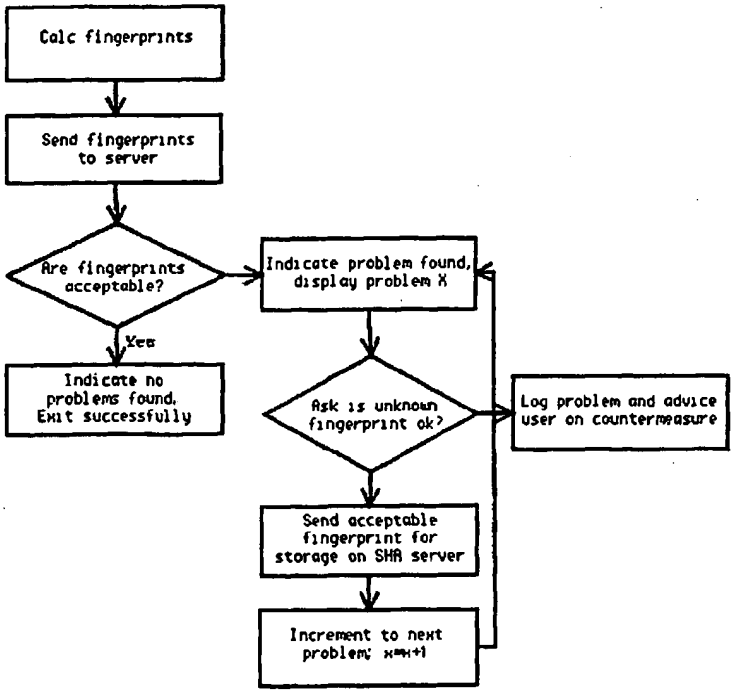
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
6 March 2003

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A METHOD AND SYSTEM FOR DETECTING ROGUE SOFTWARE



(57) Abstract: A method of detecting rogue software includes the step of creating a first database containing pre-calculated fingerprints for each file relating to typical operating systems and application software, wherein the pre-calculated fingerprints are calculated using one or more cryptographic formulae. The one or more cryptographic formulae are then used to calculate fingerprints of files on a computer system which is to be scanned for rogue software. The fingerprints calculated for the files on the computer system are compared with the fingerprints which are contained in the first database of pre-calculated fingerprints. Files on the computer system which may contain rogue software are identified by identifying files the calculated fingerprints of which do not correspond to the pre-calculated fingerprints which are stored in the first database.

WO 02/033525 A3

INTERNATIONAL SEARCH REPORT

Int al Application No
PCT/SG 01/00213

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WILLIAMS R N: "DATA INTEGRITY WITH VERACITY" INTERNET, 'Online! 12 September 1994 (1994-09-12), XP002096828 Retrieved from the Internet: <URL:ftp://ftp.rocksoft.com/clients/rocksoft/papers/vercty10.ps > 'retrieved on 1994-09-12!	1-6,9, 14-20,22
Y	page 4, left-hand column -right-hand column, paragraph 3 page 5, left-hand column, paragraph 6 - paragraph 9 page 6, right-hand column, paragraph 6 -page 8	7,8, 10-13, 20,21, 23-26

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 November 2002	09/12/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Powell, D

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	page 10	
X	"MECHANISM FOR TRUSTED COMPUTING BASE DEFINITION AND CHECKING" IBM TECHNICAL DISCLOSURE BULLETIN, IBM CORP. NEW YORK, US, vol. 34, no. 9, 1 February 1992 (1992-02-01), pages 188-191, XP000300643 ISSN: 0018-8689 page 188, paragraph 3 - paragraph 4 page 191	1-5,9, 14-19,22
Y	US 6 021 491 A (RENAUD BENJAMIN J) 1 February 2000 (2000-02-01) abstract column 3 -column 4; figures 1,3,4 column 5, line 7-23 column 6, line 29-64 column 7 -column 8	7,8, 10-13, 20,21, 23-26
A	WO 00 28420 A (SYMANTEC CORP) 18 May 2000 (2000-05-18)	

INTERNATIONAL SEARCH REPORT

Int ial Application No
PCT/SG 01/00213

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6021491 A	01-02-2000	EP 0845733 A2	03-06-1998
		JP 10326078 A	08-12-1998
		US 5958051 A	28-09-1999
WO 0028420 A	18-05-2000	US 6094731 A	25-07-2000
		EP 1129406 A1	05-09-2001
		WO 0028420 A1	18-05-2000

Form PCT/ISA/210 (patent family annex) (July 1992)

PTO/SB/81 (01-06)
Approved for use through 12/31/2006. OMB 0651-0035
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

POWER OF ATTORNEY and CORRESPONDENCE ADDRESS INDICATION FORM

Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn MORRIS et al.
Art Unit	Unknown
Examiner Name	Unknown
Attorney Docket Number	339000-P0003

I hereby revoke all previous powers of attorney given in the above-identified application.

I hereby appoint:

☒ Practitioners associated with the Customer Number: 47604

OR

☐ Practitioner(s) name below:

Name	Registration Number

as my/our attorney(s) or agent(s) to prosecute the application identified above, and to transact all business in the United States Patent and Trademark Office connected therewith.

Please recognize or change the correspondence address for the above-identified application to:

☒ The address associated with the above-mentioned Customer Number: 47604

OR

☐ The address associated with Customer Number:

OR

☐ Firm or individual name			
Address			
City		State	Zip
Country			
Telephone		Email	

I am the:

☐ Applicant/Inventor.

☒ Assignee of record of the entire interest. See 37 CFR 3.71.

Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)

SIGNATURE of Applicant or Assignee of Record

Signature		**Date**	26/3/07
Name	Paul Stubbs	**Telephone**	
Title and Company	Director, Reuk Ltd.		

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*

☒ ***Total of 1 forms are submitted.**

This collection of information is required by 37 CFR 1.31,1.32 and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 9 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/96 (12-06)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(b)

Applicant/Patent Owner: Prevx Limited

Application No./Patent No./Control No.: 11/477,807 Filed/Issue Date: June 30, 2006

Entitled: METHODS AND APPARATUS FOR DEALING WITH MALWARE

Prevx Limited, a Corporation
(Name of Assignee) (Type of Assignee: corporation, partnership, university, government agency, etc.)

States that it is:

1. ☒ the assignee of the entire right, title, and interest; or

2. ☐ an assignee of less than the entire right, title and interest
(The extent (by percentage) of its ownership interest is _____%

in the patent application/ patent identified above by virtue of either:

A. ☒ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel 018348, Frame 0840, or a true copy of the original assignment is attached.

OR

B. ☐ A chain of title from the inventor(s), of the present patent application/ patent identified above, to the current assignee as follows:

1. From: _____ To: _____
The document was recorded in the United States Patent and trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

2. From: _____ To: _____
The document was recorded in the United States Patent and trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

3. From: _____ To: _____
The document was recorded in the United States Patent and trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

☐ Additional documents in the chain of title are listed on a supplemental sheet.

As required by 37 CFR 3.73(b)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[NOTE: A separate copy (*i.e.*, a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignees.

Signature	Date: 26/3/07
Printed or Typed Name: Paul Sparks	Telephone Number
Title: Director	

This collection of information is required by 37 CFR 3.73(b). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

REST1\593015.1

Electronic Acknowledgement Receipt

EFS ID:	1735854
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	909
Filer:	Dale S. Lazar/Margaret Filipowicz
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	011765-0355986
Receipt Date:	02-MAY-2007
Filing Date:	30-JUN-2006
Time Stamp:	09:52:17
Application Type:	Utility

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	339000P0003POA.pdf	702308	no	1
Warnings:					

Information:					
2	Assignee showing of ownership per 37 CFR 3.73(b).	339000P0003StatementA.pdf	671580	no	1
Warnings:					
Information:					
			Total Files Size (in bytes):	1373888	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/477,807	06/30/2006	Melvyn Morris	011765-0355986

CONFIRMATION NO. 4785

OC000000023928527

909
PILLSBURY WINTHROP SHAW PITTMAN, LLP
P.O. BOX 10500
MCLEAN, VA 22102

Date Mailed: 05/17/2007

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 05/02/2007.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199
OFFICE COPY

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371 (c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/477,807	06/30/2006	Melvyn Morris	339000-P0003

CONFIRMATION NO. 4785

OC000000023928564

47604
DLA PIPER US LLP
P. O. BOX 9271
RESTON, VA 20195

Date Mailed: 05/17/2007

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 05/02/2007.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

Office of Initial Patent Examination (571) 272-4000, or 1-800-PTO-9199
OFFICE COPY

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	339000-000003	4785

47604 7590 01/13/2010

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

EXAMINER	
KANE, CORDELIA P	

ART UNIT	PAPER NUMBER
2432	

MAIL DATE	DELIVERY MODE
01/13/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/477,807	MORRIS ET AL.	
	Examiner	Art Unit	
	CORDELIA KANE	2432	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *1* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 June 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-47* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) _____ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☒ Claim(s) *1-47* are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Election/Restrictions

1. Restriction to one of the following inventions is required under 35 U.S.C. 121:

I. Claims 1 – 41, drawn to classifying objects.

II. Claims 42 – 47, drawn to installing software on a computer.

The inventions are distinct, each from the other because of the following reasons:

2. Inventions I, and II are directed to related processes. The related inventions are distinct if: (1) the inventions as claimed are either not capable of use together or can have a materially different design, mode of operation, function, or effect; (2) the inventions do not overlap in scope, i.e., are mutually exclusive; and (3) the inventions as claimed are not obvious variants. See MPEP § 806.05(j). In the instant case, the inventions have different functions. Invention I classifies objects as malware. Invention II installs software. Furthermore, the inventions as claimed do not encompass overlapping subject matter and there is nothing of record to show them to be obvious variants.

3. Restriction for examination purposes as indicated is proper because all these inventions listed in this action are independent or distinct for the reasons given above and there would be a serious search and examination burden if restriction were not required because one or more of the following reasons apply:

(a) the inventions have acquired a separate status in the art in view of their different classification;

(b) the inventions have acquired a separate status in the art due to their recognized divergent subject matter;

(c) the inventions require a different field of search (for example, searching different classes/subclasses or electronic resources, or employing different search queries);

(d) the prior art applicable to one invention would not likely be applicable to another invention;

(e) the inventions are likely to raise different non-prior art issues under 35 U.S.C. 101 and/or 35 U.S.C. 112, first paragraph.

Applicant is advised that the reply to this requirement to be complete must include (i) an election of a invention to be examined even though the requirement may be traversed (37 CFR 1.143) **and (ii) identification of the claims encompassing the elected invention**.

The election of an invention may be made with or without traverse. To reserve a right to petition, the election must be made with traverse. If the reply does not distinctly and specifically point out supposed errors in the restriction requirement, the election shall be treated as an election without traverse. Traversal must be presented at the time of election in order to be considered timely. Failure to timely traverse the requirement will result in the loss of right to petition under 37 CFR 1.144. If claims are added after the election, applicant must indicate which of these claims are readable on the elected invention.

If claims are added after the election, applicant must indicate which of these claims are readable upon the elected invention.

Should applicant traverse on the ground that the inventions are not patentably distinct, applicant should submit evidence or identify such evidence now of record showing the inventions to be obvious variants or clearly admit on the record that this is the case. In either instance, if the examiner finds one of the inventions unpatentable over the prior art, the evidence or admission may be used in a rejection under 35 U.S.C. 103(a) of the other invention.

4. If Invention I is elected, this invention contains claims directed to the following patentably distinct species:

a. Species 1: A first aspect of the invention comparing data about the object received from plural computers and classifying the objects as malware on the basis of the comparisons (page 2, paragraphs 14-16). (claims 1-30, 40, 41).

b. Species 2: A fourth aspect of the invention, for determining the protection that a remote computer has from malware, by identifying any malware processes that were not identified (page 3, paragraphs 37-40). (claims 31-39).

The species are independent or distinct because claims to the different species recite the mutually exclusive characteristics of such species. In addition, these species are not obvious variants of each other based on the current record.

Applicant is required under 35 U.S.C. 121 to elect a single disclosed species for prosecution on the merits to which the claims shall be restricted if no generic claim is finally held to be allowable.

There is an examination and search burden for these patentably distinct species due to their mutually exclusive characteristics. The species require a different field of search (e.g., searching different classes/subclasses or electronic resources, or employing different search queries); and/or the prior art applicable to one species would not likely be applicable to another species; and/or the species are likely to raise different non-prior art issues under 35 U.S.C. 101 and/or 35 U.S.C. 112, first paragraph.

Applicant is advised that the reply to this requirement to be complete must include (i) an election of a species to be examined even though the requirement may be traversed (37 CFR 1.143) **and (ii) identification of the claims encompassing the elected species**, including any claims subsequently added. An argument that a claim is allowable or that all claims are generic is considered nonresponsive unless accompanied by an election.

The election of the species may be made with or without traverse. To preserve a right to petition, the election must be made with traverse. If the reply does not distinctly and specifically point out supposed errors in the election of species requirement, the election shall be treated as an election without traverse. Traversal must be presented at the time of election in order to be considered timely. Failure to timely traverse the requirement will result in the loss of right to petition under 37 CFR 1.144. If claims are added after the election, applicant must indicate which of these claims are readable on the elected species.

Should applicant traverse on the ground that the species are not patentably distinct, applicant should submit evidence or identify such evidence now of record

showing the species to be obvious variants or clearly admit on the record that this is the case. In either instance, if the examiner finds one of the species unpatentable over the prior art, the evidence or admission may be used in a rejection under 35 U.S.C. 103(a) of the other species.

Upon the allowance of a generic claim, applicant will be entitled to consideration of claims to additional species which depend from or otherwise require all the limitations of an allowable generic claim as provided by 37 CFR 1.141.

5. Applicant is reminded that upon the cancellation of claims to a non-elected invention, the inventorship must be amended in compliance with 37 CFR 1.48(b) if one or more of the currently named inventors is no longer an inventor of at least one claim remaining in the application. Any amendment of inventorship must be accompanied by a request under 37 CFR 1.48(b) and by the fee required under 37 CFR 1.17(i).

Any inquiry concerning this communication or earlier communications from the examiner should be directed to CORDELIA KANE whose telephone number is (571)272-7771. The examiner can normally be reached on Monday - Thursday 8:00 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/C. K./
Examiner, Art Unit 2432

/Benjamin E Lanier/
Primary Examiner, Art Unit 2432

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:	MORRIS, Melvyn e tal.	Confirmation Number: 4785
Application No.	11/477,807	Group Art Unit: 2432
Filed:	June 30, 2006	Examiner: C. P. Kane
For:	METHODS AND APPARATUS FOR DEALING WITH MALWARE	

RESPONSE TO RESTRICTION REQUIREMENT

Assistant Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

April 9, 2010

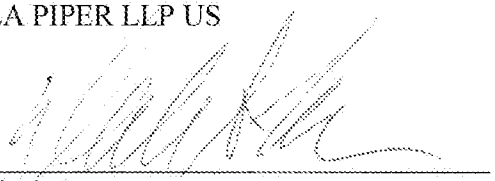
Dear Sir:

In response to the Office Action dated January 13, 2010, Applicants elect Group I, claims 1 to 41, without traverse. Applicants also wish to elect Species I directed to claims 1 to 30, 40 and 41. Applicants reserve the right to file a divisional application on the non-elected claims.

Please charge any shortage in the fees or credit any overpayment under 37 C.F.R. 1.17(a) – (d) to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US



Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/jcc
PO Box 2758
Reston, VA 20195
(703) 773.4149 Telephone
(703) 773.5200 Fax

Electronic Patent Application Fee Transmittal

Application Number:	11477807
Filing Date:	30-Jun-2006
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Filer:	Dale S. Lazar/Josefina Cheng
Attorney Docket Number:	339000-000003

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 2 months with $0 paid	1252	1	490	490

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**490**

Electronic Acknowledgement Receipt

EFS ID:	7387595
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Dale S. Lazar/Josefina Cheng
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000003
Receipt Date:	09-APR-2010
Filing Date:	30-JUN-2006
Time Stamp:	16:03:53
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$490
RAM confirmation Number	2401
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Miscellaneous Incoming Letter	339000-3Transmittal.pdf	351546 61d66b624205ed8b2d8642b280e3ab13a1f34367	no	1
Warnings:					
Information:					
2	Extension of Time	339000-3Extension.pdf	481137 7e4f3f4eb8fd0c5f45854a43f103979bf5bfda83	no	1
Warnings:					
Information:					
3	Response to Election / Restriction Filed	339000-3RR.pdf	272870 ae31c051bb53962c8e31214854150661590b709e	no	1
Warnings:					
Information:					
4	Fee Worksheet (PTO-875)	fee-info.pdf	30037 205ed9cea35697acd607dfde935969f7ae5c5550	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	1135590		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Honorable Commissioner for Patents
Post Office Box 1450
Alexandria, Virginia 22313-1450

Re: Application No.: 11/477,807
Applicants: MORRIS, Melvyn et al.
Filing Date June 30, 2006
For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

Dear Sir:

Attached hereto for filing are the following papers. The fee has been calculated as shown below.

☒ Response to the Office Action dated January 13, 2010; and

☒ Request for Extension of Time Two Months.

	CLAIMS REMAINING AFTER AMENDMENT	HIGHEST NUMBER PREVIOUSLY PAID FOR		PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
Total	47	- 47		=	x $52	
Independent	8	- 8		=	x $220	
First Presentation of Multiple Dependent Claim						$0.00
Two Months Extension of Time						$490.00
					Total Fees	$490.00

Our payment in the amount of **$490.00** covering any required fees is attached hereto. In the event any variance exists between the amount enclosed and the Patent Office charges for filing the above-noted documents, including any fees required under 37 C.F.R. 1.136 for any necessary extension of time to make the filing of the attached documents timely, please charge or credit the difference to Deposit Account No. 50-3266. Further, if these papers are not considered timely filed, then a request is hereby made under 37 C.F.R. 1.136 for the necessary extension of time.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar
Registration No. 28,872

Dated: April 9, 2010

PTO/SB/22 (10-04)
Approved for use through 7/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE
Under the paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) FY 2005 *(Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).)*	Docket Number 339000-000003
Application Number 11/477,807	Filed June 30, 2006
For METHODS AND APPARATUS FOR DEALING WITH MALWARE	
Art Unit 2432	Examiner C. P. Kane

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Fee Code	
☐	One Month (37 CFR 1.1 7(a)(1))	$130	1251	
☒	Two Month (37 CFR 1.1 7(a)(2))	$490	1252	$490.00
☐	Three Month (37 CFR 1.1 7(a)(3))	$1,110	1253	
☐	Four Month (37 CFR 1.1 7(a)(4))	$1730	1254	
☐	Five Month (37 CFR 1.1 7(a)(5))	$2,350	1255	

☐ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check in the amount of the fee is enclosed.

☒ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☐ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 50-3266. I have enclosed a duplicate copy of this sheet.

WARNING: Information **on this form may** become **public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.**

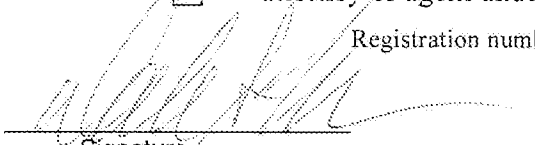
I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed (Form PTO/SB/96).

☒ attorney or agent of record. Registration Number 28,872

☐ attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 _____.



Signature

April 9, 2010
Date

Dale S. Lazar
Typed or printed name

703-773-4149
Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the more interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

☒ Total of 1 forms are submitted

This collection of information is required by 37 CFR 1.136(a). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S. C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 6 minutes to complete, including gathering, mating, and submitting the completed application form to the USPTO. Time will vary depending upon the Individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, US Patent and Trademark Office, U.S. Department OF Commerce, P.O. Box 1450, Alexandria, V.4 22313-4150. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patent, P.O. Box 1450, Alexandria, DA 22233-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2

American LegalNet, Inc
www.USCourtForms.com

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	339000-000003	4785

47604 7590 05/04/2010

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

EXAMINER	
ZECHER, CORDELIA P K	
ART UNIT	PAPER NUMBER
2432	

MAIL DATE	DELIVERY MODE
05/04/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/477,807	MORRIS ET AL.	
	Examiner	Art Unit	
	Cordelia Zecher	2432	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 April 2010*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) *31-47* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 June 2006* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Election/Restrictions

1. This application contains claims directed to the following patentably distinct species:

a. Species 1: A first aspect of the invention comparing data about the object received from plural computers and classifying the objects as malware on the basis of the comparisons (page 2, paragraphs 14-16). (claims 1-30).

b. Species 2: A fourth aspect of the invention, for determining the protection that a remote computer has from malware, by identifying any malware processes that were not identified (page 3, paragraphs 37-40). (claims 31-39).

c. Species 3: A sixth aspect of the invention, for identifying, in a key of a first computer object, a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object (page 3, paragraphs 48-51). (claims 40 and 41)

2. The species are independent or distinct because claims to the different species recite the mutually exclusive characteristics of such species. In addition, these species are not obvious variants of each other based on the current record.

3. Applicant is required under 35 U.S.C. 121 to elect a single disclosed species for prosecution on the merits to which the claims shall be restricted if no generic claim is finally held to be allowable. Currently, no claims are generic.

4. There is an examination and search burden for these patentably distinct species due to their mutually exclusive characteristics. The species require a different field of

search (e.g., searching different classes/subclasses or electronic resources, or employing different search queries); and/or the prior art applicable to one species would not likely be applicable to another species; and/or the species are likely to raise different non-prior art issues under 35 U.S.C. 101 and/or 35 U.S.C. 112, first paragraph.

5. **Applicant is advised that the reply to this requirement to be complete must include (i) an election of a species to be examined** even though the requirement may be traversed (37 CFR 1.143) **and (ii) identification of the claims encompassing the elected species**, including any claims subsequently added. An argument that a claim is allowable or that all claims are generic is considered nonresponsive unless accompanied by an election.

6. The election of the species may be made with or without traverse. To preserve a right to petition, the election must be made with traverse. If the reply does not distinctly and specifically point out supposed errors in the election of species requirement, the election shall be treated as an election without traverse. Traversal must be presented at the time of election in order to be considered timely. Failure to timely traverse the requirement will result in the loss of right to petition under 37 CFR 1.144. If claims are added after the election, applicant must indicate which of these claims are readable on the elected species.

7. Should applicant traverse on the ground that the species are not patentably distinct, applicant should submit evidence or identify such evidence now of record showing the species to be obvious variants or clearly admit on the record that this is the case. In either instance, if the examiner finds one of the species unpatentable over the

prior art, the evidence or admission may be used in a rejection under 35 U.S.C. 103(a) of the other species.

8. Upon the allowance of a generic claim, applicant will be entitled to consideration of claims to additional species which depend from or otherwise require all the limitations of an allowable generic claim as provided by 37 CFR 1.141.

9. Applicant is reminded that upon the cancellation of claims to a non-elected invention, the inventorship must be amended in compliance with 37 CFR 1.48(b) if one or more of the currently named inventors is no longer an inventor of at least one claim remaining in the application. Any amendment of inventorship must be accompanied by a request under 37 CFR 1.48(b) and by the fee required under 37 CFR 1.17(i).

10. Applicant's election without traverse of species 1 in the reply filed on April 9, 2010 is acknowledged.

Specification

The following guidelines illustrate the preferred layout for the specification of a utility application. These guidelines are suggested for the applicant's use.

Arrangement of the Specification

As provided in 37 CFR 1.77(b), the specification of a utility application should include the following sections in order. Each of the lettered items should appear in upper case, without underlining or bold type, as a section heading. If no text follows the section heading, the phrase "Not Applicable" should follow the section heading:

(a) TITLE OF THE INVENTION.
(b) CROSS-REFERENCE TO RELATED APPLICATIONS.
(c) STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT.
(d) THE NAMES OF THE PARTIES TO A JOINT RESEARCH AGREEMENT.
(e) INCORPORATION-BY-REFERENCE OF MATERIAL SUBMITTED ON A

COMPACT DISC.

(f) BACKGROUND OF THE INVENTION.

(1) Field of the Invention.

(2) Description of Related Art including information disclosed under 37 CFR 1.97 and 1.98.

(g) BRIEF SUMMARY OF THE INVENTION.

(h) BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING(S).

(i) DETAILED DESCRIPTION OF THE INVENTION.

(j) CLAIM OR CLAIMS (commencing on a separate sheet).

(k) ABSTRACT OF THE DISCLOSURE (commencing on a separate sheet).

(l) SEQUENCE LISTING (See MPEP § 2424 and 37 CFR 1.821-1.825. A "Sequence Listing" is required on paper if the application discloses a nucleotide or amino acid sequence as defined in 37 CFR 1.821(a) and if the required "Sequence Listing" is not submitted as an electronic document on compact disc).

11. The disclosure is objected to because of the following informalities: The specification is not divided into the appropriate sections as listed above.

12. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

13. Appropriate correction is required.

Claim Rejections - 35 USC § 101

14. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

15. Claim 30 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The claim is directed to a computer program. A

computer program does not fall into one of the four statutory categories of invention and would be considered software per se. Therefore the claim is non-statutory.

Claim Rejections - 35 USC § 103

16. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

17. The factual inquiries set forth in *Graham* v. *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

18. Claims 1 – 9, 12 – 21, 24 – 27, and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev's US Publication 2003/0177394 A1 in view of Sheymov et al's US Publication 2001/0052014 A1.

19. Referring to claims 1 and 13, Dozortsev teaches:

a. At a base computer, receiving data about a computer object from a remote computer on which the object or similar objects are stored (page 5, paragraph 44).

b. Comparing in the base computer the data about the computer object received from the plural computers (page 5, paragraph 44).

c. Classifying the computer object as malware on the basis of said comparison (page 5, paragraph 44).

2. Dozortsev does not explicitly teach receiving data from plural remote computers. However, Sheymov teaches having both centralized and distributed databases of information so there is a greater chance of being able to react to the attempt in real time (page 2, paragraph 23). Dozortsev and Sheymov are from the same field of endeavor, network protection. At the time of invention, it would have been obvious to modify the database of Dozortsev to include the multiple databases of Sheymov. The suggestion for doing so would have been to have a greater chance to react to the attack in real time (page 2, paragraph 23).

3. Referring to claims 2 and 14, Dozortsev teaches that the data about the computer object that is sent from the remote computer to the base computer and that is used in the comparison includes executable instructions contained within or constituted by the subject (page 3, paragraph 21).

4. Referring to claims 3 and 15, Dozortsev teaches comparing identifies relationships between the object and other objects (page 3, paragraph 22).

5. Referring to claims 4 and 16, Dozortsev teaches wherein if at least one other object to which said object is related is classified as malware, then classifying said object as malware (page 3, paragraph 25).

6. Referring to claims 5 and 17, Dozortsev teaches the other objects include objects stored on at least some of the remote computers (page 5, paragraphs 43-44).

7. Referring to claims 6 and 18, Dozortsev teaches that the other objects include otherwise process related objects (page 5, paragraph 44).

8. Referring to claims 7 and 19, Dozortsev teaches that the data is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on the remote computer (page 3, paragraph 21).

9. Referring to claims 8 and 20, Dozortsev teaches that the key has one component that represents executable instructions contained within or constituted by the object (page 3, paragraph 21).

10. Referring to claims 9 and 21, Dozortsev teaches that the key has at least one component that represents data about said object (page 3, paragraph 21).

11. Referring to claims 12 and 24, Dozortsev teaches initially classifying an object as not malware, generating a mask for said object that defines acceptable behavior for the object, and comprising monitoring operation of the object on at least one of the remote computers and reclassifying the object as malware if the actual monitored behavior extends beyond that permitted by the mask (page 4, paragraphs 37-40).

12. Referring to claims 25 and 30, Dozortsev teaches:

d. Providing from a remote computer to a base computer data about a computer object that is stored on the remote computer (page 5, paragraph 44).

e. The data including executable instructions contained within or constituted by the object (page 3, paragraph 21).

f. The data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer (page 3, paragraph 21).

13. Dozortsev does not explicitly teach receiving data from plural remote computers. However, Sheymov teaches having both centralized and distributed databases of information so there is a greater chance of being able to react to the attempt in real time (page 2, paragraph 23). Dozortsev and Sheymov are from the same field of endeavor, network protection. At the time of invention, it would have been obvious to modify the database of Dozortsev to include the multiple databases of Sheymov. The suggestion for doing so would have been to have a greater chance to react to the attack in real time (page 2, paragraph 23).

14. Referring to claim 26, Dozortsev teaches that the key has one component that represents executable instructions contained within or constituted by the object (page 3, paragraph 21).

15. Referring to claim 27, Dozortsev teaches that the key has at least one component that represents data about said object (page 3, paragraph 21).

16. Claims 10, 11, 22, 23, 28, and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Sheymov as applied above, and further in view of Barton et al's US Publication 2005/0021994 A1. Dozortsev in view of Sheymov discloses all the limitations of the parent claims. Dozortsev in view of Sheymov does not explicitly disclose the data about the object including the name of the object, the

location or the size of the object. However, Barton discloses indicating for the object the file name, size and location (pages 2-3, paragraph 39). Dozortsev in view of Sheymov and B are analogous art because they are from the same field of endeavor, malware detection. At the time of the invention, it would have been obvious to one of ordinary skill in the art, having the teachings of Dozortsev in view of Sheymov and Barton before him or her, to modify the signature of Dozortsev in view of Sheymov to include the data name and size of Barton. The suggestion/motivation for doing so would have been the characteristics are strongly indicative of a particular file being one of the candidates (page 1, paragraph 14).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cordelia Zecher whose telephone number is (571)272-7771. The examiner can normally be reached on Monday - Thursday 8:00 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/C. Z./
Examiner, Art Unit 2432

/Benjamin E Lanier/
Primary Examiner, Art Unit 2432

Notice of References Cited	Application/Control No. 11/477,807	Applicant(s)/Patent Under Reexamination MORRIS ET AL.	
	Examiner Cordelia Zecher	Art Unit 2432	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2001/0052014 A1	12-2001	Sheymov et al.	709/225
*	B	US-2003/0177394 A1	09-2003	Dozortsev, Dmitri	713/201
*	C	US-2005/0021994 A1	01-2005	Barton et al.	713/200
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	Art Unit
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010								
	1	✓								
	2	✓								
	3	✓								
	4	✓								
	5	✓								
	6	✓								
	7	✓								
	8	✓								
	9	✓								
	10	✓								
	11	✓								
	12	✓								
	13	✓								
	14	✓								
	15	✓								
	16	✓								
	17	✓								
	18	✓								
	19	✓								
	20	✓								
	21	✓								
	22	✓								
	23	✓								
	24	✓								
	25	✓								
	26	✓								
	27	✓								
	28	✓								
	29	✓								
	30	✓								
	31	N								
	32	N								
	33	N								
	34	N								
	35	N								
	36	N								

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010								
	37	N								
	38	N								
	39	N								
	40	N								
	41	N								
	42	N								
	43	N								
	44	N								
	45	N								
	46	N								
	47	N								

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	Art Unit
	Cordelia Zecher	2432

SEARCHED			
Class	Subclass	Date	Examiner
726	24	4/28/10	cz

SEARCH NOTES		
Search Notes	Date	Examiner
Searched EAST (see attached)	4/28/10	cz
Inventor Search	4/28/10	cz

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 4785

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
11/477,807	06/30/2006	726	2432	339000-000003

APPLICANTS

Melvyn Morris, Turnditch, UNITED KINGDOM;
Paul Stubbs, Wyboston, UNITED KINGDOM;
Markus Hartwig, Milton Keynes, UNITED KINGDOM;
Darren Harter, Hucclecote, UNITED KINGDOM;

**** CONTINUING DATA ***************************

**** FOREIGN APPLICATIONS ***************************

UNITED KINGDOM 0513375.6 06/30/2005

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

07/26/2006

Foreign Priority claimed	☑ Yes ☐ No	☐ Met after Allowance	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
35 USC 119(a-d) conditions met	☑ Yes ☐ No					
Verified and Acknowledged	/CORDELIA P K ZECHER/ Examiner's Signature	Initials	UNITED KINGDOM	3	47	8

ADDRESS

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195
UNITED STATES

TITLE

Methods and apparatus for dealing with malware

FILING FEE RECEIVED		
3480	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

BIB (Rev. 05/07).

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	1	("20060026684").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 08:32
L2	1	1 and malware	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
L3	1164	726/24.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
L6	2821	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:34
L7	6	(("6772346") or ("6944772") or ("20040006704") or ("20040073810") or ("20050021994") or ("20040153644")).PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 09:18
L8	1	("20030177394").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 10:48
L9	0	central near database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
L10	0	database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
L11	115859	database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
L12	2632	central near database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
L13	30	central near database with network with separate	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
L14	17	central near database with distribute with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53

L15	199	central near database with distribut$3 with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54
L16	0	central near database with distribute with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54
L17	4	central near database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55
L18	240	database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55
L19	24	database with distribut$3 with network with known and (virus or malware)	US-PGPUB; USPAT	OR	ON	2010/04/28 12:56
S1	1	"20070016953"	US-PGPUB; USPAT	OR	ON	2010/01/07 08:58

4/ 28/ 2010 1:24:47 PM
C:\ Documents and Settings\ ckane\ My Documents\ EAST\ Workspaces\ 11477807.wsp

Receipt date: 01/12/2007 11477807 - GAU: 2432

OIPE TAP 56 JAN 12 2007 PATENT & TRADEMARK OFFICE

Docket Number: 011765-0355986 PATENT APPLICATION
Client Reference: AF/JG/P10717US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re the Application of

MELVYN MORRIS, et al. Group Art Unit: 2131

Application No.: 11/477,807 Examiner: Unassigned

Filed: June 30, 2006 Confirmation No.: 4785

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

Pursuant to 37 CFR 1.56, the attention of the Patent and Trademark Office is hereby directed to the references listed on the attached PTO-1449. Copies of the required references are provided herewith. It is respectfully requested that the information be expressly considered during the prosecution of this application, and that the references be made of record therein and appear among the "References Cited" on any patent to issue therefrom. Applicants respectfully request the Examiner return an initialed copy of the enclosed Form PTO-1449 to Applicants with the next Office communication to indicate that the references have been considered, per MPEP § 609.

This Information Disclosure Statement is being within three months of the mailing date of the International Search Report in the related international application. No fee is required.

Respectfully Submitted,
PILLSBURY WINTHROP SHAW PITTMAN LLP

Christine H. McCarthy
Registration Number 41844
Customer Number: 00909

Date: January 12, 2007
Telephone: (703) 770-7900
Facsimile: (703) 770-7901
P.O. Box 10500
McLean, VA 22102

/Cordelia Zecher/ 04/28/2010

4005428051v1

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

Receipt date: 01/18/2007

OIPE JAN 18 2007 PATENT & TRADEMARK OFFICE

11477807 - GAU: 2432

FORM PTO-1449 (modified)
To: U.S. Department of Commerce
(PW FORM PAT-1449) Patent and Trademark Office

Atty. Dkt. No.	M#	Client Ref.
	11765-0355986	

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Applicant: MORRIS et al.
Appln. No.: 11/477,807
Filing Date: June 30, 2006
Examiner: Unassigned | Group Art Unit: 2131

Date: January 18, 2007 Page 1 of 1

U.S. PATENT DOCUMENTS

Examiner's Initials*		Document Number	Date MM/YYYY	Name (Family Name of First Inventor)	Class	Sub Class	Filing Date (if appropriate)
	AR	6,772,346	08/2004	CHESS			
	BR	6,944,772	09/2005	DOZORTSEV			
	CR	2004/0006704	01/2004	DAHLSTROM			
	DR	2004/0073810	05/2004	RACINE			
	ER	2005/0021994	01/2005	BARTON			
	FR	2004/0153644	08/2004	McCORKENDALE			
	GR						
	HR						
	IR						
	JR						
	KR						
	LR						
	MR						
	NR						

FOREIGN PATENT DOCUMENTS

		Document Number	Date MM/YYYY	Country	Inventor Name		English Abstract		Translation Readily Available	
							Enclosed	No	Enclose	No
	OR	1 280 040	01/2003	EP						
	PR	1 549 012	12/2003	EP						
	QR	03/021402	03/2003	WO						
	RR	2004/097602	11/2004	WO						
	SR	02/33525	04/2002	WO						

OTHER (Including in this order Author, Title, Periodical Name, Date, Pertinent Pages, etc.)

	TR				
	UR				
	VR				
	WR				

Examiner /Cordelia Zecher/ Date Considered: 04/28/2010

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP § 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to Applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

OIPE IAP44
OCT 31 2006
PATENT & TRADEMARK OFFICE

PTO/SB/08a (07-05)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	11/477,807
Filing Date	06/30/2006
First Named Inventor	MELVYN MORRIS
Art Unit	2131
Examiner Name	
Attorney Docket Number	011765-0355986

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 02/33525 A2	04/25/2002			
		WO 03/021402 A2	03/13/2003			
		WO 2004/097602 A2	11/11/2004			
		EP 1 280 040 A3	03/24/2004			
		EP 1 549 012 A1	06/29/2005			

Examiner Signature	/Cordelia Zecher/	Date Considered	04/28/2010

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

DOCKET NO. 339000-000003 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of: MORRIS, Melvyn et al. Confirmation No. 4785

Application No. 11/477,807 Group Art Unit: 2432

Filed: June 30, 2006 Examiner: C. Zecher

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

AMENDMENT UNDER 37 C.F.R. 1.111

Assistant Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

September 7, 2010

Dear Sir:

This is in response to the Office Action dated May 4, 2010, and having a period for response set to expire on August 4, 2010. A petition for a one-month extension of time, together with the requisite fee for same, is submitted herewith, thereby extending the period for response to September 4, 2010.

The following amendments and remarks are respectfully submitted. Reconsideration of the claims is respectfully requested.

Amendments to the Claims begin on page 2 of this paper.

Remarks/Arguments begin on page 15 of this paper.

Remarks/Arguments

The application has been amended to place the application condition for allowance. All amendments are supported by the specification. An early notice of allowance is earnestly solicited.

Claims 1 through 11, 13 through 23 and 25 through 30 remain pending. Claims 12 and 24 have been deleted. Claims 31 through 47 were previously withdrawn. Claim 1, 13, 25 and 30 have been amended.

The disclosure stands objected to for not being divided into appropriate sections. Applicants traverse this objection.

The Office Action correctly characterizes the arrangement specified in 37 C.F.R. 1.77 (b) as "guidelines" of a "preferred" layout that is "suggested" for the applicant's use. Thus, the Office Action itself acknowledges that the arrangement is discretionary on the part of the applicant. In this case, applicant declines the suggestion in the Office Action.

Claim 30 stands rejected under 35 U.S.C. §101 as being directed to non-statutory subject matter. Claim 30 has been amended to be directed to a manufacture, one of the four statutory categories of invention.

Claims 1-9, 12-21, 24-27 and 30 stand rejected as obvious over Dozortsev and Sheymov. The claims have been amended to obviate this rejection.

As mentioned in the introduction to the present application, the conventional approach to identifying malware is to maintain a database of signatures identifying known bad objects, i.e. malware. The database may be stored locally on the remote computer and/or at a central base computer/server. The signature is conventionally made by creating a hash or checksum of the object file, which uniquely identifies that object. The signatures of objects at remote computers are compared to the database to look up whether they match known malware. If the signature of the object is not found the database, they are assumed safe or else referred for further investigation by a human analyst if the object is thought to be suspicious. The drawback of this approach is that considerable effort must be undertaken by providers of such systems in identifying and analysing new malware and generating signatures of objects that are found to be bad. As the volume of malware increases, it is proving practically an impossible task to keep up with identifying suspect objects, and investigating whether or not they are bad. For example, large vendors of anti-malware packages using this system typically employ many thousands of human analysts working on identifying and analysing objects to keep the database of signatures of bad objects reasonably up to date.

The present invention adopts a very different and inventive approach to protecting against malware. The approach is founded on receiving information about objects present on a plurality of remote computers at a base computer. This information about objects in the community, i.e. from plural remote computers, is aggregated at the base computer. This allows analysis to be performed mapping the behaviour and attributes of objects known across the community in order to identify suspicious behaviour and to identify malware at an early stage. This approach allows among, other advantages, the number of human analysts needed to be massively reduced. This

can also reduce the latency with which new threats are identified and responding to objects exhibiting new, potentially malevolent behaviour.

Dozortsev teaches an approach using the standard technique of maintaining a database of signatures of known bad objects, i.e. malware, and comparing new objects with the database of known bad objects to see if they are malware. Dozortsev differs slightly from the normal approach of downloading the database of bad objects to the remote computer, by having a central database of bad objects. Remote computers send signatures of new objects to the base computer to be compared centrally with the database. However, this approach still fundamentally relies on using a database of signatures of bad objects. The signatures in the database must be generated by having human analysts investigating suspected bad objects. For example, as discussed at paragraph 47 and 49 of Dozortsev, the objects are analysed centrally by a human analyst by running the file in a "sandbox" environment or by examining the code or other such slow and laborious techniques. This still suffers from the problems of spotting suspicious objects in the first place, and in the huge effort required to analyse suspicious objects to keep the database of threats up to date.

In contrast, the present invention requires receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored; comparing in the base computer the data about the computer object received from the plural computers; and, classifying the computer object as malware on the basis of said comparison. In effect, the invention builds up a central picture of objects and their interrelationships and activities across the entire community. This allows automation of the process of identifying malware by aggregating and comparing activity of objects running across the community, i.e. on plural

remote computers. Many examples are given in the present description about the sophisticated analysis that may be performed on this data.

Furthermore, amended claim 1 requires when an object is not known malware, initially classifying the object as not malware, generating a mask for said object that defines acceptable behaviour for the object, and comprising monitoring operation of the object on at least one of the remote computers and reclassifying the object as malware if the actual monitored behaviour extends beyond that permitted by the mask. This has advantages in dealing with objects that do not initially exhibit suspicious behaviour, but later start to exhibit malevolent behaviour. In the prior art, such as Dozortsev, an object can only be marked as good or bad, i.e. a binary decision. This does not permit the system to automatically deal with the case where an object only starts to exhibits malevolent behaviour in the future, which is a common attribute of certain types of malware. The present invention deals with this situation by initially classifying the object as good but also generating an appropriate behaviour mask for the object which is used to monitor the permitted behaviour of the object. If the object exceeds the permitted behaviour, then this information can be sent to the base computer, which can take appropriate action, i.e. reclassifying the object as bad. Thus, the invention has the capacity of changing the classification of an object if malevolent behaviour emerges later on.

As an example, suppose a new version of Internet Explorer appeared. This could be a legitimate update to Internet Explorer released by Microsoft or alternatively it could be a file infected with a virus. In the prior art, the new object would have an unknown signature, so an in-house analyst would laboriously analyse the new object and determine whether or not it was safe. Whilst this analysis is carried out, the object would either be blocked, which would cause huge

inconvenience to users of the new object, or allowed to run, in which case there is a risk of the object performing malevolent acts. In contrast, the present invention would collect data at the base computer from remote computers running the new version of Internet Explorer. Using the information collected, the system could determine that the new object purports to be a new version of Internet Explorer. However, it may not be apparent at this point whether or not the new object is capable of malevolent behaviour. In this scenario the present invention generates an appropriate behavioural mask for the object, e.g. by using a profile of behaviour of previous versions of Internet Explorer that are known not to be malware, or by using a profile for the behaviour appropriate for a web browser. The remote computers are allowed to let the new version run whilst monitoring its behaviour against the mask. The instant the new object exhibits some new, malevolent behaviour, this can be stopped at the remote computer, as well as being flagged to the base computer and used at the base computer to change the classification of the object. Thus, the present invention allows an instant response to an object changing its behaviour to exhibit malevolent behaviour in the future.

Dozortsev does not teach or suggest anything like this scheme. The examiner referred to paragraphs 37 to 40 of Dozortsev as being relevant to the behavioural mask feature of the present invention (in previous claim 12). However, these paragraphs teach only that unknown objects received at the base computer may be allowed to run whilst being investigated by a human analyst (paragraph 39). There is no mention in Dozortsev of using a behavioural mask. Rather, in Dozortsev the investigation into the new object can take the form of allowing the program to run in a sandbox environment and/or analysing the code of the object (see paragraph 41). However, this is merely the standard technique of investigating the new object by in-house

analysts which is becoming increasingly inadequate to cope with the volume of malware and the speed at which new malware is introduced and exhibits new behaviour.

Thus, in short, Dozortsev does not teach or suggest the concepts of (i) comparing data about objects received from plural computers across the community to determine whether or not an object is malware, and (ii) monitoring the execution of an object against a mask of acceptable behaviour for that object and reclassifying an object as malware at the central computer if the object exceeds that behaviour.

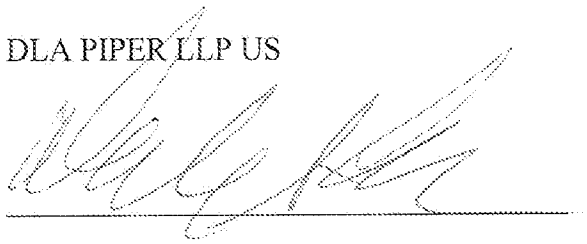
Sheymov does not compensate for the deficiencies of Dozortsev. None of the other prior art cited by the examiner teaches or suggests these concepts either. Thus, the present invention as defined by independent claims 1, 13 and 25 is novel and non-obvious over these documents.

Claims 10, 11, 22, 23, 28 and 29 stand rejected as obvious over Dozortsev, Sheymov and Barton. Barton is cited for disclosing the step of indicating the file name, size and location of an object. Barton fails to compensate for the deficiencies of Dozortsev and Sheymon noted above relative to the independent claims. Since claims 10, 11, 22, 23, 28 and 29 are dependent on the independent claims, they are patentable for the reasons discussed above for claim 1.

Please charge any shortage in the fees or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US



Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/jcc
PO Box 2758
Reston, VA 20195
(703) 773.4000 Telephone
(703) 773.5200 Fax

Electronic Patent Application Fee Transmittal

Application Number:	11477807
Filing Date:	30-Jun-2006
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Filer:	Dale S. Lazar/Josefina Cheng
Attorney Docket Number:	339000-000003

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 1 month with $0 paid	1251	1	130	130

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**130**

Electronic Acknowledgement Receipt

EFS ID:	8368847
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Dale S. Lazar/Josefina Cheng
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000003
Receipt Date:	07-SEP-2010
Filing Date:	30-JUN-2006
Time Stamp:	18:42:30
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$130
RAM confirmation Number	5133
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Extension of Time	339000-3EOT.pdf	483680 58a7dd941d5de4bed16f855d8a3dfc9abe70e6c2	no	1
Warnings:					
Information:					
2		339000-3Amendment.pdf	6679379 208823a06f186a3d269975407f0d3ee7e21c8cf5	yes	21

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Amendment/Req. Reconsideration-After Non-Final Reject	1	1
	Claims	2	14
	Applicant Arguments/Remarks Made in an Amendment	15	21

Warnings:					
Information:					
3	Fee Worksheet (PTO-875)	fee-info.pdf	29850 42a5e64b1dc661c83b04acb45983a7d7ed42726b	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	7192909		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/22 (10-04)
Approved for use through 7/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE
Under the paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless if displays a valid OMB control number

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) **FY 2005** *(Fees pursuant to me Consolidated Appropriations Act, 2005 (H.R. 4818).)*	Docket Number 339000-000003
Application Number 11/477,807	Filed June 30, 2006
For METHODS AND APPARATUS FOR DEALING WITH MALWARE	
Art Unit 2432	Examiner C. Zecher

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Fee Code	
☒	One Month (37 CFR 1.1 7(a)(1))	$130	1251	$130.00
☐	Two Month (37 CFR 1.1 7(a)(2))	$490	1252	
☐	Three Month (37 CFR 1.1 7(a)(3))	$1,110	1253	
☐	Four Month (37 CFR 1.1 7(a)(4))	$1730	1254	
☐	Five Month (37 CFR 1.1 7(a)(5))	$2,350	1255	

☐ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check in the amount of the fee is enclosed.

☒ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☐ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 50-3266. I have enclosed a duplicate copy of this sheet.

WARNING: Information **on this form may** become **public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.**

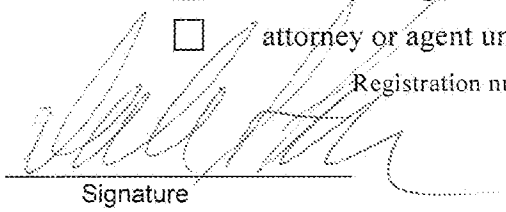
I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed (Form PTO/SB/96).

☒ attorney or agent of record. Registration Number 28,872

☐ attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 _____.



Signature

September 7, 2010
Date

Dale S. Lazar
Typed or printed name

703-773-4149
Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the more interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

☒ Total of 1 forms are submitted

This collection of information is required by 37 CFR 1.136(a). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S. C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 6 minutes to complete, including gathering, mating, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, US Patent and Trademark Office, U.S. Department OF Commerce, P.O. Box 1450, Alexandria, V.4 22313-4150. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patent. P.O. Box 1450, Alexandria, DA 22233-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2

American LegalNet, Inc
www.USCourtForms.com

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
11/477,807	06/30/2006	

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY ☐ RATE ($)	SMALL ENTITY ☐ FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT 09/07/2010	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 45	Minus	** 47	= 0	X $ =		OR	X $52=	0
Independent (37 CFR 1.16(h))	* 8	Minus	***8	= 0	X $ =		OR	X $220=	0
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	0

AMENDMENT	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/BURNELL L. ROSS/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application. Currently amended claims are shown with additions underlined and deletions in ~~strikethrough text~~ except double brackets may be placed before and after the deleted characters to show deletion of five or fewer characters.

1. (CURRENTLY AMENDED) A method of classifying a computer object as malware, the method comprising:

at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored;

comparing in the base computer the data about the computer object received from the plural computers; [[and,]]

classifying the computer object as malware on the basis of said comparison;

when the computer object is not known malware, initially classifying an object as not malware;

generating a mask for said object that defines acceptable behaviour for the object; and

monitoring operation of the object on at least one of the remote computers and reclassifying the object as malware if the actual monitored behaviour extends beyond that permitted by the mask.

2. (ORIGINAL) A method according to claim 1, wherein the data about the computer object that is sent from the plural remote computers to the base computer and that is used in the comparison includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

3. (ORIGINAL) A method according to claim 1, wherein the comparing identifies relationships between the object and other objects.

4. (ORIGINAL) A method according to claim 3, wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware.

5. (ORIGINAL) A method according to claim 3, wherein said other objects include the object or similar objects stored on at least some of the remote computers.

6. (ORIGINAL) A method according to claim 3, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

7. (ORIGINAL) A method according to claim 1, wherein the data is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on the respective remote computers.

8. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

9. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents data about said object.

10. (ORIGINAL) A method according to claim 9, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

11. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents the physical size of the object.

12. (CANCELLED)

13. (CURRENTLY AMENDED) Apparatus for classifying a computer object as malware, the apparatus comprising:

a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored;

the base computer being constructed and arranged to compare the data about the computer object received from said plural computers; and,

the base computer being constructed and arranged to classify the computer object as malware on the basis of said comparison,

wherein the base computer is constructed and arranged to generate a mask for an object that is initially classed not as malware, said mask defining acceptable behaviour for the object, and the base computer or each of the plural remote computers is constructed and arranged to monitor operation of the object on at least one of the remote computers and to reclassify the object as malware if the actual monitored behaviour extends beyond that permitted by the mask.

14. (ORIGINAL) Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to compare the data where the data includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

15. (ORIGINAL) Apparatus according to claim 13, wherein the base computer is constructed and arranged so that the comparing identifies relationships between the object and other objects.

16. (ORIGINAL) Apparatus according to claim 15, the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.

17. (ORIGINAL) Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least some of said remote computers.

18. (ORIGINAL) Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

19. (ORIGINAL) Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on said respective remote computers.

20. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

21. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents data about said object.

22. (ORIGINAL) Apparatus according to claim 21, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

23. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents the physical size of the object.

24. (CANCELLED)

25. (CURRENTLY AMENDED) A method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers, the method comprising:

providing from a remote computer to a base computer data about a computer object that is stored on the remote computer;

the data including one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers;

the data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer,

receiving at the remote computer a mask for a computer object stored at the remote computer, said mask defining acceptable behaviour for the object;

executing the computer object at the remote computer; and

monitoring operation of the computer object at the remote computer compared to the behaviour permitted by the mask and, in the event that the behaviour of the object extends beyond that permitted by the mask, sending data to the base computer that the behaviour of the object has exceeded that permitted by the mask.

26. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

27. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents data about said object.

28. (ORIGINAL) A method according to claim 27, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

29. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents the physical size of the object.

30. (CURRENTLY AMENDED) A carrier storing a computer program comprising program instructions for causing a computer to perform the method of claim 25.

31. (WITHDRAWN) A method of determining the protection that a remote computer has from malware, the method comprising:

receiving at a base computer details of all or selected security products operating at a point in time on said remote computer;

receiving similar information from other remote computers connected to the base computer; and,

identifying any malware processes that were not identified by said other remote computers having that particular combination of security products.

32. (WITHDRAWN) A method according to claim 31, comprising providing information to the user of said remote computer that said remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

33. (WITHDRAWN) A method according to claim 31, wherein the details of all or selected security products includes the name of the security products, versions, and loaded signature files.

34. (WITHDRAWN) Apparatus for determining the protection that a remote computer has from malware, the apparatus comprising:

a base computer constructed and arranged to receive computer details of all or selected security products operating at a point in time on said remote computer;

the base computer being constructed and arranged to receive similar information from other remote computers connected to the base computer; and,

the base computer being constructed and arranged to identify any malware processes that were not identified by said other remote computers having that particular combination of security products.

35. (WITHDRAWN) Apparatus according to claim 34, wherein the base computer is constructed and arranged to provide information to the user of said remote computer that said remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

36. (WITHDRAWN) Apparatus according to claim 34, wherein the details of all or selected security products includes the name of the security products, versions, and loaded signature files.

37. (WITHDRAWN) A method of classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the method comprising:

classifying a first computer object as malware or as safe;

identifying in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and,

classifying said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

38. (WITHDRAWN) A method according to claim 37, comprising monitoring activities of said first computer object and reclassifying the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the method further comprising automatically classifying as malware any computer object that has a key in which said unique identifier component is present.

39. (WITHDRAWN) A computer program comprising program instructions for causing a computer to perform the method of claim 38.

40. (WITHDRAWN) Apparatus for classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the apparatus comprising:

a computer constructed and arranged to classify a first computer object as malware or as safe;

the computer being constructed and arranged to identify in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and,

the computer being constructed and arranged to classify said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

41. (WITHDRAWN) Apparatus according to claim 40, wherein the computer is constructed and arranged to monitor activities of said first computer object and reclassify the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the computer further being constructed and arranged to automatically classify as malware any computer object that has a key in which said unique identifier component is present.

42. (WITHDRAWN) A method of installing software on a computer, the method comprising:

on initiation of installation of software on a computer, providing a computer-generated prompt on the computer to a user to ascertain whether the user authorises the installation; and,

ceasing the installation if a user authorisation is not received, else:

receiving at the computer the user's authorisation to proceed with the installation;

proceeding with the installation;

obtaining data about computer objects that are created or used during the installation;

storing said data at the local computer.

43. (WITHDRAWN) A method according to claim 42, wherein the locally stored data is referred to during the installation to ensure that all objects created or used during the installation are part of the installation process, and, if it is found that objects created or used during the installation are not part of the installation process, either or both of: (i) ceasing the installation and (ii) providing a computer-generated prompt on the computer to the user accordingly.

44. (WITHDRAWN) A method according to claim 42, comprising connecting the computer to a community database that is connectable to a plurality of computers, and uploading the stored data to the community database for comparison with similar data provided by other computers.

45. (WITHDRAWN) A method according to claim 42, comprising downloading data about trusted installers to the computer, said data about trusted installers being referred to during the installation such that any objects relating to or created by the trusted installer are automatically authorised to proceed.

46. (WITHDRAWN) A method according to claim 45, wherein said data about trusted installers is referred to only for a predetermined time period following receipt at the computer of the user's authorisation to proceed with the installation.

47. (WITHDRAWN) A computer program comprising program instructions for causing a computer to perform the method of claim 42.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	339000-000003	4785

47604 7590 09/28/2010

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

EXAMINER	
ZECHER, CORDELIA P K	
ART UNIT	PAPER NUMBER
2432	

MAIL DATE	DELIVERY MODE
09/28/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/477,807	MORRIS ET AL.	
	Examiner	Art Unit	
	Cordelia Zecher	2432	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 September 2010*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-11,13-23 and 25-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11,13-23 and 25-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Response to Arguments

1. Applicant's arguments filed September 7, 2010 have been fully considered but they are not persuasive. Applicant argues that amending claim 30 to recite a carrier storing a computer program places the claim in one of the four statutory categories of invention. However, in paragraph [0071] of applicant's specification, applicant defined carrier to include signals. Signals do not fall into one of the four statutory categories of invention and are therefore non-statutory.

2. Applicant argues that Dozortsev in view of Sheymov fails to teach or suggest comparing data about objects received from plural computers across the community to determine whether or not an object is malware. However, Dozortsev teaches comparing the executable code signature with entries in the local database (page 5, paragraph 44) and then forwarding the signature on to the central computer to compare with the central database to determine if the code is flagged as malware (page 5, paragraph 45). Dozortsev does not explicitly disclose plural remote computers. However, Sheymov teaches having both centralized and distributed databases (page 2, paragraph 23). Therefore, in combination, Dozortsev in view of Sheymov teaches the limitation of the claim.

3. Applicant argues that Dozortsev in view of Sheymov fails to teach or suggest monitoring execution of an object against a mask of acceptable behavior for that object and reclassifying an object as malware at the central computer if the object exceeds that behavior. However, Dozortsev teaches when code is not known malware, that the

code may continue to execute, and therefore not be considered malware (page 4, paragraphs 37-38), then the code is investigated and executed in an environment to see if it meets the acceptable behavior, and when the code behaves as malware, classifying the code as malware (page 4, paragraphs 40-41). Therefore Dozortsev teaches the limitation of the claim.

Specification

The following guidelines illustrate the preferred layout for the specification of a utility application. These guidelines are suggested for the applicant's use.

Arrangement of the Specification

As provided in 37 CFR 1.77(b), the specification of a utility application should include the following sections in order. Each of the lettered items should appear in upper case, without underlining or bold type, as a section heading. If no text follows the section heading, the phrase "Not Applicable" should follow the section heading:

(a) TITLE OF THE INVENTION.
(b) CROSS-REFERENCE TO RELATED APPLICATIONS.
(c) STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT.
(d) THE NAMES OF THE PARTIES TO A JOINT RESEARCH AGREEMENT.
(e) INCORPORATION-BY-REFERENCE OF MATERIAL SUBMITTED ON A COMPACT DISC.
(f) BACKGROUND OF THE INVENTION.
    (1) Field of the Invention.
    (2) Description of Related Art including information disclosed under 37 CFR 1.97 and 1.98.
(g) BRIEF SUMMARY OF THE INVENTION.
(h) BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING(S).
(i) DETAILED DESCRIPTION OF THE INVENTION.
(j) CLAIM OR CLAIMS (commencing on a separate sheet).
(k) ABSTRACT OF THE DISCLOSURE (commencing on a separate sheet).

(l) SEQUENCE LISTING (See MPEP § 2424 and 37 CFR 1.821-1.825. A "Sequence Listing" is required on paper if the application discloses a nucleotide or amino acid sequence as defined in 37 CFR 1.821(a) and if the required "Sequence Listing" is not submitted as an electronic document on compact disc).

4. The disclosure is objected to because of the following informalities: The specification is not divided into the appropriate sections as listed above.

5. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

6. Appropriate correction is required.

7. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

Claim Rejections - 35 USC § 101

8. Claim 30 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The claim is directed to a carrier. In the specification, applicant defines a carrier to include a signal ([0071]). *See* MPEP 2111.01. When the broadest reasonable interpretation of a claim covers a signal *per se*, the claim must be rejected under 35 U.S.C. §101 as covering non-statutory subject matter. *See In re Nuijten*, 500 F.3d 1346, 1356-57 (Fed. Cir. 2007) (transitory embodiments are not direct to statutory subject matter) and *Interim Examination*

Instructions for Evaluating Subject Matter Eligibility Under 35 U.S.C §101, Aug. 24, 2009; p. 2.

Claim Rejections - 35 USC § 103

9. Claims 1 – 9, 12 – 21, 24 – 27, and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev's US Publication 2003/0177394 A1 in view of Sheymov et al's US Publication 2001/0052014 A1.

10. Referring to claims 1 and 13, Dozortsev teaches:

a. At a base computer, receiving data about a computer object from a remote computer on which the object or similar objects are stored (page 5, paragraph 44).

b. Comparing in the base computer the data about the computer object received from the plural computers (page 5, paragraph 44).

c. Classifying the computer object as malware on the basis of said comparison (page 5, paragraph 44).

d. When the computer object is not known malware, initially classifying an object as not malware (page 4, paragraphs 37-38).

e. Generating a mask for said object that defines acceptable behavior for the object (page 4, paragraph 40).

f. Monitoring operation of said object and reclassifying the object as malware if the actual monitored behavior extends beyond that permitted by the mask (page 4, paragraphs 40-41).

2. Dozortsev does not explicitly teach receiving data from plural remote computers. However, Sheymov teaches having both centralized and distributed databases of information so there is a greater chance of being able to react to the attempt in real time (page 2, paragraph 23). Dozortsev and Sheymov are from the same field of endeavor, network protection. At the time of invention, it would have been obvious to modify the database of Dozortsev to include the multiple databases of Sheymov. The suggestion for doing so would have been to have a greater chance to react to the attack in real time (page 2, paragraph 23).

3. Referring to claims 2 and 14, Dozortsev teaches that the data about the computer object that is sent from the remote computer to the base computer and that is used in the comparison includes executable instructions contained within or constituted by the subject (page 3, paragraph 21).

4. Referring to claims 3 and 15, Dozortsev teaches comparing identifies relationships between the object and other objects (page 3, paragraph 22).

5. Referring to claims 4 and 16, Dozortsev teaches wherein if at least one other object to which said object is related is classified as malware, then classifying said object as malware (page 3, paragraph 25).

6. Referring to claims 5 and 17, Dozortsev teaches the other objects include objects stored on at least some of the remote computers (page 5, paragraphs 43-44).

7. Referring to claims 6 and 18, Dozortsev teaches that the other objects include otherwise process related objects (page 5, paragraph 44).

8. Referring to claims 7 and 19, Dozortsev teaches that the data is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on the remote computer (page 3, paragraph 21).

9. Referring to claims 8 and 20, Dozortsev teaches that the key has one component that represents executable instructions contained within or constituted by the object (page 3, paragraph 21).

10. Referring to claims 9 and 21, Dozortsev teaches that the key has at least one component that represents data about said object (page 3, paragraph 21).

11. Referring to claims 25 and 30, Dozortsev teaches:

g. Providing from a remote computer to a base computer data about a computer object that is stored on the remote computer (page 5, paragraph 44).

h. The data including executable instructions contained within or constituted by the object (page 3, paragraph 21).

i. The data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer (page 3, paragraph 21).

j. Receiving at the remote computer a mask for a computer object stored at the remote computer, said mask defining acceptable behavior of the object (page 4, paragraph 40).

k. Executing the computer object at the remote computer (page 4, paragraph 40).

l. Monitoring operation of the computer object at the remote computer compared to the behavior permitted by the mask, and in the event that the behavior of the object extends beyond that permitted by the mask, sending data to the base computer that the behavior of the object has exceeded that permitted by the mask (page 4, paragraphs 40-41).

12. Dozortsev does not explicitly teach receiving data from plural remote computers. However, Sheymov teaches having both centralized and distributed databases of information so there is a greater chance of being able to react to the attempt in real time (page 2, paragraph 23). Dozortsev and Sheymov are from the same field of endeavor, network protection. At the time of invention, it would have been obvious to modify the database of Dozortsev to include the multiple databases of Sheymov. The suggestion for doing so would have been to have a greater chance to react to the attack in real time (page 2, paragraph 23).

13. Referring to claim 26, Dozortsev teaches that the key has one component that represents executable instructions contained within or constituted by the object (page 3, paragraph 21).

14. Referring to claim 27, Dozortsev teaches that the key has at least one component that represents data about said object (page 3, paragraph 21).

15. Claims 10, 11, 22, 23, 28, and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Sheymov as applied above, and further in view of Barton et al's US Publication 2005/0021994 A1. Dozortsev in view of Sheymov

discloses all the limitations of the parent claims. Dozortsev in view of Sheymov does not explicitly disclose the data about the object including the name of the object, the location or the size of the object. However, Barton discloses indicating for the object the file name, size and location (pages 2-3, paragraph 39). Dozortsev in view of Sheymov and Barton are analogous art because they are from the same field of endeavor, malware detection. At the time of the invention, it would have been obvious to one of ordinary skill in the art, having the teachings of Dozortsev in view of Sheymov and Barton before him or her, to modify the signature of Dozortsev in view of Sheymov to include the data name and size of Barton. The suggestion/motivation for doing so would have been the characteristics are strongly indicative of a particular file being one of the candidates (page 1, paragraph 14).

Conclusion

16. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cordelia Zecher whose telephone number is (571)272-7771. The examiner can normally be reached on Monday - Thursday 8:00 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/C. Z./
Examiner, Art Unit 2432

/Benjamin E Lanier/
Primary Examiner, Art Unit 2432

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010							
	1	✓	✓							
	2	✓	✓							
	3	✓	✓							
	4	✓	✓							
	5	✓	✓							
	6	✓	✓							
	7	✓	✓							
	8	✓	✓							
	9	✓	✓							
	10	✓	✓							
	11	✓	✓							
	12	✓	-							
	13	✓	✓							
	14	✓	✓							
	15	✓	✓							
	16	✓	✓							
	17	✓	✓							
	18	✓	✓							
	19	✓	✓							
	20	✓	✓							
	21	✓	✓							
	22	✓	✓							
	23	✓	✓							
	24	✓	-							
	25	✓	✓							
	26	✓	✓							
	27	✓	✓							
	28	✓	✓							
	29	✓	✓							
	30	✓	✓							
	31	N								
	32	N								
	33	N								
	34	N								
	35	N								
	36	N								

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010							
	37	N								
	38	N								
	39	N								
	40	N								
	41	N								
	42	N								
	43	N								
	44	N								
	45	N								
	46	N								
	47	N								

Interview Summary	Application No.	Applicant(s)	
	11/477,807	MORRIS ET AL.	
	Examiner	Art Unit	
	Cordelia Zecher	2432	

All participants (applicant, applicant's representative, PTO personnel):

(1) *Cordelia Zecher*.

(2) *Dale Lazar*.

(3) *Nicholas Panno*.

(4) _____.

Date of Interview: *12 January 2011*.

Type: a)☐ Telephonic b)☐ Video Conference
c)☒ Personal [copy given to: 1)☐ applicant 2)☒ applicant's representative]

Exhibit shown or demonstration conducted: d)☐ Yes e)☒ No.
If Yes, brief description: _____.

Claim(s) discussed: *1,13 and 25*.

Identification of prior art discussed: *Dozortsev*.

Agreement with respect to the claims f)☐ was reached. g)☒ was not reached. h)☐ N/A.

Substance of Interview including description of the general nature of what was agreed to if an agreement was reached, or any other comments: *Possible claim amendments were discussed to clarify the mask and allowing the object to run when the behaviour is permitted by the mask. Such amendments would require further search and consideration*.

(A fuller description, if necessary, and a copy of the amendments which the examiner agreed would render the claims allowable, if available, must be attached. Also, where no copy of the amendments that would render the claims allowable is available, a summary thereof must be attached.)

THE FORMAL WRITTEN REPLY TO THE LAST OFFICE ACTION MUST INCLUDE THE SUBSTANCE OF THE INTERVIEW. (See MPEP Section 713.04). If a reply to the last Office action has already been filed, APPLICANT IS GIVEN A NON-EXTENDABLE PERIOD OF THE LONGER OF ONE MONTH OR THIRTY DAYS FROM THIS INTERVIEW DATE, OR THE MAILING DATE OF THIS INTERVIEW SUMMARY FORM, WHICHEVER IS LATER, TO FILE A STATEMENT OF THE SUBSTANCE OF THE INTERVIEW. See Summary of Record of Interview requirements on reverse side or on attached sheet.

Summary of Record of Interview Requirements

Manual of Patent Examining Procedure (MPEP), Section 713.04, Substance of Interview Must be Made of Record
A complete written statement as to the substance of any face-to-face, video conference, or telephone interview with regard to an application must be made of record in the application whether or not an agreement with the examiner was reached at the interview.

Title 37 Code of Federal Regulations (CFR) § 1.133 Interviews
Paragraph (b)

In every instance where reconsideration is requested in view of an interview with an examiner, a complete written statement of the reasons presented at the interview as warranting favorable action must be filed by the applicant. An interview does not remove the necessity for reply to Office action as specified in §§ 1.111, 1.135. (35 U.S.C. 132)

37 CFR §1.2 Business to be transacted in writing.
All business with the Patent or Trademark Office should be transacted in writing. The personal attendance of applicants or their attorneys or agents at the Patent and Trademark Office is unnecessary. The action of the Patent and Trademark Office will be based exclusively on the written record in the Office. No attention will be paid to any alleged oral promise, stipulation, or understanding in relation to which there is disagreement or doubt.

The action of the Patent and Trademark Office cannot be based exclusively on the written record in the Office if that record is itself incomplete through the failure to record the substance of interviews.

It is the responsibility of the applicant or the attorney or agent to make the substance of an interview of record in the application file, unless the examiner indicates he or she will do so. It is the examiner's responsibility to see that such a record is made and to correct material inaccuracies which bear directly on the question of patentability.

Examiners must complete an Interview Summary Form for each interview held where a matter of substance has been discussed during the interview by checking the appropriate boxes and filling in the blanks. Discussions regarding only procedural matters, directed solely to restriction requirements for which interview recordation is otherwise provided for in Section 812.01 of the Manual of Patent Examining Procedure, or pointing out typographical errors or unreadable script in Office actions or the like, are excluded from the interview recordation procedures below. Where the substance of an interview is completely recorded in an Examiners Amendment, no separate Interview Summary Record is required.

The Interview Summary Form shall be given an appropriate Paper No., placed in the right hand portion of the file, and listed on the "Contents" section of the file wrapper. In a personal interview, a duplicate of the Form is given to the applicant (or attorney or agent) at the conclusion of the interview. In the case of a telephone or video-conference interview, the copy is mailed to the applicant's correspondence address either with or prior to the next official communication. If additional correspondence from the examiner is not likely before an allowance or if other circumstances dictate, the Form should be mailed promptly after the interview rather than with the next official communication.

The Form provides for recordation of the following information:
- Application Number (Series Code and Serial Number)
- Name of applicant
- Name of examiner
- Date of interview
- Type of interview (telephonic, video-conference, or personal)
- Name of participant(s) (applicant, attorney or agent, examiner, other PTO personnel, etc.)
- An indication whether or not an exhibit was shown or a demonstration conducted
- An identification of the specific prior art discussed
- An indication whether an agreement was reached and if so, a description of the general nature of the agreement (may be by attachment of a copy of amendments or claims agreed as being allowable). Note: Agreement as to allowability is tentative and does not restrict further action by the examiner to the contrary.
- The signature of the examiner who conducted the interview (if Form is not an attachment to a signed Office action)

It is desirable that the examiner orally remind the applicant of his or her obligation to record the substance of the interview of each case. It should be noted, however, that the Interview Summary Form will not normally be considered a complete and proper recordation of the interview unless it includes, or is supplemented by the applicant or the examiner to include, all of the applicable items required below concerning the substance of the interview.

A complete and proper recordation of the substance of any interview should include at least the following applicable items:

1) A brief description of the nature of any exhibit shown or any demonstration conducted,
2) an identification of the claims discussed,
3) an identification of the specific prior art discussed,
4) an identification of the principal proposed amendments of a substantive nature discussed, unless these are already described on the Interview Summary Form completed by the Examiner,
5) a brief identification of the general thrust of the principal arguments presented to the examiner,
   (The identification of arguments need not be lengthy or elaborate. A verbatim or highly detailed description of the arguments is not required. The identification of the arguments is sufficient if the general nature or thrust of the principal arguments made to the examiner can be understood in the context of the application file. Of course, the applicant may desire to emphasize and fully describe those arguments which he or she feels were or might be persuasive to the examiner.)
6) a general indication of any other pertinent matters discussed, and
7) if appropriate, the general results or outcome of the interview unless already described in the Interview Summary Form completed by the examiner.

Examiners are expected to carefully review the applicant's record of the substance of an interview. If the record is not complete and accurate, the examiner will give the applicant an extendable one month time period to correct the record.

Examiner to Check for Accuracy

If the claims are allowable for other reasons of record, the examiner should send a letter setting forth the examiner's version of the statement attributed to him or her. If the record is complete and accurate, the examiner should place the indication, "Interview Record OK" on the paper recording the substance of the interview along with the date and the examiner's initials.

Interview Summary	Application No.	Applicant(s)
	11/477,807	MORRIS ET AL.
	Examiner	Art Unit
	Cordelia Zecher	2432

All participants (applicant, applicant's representative, PTO personnel):

(1) *Cordelia Zecher*.

(2) *Dale Lazar*.

(3) *Nicholas Panno*.

(4)_____.

Date of Interview: *12 January 2011*.

Type: a)☐ Telephonic b)☐ Video Conference
c)☒ Personal [copy given to: 1)☐ applicant 2)☒ applicant's representative]

Exhibit shown or demonstration conducted: d)☐ Yes e)☒ No.
If Yes, brief description: _____.

Claim(s) discussed: *1,13 and 25*.

Identification of prior art discussed: *Dozortsev*.

Agreement with respect to the claims f)☐ was reached. g)☒ was not reached. h)☐ N/A.

Substance of Interview including description of the general nature of what was agreed to if an agreement was reached, or any other comments: *Possible claim amendments were discussed to clarify the mask and allowing the object to run when the behaviour is permitted by the mask. Such amendments would require further search and consideration*.

(A fuller description, if necessary, and a copy of the amendments which the examiner agreed would render the claims allowable, if available, must be attached. Also, where no copy of the amendments that would render the claims allowable is available, a summary thereof must be attached.)

THE FORMAL WRITTEN REPLY TO THE LAST OFFICE ACTION MUST INCLUDE THE SUBSTANCE OF THE INTERVIEW. (See MPEP Section 713.04). If a reply to the last Office action has already been filed, APPLICANT IS GIVEN A NON-EXTENDABLE PERIOD OF THE LONGER OF ONE MONTH OR THIRTY DAYS FROM THIS INTERVIEW DATE, OR THE MAILING DATE OF THIS INTERVIEW SUMMARY FORM, WHICHEVER IS LATER, TO FILE A STATEMENT OF THE SUBSTANCE OF THE INTERVIEW. See Summary of Record of Interview requirements on reverse side or on attached sheet.

PTO/SB/30 (04-05)
Approved for use through 02/29/2008. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number

REQUEST FOR CONTINUED EXAMINATION (RCE) TRANSMITTAL

Address to:
Mail Stop RCE
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	MORRIS, Melvyn et al.
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

This is a Request for Continued Examination (RCE) under 37 CFR 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. See Instruction Sheet for RCEs (not to be submitted to the USPTO) on page 2.

1. **Submission required under 37 CFR 1.114** Note: If the RCE is proper, any previously filed unentered amendments and amendments enclosed with the RCE will be entered in the order in which they were filed unless applicant instructs otherwise. If applicant does not wish to have any previously filed unentered amendment(s) entered, applicant must request non-entry of such amendment(s).

   a ☐ Previously submitted. If a final Office action is outstanding, any amendments filed after the final Office action may be considered as a submission even if this box is not checked.
   - (i) ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on
   - (ii) ☐ Other

   b ☒ Enclosed
   - (i) ☒ Amendment/Reply
   - (ii) ☒ Information Disclosure Statement (IDS)
   - (iii) ☐ Affidavit(s)/Declarations
   - (iv) ☐ Other _____

2. **Miscellaneous**

   a ☐ Suspension of action on the above-identified application is requested under 37 CFR 1.103(c) for a period of _____ months. (Period of suspension shall not exceed 3 months; Fee under 37 CFR 1.17(i) required)

   b ☐ Other _____

3. **Fees** The RCE fee under 37 CFR 1.17(e) is required by 37 CFR 1.114 when the RCE is filed.

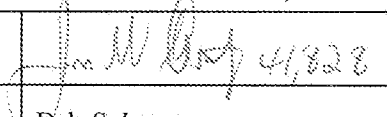
   a ☐ The Director is hereby authorized to charge the following fees, any underpayment of fees, or credit any overpayments, to Deposit Account No. 50-3266. I have enclosed a duplicate copy of this sheet.
   - (i) ☐ RCE fee required under 37 CFR 1.17(e)
   - (ii) ☐ Extension of time fee (37 CFR 1.136 and 1.17)
   - (iii) ☐ Other _____

   b ☐ Check in the amount of _____ enclosed

   c ☒ Payment by credit card (Form PTO-2038 enclosed)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT, REQUIRED

Signature		Date	January 28, 2011
Name (Print/Type)	Dale S. Lazar	Registration No.	28,872

This collection of information is required by 37 CFR 1.114. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)
FY 2005
(Fees pursuant to me Consolidated Appropriations Act, 2005 (H.R. 4818).)

Docket Number 339000-000003

Application Number 11/477,807 | Filed June 30, 2006

For METHODS AND APPARATUS FOR DEALING WITH MALWARE

Art Unit 2432 | Examiner C. Zecher

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Fee Code	
☒	One Month (37 CFR 1.1 7(a)(1))	$130	1251	130.00
☐	Two Month (37 CFR 1.1 7(a)(2))	$490	1252	
☐	Three Month (37 CFR 1.1 7(a)(3))	$1,110	1253	
☐	Four Month (37 CFR 1.1 7(a)(4))	$1730	1254	
☐	Five Month (37 CFR 1.1 7(a)(5))	$2,350	1255	

☐ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check in the amount of the fee is enclosed.

☒ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☐ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 50-3266. I have enclosed a duplicate copy of this sheet.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

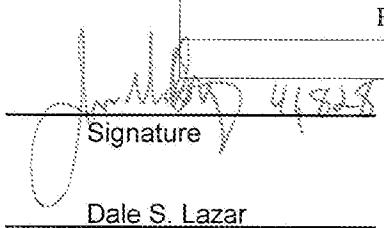
I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed (Form PTO/SB/96).

☒ attorney or agent of record. Registration Number 28,872

☐ attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 ___.



Signature

January 28, 2011
Date

Dale S. Lazar
Typed or printed name

703-773-4149
Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the more interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

☒ Total of 1 forms are submitted

This collection of information is required by 37 CFR 1.136(a). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S. C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 6 minutes to complete, including gathering, mating, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, US Patent and Trademark Office. U.S. Department OF Commerce, P.O. Box 1450, Alexandria, V.4 22313-4150. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS.
SEND TO: Commissioner for Patent. P.O. Box 1450, Alexandria, DA 22233-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2

American LegalNet, Inc
www.USCourtForms.com

DOCKET NO. 339000-000003 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:	MORRIS, Melvyn et al.	Confirmation No.	4785
Application No.	11/477,807	Group Art Unit:	2432
Filed:	June 30, 2006	Examiner:	C. Zecher

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

AMENDMENT UNDER 37 C.F.R. 1.114

Assistant Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

January 28, 2011

Dear Sir:

This is in response to the Final Rejection dated September 28, 2010, and having a period for response set to expire on December 28, 2010. This submission is believed to be timely because it is being submitted in conjunction with a Request for Continued Examination and a petition for a one-month extension of time, together with the requisite fee for same, is submitted herewith, thereby extending the period for response to January 28, 2011.

The following amendments and remarks are respectfully submitted. Reconsideration of the claims is respectfully requested.

Amendments to the Claims begin on page 2 of this paper.

Remarks/Arguments begin on page 18 of this paper.

Remarks/Arguments

We appreciate Examiner Zecher time and helpfulness during the personal interview on January 12, 2011.

Claims 1-11, 13-23, and 25-30 remain pending. Claims 12, 24, and 31-47 were previously deleted or withdrawn. Claims 1, 13, 25, and 30 have been amended. Claims 48-51 have been added and are fully supported by the specification (for example, see the first paragraph of page 36 for claims 48 and 50 and page 34 for claims 49 and 51).

On page 3 of the Office Action, the disclosure stands objected to for not being divided into the sections suggested by 37 CFR 1.77(b). The examiner correctly characterizes the arrangement specified in 37 CFR. 1.77(b) as a set of "guidelines" of a "preferred" layout that is "suggested" for the applicant's use. Thus, the Office Action itself acknowledges that the arrangement is discretionary on the part of the applicant. Also, 37 CFR 1.77(b) states that the specification "should," as opposed to "shall" or "must," follow the arrangement disclosed therein. This language clearly makes this arrangement optional, particularly when read in the context of other sections of 37 CFR dealing with the specification (e.g. 37 CFR 1.71(a) "(t)he specification must include a written description of the invention..," and 37 CFR 1.75(a) "(t)he specification must conclude with a claim..."). In this case, applicant declines the suggestion in the Office Action.

On page 4 of the Office Action, claim 30 stands rejected under 35 U.S.C. 101 as being directed to non-statutory subject matter. Claim 30 has been amended to be directed to a storage medium, which is a manufacture and therefore statutory subject matter.

On page 5 of the Office Action, claims 1-9, 12-21, 24-27, and 30 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev (US Pub. 2003/0177394) in view of Sheymov et al. (US Pub. 2001/0052014). Claims 1, 13, and 25 are independent claims, and claims 2-9, 12, 14-21, 24, 26, 27, and 30 depend on either claim 1, 13, or 25. Claims 1, 13, and 25 have been amended to overcome this rejection.

Claim 1 has been amended to read, *inter alia*, "when the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware; generating a mask for the computer object that defines acceptable behaviour for the computer object; running the computer object on at least one of the remote computers; automatically monitoring operation of the computer object on the at least one of the remote computers; allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask; disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask; and reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask." Claims 13 and 25 have been amended to include similar limitations. Dozortsev and Sheymov fail to disclose at least these limitations.

In the method of claim 1, when an unknown object is encountered, an initial determination may indicate that it is not known malware. In this event, the object is initially classified as not malware, and a mask is generated for the object. The mask defines operational bounds for the object. For example, if the unknown object purports to be a web browser, the mask may define a set of bounds that correspond to normal operation for web browsers. The unknown object may run on the remote computer, but only as long as its behaviour is permitted

by the mask. Therefore, if the unknown object does not exhibit malevolent behaviour, it may continue to run. If the unknown object attempts to exceed the bounds of the mask, this activity is detected, the object is prevented from running, and the object is reclassified as malware. User inconvenience is minimized because an object can run as long as no malevolent behaviour is exhibited, even if the object is unknown. Also note that the behaviour of the object running on a remote computer (rather than running in a sandbox environment on a central computer) is automatically monitored (not monitored by a human operator).

The examiner notes that Dozortsev teaches that when code is not known malware, it may continue to execute. See Office Action pages 3-4; Dozortsev par. 37-38. In Dorzotsev, when an object is unknown, it is forwarded to a central computer to be examined. While this examination takes place, a client computer may choose to allow the entire executable code to operate or prevent the entire executable code from operating, depending on configuration. See Dozortsev par. 37-38. However, the examination is carried out by execution of the code at the central location. See Dozortsev par. 37-38. When the examination is complete, the results are sent to the client computer and the code operation is allowed or terminated accordingly. See Dozortsev par. 40-42. This is clearly different from the present invention, wherein an unknown object is allowed to operate on the remote computer within limited bounds defined by a mask and is automatically prevented from operating and classified as malware if it tries to exceed those bounds. The examiner cites paragraph 40 of Dozortsev to find a mask. See Office Action page 5. Paragraph 40 merely indicates that file investigation may utilize several listed methods at a central location, making no mention of a behaviour-based mask or monitoring performance at a

remote computer. Dozortsev does not teach a behaviour-based mask at paragraph 40 or elsewhere.

The method of analysis in Dozortsev is different from the present invention as well. File investigation in Dozortsev may include executing the code in a sandbox environment. See Dozortsev par. 40. All behaviour can safely be allowed to run in an isolated sandbox environment on a central computer. Indeed, it is necessary for all behaviour to be fully exhibited so that a human operator can observe it and determine whether it is malicious. In contrast, the present invention establishes a behaviour-based mask. While the object is running on the remote computer, any behaviour exceeding the limits of the mask is automatically considered sufficient to classify an unknown object as malware. As noted above, this classification can be made automatically while the object is running on the remote computer. This further distinguishes the present invention from Dozortsev.

Thus, Dozortsev does not disclose every element of claim 1 for at least three reasons. Claim 1 requires running objects on a remote computer while behaviour is monitored. Dozortsev teaches receiving an object at a central computer and investigating the object centrally. See Dozortsev par. 38. Claim 1 requires monitoring objects for behaviour that exceeds the limits of a behaviour-based mask while the object is running. Dozortsev does not disclose any behaviour-based mask. Claim 1 requires monitoring the operation of objects running on a remote computer automatically. Dozortsev teaches the use of a sandbox environment wherein behaviour of an object can be observed by a human analyst. See Dozortsev par. 40.

The examiner cites Sheymov to teach having both centralized and distributed databases in a network access monitoring method. See Office Action page 2; Sheymov par. 23. Sheymov

does not compensate for any of the deficiencies of Dozortsev discussed above. Specifically, Sheymov does not discuss establishing a mask for an unknown object, running the object on a remote computer, automatically monitoring the object, allowing the object to run if its operation does not exceed the bounds of the mask, and preventing the object from running and reclassifying it as malware if its operation exceeds the bounds of the mask. Because Dozortsev and Sheymov fail to teach at least the limitations of claims 1, 13, and 25 discussed above, either alone or in combination, claims 1-9, 12-21, 24-27, and 30 should be allowed.

On page 8 of the Office Action, claims 10, 11, 22, 23, 28, and 29 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Sheymov and Barton et al. (US Pub. 2005/0021994). Barton is cited for disclosing the step of indicating the file name, size and location of an object. Barton fails to compensate for the deficiencies of Dozortsev and Sheymon noted above in the analysis of claims 1, 13, and 25. Since claims 10, 11, 22, 23, 28, and 29 depend on either claim 1, 13, or 25, they are patentable for the reasons discussed above with respect to claims 1, 13, and 25.

The application has been amended to place the application condition for allowance. All amendments are supported by the specification. An early notice of allowance is earnestly solicited.

Please charge any shortage in the fees or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

41,828

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/NP/jcc
PO Box 2758
Reston, VA 20195
(703) 773.4000 Telephone
(703) 773.5200 Fax

Docket No.: 339000-000003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Melvyn Morris	Conf. No.	4785
SERIAL NUMBER:	11/477,807	Group:	2432
FILED:	June 30, 2006	Examiner:	C. Zecher
FOR:	METHODS AND APPARATUS FOR DEALING WITH MALWARE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

January 28, 2011

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☒ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☐ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☒ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

41,828

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/jcc
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 2

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	MORRIS, Melvyn et al.
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -60/789,156			
		US -7,013,483	3/14/2006		
		US -2003/0023857	1/30/2003	Hinchlife	
		US -2005/0027686	2/3/2005	Shipp	
		US -6,338,141	1/8/2002	Wells	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 96/30829	10/3/1996	APPLE COMPUTER, INC.		
		WO 99/15966	4/1/1999	SYMANTEC CORPORATION		
		EP 1536341 A2	6/1/2005	MICROSOFT CORPORATION		

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional). 2 See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. 3 Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). 4 For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. 5 Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. 6 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	2	of	2

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	MORRIS, Melvyn et al.
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Image File Wrapper of U.S. Appl 11/694,261 as of January 28, 2011	
		"GHOSTBUSTER 3", Mihai CHIRIAC, pages 1-7, Prevx1 September 2005	
		A. Stephan; "Defeating Polymorphism: Beyond Emulation"; Virus Bulletin Conference; October 2005	

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
[1] Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Docket No.: 339000-000003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Melvyn Morris	Conf. No.	4785
SERIAL NUMBER:	11/477,807	Group:	2432
FILED:	June 30, 2006	Examiner:	C. Zecher
FOR:	METHODS AND APPARATUS FOR DEALING WITH MALWARE		

LIST OF RELATED CASES

Serial Number	Publication Number	Filing or Publication Date	Status
11/694,261	US 2008-0040710	February 14, 2008	Pending

The cases listed on this Notice of Related Cases include cases which may contain information that is material to patentability. The listing of a case on this Notice should not be taken as an indication or admission that any information contained therein is material. Prior art for each case listed on this Notice may have been cited. **The files corresponding to the listed cases, which are available to the Examiner, may not have been examined to ascertain the materiality of any prior art therein.** Accordingly, the Examiner is requested to review the file for each case listed on this Notice in order to assess the materiality of such prior art.

Respectfully submitted,

DLA PIPER LLP US

41,328

Dale S. Lazar

P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

Reg. No. 28,872
Attorney of Record
Date: January 28, 2011

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/455

A1

(11) International Publication Number: WO 96/30829

(43) International Publication Date: 3 October 1996 (03.10.96)

(21) International Application Number: PCT/US96/03769

(22) International Filing Date: 20 March 1996 (20.03.96)

(30) Priority Data:
08/414,877 31 March 1995 (31.03.95) US

(71) Applicant: APPLE COMPUTER, INC. [US/US]; MS 38-PAT, 1 Infinite Loop, Cupertino, CA 95014 (US).

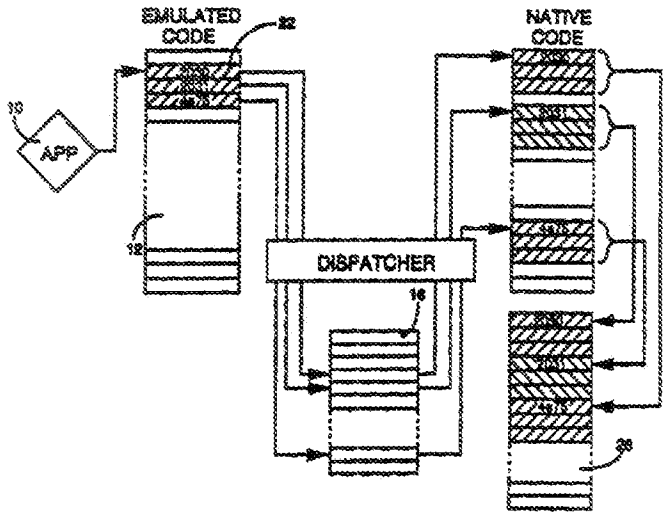
(72) Inventor: MORLEY, John, E.; 5389 Kula Mau'u Street, Kapaa, HI 96746 (US).

(74) Agents: CARMICHAEL, Paul, D. et al.; Apple Computer, Inc., MS 38-PAT, 1 Infinite Loop, Cupertino, CA 95014 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SOFTWARE EMULATION SYSTEM WITH DYNAMIC TRANSLATION OF EMULATED INSTRUCTIONS FOR INCREASED PROCESSING SPEED



(57) Abstract

The execution time overhead of software emulation is reduced by selecting frequently emulated instruction sequences in the software being emulated, and translating those instruction sequences into the instruction set of the emulating processor. In a first phase, frequently repeated sequences of emulated computer code are identified and selected for translation. In a second phase, the selected sequences from the instruction set of the emulated processor are translated into equivalent sequences for the instruction set of the emulating processor. In a third phase, the instruction sequence of the emulating processor is executed in lieu of emulating the original instructions from the emulated software.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgystan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SG	Singapore
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LR	Liberia	SZ	Swaziland
CS	Czechoslovakia	LT	Lithuania	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	LV	Latvia	TJ	Tajikistan
DK	Denmark	MC	Monaco	TT	Trinidad and Tobago
EE	Estonia	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	UG	Uganda
FI	Finland	ML	Mali	US	United States of America
FR	France	MN	Mongolia	UZ	Uzbekistan
GA	Gabon	MR	Mauritania	VN	Viet Nam

SOFTWARE EMULATION SYSTEM WITH DYNAMIC TRANSLATION OF EMULATED INSTRUCTIONS FOR INCREASED PROCESSING SPEED

Field of the Invention

The present invention is directed to emulation systems which execute software instructions designed for a specific instruction set on a processor which supports a different instruction set, and more particularly to the use of dynamic translation of instructions to increase emulation performance.

Background of the Invention

The central processing unit (CPU) of a computer is designed to execute a particular set of software instructions, which form the basis for the computer's operating system. Similarly, most application programs are designed to work with specific operating systems. In the past, for example, many computers were designed for complex instruction set computing, and are identified as CISC processors. Accordingly, application programs designed to run on these computers consist of commands taken from the instruction set for their processors. More recently, there has been a move toward reduced instruction set computing, so-called RISC. Application programs which are designed for a CISC class of processor will not normally run on a RISC-based processor. In order to utilize these application programs on a RISC processor, it is necessary for the RISC processor to emulate the operation of a CISC processor.

The instruction set of a typical processor defines a set of basic operations that can be performed on one or more operands. Software emulation that functionally implements the operation of a particular processor, for example a CISC processor, must be able to decode an instruction from the instruction set for the emulated processor and execute an equivalent sequence of instructions from the instruction set of a different processor, e.g. a RISC processor, on which the emulation is being performed.

In one of the more efficient approaches to software emulation, a jump table, also known as a dispatch table, is employed to decode instructions in an emulated processor's instruction set. Generally speaking, an instruction being emulated provides an address to a location in a dispatch table. This entry in the dispatch table contains a pointer to a sequence of equivalent instructions in the instruction set for the processor. This sequence of instructions forms a semantic routine whose execution results in the desired emulation of the original instruction.

One characteristic of an emulation system which has a significant impact on its overall performance is the considerable execution time overhead which the emulation system imposes. In particular, a good percentage of the time required for emulation is spent in the dispatching operations. In general, each instruction generated by an application program, in the instruction set for the processor being emulated, causes the emulator to address the dispatch table, which results in a jump to the corresponding semantic routine in the native, or emulation, code. Thus, for each instruction in the emulated code, the following sequence of actions occurs: (a) fetching the instruction to be implemented, (b) addressing the dispatch table, (c) obtaining the pointer to the native code, (d) fetching the first instruction for the corresponding semantic routine in the native code, and (e) executing the semantic routine.

It can be appreciated that the need to perform this sequence of operations for every instruction that is issued in the emulated code presents a substantial amount of processing overhead, which can significantly affect the overall performance of the emulator. It is desirable, therefore, to decrease the processing overhead that is required to emulate instructions, and thereby improve the overall performance of the emulator.

Summary of the Invention

In accordance with the present invention, the processing overhead of an emulating system is reduced by selecting frequently emulated instruction sequences in the software being emulated and translating those instruction sequences into the native instruction set of the emulating processor. The dynamic translation process of the present invention is implemented in three main phases. In the first phase, an identification is made of those code sequences in the emulated software that are frequently repeated. This can be done, for example, by recording all program counter values that are produced by instructions that cause a non-sequential change in the value of the program counter for the emulated software. On a periodic basis, the recorded program counter values are analyzed, to identify code sequences which are emulated frequently enough to warrant dynamic translation. The program counter values which are selected identify the starting point of emulated code sequences to be translated.

In the second phase, the selected code sequences are translated from the instruction set of the emulated processor into the instruction set of the emulating processor. For each emulated instruction in a selected code

sequence, its equivalent code sequence in the native instruction set is obtained from the emulator's set of semantic routines, by indexing into the dispatch table with a binary code for the emulated instruction. The successively retrieved code sequences are cumulatively stored in an instruction buffer, until each instruction in the selected sequence has been translated.

In the final phase, the translated sequence is executed in place of emulating the untranslated sequence. This is done by assigning each newly translated code sequence an available operation code from the emulated instruction set. The start of the untranslated instruction sequence is replaced with the value of this operation code, and the address of the translated instruction sequence is inserted in the dispatch table as the corresponding entry for this operation code. From that point on, the emulation software automatically executes the translated code sequence in lieu of emulating the untranslated code sequence, resulting in a substantial improvement in the time required to perform the emulation.

As a further feature of the invention, the computation of gratuitous values that are not employed during execution, such as flag values for emulated condition code registers, can be eliminated to further optimize the translated code sequence. In the implementation of this feature, an analysis of the emulated code sequence being translated is performed to identify any emulated instructions in the sequence which unnecessarily modify emulated condition flags. The modification of an emulated condition flag is unnecessary when a subsequent emulated instruction in the sequence modifies the same flag and no intervening emulated instruction either refers to the flag value or causes a non-sequential change to the emulated program counter value. For each emulated instruction where these conditions are met, an alternative native code sequence, which does not include the instructions to compute the unnecessary condition flag values, is employed in the translation of that emulated instruction.

Further features of the invention, as well as the advantages offered thereby, are explained in greater detail hereinafter with reference to specific embodiments illustrated in the accompanying drawings.

Brief Description of the Drawings

Figure 1 is a block diagram of the software components of an emulator system;

Figure 2 is a block diagram similar to Figure 1, which depicts the conventional translation of a code block;

Figure 3 is another block diagram similar to Figure 1, depicting dynamic translation in accordance with the present invention; and

Figure 4 is a block diagram depicting the execution of a translated code block.

Detailed Description

To facilitate an understanding of the present invention, it is described hereinafter with reference to its implementation in specific embodiments. In particular, the features of the invention are described in the context of an emulator system in which the instruction set of a processor designed for complex instruction set computing (CISC) is emulated by a processor which is designed for reduced instruction set computing (RISC). In this context, the instruction set for the CISC processor is identified as the "emulated code", and the instruction set for the RISC processor is labeled the "native code". It will be appreciated, however, that the practical applications of the invention are not limited to this particular embodiment. Rather, the invention will find utility in any emulator system in which a processor of one type functions as a virtual processor of another type.

Generally speaking, an emulator is a software program that runs on a computer and enables the computer to behave as though it were a different type of computer. In other words, it enables a computer of one type to function as a virtual processor of another type. Depending upon the portions of the program that are being executed at any particular time, as well as the architecture of the computer itself, various parts of the program may reside in different types of memory, including main memory, cache memory and/or permanent storage. In the following description, the term memory is employed in a generic sense to refer to any and all types of memory in which the emulator may reside. The particular forms of memory that are used for any particular implementation of the invention will be dependent upon choices made by the programmer, as well as the architecture of the computer system in which the implementation takes place. For this reason, discussion of specific registers and the like is omitted from the following discussion, so as not to obscure an understanding of the basic principles of the invention.

The basic operations that are carried out in an emulator are illustrated in the block diagram of Figure 1. The blocks in this diagram represent different

items of software that are stored in the memory of the computer which is functioning as an emulator. Referring to Figure 1, an application program 10 consists of commands that are designed for the instruction set of the emulated processor, in this case the CISC processor. In Figure 1, the complete set of instructions for the emulated processor are represented by a list 12. In practice, the application program issues individual instructions for execution by the computer's CPU. For example, the instructions might be issued in response to user inputs. A particular instruction to be emulated by the processor is forwarded to a dispatcher 14. Associated with the dispatcher is a dispatch table 16, which contains pointers to sequences of instructions in the native code 18 that functionally correspond to each of the emulated code instructions. In response to information obtained from the dispatch table, the dispatcher 14 calls one or more corresponding instructions in the native code. The processor, in this case the RISC processor, executes the called instruction(s), and thereby emulates the operation of the CISC processor. In the particular example illustrated in Figure 1, a single instruction issued by the application program 10 results in the execution of three consecutive instructions in the native code (represented by the shaded area 20). This set of three instructions in the native code which emulates the instructions in the emulated code is referred to as a "semantic routine".

Typically, instructions in the emulated code are not issued in some random order. Rather, groups of sequentially stored instructions tend to be repeatedly executed to perform an operation. In the context of the present invention, the term "code block" is employed to denote a series of sequential instructions in the emulated code which begins with the target of some non-sequential branch operation and whose extent is the maximum number of sequential instructions that can be executed by following code paths that begin with the first instruction in the sequence and are wholly contained within the sequence. A "code block", as defined herein, may contain conditional and unconditional branch instructions.

Referring to Figure 2, an example of a code block is represented by the shaded area 22. In a conventional emulation process, the instructions in a code block are individually translated each time they are encountered during the emulation process. As each instruction is fetched by the dispatcher 14, it functions as an address to the dispatch table 16. The dispatcher 14 retrieves the pointer stored at the addressed portion of the table 16, and calls the corresponding semantic routine to which the pointer refers, for execution. In the specific example of Figure 2, the code block 22 consists of three instructions.

As is conventional in most programs, each instruction is represented by a unique numerical value, known as its operation code, or opcode. For this example, the three instructions have the opcodes 2030, 2031 and 4e75. these three instructions respectively result in three dispatch operations to separately retrieve and execute three semantic routines for those opcodes, respectively. This need to continually dispatch each instruction as it is issued by the application program presents a significant amount of processing overhead in the overall operation of the emulator.

In accordance with the present invention, this processing overhead can be considerably reduced through dynamic translation of selected code blocks in the emulated application program. To implement this feature, the code blocks which are emulated frequently enough to warrant dynamic translation are first identified. This can be carried out by recording program counter values that produce non-sequential changes in the sequence of instructions being emulated. Whenever an instruction from the CISC code is emulated that results in a non-sequential change to the program counter, the new program counter value is recorded, for example by pushing its value onto a programmatic stack. In essence, each recorded program counter value represents the starting point of a new code block. Whenever there is a break in the operation of the emulator, for example as the processor stops emulation to service a special event or an interruption, the accumulated values are removed from the stack and analyzed to identify code blocks that are emulated more than a defined number of times within a predetermined time window. For example, if a particular block is emulated more than 256 times within a period of about 16 milliseconds, it may be selected. Any suitable approach can be employed to select the program counter values that identify the code blocks which occur with sufficient frequency. In the preferred embodiment of the invention, as each recorded program counter value is removed from the stack, its value is used as a hash index into a table of frequency counts, and the corresponding entry in the table is incremented by one. When the count is incremented beyond a predetermined threshold value, the program counter value corresponding to that table entry is placed on a list of code blocks to be dynamically translated.

The selected values identify the starting points of emulated code sequences, or code blocks, that are to be dynamically translated. Each code block begins with an instruction that is the target of a non-sequential change in the value of the program counter, and ends with the last instruction that can be reached by any code path that is wholly contained within the code block. Once

a code block has been selected for dynamic translation, each instruction in the selected code block is translated from the instruction set of the emulated processor into the native code. For each instruction in the code block, its equivalent semantic routine is identified in the native code, using the dispatch table 16. Referring to Figure 3, as the first instruction in the code block, in this example 2030, is translated, its corresponding semantic routine 2030 is stored in a buffer 26. As the next instruction in the code block is translated, its semantic routine 2031 is appended to the instructions stored in the buffer 26. This process is repeated until each instruction in the selected code block has been translated and its corresponding semantic routine stored in the instruction buffer 26.

This entire dynamic translation procedure, namely (a) the analysis of program counter values, (b) the identification of frequently emulated code blocks, and (c) the translation and storage of selected code blocks in the buffer, preferably takes place during the interruption of the emulation, i.e. prior to the time that the event which interrupted the emulation is serviced.

Once the entire code block has been translated in this manner, the translated sequence is executed in place of emulating the individual instructions of the untranslated sequence. To do so, the newly translated code sequence is assigned an opcode from the instruction set of the emulated processor. For example, the instruction set for the Motorola 68000 series of microprocessors employs a 16-bit opcode. This presents the capability for over 65,536 different possible opcodes that can be represented. Of this number, about 50,000 are actually employed for the instruction set, leaving a remainder of about 15,000 unassigned operation codes that are available for other uses. In the implementation of the present invention, each newly translated code sequence that is stored in an instruction buffer 26 is assigned one of the available operation codes from the emulated instruction set. The beginning instruction of the untranslated instruction sequence 22 is overwritten with the value of this new, unassigned operation code. Referring to the example of Figure 4, the opcode 6001 is assumed to be one such unassigned, and hence available, opcode. It is substituted for the opcode 2030 of the first instruction in the code block 22. In the dispatch table 16, the address of the translated instruction sequence in the buffer 26 is inserted as the corresponding entry for the new operation code. Subsequently, whenever the application program issues the first instruction in the translated code block, the dispatcher 14 will point to the translated code sequence, which will be automatically executed by the emulation software, in lieu of emulating the untranslated code sequence.

Consequently, only a single call to the dispatcher is made, rather than a separate call for each instruction in the code block, resulting in a substantial improvement in the time required to perform the emulation.

As a side-effect of most instructions performed by a CISC class of processor, a register known as the Condition Code Register is modified to reflect certain data conditions that resulted from the execution of that instruction. These data conditions are typically referred to as flags. One of the more time-consuming aspects of performing a software emulation is the computation of new values for the flags contained in the emulated processor's Condition Code Register. Often, the computation of new condition code flags in response to a particular instruction is a gratuitous exercise, in the sense that a subsequent instruction typically replaces the previous instruction's condition code flag values with its own.

As a feature of the present invention, the performance of the emulator can be further improved by eliminating the computation of gratuitous condition code values under those conditions in which their elimination results in semantically identical results in the emulation of the code block. This feature of the invention can be implemented during the second phase of the dynamic translation process, i.e. while the instructions of the selected code block are being translated. During this phase, the instructions in the emulated code are analyzed to identify those situations in which a subsequent instruction, or set of instructions, modifies all of the condition code flags that are set by the instruction under analysis. If no intervening instructions refer to the condition code flags, or cause a potential non-sequential change in the program counter value, the computation of the condition code flags can be eliminated. In this situation, an alternative semantic routine, which does not include computation of the condition code flag values, is employed for the translation of the instruction under analysis. The alternate semantic routine for an emulated instruction can be stored immediately following the primary semantic routine for that instruction. In the case of some CISC instructions, the elimination of condition code flag computations can reduce the number of native instructions in the semantic routine by two-thirds, resulting in appreciable speed and code size savings.

From the foregoing, it can be seen that the present invention employs dynamic translation to significantly improve the execution time of a software emulator. The dynamic translation results in a single dispatch operation for an entire code block of the emulated code, thereby significantly reducing the dispatch overhead associated with the emulation process. In the

implementation of the invention, it is desirable to translate as many instructions as possible in a code block. The longer the sequence of instructions that are dynamically translated, the greater the increase in the emulation speed. This increased performance is due to the elimination of instruction decoding overhead, as well as increased locality within the target processor's instruction caches. More particularly, the sequence of native instructions that comprises a translated code block will generally fit within the available instruction cache memory of a RISC processor. When the translated block contains frequently iterated loops of code, a large stream of instructions will be executed directly out of cache memory at a much greater overall instruction throughput. Conversely, if that same block were to be emulated, there will likely be cache conflicts generated between the emulator's dispatcher, the dispatch table, and the semantic routines, and these delays will be multiplied by the number of loop iterations within the block.

It will be appreciated by those of ordinary skill in the art that the present invention can be embodied in other specific forms without departing from the spirit or essential characteristics thereof. The presently disclosed embodiments are considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated b #`pended claims, rather than the foregoing description, and all changes that come within the meaning and range of equivalents thereof are intended to be embraced therein.

What is claimed is:

1. In a computer having a processor which executes instructions from a first set of instructions, a method for emulating the operation of a different processor which executes a second set of instructions, comprising the steps of:

identifying a sequence of plural instructions in said second set of instructions which are repeatedly called with a predetermined frequency;

translating each of the instructions in said sequence into one or more corresponding instructions in said first set of instructions;

cumulatively storing the corresponding instructions from said translated instructions as a group; and

executing the group of stored instructions when an initial instruction in said sequence is issued during subsequent operation of the computer.

2. The method of claim 1 wherein each instruction in said second set of instructions has an associated operation code, and further including the steps of assigning an operation code to said group of stored instructions, and substituting said assigned operation code for the operation code associated with said initial instruction.

3. The method of claim 1 wherein said computer includes a program counter which identifies instructions from said second set to be executed, and wherein the step of identifying frequently called instructions comprises the steps of detecting each occurrence of a non-sequential change of value in said program counter, and counting the number of times that each non-sequential change of value is detected within a predetermined time period.

4. The method of claim 3 wherein said counting step comprises the steps of loading a value associated with said non-sequential change in a stack upon each detected occurrence, emptying said stack in response to an interruption in an emulation operation, and counting the number of occurrences of each value in said stack.

5. The method of claim 1 further including the steps of analyzing the instructions in said sequence to determine whether a given instruction calculates values that are recalculated by a subsequent instruction in said sequence and which are not utilized prior to said recalculation, and translating said given instruction into a corresponding instruction in said first set which does not calculate said values.

6. The method of claim 5 wherein said calculated values are condition code flag values.

7. A software emulation system for emulating the operation of a first processor which executes a first set of instructions by means of a different processor which executes a second set of instructions, comprising:

a dispatcher for translating instructions issued from said first set of instructions into one or more corresponding instructions in said second set of instructions;

means for identifying a sequence of plural instructions in said first set of instructions which are repeatedly called with a predetermined frequency;

a buffer which cumulatively stores instructions from said translated instructions that respectively correspond to the instructions in an identified sequence; and

means for causing the instructions stored in said buffer to be executed when an initial instruction in said sequence is issued.

8. The system of claim 7 wherein each instruction in said first set of instructions has an associated operation code, and wherein said causing means assigns an operation code to the instructions stored in said buffer, and substitutes said assigned operation code for the operation code associated with said initial instruction.

9. The system of claim 7 wherein said identifying means includes a program counter which identifies instructions from said first set to be executed, means for detecting each occurrence of a non-sequential change of value in said program counter, and means for counting the number of times that each non-sequential change of value is detected within a predetermined time period.

10. The system of claim 7 further including means for analyzing the instructions in said sequence to determine whether a given instruction calculates values that are recalculated by a subsequent instruction in said sequence and which are not utilized prior to said recalculation, and means for translating said given instruction into a corresponding instruction in said first set which does not calculate said values.

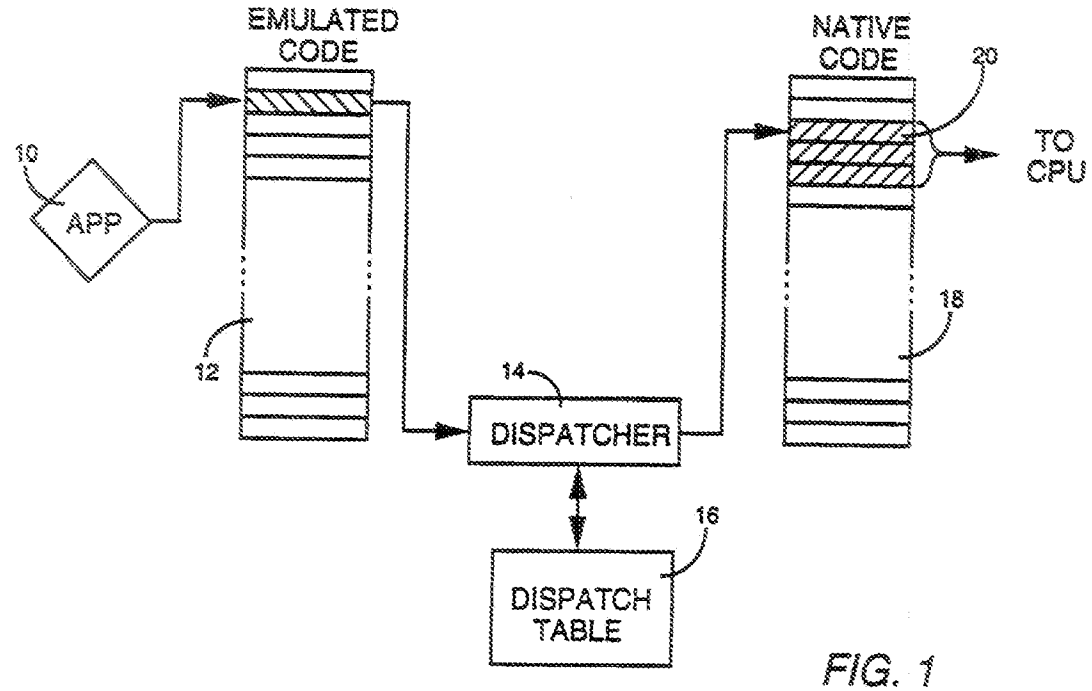

EMULATED CODE
NATIVE CODE
20
10
APP
TO CPU
18
12
14
DISPATCHER
16
DISPATCH TABLE

FIG. 1

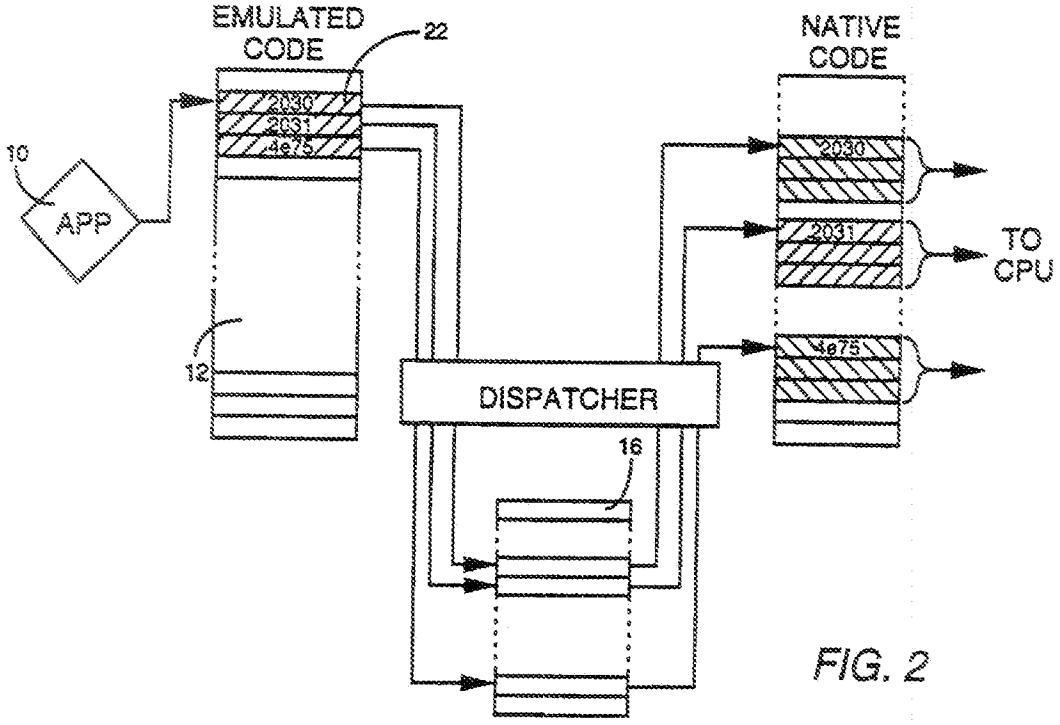

EMULATED CODE
22
NATIVE CODE
2030
2031
4e75
10
APP
TO CPU
12
DISPATCHER
16

FIG. 2

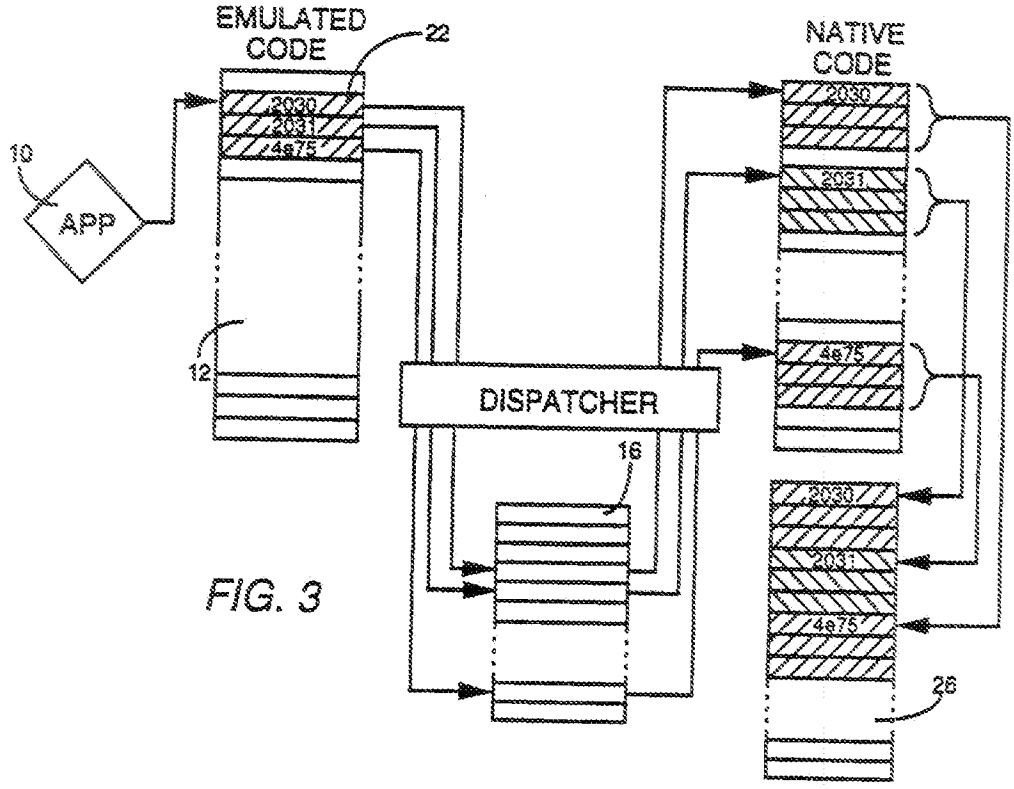

EMULATED CODE
22
NATIVE CODE
2030
2031
4e75
10
APP
12
DISPATCHER
16
2030
2031
4e75
26

FIG. 3

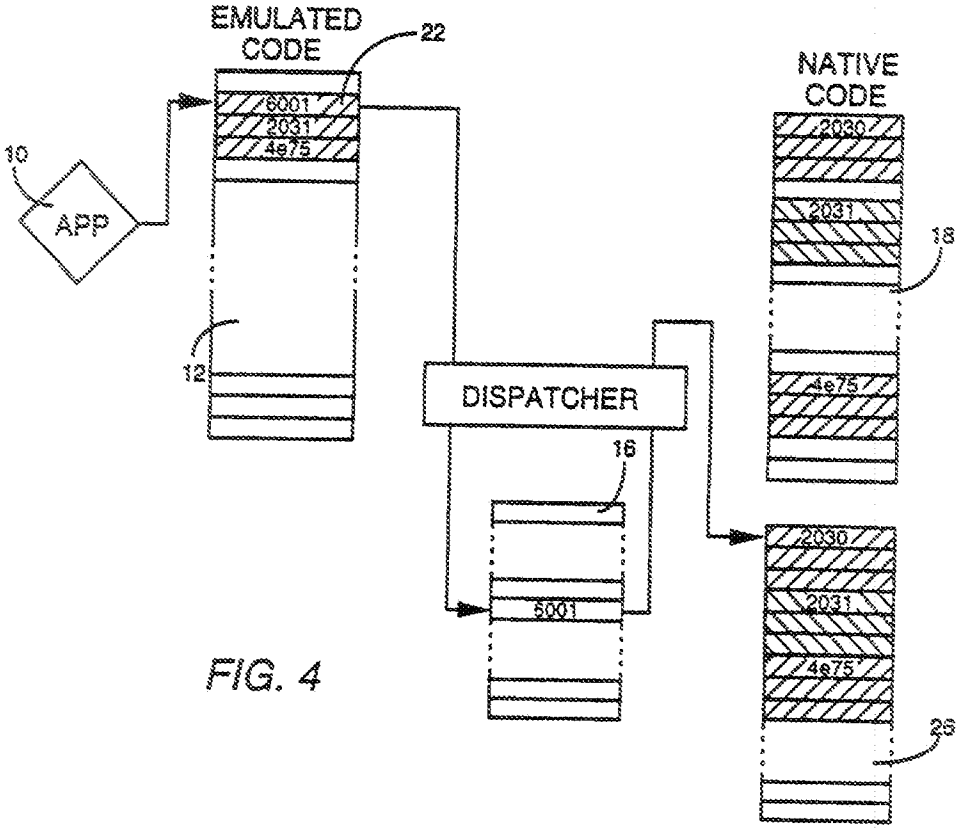

EMULATED CODE
22
NATIVE CODE
6001
2031
4e75
2030
2031
18
10
APP
12
DISPATCHER
16
5001
2030
2031
4e75
28

FIG. 4

INTERNATIONAL SEARCH REPORT

I ational Application No
PCT/US 96/03769

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/455

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category '	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BYTE, vol. 19, no. 4, 1 April 1994, page 119/120, 122, 124, 126, 128, 130 XP000435283 HALFHILL T R: "EMULATION: RISC'S SECRET WEAPON"	1,5-7,10
Y	see page 126, right-hand column, line 16 - page 130, left-hand column, line 13	2,8
A	---	3,9
Y	IBM TECHNICAL DISCLOSURE BULLETIN, vol. 36, no. 6A, 1 June 1993, pages 35-38, XP000370750 "SELECTING PREDECODED INSTRUCTIONS WITH A SURROGATE" see the whole document	2,8
	--- -/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

' Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 July 1996	09.08.96

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Klocke, L

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO,A,88 07718 (INSIGNIA SOLUTIONS LIMITED) 6 October 1988 see page 11, line 24 - page 12, line 2 ; page 15, lines 9-17; page 18, lines 15-19 ---	1,2,5-8, 10
A	WO,A,94 27214 (APPLE COMPUTER) 24 November 1994 see the whole document ---	1,7
A	SIGPLAN '87 SYMPOSIUM ON INTERPRETERS AND INTERPRETIVE TECHNIQUES, 24 - 26 June 1987, ST. PAUL,US, pages 1-13, XP002009322 C.MAY: "MIMIC: A fast system/S70 simulator" see the whole document -----	1-10

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 96/03769

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
WO-A-8807718	06-10-88	GB-A,B	2203572	19-10-88
		US-A-	5406644	11-04-95
		DE-T-	3788877	23-06-94
		EP-A-	0310600	12-04-89
		JP-T-	1503181	26-10-89
WO-A-9427214	24-11-94	AU-B-	6701594	12-12-94

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/00	A1	(11) International Publication Number: **WO 99/15966**
		(43) International Publication Date: 1 April 1999 (01.04.99)

(21) International Application Number: PCT/US98/17609

(22) International Filing Date: 25 August 1998 (25.08.98)

(30) Priority Data:
08/935,577 23 September 1997 (23.09.97) US

(71) Applicant: SYMANTEC CORPORATION [US/US]; 10201 Torre Avenue, Cupertino, CA 95014 (US).

(72) Inventor: NACHENBERG, Carey, S.; 19585 Shadow Glen Circle, Northridge, CA 91326 (US).

(74) Agents: OKAMOTO, James, K. et al.; Fenwick & West LLP, Two Palo Alto Square, Palo Alto, CA 94306 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: DYNAMIC HEURISTIC METHOD FOR DETECTING COMPUTER VIRUSES

(57) Abstract

A dynamic heuristic method for detecting computer viruses comprising three phases: a decryption phase, an exploration phase, and an evaluation phase. A purpose of the decryption phase is to emulate a sufficient number of instructions to allow an encrypted virus to decrypt its viral body. A purpose of the exploration phase is to emulate at least once all sections of code within a region deemed likely to contain any virus present in the target program. A purpose of the evaluation phase is to analyze any suspicious behavior observed during the decryption and exploration phases to determine whether the target appears to be infected.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

DYNAMIC HEURISTIC METHOD FOR DETECTING COMPUTER VIRUSES

I. BACKGROUND

A. Technical Field

This invention relates to computer antivirus software. More particularly, this invention relates to software for detecting unknown computer viruses using emulation and artificial intelligence.

B. Related Art

Computer virus detection technology may be divided into categories such as signature scanning, integrity checking, and non-integrity-based unknown virus detection (also called heuristics). This section discusses these categories of antivirus technology.

Signature scanning antivirus programs work by scanning files for signatures of known viruses. A signature is a sequence of bytes that may be found in a virus program code, yet is unlikely to be found elsewhere. To "extract" a signature, an antivirus researcher must analyze the virus. Once this signature is determined, it is recorded in a database of virus signatures to be used by an antivirus program. The antivirus program scans a target program (executable file, boot record, or possibly document file with a macro) to detect the presence of a virus signature. If a signature is found, then the target program is deemed infected. Otherwise, the target program is considered uninfected.

A signature scanning antivirus program can identify particular virus strains for removal and may have a low "false-positive" rate if properly implemented. However, only viruses whose signatures have already been determined and stored in the signature database may be detected using signature scanning. Moreover, the signature database must be updated frequently to detect the latest viruses.

Integrity checking (called "inoculation" by the commercial Norton Anti-Virus product from Symantec Corp.) is a technique in which "snapshots" or "fingerprints" are taken of programs (executable files, boot records) on the computer under the assumption that all these files are in an uninfected state. These fingerprints are typically taken after the computer has

been scanned with a virus scanner that reasonably assures the computer is virus-free. These fingerprints are then saved into a database for later integrity-based scans.

During subsequent integrity-based scans of the computer, the antivirus program verifies that each previously fingerprinted program on the computer matches its fingerprint. If a program does not match its fingerprint, then the antivirus program typically uses artificial intelligence to determine if the modification is "virus-like" or merely a valid program update. If the modification appears due to an infection by a virus, the antivirus program typically alerts the user to the modification and gives the user the option to repair the damage, if possible.

Because integrity checking does not scan for virus signatures, it can be used to detect new and (as yet) unknown virus strains. Integrity checking works because viruses must generally make changes to their host program, and these changes can be detected if the database of fingerprints of clean programs is properly created and maintained. However, integrity checking does not work if the computer is not virus-free when the programs are fingerprinted. A virus-infected program that is "inoculated" along with other clean programs would be a safe haven from where the virus can infect other programs. Furthermore, when a change is detected by integrity checking, it is often difficult for the antivirus program to determine if the change was virus-induced or user-induced (e.g., the user may update a program by installing a new version or copying an updated file). If this determination cannot be made by the antivirus program, the user must be called upon to make this determination, and many users are not knowledgeable enough to do so.

Non-integrity-based (also called "heuristic") unknown virus detection is used to detect new and unknown viruses without any integrity information. A heuristic antivirus program examines a target program (executable file, boot record, or possibly document file with a macro) and analyzes its program code to determine if the code appears virus-like. If the target program's code appears virus-like, then the possible infection is reported to the user.

Heuristic virus detection can detect new and unknown viruses that have not yet been analyzed by antivirus researchers since it does not use virus signatures. Because the heuristic technique does not use integrity information, it does not require fingerprints of programs to be taken and saved when the computer is in a known clean state.

Heuristic virus detection can be classified as either static or dynamic. The primary difference between these two detection schemes is that the dynamic method uses CPU emulation while the static method does not.

i. Static Heuristic Detection

In static heuristic virus detection, the antivirus program searches the instructions of a target program for sequences of instructions that perform operations typically used by viruses. Unlike virus signatures, these sequences are not designed to be specific to a single virus. Instead, they are meant to be as general as possible in order to detect the operation of many different viruses.

For example, the following sequence X86 (Intel microprocessor) machine code instructions may be used to open a file:

assembly language	machine code
MOV DX, ????	BA ?? ??
MOV AX, 3D02	B8 02 3D
INT 21	CD 21

where ?? indicates that the byte may vary in different viruses. Similarly, the following sequence of X86 machine code instructions may be used to write to a file:

assembly language	machine code
MOV DX, ????	BA ?? ??
MOV CX, ????	B9 ?? ??
MOV AX, 4000	B8 00 40
INT 21	CD 21

where again ?? indicates that the byte may vary in different viruses.

Static heuristic antivirus programs search for sequences of bytes like those shown above, then makes an assessment of viral infection based on the sequences it found. For example, if the

static heuristic antivirus program finds a file open operation, followed by file read and write operations, and also finds a character (ASC11) string "VIRUS" in the program, it may report that the file is infected by an unknown virus.

Some (self-decrypting) computer viruses have encrypted viral bodies. Sequences of instructions that exhibit virus-like behavior are not identifiable while they are encrypted. Therefore, some static heuristic detection programs precede the behavior searching phase with a decryption phase which is typically performed using a CPU emulator.

Although static heuristic detection programs can be relatively fast, they may recognize only some of the numerous different ways of performing various virus-like operations. For example, a virus writer may re-order the instructions of the file open sequence above as follows:

assembly language	machine code
MOV AX, 3D02	B8 02 3D
MOV DX, ????	BA ?? ??
INT 21	CD 21.

As a further example, a virus written may more radically change the instructions for a file open as follows:

assembly language	machine code
MOV AX, 3E00	B8 00 3E
SUB AX, 0100	2D 00 01
MOV DX, 1234	BA 34 12
ADD AL, 02	04 02
INT 21	CD 21

Thus, the static heuristic detection program must look for a large number of different ways each virus-like operation may be implemented in order to reliably detect virus-like behavior. A data-base covering large number of possible permutations of these operations may become unmanageable. This problem would be particularly acute if a virus writer wrote a

"virus generator" program which generated thousands of viruses at a time, permuting the order of its sections of code, but not changing its effective behavior. Such a multitude of viruses would be very difficult to deal with for static heuristic detection programs.

ii. Dynamic Heuristic Detection

In dynamic heuristic virus detection, the antivirus program emulates the target program in a virtual environment and observes the emulated instructions for virus-like operations. As the target program is emulated, its virus-like operations are identified and catalogued. From the catalog of virus-like operations, the dynamic heuristic antivirus program can determine if the target program looks like a virus. Naturally, if the virus has an encrypted viral body, this emulation-based dynamic method can allow the virus to decrypt before observing its virus-like operations (opening files, finding files, etc.).

Dynamic heuristic virus detection can detect many different permutations of a given operation more easily than the static heuristic method. For example, consider the dynamic heuristic detection of a file open operation. Any time an interrupt is called during the emulation, the dynamic heuristic antivirus program checks the values in the registers. These values specify the task that the target program wants the operating system to perform on its behalf. As discussed above regarding static heuristics, a virus infecting the target program may choose to put certain values in the registers in a great variety of ways. However, when the interrupt is finally called, the registers must contain the certain values that correspond to the desired operation. A dynamic heuristic antivirus program is only concerned with the values of the registers at the time of the interrupt call.

While the dynamic heuristic technique is superior in detecting virus-like operations, there are at least three problems to overcome in its implementation. The following is a discussion of these three problems.

First, extensive emulation may be required before the virus-like operations occur. For example, a virus may idle-loop 50,000 times before a file open operation. In that case, a very large number of instructions would have to be emulated before the file open operation is reached. This would greatly slow down the antivirus program.

Second, some viruses activate only when certain arbitrary conditions are met. For example, consider the following pseudo-code of a virus:

1. Find the first file in the current directory that has a ".com" extension (*.com).
2. If a file was found, go to Step 4.
3. Return control to the host program.
4. If the file is less than 1000 bytes long, go to Step 3.
5. If the file name does not end in "EL", go to Step 3.
6. Open the file.
7. Read the first 3 bytes.
8. Seek to the end of the file.
9. Write virus bytes to the file.
10. etc.

If a dynamic heuristic antivirus program were to emulate a host program infected with such a virus, it would encounter first in the virus Step 1 which instructs to find the first *.com file in the current directory. Here, the antivirus program can simulate the DOS call and indicate to the virus that a mock *.com program was found.

Subsequently, in Step 4, the emulator is instructed to return control from the virus to the host program if the mock *.com program did not have a file size of at least 1000 bytes. How is the antivirus program going to anticipate such an arbitrary condition?

Perhaps the antivirus program will be lucky and the mock *.com program had a file size of at least 1000 bytes. Subsequently, in Step 5, the emulator is instructed to return control from the virus to the host program if the file name does not end in "EL". Once again, if this criterion is not met (e.g., the file name is "FOO.COM," ending in "OO"), the virus will immediately terminate and return control to the host program.

Thus, a virus may be designed to be arbitrarily "picky" in its infection process and if any one criterion (such as the date being the 5th of the month) is not met, the virus will fail to execute its infectious behavior. Consequently, a dynamic heuristic antivirus program will not observe the infectious behavior and will not detect the virus.

Third, while a "direct action" virus (such as the examples discussed above) infects other programs as soon as an infected host program is launched, a "memory resident" virus installs itself as a resident interrupt handler and remains dormant until the appropriate interrupt is called. After installing itself as a resident interrupt handler, the memory resident virus returns control to the host program.

A dynamic heuristic antivirus program begins emulation at the main entry-point of a target program. However, the infectious viral code (the part of the virus that infects other programs) of a memory resident virus is not reached via the main entry-point of its host program. Instead, the infectious viral code is executed only when the interrupt into which the virus is hooked is called, and such a call to the operating system may be made by a different program other than the infected host program.

So, even if the dynamic heuristic antivirus program emulates the infected host program for a very long time, the infectious viral code may not be reached, and thus the suspicious viral operations may go undetected.

II. SUMMARY OF THE INVENTION

The above described problems are overcome by the present invention. The present invention relates to a dynamic heuristic method for detecting computer viruses comprising three phases: a decryption phase, an exploration phase, and an evaluation phase. A purpose of the decryption phase is to emulate a sufficient number of instructions to allow an encrypted virus to decrypt its viral body. A purpose of the exploration phase is to emulate at least once all substantial sections of code within a region deemed likely to contain any virus present in the target program. A purpose of the evaluation phase is to analyze any suspicious behavior observed during the decryption and exploration phases to determine whether the target appears to be infected.

III. BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating a structure of a computer platform which includes a preferred embodiment of the present invention.

Fig. 2A is a high-level flow diagram of a first and conventional dynamic heuristic method for detecting computer viruses.

Fig. 2B is a high-level flow diagram of a second and preferred dynamic heuristic method for detecting computer viruses according to the present invention.

Fig. 3 is a flow diagram of the decryption phase of the second method according to a preferred embodiment of the invention.

Fig. 4A is a flow diagram of the ninth (analyze/record) procedure of the decryption phase according to a preferred embodiment of this invention.

Fig. 4B is a flow diagram of the tenth (set virus proper) procedure of the decryption phase according to a preferred embodiment of the present invention.

Fig. 5 is a flow diagram of the exploration phase of the second method according to a preferred embodiment of the present invention.

Fig. 6 is a flow diagram of the first (set virtual CS:IP) procedure of the exploration phase according to a preferred embodiment of the present invention.

Fig. 7A is a flow diagram of the fourth (fetch/emulate) procedure of the exploration phase according to a preferred embodiment of the present invention.

Fig. 7B is a flow diagram of the fourth (process instruction) step of the fourth procedure of the exploration phase according to a preferred embodiment of the present invention.

Fig. 8 is a flow diagram of the evaluation phase of the second method according to a preferred embodiment of the present invention.

IV. DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 1 is a block diagram illustrating a structure of a computer platform which includes a preferred embodiment of the present invention. The computer platform 100 includes hardware devices 102 and a memory module 104 for storing programs and data. Communications between the memory module 104 and the hardware devices 102 occurs through a communication mechanism 106 (e.g., a bus).

Preferably, the hardware devices 102 include, but are not limited to: a central processing unit (CPU) 108 (e.g., an Intel "x86" or a Motorola PowerPC processor) communicating with: a storage device 110 (e.g., a hard disk drive) and an input/output (I/O) interface 112 (e.g., a small

computer system interface (SCSI) or a Personal Computer Memory Card International Association (PCMCIA) interface). Communicating with the I/O interface 112 are: a storage drive 114 (e.g., a floppy disk drive), which can read programs and data from a storage medium 116 (e.g., a floppy disk), and a network interface 118 (e.g., an ethernet card or a modem), which communicates with a remote system 120 via a network 122 (e.g., a local area network, a wide area network, an intranet, or the global public Internet). A target software program (or a target data file) suspected of having a computer virus may be stored either on the storage medium 116 (A), the storage device 110 (B), or the remote system 120 (C).

Preferably, the programs and data stored in the memory module 104 include, but are not limited to: an operating system (OS) 150, such as DOS, Windows 3.1, or Windows 95 from Micorsoft (or even Windows NT from Microsoft, or OS/2 from IBM, or Macintosh OS from Apple Computer, or a flavor of UNIX), an antivirus main module 151, a decryption module 152, an exploration module 154, an evaluation module 156, a CPU emulator 158, and virtual memory 160.

A description of the operation of the decryption module 152, the exploration module 154, and the evaluation module 156 begins below in the discussion relating to Fig. 2B.

The CPU emulator 158 executes files in a fully contained virtual environment. The virtual environment includes the virtual memory 160 which is effectively independent of the memory of the host computer system. In this way, the CPU emulator 158 is effectively isolated from the actual hardware devices 102 so that no harm can be done by a virus while a file is being simulated.

Fig. 2A is a high-level flow diagram of a first and conventional dynamic heuristic method 200 for detecting computer viruses. In the first method 200, a single phase 202 emulates the target program and monitors the emulation for virus-like operations. Such a prior art method is described in U.S. Patent No. 5,398,196 which issued on March 14, 1995 to David A. Chambers.

Fig. 2B is a high-level flow diagram of a second and preferred dynamic heuristic method 250 for detecting computer viruses according to the present invention. The second method 250 begins with a decryption phase 252 which involves emulation. A purpose of the decryption phase 252 is to emulate a sufficient number of instructions to allow an encrypted virus to decrypt its viral body. The decryption phase 252 is described below in relation to Fig. 3. When

the decryption phase 252 is completed, the second method 250 continues with an exploration phase 254 which also involves emulation. A purpose of the exploration phase 254 is to emulate at least once all sections of code within a region likely to contain any virus present, said region being referred to as the virus proper region. In contrast, the emulation phase 202 in Fig. 1 is not so comprehensive as to emulate all sections of code in such a region. The exploration phase 254 is described below in relation to Fig. 5. Following the exploration phase 254, the second method 250 continues with an evaluation phase 256. A purpose of the evaluation phase 256 is to analyze any suspicious operations observed during the decryption and exploration phases 252 and 254 to determine whether or not the target program appears to be infected by a computer virus. The evaluation phase 256 is described below in relation to Fig. 8.

Fig. 3 is a flow diagram of the decryption phase 252 according to a preferred embodiment of the invention. When the antivirus main module 151 is set to scan a target program (A, B, or C in Fig. 1) to determine heuristically whether or not the target program contains virus-like code, the main module 151 begins the decryption phase 252 which includes the following ten procedures 302, 304, 306, 308, 310, 312, 314, 316, 318, and 320..

In a first procedure 302, the decryption module 152 sets the virtual code segment:instruction pointer (CS:IP) of the CPU emulator 158 to the entry-point address of the target program. For example, if the target program is a *.com program, then the virtual IP would be reset to point to the word at 100h in the code segment of the *.com program which is the location of the first executable address of the program. As another example, if the target program is an *.exe program, then the virtual CS would be set to the value of the word at 16h and the virtual IP would be set to the value of the word at 18h in the EXE header as that virtual CS:IP should point to the first executable address of the program.

In a second procedure 304, the decryption module 152 determines whether or not the instruction at the virtual CS:IP was previously modified by another emulated instruction (i.e., whether or not the address of the instruction had been previously written over by a prior emulated instruction). If the instruction at the virtual CS:IP was previously modified, this indicates that a decryption routine of a self-decrypting virus appears to be finished decrypting the viral body and is now trying to execute a decrypted instruction from the viral body. Therefore, if the instruction at the virtual CS:IP was previously modified, then the decryption phase 252 ends and the exploration phase 254 begins.

If the instruction at the virtual CS:IP was not previously modified, then the decryption module 152 performs an third procedure 306 of checking whether or not a first threshold number of instructions have been emulated. The first threshold number of emulated instructions is predetermined such that a typical encrypted viruses would have started to decrypt by the time the first threshold number of instructions have been emulated. In a preferred embodiment, the first threshold number of emulated instructions is three hundred eighty-four (384) instructions. If the first threshold has not yet been reached, then the decryption module 152 continues the emulation by going to a sixth procedure 312, described in more detail below, which fetches the instruction at the CS:IP..

On the other hand, if the first threshold number has been reached, then the decryption module 152 determines in a fourth procedure 308 whether a region of a certain minimum size or larger appears to have been decrypted. Instructions appear to have been decrypted if they have been overwritten. In a preferred embodiment, this certain minimum size is 24 bytes. This certain minimum size of the region is only a fraction (one sixteenth in this case) of the first threshold number of emulated instructions. If no such region appears to have been decrypted, then under the assumption that any virus present is unlikely to be an encrypted virus, the decryption phase 252 ends and the exploration phase 254 begins.

On the other hand, if such a region appears to have been decrypted, then emulation in the decryption phase 252 continues to allow further decryption by fetching the instruction at the virtual CS:IP in the sixth procedure 312 unless a second threshold number of emulated instructions has been reached.

Checking for whether the second threshold has been reached is performed in a fifth procedure 310. In a preferred embodiment, the second threshold is set to about thirty thoursand (30,000) instructions. Once the second threshold number of emulated instructions has been reached, then the decryption phase 252 ends and the exploration phase 254 begins. The purpose of the second threshold is to limit the time taken to allow a virus to decrypt. If the second threshold is reached, the exploration phase 254 begins under the assumption that a significant region of the viral body has been decrypted and exploration of that region is warranted.

In the sixth procedure 312, the decryption module 152 causes the CPU emulator 158 to fetch the instruction stored at the memory address indicated by the virtual CS:IP.

In a seventh procedure 314, the decryption module 152 causes the CPU emulator 158 to increment the virtual CS:IP so that it points to the next instruction in the target program (i.e., to the instruction immediately following the one just fetched).

In a eighth procedure 316, the decryption module 152 causes the CPU emulator 158 to emulate the just-fetched instruction. Because the just-fetched instruction is being emulated by the CPU emulator 158 in a virtual environment, rather than being executed by the actual CPU 108 of the computer system 100, the computer system 100 and its files are protected from any harm from the instruction being emulated.

In a ninth procedure 318, the decryption module 152 analyzes the just-emulated instruction and records any "suspicious" operations indicated by the analysis. This analysis/recording procedure 318 is described in further detail below in relation to Fig. 4A.

In a tenth procedure 320, the decryption module 152 sets the virus proper region. This setting procedure 320 is described in further detail below in relation to Fig. 4B. Following the tenth procedure 320, the decryption module 152 loops back to the second procedure 304.

An advantage of the decryption phase 252 described above is that the substantial majority of self-decrypting viruses are allowed to decrypt much or all of their viral bodies, and this is done without spending too much processing time per file. The decrypted portions of any encrytped viral bodies can then be explored in the subsequent exploration phase 254.

Fig. 4A is a flow diagram which describes a process for the ninth procedure (the analysis/recording procedure) 318 of the decryption phase 252 according to a preferred embodiment of this invention. This analysis/recording procedure 318 begins after the CPU emulator 158 emulates the just-fetched instruction in the eighth procedure 316 of the decryption phase 252.

The ninth procedure 318 begins with a first step 402 in which the decryption module 152 determines whether or not the just-emulated instruction modifies the content stored at a (virtual) memory address within the (virtual) address space of the target program. Such a modification is indicative of decryption taking place.

If such a modification is performed by the just-emulated instruction, then the decryption module 152 performs a second step 404 of recording that modification for future reference (such a reference occurring in the second procedure 304 of the decryption phase 252) before

performing a third step 406. Otherwise, if such a modification is not performed, then the decryption module 152 goes straight from the first step 402 to the third step 406.

In the third step 406, the decryption module 152 determines whether or not any suspicious operation from a predetermined list of suspicious operations is performed by the just-emulated instruction. The predetermined list of suspicious operations include operations characteristically performed by viruses but not typically performed by "legitimate" uninfected programs.

In this embodiment, the list of suspicious operations includes, but is not limited to, the following suspicious operations:

1. File opens (read-only, read-write, etc.);
2. File reads (sub-categories: 3-4 bytes, between 5 and 8 bytes, 12-14 bytes, 128 or more bytes, etc. These values are chosen because of their particular significance in certain computer viruses.);
3. File writes (sub-categories: 3-4 bytes, between 5 and 8 bytes, 12-14 bytes, 1Ch to 1Eh bytes, 128 or more bytes, etc.);
4. File seeks (to the top of the file, to the end of the file, etc.);
5. File get/set attribute;
6. File get/set date/time;
7. File find-first/find-next;
8. File delete;
9. File create;
10. File rename;
11. Change directory, get current directory, set current directory;
12. File get/set DTA;

13. Interrupt revectoring (interrupt 13 for disk services, interrupt 21 for DOS services, interrupt 24 for error handling, etc.)

14. Modification of memory allocation (in particular, reads/writes to program segment prefix);

15. Program execution;

16. Program residency self-check calls (typically used by viruses);

17. Read/write to master boot record (MBR) using BIOS disk services (typically used in multipartite and dropper viruses);

18. In-memory modification of CS:100 in *.com programs (typically done in viruses infecting *.com programs).

If it is determined that suspicious operation is being performed by the just-emulated instruction, the suspicious operation is recorded by the decryption module 152 in a fourth step.

In the fifth step 410, the decryption module 152 determines whether or not the just-emulated instruction comprises a branch point. In this specification, a branch point is an instruction which conditionally or unconditionally transfers control to an instruction not sequentially following the branch point instruction. Such a branch point indicates that while one branch of instructions is taken, an alternate branch of instructions is not being taken.

As described above in the background section with respect to the second problem of prior dynamic heuristic antivirus methods, such untaken branches may contain infectious viral code which activates only upon satisfaction of certain arbitrary conditions. For example, consider the following section of assembly language instructions from a hypothetical virus:

```
180   MOV  AX, [TargetHostProgramSize]
185   CMP  AX,1000
188   JLE  210
...
210   HLT
```

The above section of instructions checks to see if the potential host program that it has located (the target host program) is greater than a minimum size (1000h bytes). If the potential host program is bigger, then it will be considered for infection (i.e. the jump is not taken). Otherwise, the potential host program is not considered for infection and the virus merely halts (i.e. the jump is taken). Branch points such as this one are critical because if the virus does not have its criteria for a host program satisfied, the infectious code may remain in an untaken branch which is not explored by a prior art dynamic heuristic antivirus program.

In light of a need to eventually analyze such untaken branches, if the just-emulated instruction comprises a branch point, then the decryption module 152 in a sixth step 412 stores onto a queue the destination address of the untaken branch (i.e. the address of the first instruction of the untaken branch) and the current state of the CPU emulator 158, before it ends the ninth procedure 318 and begins the tenth procedure 320. The state of the CPU emulator 158 does not refer to the entire virtual CPU/memory state, which is prohibitively large, but rather to a small subset of that state. This subset is sufficient to detect the majority of viruses of current concern. In this embodiment, the state of the CPU emulator 158 includes, but is not limited to, the virtual CS:IP, certain commonly used registers such as the AX, BX, CX, DX registers, and several words at the top of the SS:SP stack.

If only the destination address of the untaken branch, but not the CPU emulator state, were stored, then certain operations would go undetected when that branch of instructions was later explored during the exploration phase 254. For example, consider the following hypothetical section of code from a virus:

```
100   MOV  AX,4200
103   MOV  CX,0000
106   MOV  DX,0000
109   INT  21        INT 21, AX=4200:  SEEK TOP OF FILE
10B   JC   120
10D   MOV  AL,02     INT 21, AX=4202:  SEEK TOP OF FILE
10E   INT  21
```

```
...

120    MOV  AX,FFFF

123    RET
```

If the above code were to execute in the emulator and seek instruction at offset 109 were to yield a CARRY result (indicating a seek error), then the jump at offset 10B would be taken to the instruction at offset 120. In that case, the untaken destination address saved upon the queue would be 10D and the state of the CPU emulator 158 saved upon the queue would be CS:10D, AX = 4200, BX = ????, CX=DX=0, the top 8 words of the stack, etc. Later when that untaken branch is explored, the instruction at offset 10D would change AX to 4202, and the instruction at offset 10E would be a seek instruction which is a suspicious operation. However, if the state of the CPU emulator 158 were not saved upon the queue, then the AX register might contain a value such as the FFFF due to an instruction such as the one at offset 120. In that case, when the untaken branch is explored, the instruction at offset 10D would change AX to FF02, and the instruction at offset 10E would not accurately exhibit the operation of the viral code.

Otherwise, if the just-emulated instruction is not a branch point, then the decryption module 152 ends the ninth procedure 318 and begins the tenth procedure 320. The tenth procedure 320 is described in further detail below in relation to Fig. 4B.

Fig. 4B is a flow diagram of the tenth procedure 320 of the decryption phase 252 according to a preferred embodiment of the present invention. This tenth procedure 320 determines a relatively likely location of the virus in the target program.

In a first step 420, the decryption module 152 determines whether or not the virus proper region has already been set (i.e., during a prior performance of either a third step 424 or a sixth step 430 as discussed further below). If the region has already been set, then the tenth procedure 320 ends and the decryption module 152 loops back to the second procedure 304 of the decryption phase 252.

Otherwise, in a second step 422 the decryption module 152 determines whether the target program is an EXE file. If it is an EXE file, then the decryption module 152 in a third step 424 sets the virus proper region to be the region from the start of the beginning code segment (CS) to the end of file for the target program. Following the third step 424, the tenth

procedure 320 ends and the decryption module 152 loops back to the second procedure 304 of the decryption phase 252.

Otherwise, if the target program is not an EXE file, it is generally a COM file. For COM files, the decryption module 152 in a fourth step 426 determines whether the number of instructions of the target program which have been emulated is greater than n, where n is a predetermined number. In a preferred embodiment, n=10.

If the number of instructions emulated is greater than n, then the decryption module 152 in a fifth step 428 determines whether the current virtual CS:IP is within the beginning memory segment of the target program. If the current virtual CS:IP is outside the beginning segment, then the tenth procedure 320 ends and the decryption module 152 loops back to the second procedure 304 of the decryption phase 252.

Otherwise, if the current virtual CS:IP is within the beginning segment, then in a sixth step 430 the virus proper region is set based upon the current virtual CS:IP. In a preferred embodiment, the region is set to include the 4k (4096) bytes centered about the current virtual CS:IP. Subsequently, the tenth procedure 320 ends and the decryption module 152 loops back to the second procedure 304 of the decryption phase 252.

Fig. 5 is a flow diagram of the exploration phase 254 of the second method according to a preferred embodiment of the present invention. The exploration phase 254 is initiated by the main module 151 after the decryption phase 252, unless the decryption module 152 had returned that the target program appeared to be virus-free. Emulation during the exploration phase 254 is limited to those instructions in the virus proper region. The exploration phase 254 includes the following eight procedures 502, 504, 506, 508, 510, 512, 514, and 516.

In a first procedure 502, the exploration module 154 sets the virtual CS:IP of the CPU emulator 158. The first procedure 502 is described in further detail below in relation to Fig. 6.

In a second procedure 504, the exploration module 154 determines whether the instruction at the virtual CS:IP was already fetched during this exploration phase 504. If the instruction at the virtual CS:IP was not already fetched in this exploration phase 504, then the exploration module 154 determines in a third procedure 506 whether the virtual CS:IP is outside the virus proper region.

If the virtual CS:IP is determined in the third procedure not to be outside the virus proper region, then the exploration module 154 in a fourth procedure 508 causes the CPU emulator 158 to fetch the instruction at the virtual CS:IP and emulate the fetched instruction. The fourth procedure 508 is described in further detail below in relation to Figs. 7A and 7B. After the fourth procedure 508, either the exploration module 154 loops back to the second procedure 504 (if a threshold number of instructions have not yet been emulated), or the exploration phase 254 terminates and the evaluation phase 256 begins (if the threshold number of instruction have been emulated).

On the other hand, if the virtual CS:IP is determined in the third procedure 506 to be outside of the virus proper region, then the exploration module 154 determines in a fifth procedure 510 whether an untaken branch queue is empty. This fifth procedure 510 is also reached if the exploration module determines in the second procedure 504 that the instruction at the virtual CS:IP was already fetched during this exploration phase 254.

If the untaken branch queue is determined in the fifth procedure 510 not to be empty, then the exploration module 154 in a sixth procedure 512 "dequeues" the next untaken branch from the untaken branch queue. Dequeueing the next untaken branch includes setting the virtual CS:IP to the stored destination address of the branch and setting the state of the CPU emulator 158 to the corresponding stored state (both being stored previously during the eighth substep 754 of the fourth step 708 of the fourth procedure 508, see Fig. 7B and the corresponding description below). After the sixth procedure 512, the exploration module 154 loops back to the second procedure 504.

On the other hand, if the untaken branch queue is determined in the fifth procedure 510 to be empty, then the exploration module 154 in a seventh procedure 514 determines whether any unemulated section of code remains within the virus proper region. If such an unemulated section of code remains within the virus proper region, then the exploration module 154 in an eighth procedure 516 resets major virtual CPU registers (to zero) and set the virtual CS:IP to point to the start of the unemulated section of code. After the eighth procedure 516, the exploration module 154 loops back and performs the second procedure 504. On the other hand, if no unemulated section of code remains within the virus proper region, then the exploration phase 254 terminates and the evaluation phase 256 begins. In a preferred embodiment of the present invention, an unemulated section of code will be emulated in the exploration phase 254

even if the section is only one byte (i.e. one instruction) in size. However, this minimum size may be made larger.

Fig. 6 is a flow diagram of the first procedure 502 of the exploration phase 254 according to a preferred embodiment of the present invention. If no decryption was observed during the decryption phase 252, then the first procedure 502 resets the virtual CS:IP of the CPU emulator 158 to begin the exploration phase 254.

The first procedure 502 begins with a first step 602 in which the exploration module 154 determines whether the decryption was observed during the decryption phase 252. In particular, if the decryption module 152 determined in the sixth procedure 312 of the decryption phase 252 that the just-emulated instruction was previously modified, then the decryption is considered to have been observed.

If decryption was observed in the first step 602, then the virtual CS:IP remains unchanged so that emulation of the target program will continue from the instruction following the last instruction emulated during the decryption phase 252. In this case, after the first step 602, the first procedure 502 of the exploration phase 254 ends and the second procedure 254 begins.

On the other hand, if decryption was not observed in the first step 602, then the exploration module 154 in a second step 604 resets the virtual CS:IP to the entry-point address of the target program.

After the virtual CS:IP is reset in the second step 604, the exploration module 154 in a third step 606 re-establishes the virus proper region. This re-establishing of the virus proper region may be implemented by incorporating a procedure similar to the procedure 320 shown in Fig. 4B into a loop including procedures similar to the procedures 312, 314, and 316. After the virus proper region is re-established in the third step 606, the first procedure 502 of the exploration phase 254 ends and the second procedure 504 begins.

Fig. 7A is a flow diagram which describes a process for the fourth procedure 508 of the exploration phase 254 according to a preferred embodiment of the present invention. The fourth procedure 508 fetches and emulates the instruction at the virtual CS:IP if the virtual CS:IP was determined 506 to be within the virus proper region.

The fourth procedure 508 begins with a first step 702 in which the exploration module 154 updates a bitmap record of instructions fetched from the virus proper region. Such a bitmap record comprises one bit per byte of the virus proper region. If the bit is zero, then the corresponding instruction byte has not yet been fetched. On the other hand, if the bit is one, then the corresponding instruction byte has already been fetched (and emulated). Thus, the bit that corresponds to the instruction byte pointed to by the virtual CS:IP is set to one in this first step 702. Note that this first step 702 need not occur at the very beginning of the fourth procedure 508 and in another embodiment may occur at another point in the fourth procedure 508.

Following the first step 702, the exploration module 154 in a second step 704 causes the CPU emulator 158 to fetch the instruction stored at the virtual memory address pointed to by the virtual CS:IP.

Following the second step 704, the exploration module 154 in a third step 706 causes the CPU emulator 158 to increment the virtual CS:IP so that it points to the next instruction in the target program (i.e., to the instruction immediately following the one just fetched).

Following the third step 706, the exploration module 154 in a fourth step 708 processes the instruction before emulation occurs. This processing 708 exerts control over certain instructions, such as interrupts and memory writes within the virus proper region, and records the presence of these certain instructions for later use if necessary. This fourth step 708 is described in more detail below in relation to Fig. 7B.

Following the fourth step 708, the exploration module 154 in a fifth step 710 determines whether the just-fetched instruction comprises a suspicious operation not already recorded in the fourth step 708. If such suspicious operation was observed, then the exploration module 154 in a sixth step 712 records the suspicious operation. A list of such suspicious operations is predetermined and was generated from an analysis to find viral behaviors or combinations thereof that are very unlikely to be used in non-infected programs. A list of suspicious operations was given above in relation to Fig. 4A. After the sixth step 712, or directly after the fifth step 710 if no suspicious operation was observed, the exploration module 154 in a seventh step 714 causes the CPU emulator 158 to emulate the just-fetched instruction.

After the seventh step 714, the exploration module 154 in an eighth step 716 determines whether a threshold number of instructions have been emulated after the entry-point of the target

program. In a preferred embodiment, the threshold number is set to the number of bytes in the virus proper region. Since for an Intel X86 microprocessor an instruction is at minimum one byte in length, this threshold number is equal to the maximum number of instructions in the virus proper region. This threshold number ensures that the emulation does not continue much beyond the number of instructions in the virus proper region. If the exploration module 154 determines that the number of instructions emulated since the entry-point has reached the threshold number, then it terminates the exploration phase 254, and the main module 151 causes the evaluation module 156 to begin the evaluation phase 256. On the other hand, if the number of instructions emulated since the entry-point has not yet reached the threshold number, then the exploration module 154 loops back to the second procedure 504 where it determines whether the next instruction was already fetched during the exploration phase 254.

Fig. 7B is a flow diagram which describes a process for the fourth step 708 of the fourth procedure 508 of the exploration phase 254 according to a preferred embodiment of the present invention. This fourth step 708 processes the instruction prior to emulation.

The fourth step 708 begins with a first substep 740 in which the exploration module 154 determines whether or not the just-fetched instruction is a call to an interrupt vector. If it is a call to an interrupt, then the exploration module 154 in a second substep 742 treats the interrupt as a NOP (no operation) instruction but "fakes" completion of the interrupt by changing appropriate registers of the CPU emulator 158, popping appropriate values off of the stack, and so on. In the second substep 742, the exploration module 154 also checks the appropriate register to determine the nature of the actual instruction called via the interrupt, and the actual instruction is recorded if it is deemed to be suspicious.

In a third substep 744, the exploration module 154 determines whether the just-fetched instruction modifies (i.e. writes to) virtual memory within the virus proper region. Such a memory write, for example, may indicates that the virus appears to be relocating its code, or that the virus is storing data within the virus proper region. In this embodiment, such writes are prevented. If the just-fetched instruction does write to memory within the virus proper region, then the exploration module 154 in a fourth substep 746 redirects the memory write to a virtual memory address in a different virtual segment, defining that virtual segment as segment S, and records such a write to segment S. . As discussed below, subsequent data reads from that memory address are redirected to segment S.

The exploration module 154 implements such redirection of subsequent memory reads via fifth and sixth substeps 748 and 750. In the fifth substep 748, the exploration module 154 determines whether the just-fetched instruction initiates a read of data from a location in the virus proper region of memory that the target program previously attempted to modify. If it is determined in the fifth substep 748 that the just-fetched instruction initiates such a memory read of data, then the exploration module 154 in the sixth substep 750 redirects that memory read to segment S. Note, in this embodiment, that fetches from the instruction stream are not to be redirected.

In a seventh substep 752, the exploration module 154 determines whether the just-fetched instruction comprises a branch point. A branch point is defined here as an instruction in a computer program at which a jump occurs or may occur.

If it is determined in the seventh substep 752 that the just-fetched instruction does not comprise a branch point, then the fourth step 708 in Fig. 7A ends and the exploration module 154 goes on to perform the fifth step 710. On the other hand, if it is determined in the seventh substep 752 that the just-fetched instruction is a branch point, then the exploration module 154 in an eight substep 754 stores the current state of the CPU emulator 158 and the destination (also called target) address of the (untaken) branch onto a queue of untaken branches. Subsequently, the fourth step 708 in Fig. 7A ends and the exploration module 154 goes on to perform the fifth step 710.

Fig. 8 is a flow diagram of the evaluation phase 256 which occurs after the exploration phase 254 terminates according to a preferred embodiment of the present invention. The evaluation phase 256 analyzes the suspicious operations observed and recorded during the decryption and exploration phases 252 and 254 to determine whether or not the target program appears to be infected by a computer virus. While this embodiment uses a particular approach to evaluate whether or not the target program is likely to be infected, many other "expert system" approaches may be used in the evaluation phase 256.

In this embodiment, the evaluation phase 256 begins with a first procedure 802 in which the evaluation module 156 determines whether a so-called innocent operation was observed during the decryption or exploration phases 252 or 254. In a preferred embodiment, innocent operations include, but are not limited to:

1. printing a character to a printer or displaying a character to a monitor or other conspicuous output device;

2. a program termination instruction in the virus proper region that is directly reachable from the entry point;

3. re-vectoring of the divide-by-zero error vector (viruses don't generally need to do division); and

4. calling from within the virus proper region to a function located far (more than four kilobytes in a preferred embodiment) away from the current position in the file (since this would indicate a routine larger than most viruses).

If any of the latter three innocent operations was observed, then the evaluation module 156 determines in a second procedure 804 whether or not specific highly suspicious operations or features were observed. In a preferred embodiment, highly suspicious operations or features include, but are not limited to, the following:

1. comparing a word in any register or memory location with "MZ" (such a comparison may be used to determine the presence of an EXE header since "MZ" is an identifying marker or tag or label of such a header);

2. a file write of the same or a suspiciously similar size in bytes as the distance in bytes from the entry-point of the target program to the end-of-file of the target program (a file write of such a size may be used in the process of appending the virus to the next host program);

3. a seek to the end-of-file to determine file size followed by a file write of a jump instruction whose jump distance is suspiciously similar to the file size (such a jump instruction may be used by an appending virus to bypass the clean original body of the next host program);

4. a file write of a number of bytes similar to the number of bytes in an EXE header where the bytes written include "MZ" (such a file write may be used to write a new header to the next host program);

5. having an EXE header in the body of an EXE file that is suspiciously similar to the EXE header of the file (such a header may be used to substitute for the original header of the next host program);

6. a command to find executable (e.g., *.com or *.exe) files in a directory (such a command may be used to find the next host program);

7. examination and modification of the (DOS) memory allocation scheme (not uncommon in viruses, but rare in clean programs);

8. repeat move string operations that move approximately the same number of bytes as the distance in bytes from the entry-point to the end-of-file of the target program (string moves of such a total size may be used in the process of appending the virus to the next host program); and

9. reading/writing to the program segment prefix (indicating a modification of the memory allocation scheme in a backhand way).

If no such highly suspicious operation or combination of operations was observed or the innocent operation was printing or displaying a character, then the evaluation module 156 in a third procedure 806 returns a result to the anti-virus main module 151 that indicates that the target program appears to be virus-free.

Otherwise, if any highly suspicious operation was observed 804 or if no innocent operation was observed 802, then the evaluation module 156 in a fourth procedure 808 calculates the extent to which various suspicious operations were observed during the decryption 252 and/or exploration 254 phases. In a preferred embodiment, the fourth procedure 808 involves calculating for each of several suspicious "behaviors" a value based on the number of occurrences of select suspicious operations indicative of the suspicious behavior, where the number of occurrences is weighted depending on the operation's indicativeness regarding the behavior and where the calculated value is normalized so as not to exceed 100%.

In a preferred embodiment, suspicious behaviors include, but are not limited to, the following:

1. encryption;

2. appending bytes onto a COM file;

3. appending bytes onto an EXE file;

4. opening/closing of a file in standard and non-standard ways;

5. hooking to a memory resident code or routine;

6. critical error handling (re-vectoring a critical error vector);

7. string moves or file writes that are suspiciously similar in size to the target file size;

8. finding a file or finding a first file in a directory; and

9. changing file date/time stamps or other file attributes.

Following the fourth procedure 808, the evaluation module 156 in a fifth procedure 810 determines for each of several types of viruses whether a measure of the suspicious behaviors reaches or surpasses a threshold indicative of infection. In particular, each suspicious behavior is assigned a weight that multiplies its normalized value to arrive at a contribution to the measure. If the measure surpasses a predetermined threshold and every suspicious behavior designated as mandatory for that type of virus is present, then infection by that type of virus is indicated.

In a preferred embodiment, types of viruses include, but are not limited to, the following:

1. memory resident viruses appending to COM files;

2. direct action viruses appending to COM files;

3. memory resident viruses prepending to COM files;

4. direct action viruses prepending to COM files;

5. memory resident viruses appending to EXE files;

6. direct action viruses appending to EXE files;

7. memory resident viruses in general; and

8. direct action viruses in general.

If one (or possibly more) of the thresholds are reached, then in a sixth procedure 812 the evaluation module 156 determines whether any highly suspicious combination of three or more suspicious operations were detected. In a preferred embodiment, these highly suspicious combinations are determined by analyzing various combinations with respect to the universe of known computer viruses and selecting as highly suspicious those combinations which appear in several (e.g., five) percent or more of the known viruses. Other ways of selecting the highly suspicious combinations are possible. When a highly suspicious combination generates a false positive by incorrectly identifying a clean program as infected, then that highly suspicious combination may be removed from the list of highly suspicious combinations. Such a list should be quite robust since more than one highly suspicious combination may be expected to detect a typical virus.

If a highly suspicious combination is detected 814, then the evaluation module 156 performs the sixth procedure 812 of returning to the anti-virus main module 151 the type of infection apparently present in the file. In particular, a label corresponding to the combination(s) detected is returned.

Otherwise, if no highly suspicious combination is detected, then the evaluation module 156 performs the third procedure 806 of returning to the anti-virus main module 151 the result that the file appears to be virus-free.

The above description is included to illustrate the operation of the preferred embodiments and is not meant to limit the scope of the invention. From the above description, many variations will be apparent to one skilled in the art that would yet be encompassed by the spirit and scope of the invention. First, groups of blocks in Figs. 4A and 7B may be reordered without changing the functionality of the embodiment. (For example, steps 402 and 404 may be moved to be after steps 406 and 408. Or, steps 740 and 742 in Fig. 7B may be moved to be after steps 748 and 750. And so on.) Second, the determination of when to end the decryption phase 252 may be done in alternate ways different from the way using two thresholds which is shown in Fig. 3. Third, the determination of the virus proper region may be done in many alternate ways different from the specific way which is shown in Fig. 4B. Fourth, the evaluation of whether or not the file appears infected may be done in alternate ways different from the vector approach of the evaluation phase 256 illustrated in Fig. 8. Fifth, different criteria, or combinations of criteria, for suspicious operations and behaviors may be used in the evaluation phase 256 that vary from the criteria given in the specification. These are only some of the

variations that are rightly encompassed by the present invention. The scope of the invention is to be limited only by the following claims. What is claimed is:

CLAIMS

1. A method for detecting undesirable code in a digital file comprising:

determining a region of the digital file in which the undesirable code is likely to be present; and

emulating at least once all substantial sections of code within the region in order to fully explore the region during an exploration phase.

2. The method of claim 1, wherein the region includes an end portion of the digital file.

3. The method of claim 1, wherein the region includes a beginning portion of the digital file.

4. The method of claim 1, further comprising:

pointing to an instruction to be emulated from a section of code;

determining whether during the exploration phase the instruction to be emulated has already been emulated; and

if the instruction to be emulated has already been emulated during the exploration phase, then not emulating the instruction to be emulated.

5. The method of claim 4, wherein the emulating further comprises:

determining whether the instruction to be emulated is outside of the region; and

if the instruction to be emulated is outside the region, then not emulating the instruction to be emulated.

6. The method of claim 5, further comprising:

determining whether any untaken branch destination remains; and

if any untaken branch destination remains, then pointing to a next untaken branch destination as the instruction to be emulated and setting a state of a CPU emulator to a stored state which corresponds to the next untaken branch destination.

7. The method of claim 6, further comprising:

if no untaken branch destinations remain, then determining whether any unemulated section of code remains within the region; and

if any unemulated section of code remains within the region, then pointing to a beginning instruction of an unemulated section of code as the instruction to be emulated and resetting the state of the CPU emulator.

8. The method of claim 5, wherein emulating the instruction to be emulated comprises:

determining whether the instruction to be emulated performs a suspicious operation; and

recording the suspicious operation if the instruction to be emulated performs the suspicious operation.

9. The method of claim 5, wherein emulating the instruction to be emulated comprises:

determining whether the instruction to be emulated comprises an interrupt; and

if the instruction to be emulated comprises an interrupt, then faking completion of the interrupt but not actually performing the interrupt.

10. The method of claim 5, wherein emulating the instruction to be emulated comprises:

determining whether the instruction to be emulated comprises a memory write to a location within the region;

redirecting the memory write to a different segment S if the instruction to be emulated comprises a memory write to a location within the region.

11. The method of claim 10, wherein emulating the instruction to be emulated further comprises:

determining whether the instruction to be emulated comprises initiating a memory read of data previously written by redirection to segment S; and

redirecting the memory read of data to the segment S if the instruction to be emulated comprises the memory read of data previously written by redirection to the segment S.

12. The method of claim 5, wherein emulating the instruction to be emulated comprises:

determining whether the instruction to be emulated comprises a branch point;

storing an untaken destination address of the branch point and a CPU emulator state if the instruction to be emulated comprises a branch point.

13. A method for detecting a computer virus in a target program comprising:

a first emulation of instructions of the target program to allow the computer virus to decrypt a viral body; and

a second emulation of instructions of a region of the target program which is likely to contain the viral body.

14. The method of claim 13, wherein the first emulation begins at an entry-point instruction of the target program.

15. The method of claim 13, wherein the second emulation begins at an instruction immediately following a last instruction emulated during the first emulation.

16. The method of claim 13, wherein during the first emulation the computer virus does not decrypt the viral body because the viral body is not encrypted; and wherein the second emulation begins at an entry-point instruction of the target program.

17. The method of claim 13, wherein whether the viral body has decrypted is determined by identifying a contiguous section of bytes in memory which were overwritten during the first emulation.

18. The method of claim 13, further comprising an evaluation of suspicious operations observed during the first and second emulation.

19. The method of claim 18, wherein the evaluation of suspicious operations results in a determination of whether the target program appears to be infected.

20. A method for detecting undesirable code in a program comprising: a decryption phase, an exploration phase; and an evaluation phase.

21. A method for detecting computer viruses in a target program comprising:

emulation of instructions of the target program;

evaluation of operations observed during the emulation; and

stopping the evaluation if an operation was observed which is rarely found in the computer viruses.

22. The method of claim 21, wherein the operation comprises printing or displaying a character.

23. The method of claim 21, wherein the operation comprises a program termination instruction within a particular region of the program.

24. The method of claim 21, wherein the operation comprises a re-vectoring of an error vector.

25. The method of claim 21, wherein the operation comprises calling from within a particular region of the program a function far away from a present location.

26. The method of claim 21, wherein the stopping is avoided if a highly suspicious operation is observed.

27. The method of claim 26, wherein the highly suspicious operation comprises comparing contents of a register or memory location with an identifier of a file header.

28. The method of claim 26, wherein the highly suspicious operation comprises a file write of similar size in bytes as a distance in bytes from an entry-point of the target program to an end-of-file of the target program.

29. The method of claim 26, wherein the highly suspicious operation comprises a seek to an end-of-file followed by a file write of a jump instruction.

30. The method of claim 29, wherein the seek determines a size in bytes and the jump instruction has a jump distance of a similar size in bytes.

31. The method of claim 26, wherein the highly suspicious operation comprises a file write of a similar size as a size of a header, and the file write includes a tag that is characteristic of the header.

32. The method of claim 26, wherein the highly suspicious operation comprises having a header of an executable file within a body of the target program.

33. The method of claim 26, wherein the highly suspicious behavior comprises finding an executable file in a directory.

34. The method of claim 26, wherein the highly suspicious behavior comprises examination or modification of the memory allocation scheme.

35. The method of claim 26, wherein the highly suspicious behavior comprises one or more move string operations that move approximately a same number of bytes as a distance in bytes from an entry-point of the target program to an end-of-file of the target program.

36. A method for detecting undesirable code, such as a computer virus, trojan horse, or worm, in a computer file comprising:

determining a region of the computer file which is likely to contain the undesirable computer code; and

emulating substantially all code in the region.

37. A method for detecting undesirable code in digital code comprising:

determining a presence of a first feature in the digital code;

determining a presence of a second feature in the digital code;

determining a presence of a third feature in the digital code;

determining a logical value based upon a logical combination of the presence of the first, second, and third features; and

indicating the presence of the undesirable code if the logical value is true.

38. A method for detecting undesirable code in a target program including detecting a presence of an operation comprising comparing contents of a register or memory location with an identifier of a file header.

39. A method for detecting undesirable code in a target program including detecting a presence of an operation comprising a file write of a similar size in bytes as a distance in bytes from an entry-point of the target program to an end-of-file of the target program.

40. A method for detecting undesirable code in a target program including detecting a presence of an operation comprising a seek to an end-of-file followed by a file write of a jump instruction.

41. The method of claim 40, wherein the seek determines a size in bytes and the jump instruction has a jump distance of a similar size in bytes.

42. A method for detecting undesirable code in a target program including detecting a presence of an operation comprising a file write of a similar size as a size of a header, and the file write includes a tag that is characteristic of the header.

43. A method for detecting undesirable code in a target program including detecting a presence of an operation comprising having a header of an executable file within a body of the target program.

44. A method for detecting undesirable code in a target program including detecting a presence of an operation comprising finding an executable file in a directory.

45. A method for detecting undesirable code in a target program including detecting a presence of an operation comprising examination or modification of the memory allocation scheme.

46. A method for detecting undesirable code in a target program including detecting a presence of an operation comprising one or more move string operations that move approximately a same number of bytes as a distance in bytes from an entry-point of the target program to an end-of-file of the target program.

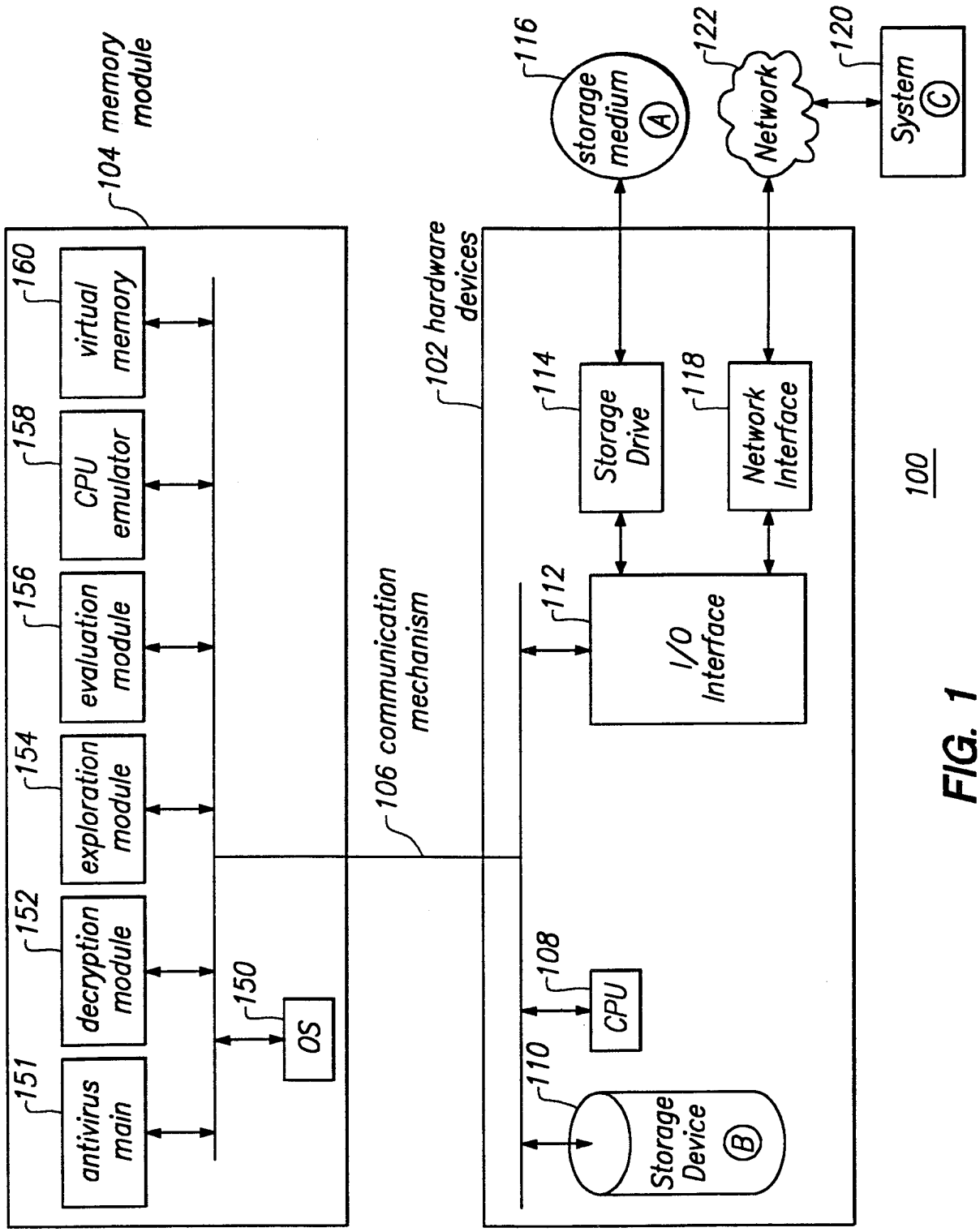

104 memory module
151 antivirus main
152 decryption module
154 exploration module
156 evaluation module
158 CPU emulator
160 virtual memory
150 OS
106 communication mechanism
102 hardware devices
108 CPU
110
Storage Device (B)
112 I/O Interface
114 Storage Drive
118 Network Interface
116 storage medium (A)
122 Network
120 System (C)
100

FIG. 1

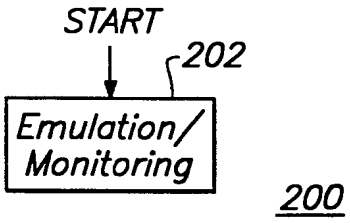


FIG. 2A (Prior Art)

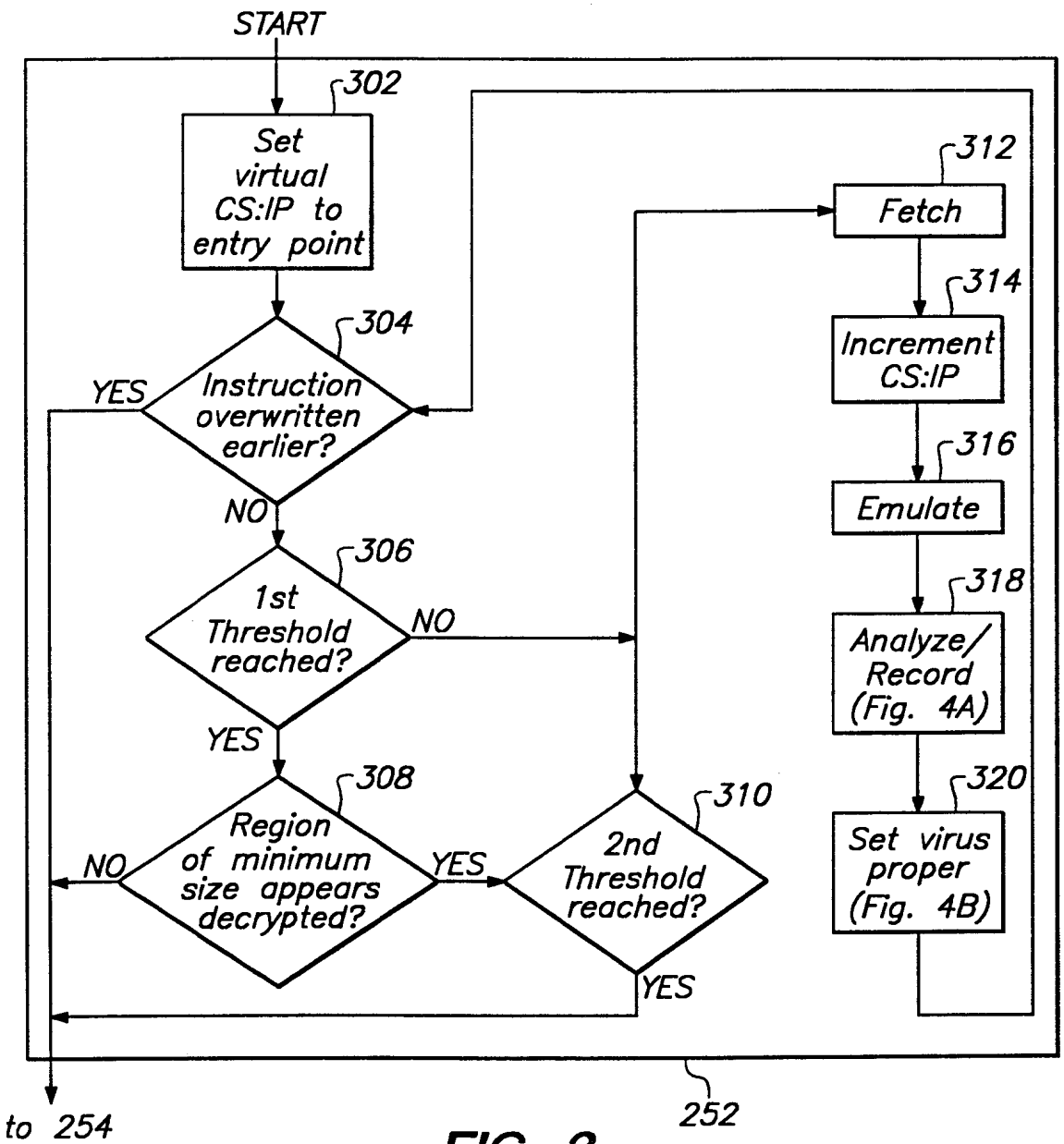


FIG. 3

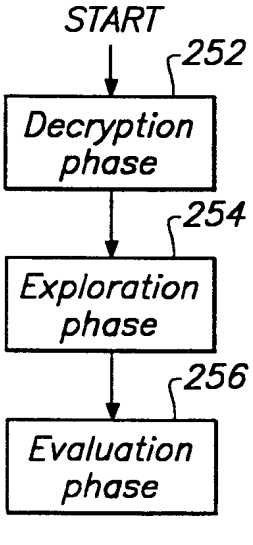

START
252
Decryption phase
254
Exploration phase
256
Evaluation phase
250

FIG. 2B

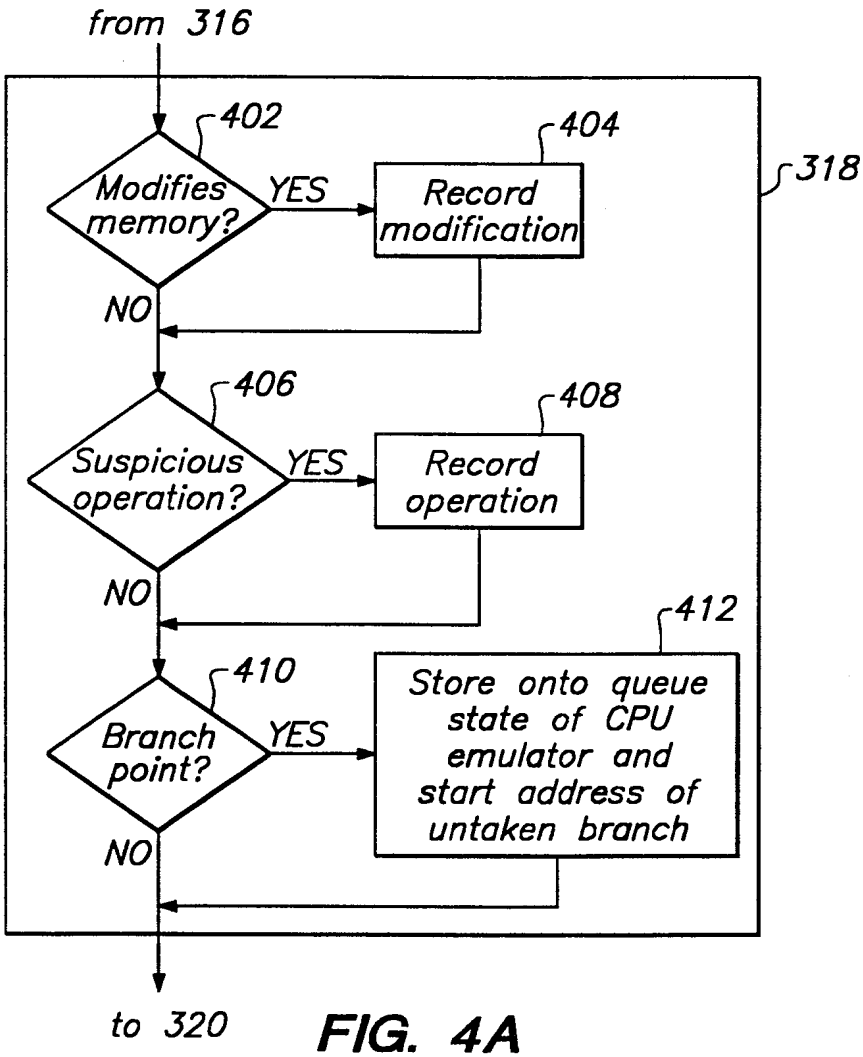

from 316
402
Modifies memory?
YES
404
Record modification
318
NO
406
Suspicious operation?
YES
408
Record operation
NO
412
410
Branch point?
YES
Store onto queue state of CPU emulator and start address of untaken branch
NO
to 320

FIG. 4A

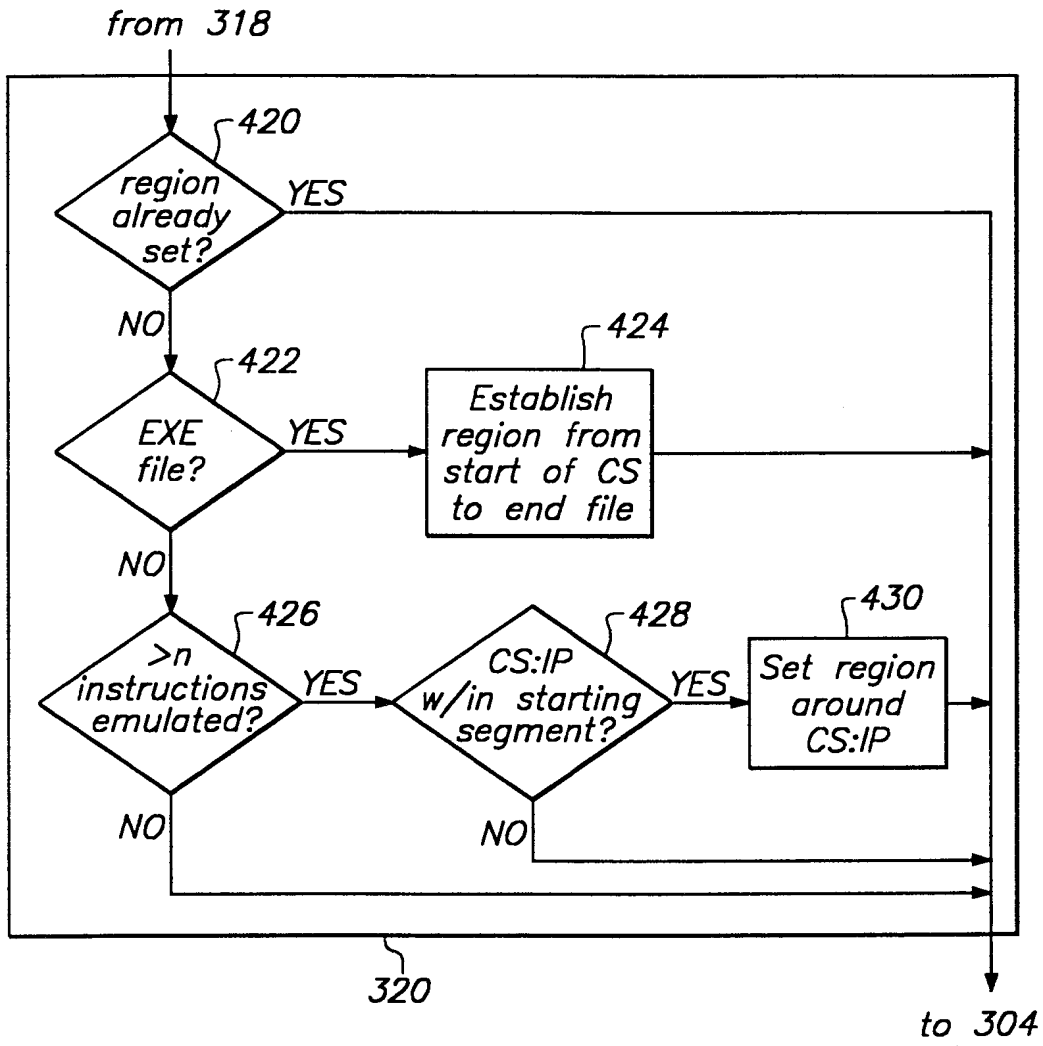
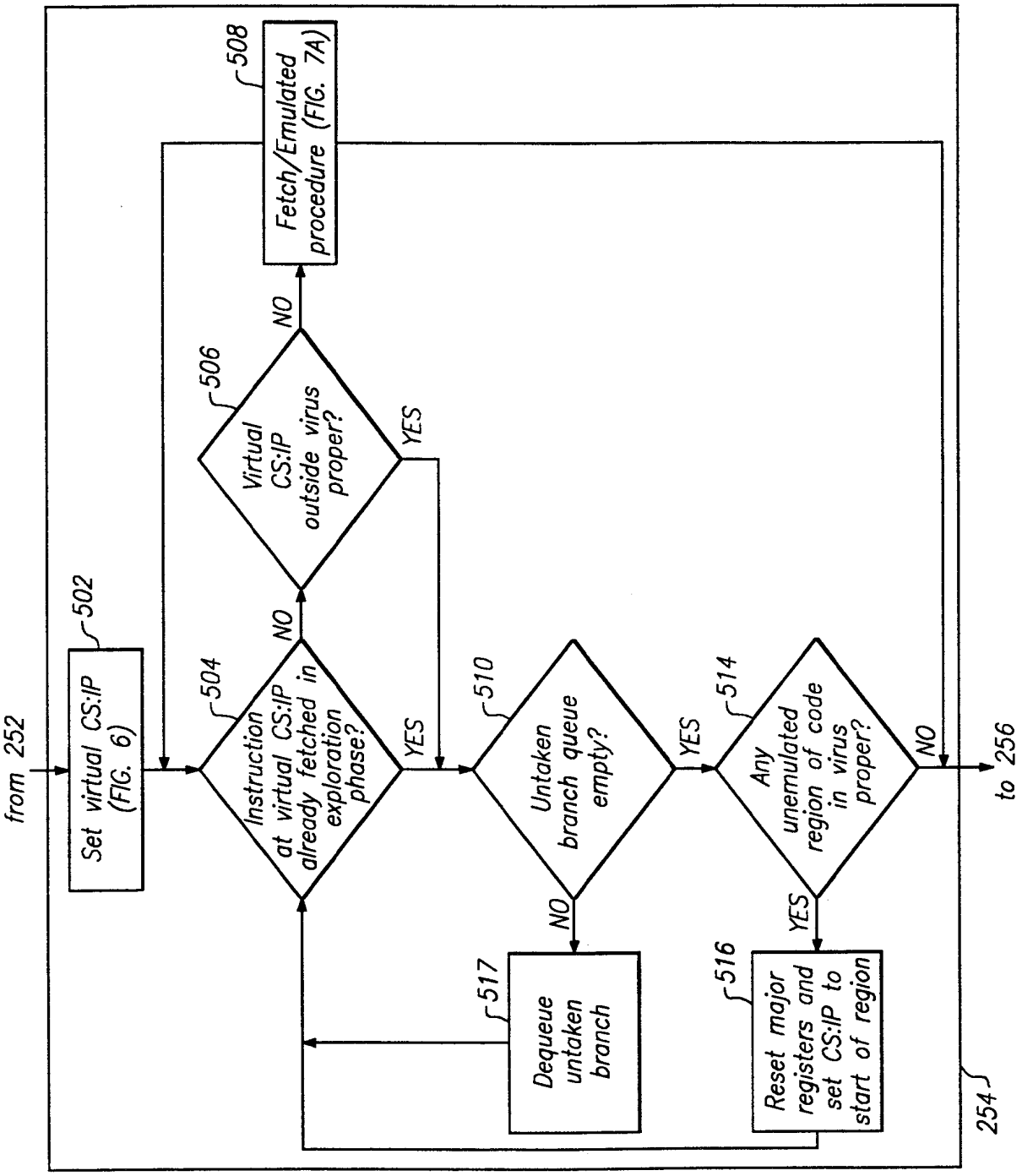

from 318
420
region already set?
YES
NO
422
EXE file?
YES
424
Establish region from start of CS to end file
NO
426
>n instructions emulated?
YES
428
CS:IP w/in starting segment?
YES
430
Set region around CS:IP
NO
NO
320
to 304

FIG. 4B

FIG. 5


from 252
Set virtual CS:IP (FIG. 6)
502
Instruction at virtual CS:IP already fetched in exploration phase?
504
NO
YES
Virtual CS:IP outside virus proper?
506
NO
YES
Fetch/Emulated procedure (FIG. 7A)
508
Untaken branch queue empty?
510
NO
YES
Dequeue untaken branch
517
Any unemulated region of code in virus proper?
514
NO
YES
Reset major registers and set CS:IP to start of region
516
254
to 256

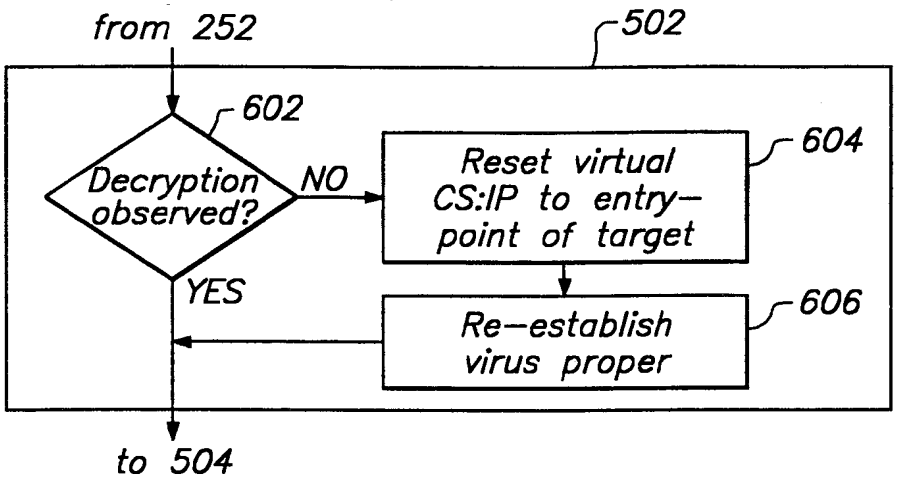

from 252
502
602
Decryption observed?
NO
Reset virtual CS:IP to entry-point of target
604
YES
Re-establish virus proper
606
to 504

FIG. 6

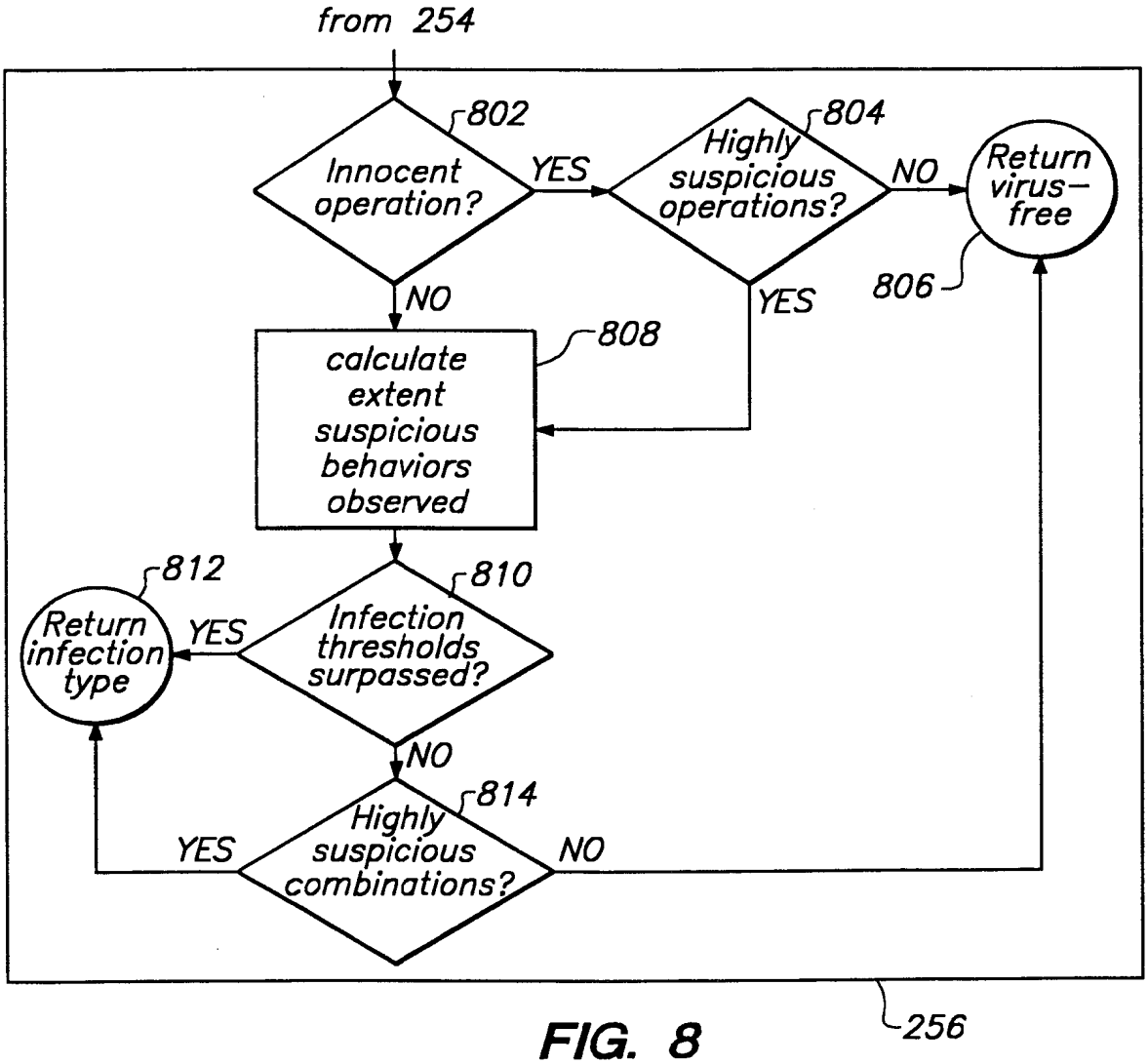

from 254
802
Innocent operation?
YES
804
Highly suspicious operations?
NO
Return virus-free
806
NO
YES
808
calculate extent suspicious behaviors observed
812
Return infection type
YES
810
Infection thresholds surpassed?
NO
814
Highly suspicious combinations?
YES
NO
256

FIG. 8

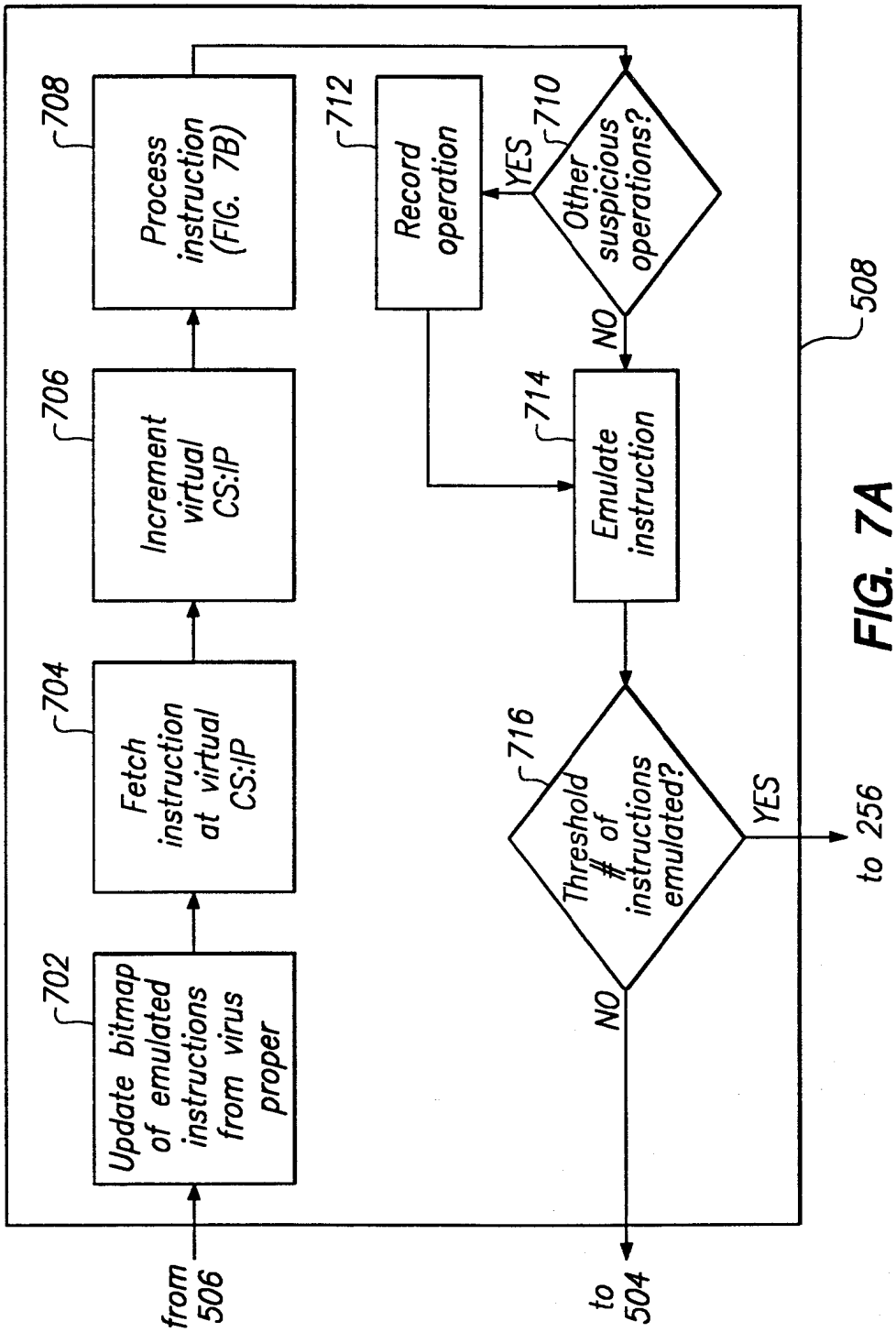

from 506
Update bitmap of emulated instructions from virus proper
702
Fetch instruction at virtual CS:IP
704
Increment virtual CS:IP
706
Process instruction (FIG. 7B)
708
Record operation
712
YES
Other suspicious operations?
710
NO
Emulate instruction
714
Threshold # of instructions emulated?
716
YES
to 256
NO
to 504
508

FIG. 7A

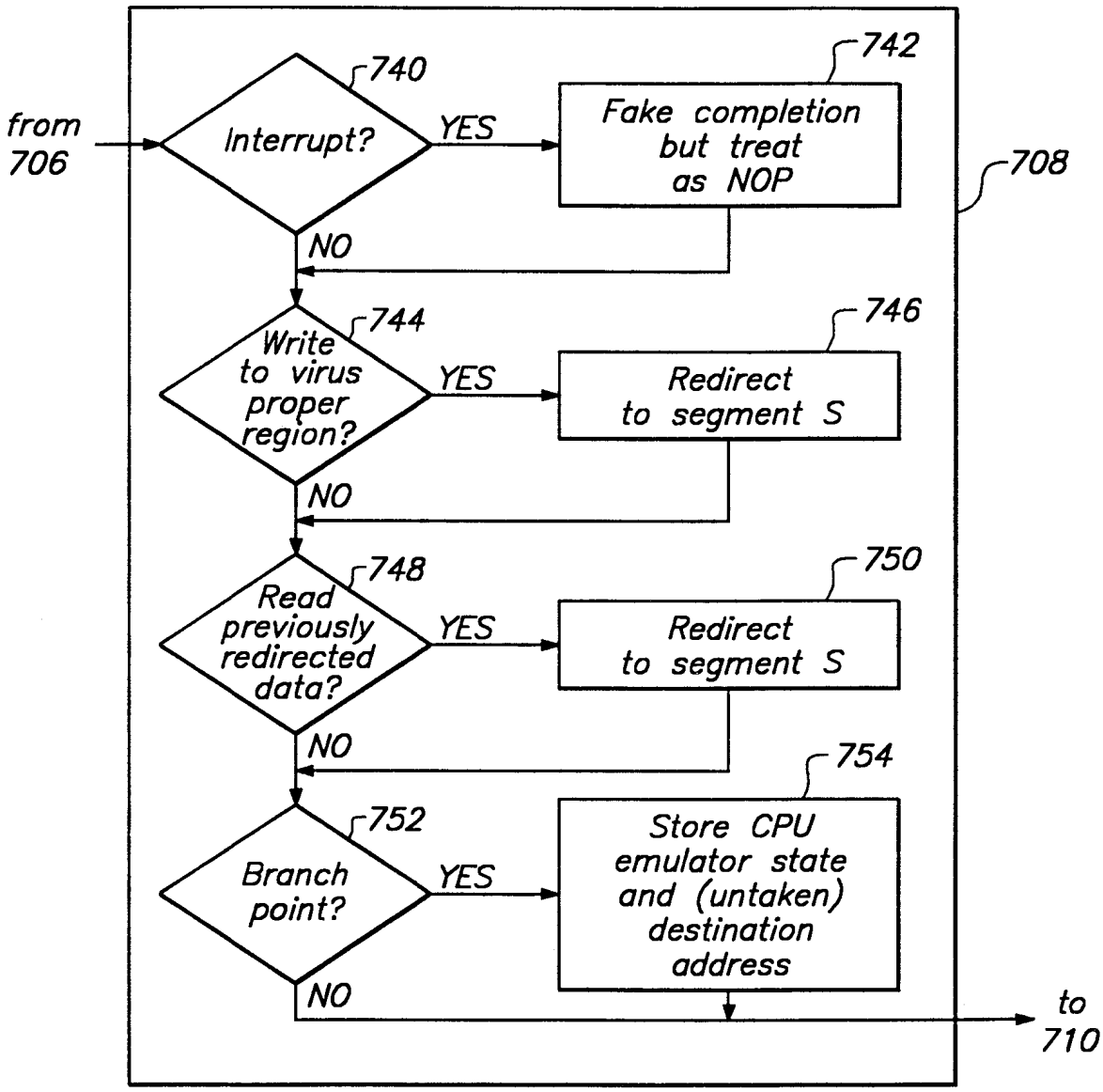

from 706
740
Interrupt?
YES
742
Fake completion but treat as NOP
708
NO
744
Write to virus proper region?
YES
746
Redirect to segment S
NO
748
Read previously redirected data?
YES
750
Redirect to segment S
NO
754
752
Branch point?
YES
Store CPU emulator state and (untaken) destination address
NO
to 710

FIG. 7B

INTERNATIONAL SEARCH REPORT

Interna .l Application No
PCT/US 98/17609

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F11/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CAREY STOVER NACHENBERG: "A New Technique For Detecting Polymorphic Computer Viruses" 1995 , UNIVERSITY OF CALIFORNIA , LOS ANGELES XP000197628 see page 70, line 1 - page 82, line 6	1-5, 8-11, 13-20,36
Y	---	6,7,12
X	EP 0 636 977 A (DAVID ALAN CHAMBERS) 1 February 1995 see column 8, line 15 - column 12, line 58	1
Y		6,7,12
A	see column 14, line 5 - line 17	21-35, 38-46
	--- -/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 January 1999	21/01/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Corremans, G

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Interna l Application No
PCT/US 98/17609

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	WO 95 33237 A (QUANTUM LEAP INNOVATIONS INC.) 7 December 1995 see page 12, line 23 - page 13, line 24	

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Interna al Application No
PCT/US 98/17609

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 636977 A	01-02-1995	US 5398196 A	14-03-1995
WO 9533237 A	07-12-1995	CA 2191205 A	07-12-1995
		EP 0769170 A	23-04-1997
		JP 10501354 T	03-02-1998
		US 5842002 A	24-11-1998

Form PCT/ISA/210 (patent family annex) (July 1992)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 536 341 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
01.06.2005 Bulletin 2005/22

(51) Int Cl.[7]: **G06F 12/10**

(21) Application number: **04019312.0**

(22) Date of filing: **13.08.2004**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PL PT RO SE SI SK TR
Designated Extension States:
AL HR LT LV MK

(30) Priority: **26.11.2003 US 723823**

(71) Applicant: **MICROSOFT CORPORATION**
Redmond, Washington 98052 (US)

(72) Inventor: **Cohen, Ernest S.**
Redmond, WA 98052 (US)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) Lazy flushing of translation lookaside buffers

(57) Address translation control (ATC) limits the mappings between virtual and physical addresses in order to implement a memory access policy. Each processor in a multi-processor system maintains a translation lookaside buffer (TLB) that caches mappings to speed translation of virtual addresses. Each processor also maintains a counter. Each time a processor's TLB is flushed, the processor's counter is incremented. When a link to a page is removed from an address translation map, the counter values for all of the processors are recorded. When that page is accessed by a processor, the recorded counter values are compared with the processor's current counter value to determine whether the processor's TLB has been flushed since the link to the page was removed from the map. An expensive TLB flush operation is delayed until needed, but still occurs early enough to prevent an invalid TLB entry from being used to violate the access policy.

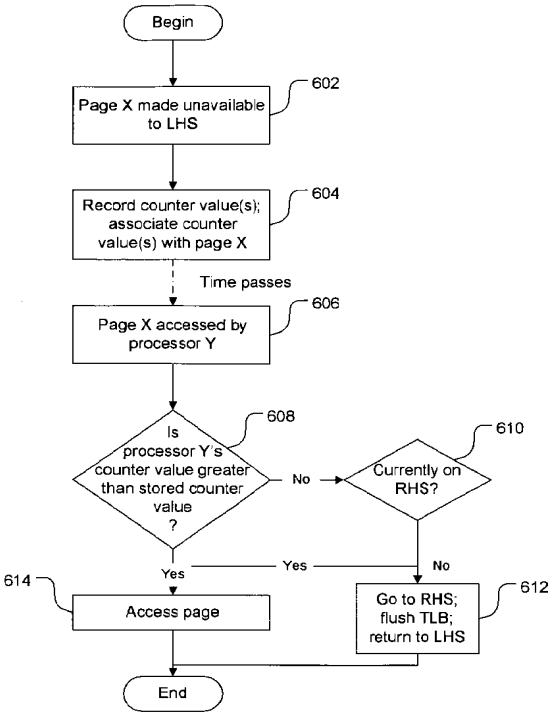


FIG. 6

EP 1 536 341 A2

Description

FIELD OF THE INVENTION

[0001] The present invention relates generally to the field of memory management, and, more particularly, to the flushing of address translation caches.

BACKGROUND OF THE INVENTION

[0002] Most computer systems provide a virtual address mechanism, whereby virtual addresses are mapped to physical addresses. When a request to access a memory location is made using a virtual address, the virtual address is translated into the corresponding physical address of the target memory location to which access is sought. A set of address translation tables defines the mapping between virtual addresses and physical addresses. The translation tables are normally stored in memory, so the translation of an address requires a memory access in order to read the tables. This memory access operation required to read the tables is in addition to the access operation to be performed on the target location. Thus, when virtual addressing is used, the number of memory accesses performed by a system may double relative to the number that would take place if all access requests were made by physical address. Some virtual addresses are multi-leveled in the sense that they require the mapping to be dereferenced in stages, which means that it may take two or more memory accesses to perform an address translation (thereby tripling or more the number of memory accesses that are needed to carry out one underlying access request).

[0003] In order to reduce the number of memory accesses that must take place to translate an address, many virtual address systems employ a type of cache called a translation lookaside buffer (TLB). Since memory pages that have been accessed recently are likely to be accessed again in the near future, once the address translation tables have been used to translate a virtual page descriptor into a physical page location, the correspondence between the virtual page and physical page is cached in the TLB. Every time an address translation needs to be performed, the TLB is checked to determine whether the TLB contains a cached mapping for the page on which the requested unit of memory is located. If the relevant mapping has been cached in the TLB, then the cache copy is used; otherwise, the address is translated from the translation tables. Since accessing the TLB is faster than accessing the translation tables in memory, use of a TLB speeds up performance when successive memory accesses are located on the same group of pages - which is normally the case.

[0004] TLBs create some additional issues when virtual memory is used to provide memory protection. Memory protection seeks to enforce a security policy governing which software components can perform which kinds of access (e.g. reading, writing) to which physical memory pages; this protection can be enforced by the virtual memory by controlling edits to the virtual-to-physical address translation. (This control can be exercised either by the operating system that creates the mappings or by an address translation control (ATC) system that filters changes to such mappings.) However, when an address translation is modified, old mappings may still exist in the TLB. Thus, when the address translation tables are edited to revoke some access right to a page for some software component, the component might retain access to the page until these old mappings have been flushed from the TLBs. The usual way to do this is to force any relevant TLBs to be flushed as part of such an operation.

[0005] However, flushing a TLB is expensive, particularly on shared memory multiprocessors. Every processor that might contain a stale mapping violating the new security policy has to be signaled to flush its TLB; this signaling typically requires a relatively slow interprocessor interrupt (IPI). In addition, the flush itself is relatively expensive.

[0006] In view of the foregoing, there is a need for a mechanism that overcomes the drawbacks of the prior art.

SUMMARY OF THE INVENTION

[0007] The present invention provides a mechanism that supports the lazy flushing of TLBs. For each TLB, a counter is maintained, and every time a TLB is flushed, the counter is incremented. In a multi-processor system where each processor maintains its own TLB, there may be a separate counter for each processor, where the counter for a specific processor is incremented when that processor flushes its TLB. When a triggering event occurs, the values of any relevant counters are recorded after the triggering event has completed. A "triggering event" is an event that affects the address translation map, or the memory access policy under which the map is constrained, in such a way that stale TLB entries could result in violation of the policy; a relevant counter is the counter of a TLB that might contain such stale entries. For example, a given page of memory may be declared off-limits under a policy, and all mappings to that page may be removed from the address translation map; this de-linking of the page from the map is an example of a triggering event, and the counters for any TLBs that might contain a mapping into the page are relevant. It may be the case that a triggering event is performed and that substantial time passes before the effects or results of that triggering event are actually used in the address translation process. (For example, the mappings to a given page of memory may change at a given point in time, but millions of operations may take place before any attempt is made to reuse the de-linked page.) When the results of a triggering event are to be used, the stored counter values are compared with the current

counter values, in order to determine which (if any) relevant TLBs might not have been flushed since the triggering event. If any relevant TLBs have counter values that match their recorded value, all such TLBs are flushed in the conventional way.

[0008] It should be noted that a TLB is safe - i.e., definitely free of stale TLB entries that could result in violation of the applicable access policy - if the counter value has changed, and the TLB is unsafe if the counter value has not changed. Thus, it is safe to use any size of counter, and to change the counter in an arbitrary way (or even not change it al all) on a flush. Another possibility is to use a real-time counter to timestamp the time of the last flush; if clocks are synchronized to within some time skew, the initiator of the flush can simply record the time at which the translation was modified, and assume completion of any flushes whose timestamps exceed the translation modification time by at least the time skew.

[0009] In a system where high-assurance and non-high-assurance environments (the "right side" and "left side," respectively) exist side-by-side, and in which every TLB flush requires a trip to the right side, the mechanism of the present invention can be used to avoid unnecessary trips to the right side. For example, each change between the right side and the left side may result in a TLB flush, and the counter for each processor may be incremented each time the processor moves from the right side to the left side. In other words, since the right side reliably performs (or verifies) the flushing of the TLB, the counter reflects the last time that the TLB could have contained entries that could be damaging to the memory access control scheme. Each time a page is made inaccessible to (or non-writeable by) the left side, that page is essentially "timestamped" with the value of the counter(s). When that page is accessed by a processor, it is determined whether the processor either: (1) is currently in the right side, or (2) has reliably flushed its TLB since the page's status changed. The latter determination can be performed by ensuring that the processor's counter value is greater than the saved value for that page.

[0010] Other features of the invention are described below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] The foregoing summary, as well as the following detailed description of preferred embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings exemplary constructions of the invention; however, the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

[0012] FIG. 1 is a block diagram of an example computing environment in which aspects of the invention may be implemented;

[0013] FIG. 2 is a block diagram of an example virtual address system;

[0014] FIG. 3 is a block diagram of two environments that co-exist on a machine, and a memory part of which is curtained;

[0015] FIG. 4 is a block diagram showing a translation lookaside buffer (TLB) being used in an address translation process;

[0016] FIG. 5 is a block diagram of a system that comprises plural processors, where each processor is associated with a TLB and a counter; and

[0017] FIG. 6 is a flow diagram of a process for lazy flushing of a TLB in accordance with aspects of the invention.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

Overview

[0018] Address translation control (ATC) may be used to implement a memory access policy by dynamically controlling the mappings that are used to translate virtual addresses to physical addresses. When a translation lookaside buffer (TLB) is used to cache mappings, old mappings that are inconsistent with the current state of the access policy may remain in the TLB, thereby exposing the memory to use that runs counter to the policy. The TLB can be flushed, but flushing the TLB is an expensive operation, which is preferably not performed more often than necessary. The present invention provides a mechanism for a lazy TLB flush, that allows a flush of a TLB to be delayed until the flush is needed. The mechanism is preferably used in a multi-processor system in which each processor maintains its own TLB, which can be flushed separately from the other processors' TLBs.

Example Computing Arrangement

[0019] FIG. 1 shows an exemplary computing environment in which aspects of the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0020] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based sys-

tems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, embedded systems, distributed computing environments that include any of the above systems or devices, and the like.

[0021] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network or other data transmission medium. In a distributed computing environment, program modules and other data may be located in both local and remote computer storage media including memory storage devices.

[0022] With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The processing unit 120 may represent multiple logical processing units such as those supported on a multi-threaded processor. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus). The system bus 121 may also be implemented as a point-to-point connection, switching fabric, or the like, among the communicating devices.

[0023] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0024] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in-ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0025] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 1 illustrates a hard disk drive 140 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0026] The drives and their associated computer storage media discussed above and illustrated in FIG. 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In FIG. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can

either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 20 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195

[0027] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0028] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Example Virtual Address Scheme

[0029] FIG. 2 shows an example of a virtual address system. The example depicted in FIG. 2 is a page-type virtual address scheme, although it will be understood that virtual addressing can be based on other models, such as segmentation. The scheme shown in FIG. 2 is a two-level address scheme, such as one of the virtual addressing schemes available on the INTEL x86 processor. The scheme is "two-level" in the sense that one must use two levels of indirection to translate a virtual page identifier into a physical page, as described below.

[0030] In this paging scheme, page directory 202 contains a set of entries. An example structure of an entry is more particularly described below in connection with FIG. 3, but, in essence, each entry identifies the physical location (i.e., page frame number or "PFN") of a particular page table, such as page tables 204(1), 204(2), or 204(3). Each page table, in turn, contains a set of entries, where each entry identifies the physical location (again, the page frame number) a particular data page, such as pages 206(1), 206(2), 206(3), or 206(4). Data pages are defined-length contiguous portions of RAM 132. Data pages may store any type of data, and, it should be noted that, in addition to storing ordinary data, data pages are also used to store the contents of page directory 202 and pages 204(1) through 204(3). Thus, a given page could be a directory, a table, a data page, or could play multiple roles as any combination of those three structures.

[0031] The virtual address scheme depicted in FIG. 2 is a two-level virtual address scheme, since it is necessary to go through both a page directory (level 1) and a page table (level 2) in order to locate a particular page. It will be appreciated by those of skill in the art that it is possible to design a virtual address system with an arbitrary number of levels, and the principles of the present invention can be applied to all such virtual address schemes. As is known in the art, the INTEL x86 processor supports virtual addresses having one, two, or three levels, and typically employs a "hybrid" scheme, in which "small" pages (i.e., pages that are four kilobytes in length) use two-level virtual addresses, while "large" pages (i.e., pages that are four megabytes in length) use one-level virtual addresses.

[0032] In the paging scheme of FIG. 2, any byte on a page can be identified by a virtual address 210, comprising a page directory offset 211, a page table offset 212, and a page offset 213. Thus, in order to locate a physical address, a memory management unit (MMU) 220 that performs the translation of addresses uses page directory offset 211 to locate a particular entry in page directory 202. For example, offset 211 may be equal to zero, indicating that the zero-th entry in page directory 202 should be consulted. This entry contains the PFN at which a page table is stored, so MMU 220 uses this PFN to locate one of the page tables (e.g., page table 204(1)). MMU 220 then uses page table off-

set 212 as an index into the identified page table, and retrieve the entry found at that offset. The entry contains the PFN of a data page (e.g., page 206(1)), so MMU 220 adds page offset 213 to the base address of the identified page in order to locate a particular byte of physical memory. MMU 220 may also be adapted to perform various other functions in addition to the mere translation of addresses: e.g., MMU 220 may load the page from disk if the page's entry in the table is marked "not present"; MMU 220 may disallow write access if the page is marked "read-only," etc.

[0033] In the virtual address scheme of FIG. 2, the location (i.e., PFN) of the page directory itself is stored in a storage location 201. MMU 220 uses the contents of this storage location to locate the page directory 202 when it begins to translate virtual address 210. Thus, there can be multiple page maps in existence, and a particular map can be selected for current use by setting the contents of storage location 201 to contain the PFN of a given map's page directory. In the example of an INTEL x86 processor, storage location 201 corresponds to the register named CR3.

High Assurance Environment and Curtained Memory through Address Translation Control

[0034] It will be appreciated that one feature of the virtual address system described above is that it is possible for there to be a physical address that is not mapped into by any virtual address under a given address translation map. Thus, in a system such as the INTEL x86 processor where nearly all access requests are made by virtual address, it is possible to make a portion of the physical memory off-limits to a given source by ensuring that the source's address translation map does not point to the off-limits memory. Exerting control over an address translation map to achieve such off-limits memory is referred to as Address Translation Control (ATC). The off-limits portion of memory is sometimes referred to as "curtained memory," and ATC is one way to achieve a curtained memory.

[0035] One use of a curtained memory is when a high-assurance environment and a non-high-assurance environment co-exist on the same machine. The high-assurance environment can be provided with a curtained memory that cannot be accessed by the non-high-assurance environment. The concept of a high-assurance environment - and its relationship to curtained memory - is described below with reference to FIG. 3.

[0036] FIG. 3 shows two environments that co-exist on the same machine: a high-assurance environment 360, and a non-high-assurance environment 350. For convenience, these environments shall be referred to as the right-hand-side (RHS) and left-hand-side (LHS) respectively; RHS 360 is high-assurance, and LHS 350 is non-high-assurance. What it means for an environment to be "high-assurance" is that there is a high level of assurance that functions performed in the environment will be performed correctly. Thus, LHS 350 performs various functions 306(1), 306(2), ... 306(n), and RHS 360 performs various functions 308(1), 308(2), ..., 308(m). LHS 350 is associated with a specification 302 that describes how functions 306(1) through 306(n) are supposed to behave. Likewise, RHS 360 is associated with a specification 304 that describes how functions 308(1) through 308(m) are supposed to behave. Specifications 302 and 304 may or may not be written. For example, LHS 350 may be a commercial operating system that comes with a detailed written manual explaining how its various services, drivers, system calls, etc., are supposed to behave. Or, there may simply be a general understanding as to how a given environment is supposed to behave, and this understanding may constitute a specification.

[0037] Regardless of what form a specification takes, it will be understood that nearly all software contains bugs, backdoors (both known and undiscovered), logical errors, etc., that can cause the software to behave in some unexpected manner. However, it is possible to evaluate the level of assurance that a given piece of software will behave in the expected manner - i.e., the level of assurance that the software's behavior will actually match that which is described in its specification. In the example of FIG. 3, RHS 360 is "high-assurance" in the sense that there is a relatively high level of assurance that the functions 308(1) through 308(m), which RHS 360 is designed to perform, will actually be performed in conformance with specification 304. Likewise, LHS 350 is "non-high-assurance" in the sense that there is a relatively low level of assurance that LHS 350 will perform functions 306(1) through 306(n) in conformance with specification 302. "Relatively high" and "relatively low," in this context, mean that the level of assurance that RHS 360 will behave according to specification 304 is higher than the level of assurance that LHS 350 will behave according to specification 302.

[0038] It is generally the case that a full-featured commercial operating system, such as one of the MICROSOFT WINDOWS operating systems, is a non-high-assurance environment. Such environments support an open architecture in which device drivers, plug-ins, and other types of extensions can be added freely, which makes the behavior of such an operating system under all imaginable circumstances difficult to verify. Therefore, when some type of security is desired, it is useful to run such a full-featured operating system side-by-side with a high-assurance operating system. The high-assurance operating system is a small operating system that provides a small number of features. Because the functionality provided by the high-assurance operating system is small, there are fewer variables that can affect its operation, so its behavior is easier to verify to a high degree of certainty.

[0039] In a preferred embodiment, a high-assurance environment includes a curtained memory - i.e., a portion of memory that is inaccessible to the non-high-as-

surance environment. Thus, RHS 360 can store secret data (e.g., cryptographic keys) in the curtained memory, without the danger of that danger being read or written by processes that run under LHS 350. For example, specification 304 may provide that RHS 360 has the capability to protect information from outside tampering, and the curtained memory allows RHS 360 to perform this function. Moreover, the code that RHS 360 uses to perform its various functions may be stored in the curtained memory, in order to prevent processes running under LHS 350 from overwriting this code with different code (which might cause RHS 360 to behave outside of its specification). FIG. 3 shows the curtained memory 312 that is available for use by RHS 360. Physical address space 310 includes all of the physical memory locations available on a given computing device. Curtained memory 312 is a subset of those locations. As shown in FIG. 3, RHS 360 has access to all of physical address space 310, but LHS 350 lacks access to the portions of physical address space 310 that constitute curtained memory 312. (It should be noted that, while FIG. 3 shows the curtained and non-curtained portions of the address space as each being contiguous, there is no requirement for such contiguousness. Moreover, it is not necessary for RHS 360 to have access to the entire physical address space, or for LHS 350 to have access to every portion of the physical address space that is outside of curtained memory 312.)

[0040] As noted above, there are certain systems where nearly all memory access requests are made by physical address. One way to implement curtained memory 312 on such a system is to control the content of the address translation maps in such a way that no virtual addresses for curtained memory 312 are exposed to LHS 350. (Where there is a potential for some types of access requests identify their target by physical address, access to curtained memory can be limited by some auxiliary cooperating mechanism, such as an exclusion vector that filters physical access requests from direct memory access devices or from other sources.) A number of algorithms are available to ensure that LHS 350 cannot use an address translation map that would lead to curtained memory 312, but the central idea behind these algorithms is: (1) when a processor is operating in LHS 350, ensure that any map loaded into CR3 (storage location 201, shown in FIG. 2) does not lead to pages contained in curtained memory 312; and (2) for any attempt to edit an active map in LHS 350, evaluate the proposed edit to ensure that the edit will not result in a link to a page of curtained memory 312.

[0041] The following is an example algorithm for implementing curtained memory through ATC:

[0042] Let D1 be the set of pages that can be used as page directories. Let D2 be the set of pages that can be used as page tables. Let D be the union of D1 and D2 (i.e., $D = D1 \cup D2$). Each entry in a page directory or page table that is marked "present" (i.e., whose present bit is set) is called a "link." A page in D2 is "write-active" if there is a small read-write link from some page in D1 to the D2 page in question. (A "small" link is a link from a directory to a table - i.e., a link in a directory that will ultimately lead to a small page. A "large" link is a link in a directory that points to a large page.) It is assumed that there is a policy that defines the pages to which some entity is permitted read and/or write access.

[0043] The following invariants are maintained:

- CR3 is in D1;
- all D1 and D2 pages are readable under the relevant policy;
- every small link from a D1 page points to a D2 page;
- links from D2 pages point to a page that is readable under the policy;
- every read-write link from a write-active D2 page points to a page that is writeable under the policy and that is not in D;
- every small page contained in the large page target of a large link from a D1 page is readable under the policy; if the link is read-write, then the small page is also writable under the policy and is not in D.

[0044] If, for example, the policy defines curtained memory 312 as being inaccessible, then maintaining the above invariants for all address translation tables that are usable by LHS 350 will ensure that no access request that identifies its target by virtual address and arises in LHS 350 can ever reach curtained memory 312.

Use of Translation Lookaside Buffers with ATC

[0045] A Translation Lookaside Buffer (TLB) is essentially a cache in which mapping data is stored. The basic idea behind a TLB is that pages that have been recently accessed are likely to be accessed again, so the recently used mappings from a given virtual page to the corresponding physical page are stored in the TLB. Computing a mapping from an address translation table (at least in the most common two-level virtual address scheme employed on an INTEL x86 processor) requires two memory accesses: one access to the page directory to find the location of the relevant page table, and then a second access to the page table to find the relevant data page. The actual target location to which access is being sought can only be accessed after these two (other) memory accesses have been performed to find the target data's physical location. A TLB reduces the need for such additional memory accesses by providing fast memory in which recently used mapping information is maintained. Thus, for every address translation that needs to be performed, the processor consults its TLB first to determine whether the mapping data for a given virtual page is cached in the TLB. The address translation table is consulted if the mapping data is not present in the TLB.

[0046] FIG. 4 shows how a processor uses a TLB 402. Processor 120 executes a piece of code, and, during

that execution, an instruction arises to read or write target location 406 of memory 400. The instruction does not identify the physical address of target location 406, but rather specifies the virtual address of target location 406. Thus, processor 120 (or, in many cases, the memory management unit associated with processor 120) must translate the virtual address into a physical address. Processor 120 first looks in TLB 402 to determine whether a mapping has been cached that can be used in translating the virtual address (circled 1). If such a mapping exists, then this mapping is used to translate the address. If no such mapping exists, then the relevant portions of address translation map 404 are read from memory 400 (circled 2), and these map portions are used to translate the virtual address. Regardless of how the address is translated, the translated address is used to access the physical target location 406 (circled 3). The reason that a TLB provides efficiency is that it is faster to look up information in TLB 402 than it is to read address translation map 404 in memory 400. Additionally, in some designs the data stored in TLB 402 can collapse the two-step translation process into one step: i.e., using the address translation map requires reading the page directory and page table in separate steps; however, a single PD offset/PT offset combination could be cached in TLB 402, thereby allowing a virtual page to be identified by the PD/PT offset combination in a single step.

[0047] When using TLBs in an ATC system, the ATC invariants are modified to take into account translations available through the TLB. For example, in the previously described example ATC algorithm, the definition of a link can be modified as follows: a link from one page to another exists if it exists either in the physical memory or if it is cached in some TLB. Although the contents of the TLBs are not directly observable in most architectures (e.g. in x86 processors), its contents can be bounded because translations enter TLBs only through memory. Using the invariants in the example algorithm, the TLBs then have to be flushed on removing a page from D1 or D2, giving up read or read-write access to a page, and on writes or additions to D2 that change the status of a page from write-active to write-inactive or vice-versa.

[0048] However, there is substantial incentive to reduce TLB flushes even further, because flushing the TLB is an expensive operation that slows performance. First, the TLB only improves efficiency to the extent that it is populated with useful mapping data, and flushing the TLB eliminates this mapping data from the TLB. Second, the actual flush itself may be expensive for the following reason: In the architecture described in FIG. 3, the maintenance of curtained memory 312 depends on the absence of counter-policy mappings. Since the absence of counter-policy mappings is, in a preferred embodiment, an element of providing a high-assurance environment, control over the mappings must, itself, be performed in a high-assurance environment. Thus, not only must control over the address translation map itself be performed in a high-assurance environment, but also the flushing of the TLB must be performed (or at least verified) in a high-assurance environment. Therefore, every TLB flush that is being relied upon as part of an ATC system requires that the current environment be changed from LHS 350 to RHS 360. The process of changing context from one environment to another is, itself, expensive, which increases the expense of a flush. Therefore, it is desirable to avoid flushing the TLB more often than necessary.

[0049] The present invention provides mechanism for delaying and possibly avoiding some of these flushes. The invention, in one embodiment, is based on two submechanisms. The first submechanism provides that operations that do not modify the actual memory map (e. g. adding a page to D2, or making the page inaccessible to writes), can be conceptually delayed until their completion is required to permit some modification to memory. For example, the act of giving up read access to a page (in the security policy) can be delayed until there is the possibility of that page being modified (when. another computational entity creating a read-write mapping to the page). The second submechanism uses timestamps to track more precisely whether a particular mapping to a page might exist in a particular TLB.

[0050] In the example algorithm presented earlier, delayable operations requiring flushes can be partitioned into two classes: those requiring removal from the TLBs any translations that allow a page to be read (giving up read access to a page, or removing the page from either D1 or D2), and those requiring removal from the TLBs any translations that allow a page to be modified (removing write access to a page). Calling the former "read flushing operations" and the latter "write flushing operations", these operations can be delayed as follows:

- read-flushing operations can be delayed until some computational entity tries to create a read-write mapping to the page, or until some device might be able to modify the page through direct memory access, or until the page is modified by the ATC algorithm itself;
- write-flushing operations can be delayed until some computational entity tries to create a read or read-write mapping to the page, or until some device might be able to modify the page through direct memory access, or until the page is modified by the ATC algorithm itself.

[0051] When performing a read-flushing operation, it is only necessary to flush a TLB if it still potentially contains a read mapping to the page in question. This can only be the case if the TLB has not been flushed since the last time that there was a read mapping to the page. Similarly, when performing a write-flushing operation, it is only necessary to flush a TLB if it has not been flushed since there was last a read-write mapping to the page.

[0052] To determine whether a TLB was flushed since there was a read/read-write mapping to a page, we maintain two timestamps for each page (its read timestamp and its write timestamp) and one timestamp for each TLB (its flush timestamp). Whenever the last read/read-write mapping to a page is removed, the current time is recorded in its read/read-write timetamp. Whenever the TLB is flushed, the current time is recorded in its flush timestamp.

[0053] Time can be measured in several ways. In a first example approach, the clock used for a TLB is just the flush timestamp. Thus, the read and read-write timestamp for a page is copied from the TLB flush timestamp. When the TLB is flushed, the flush timestamp is changed (it is not important how). If the read/read-write timestamp on a page differs from the flush timestamp, the read/read-write mapping is guaranteed not to be in the TLB. If by coincidence, the flush timestamp has changed (perhaps several times) but happens to coincide with the read/read-write timestamp (e.g. because of counter wraparound), no harm is done; there will just be an unnecessary flush.

[0054] In a second example approach, time is measured with real-time or virtual clocks. If the clocks used to record the read/read-write timestamps and flush timestamps are guaranteed to always agree within some clock skew, the read/read-write mapping is guaranteed not to be in the TLB if the read/read-write timestamp precedes the flush timestamp by at least the clock skew. One way to avoid clock skew entirely is to use a global clock, a shared counter that is incremented when any TLB is flushed; however, such a clock may become a hotspot.

[0055] The first approach has the advantage that it does not require synchronized clocks. However, if there are a large number of TLBs to be considered, the first approach requires a separate read/read-write timestamp for each TLB, whereas the second approach requires only a single clock read. Moreover, it eliminates memory contention over the clocks themselves. The two approaches can also be combined, and other mechanisms (e.g. vector clocks) can also be employed.

[0056] The present invention provides a mechanism that allows for lazy flushing of a TLB, so that explicit flushes of a TLB are delayed until there is a potential that LHS 350 is actually accessing a page using an old TLB entry that no longer conforms to the address translation map. Essentially, when the last mapping to a page is removed, the page is "timestamped," (As discussed below, the "timestamps" may not actually contain the time, but rather the value of sequential counters.) When the page is subsequently accessed, a determination is made as to whether to flush the TLB depending upon whether the TLB has been flushed since the page was timestamped.

[0057] In a multi-processor computer system, as shown in FIG. 5, each of processors 120(1) through 120 (n) maintains its own TLB 402(1) through 402(n). In other words, processors 120(1) through 120(n) can each access the same underlying address translation tables, but they can separately cache the mappings from those tables. Additionally, in one embodiment each processor can operate in LHS 350 or RHS 360. In accordance with the invention, a counter 502(1) through 502(n) is associated with each of processors 120(1) through 120(n). Each time that the environment on a processor changes from RHS 360 to LHS 350, the counter associated with that processor is incremented.

[0058] Whenever a page is made unavailable to LHS 350 (i.e., when the page is made unavailable to LHS 350 under the relevant access policy and is de-linked from all mappings available to LHS 350), the page is effectively "time-stamped" with the counter values. (The counters do not count "time" in the physical sense, but can be viewed of as a type of timekeeping, since the counters move forward in response to certain events as they occur.) Thus, by comparing the stored counter values for a page with the current counter value for the processor that is attempting to access the page, it can be determined whether the processor has entered RHS 360 since the page was made unavailable to LHS 350. (As explained above, entering RHS 360 causes the TLB to be reliably flushed.) If the processor has not entered RHS 360 since the page was de-linked, then the processor enters RHS 360 and flushes its TLB. If the attempt to access the page arose in LHS 350, then the processor returns to LHS 350 and attempts again to access the page. If the processor has entered RHS 360 since the page was de-linked (i.e., if the processor is currently in RHS 360, or has, at some point after the page was de-linked, entered RHS 360), then it is known that the TLB was reliably flushed since the last time the page was available to LHS 350, and thus the TLB cannot contain any mapping to the page, so the TLB does not need to be flushed at that time.

[0059] FIG. 6 is a flow diagram showing the process by which a TLB is flushed following a page being made unavailable to LHS 350. At the outset of the process shown in FIG. 6, it is presumed that a given page (page X) is available to be accessed from LHS 350 under an applicable access policy. At some point, page X is made unavailable to LHS 350 under the access policy, and the address translation maps available to LHS 350 are adjusted as needed to ensure that page X is de-linked from those maps (602). At the time that page X is de-linked, the counter values for all of the processors on the machine are recorded, and this record is associated with page X for later retrieval (604). The record preferably contains the current counter value for each of the processors on the machine.

[0060] After page X is de-linked from maps available to LHS 350, some amount of time may pass before page X is actually accessed. At some point, however, an access request for page X will originate from a particular processor (processor Y) (606). The recorded counter values associated with page are then retrieved, and the

stored value for processor Y is compared with processor Y's current counter value. (The record will also contain the stored counter values for other processors, but these other values are disregarded in determining whether processor Y needs to flush its TLB.) If processor Y's current counter value is greater than the stored counter value, then access to the page is permitted (614) without further inquiry, since it must be the case that processor Y entered RHS 360 and flushed its TLB since page X was made unavailable to LHS 350. On the other hand, if the stored counter value for processor Y is the same as processor Y's current counter value, then RHS 360 has not been entered since page X was de-linked, which means that processor Y's TLB may still contain a mapping to page X.

[0061] If processor Y is running in RHS 360 (610), then TLB can contain a mapping to page X without violation of the access policy, so access to page X is permitted (614). If, however, processor Y is running in LHS 350 (and it is known from 608 that processor Y has not flushed its TLB since page X was de-linked), there is a possibility that the contents of processor Y's TLB would expose, to LHS 350, a mapping to page X. Thus, processor Y enters RHS 360 and flushes its TLB (612). After the TLB is flushed, processor Y returns to LHS 350 and the access request is reexecuted (614) (thereby requiring re-translation of the requested address, with the newly-empty TLB.)

[0062] It should be noted that making a page inaccessible to LHS 350 is not the only event that can trigger the need to flush a TLB, and the mechanism of the present invention can also be used to trigger TLB flushes that arise for other reasons. For example, removing a page from one of the sets D1 or D2 (as defined above), or relinquishing read access (or write access) for a page, may also give rise to the need for a TLB flush, since these events may affect what mappings are legitimate under the relevant policy. In such a case, the mechanism of the present invention may be used in a manner similar to that described above - e.g., the counter values may be recorded when a page is removed from D1 or D2, or when read (or write) access is relinquished, and any subsequent attempt by a processor to access the page may trigger a comparison between the processor's counter value and the stored counter values for the page.

[0063] It should be further noted that the mechanisms of the present invention are not limited to the case where a flush requires a trip to RHS 360 - or even to the case where both LHS 350 and RHS 360 are available. In greater generality, the mechanisms of the present invention can be used in any case where there is a possibility that stale entries exist in the TLB that are no longer reflective of the state of the map - e.g., in the case of ordinary process isolation performed by an operating system.

[0064] It is noted that the foregoing examples have been provided merely for the purpose of explanation and are in no way to be construed as limiting of the present invention. While the invention has been described with reference to various embodiments, it is understood that the words which have been used herein are words of description and illustration, rather than words of limitations. Further, although the invention has been described herein with reference to particular means, materials and embodiments, the invention is not intended to be limited to the particulars disclosed herein; rather, the invention extends to all functionally equivalent structures, methods and uses, such as are within the scope of the appended claims. Those skilled in the art, having the benefit of the teachings of this specification, may effect numerous modifications thereto and changes may be made without departing from the scope and spirit of the invention in its aspects.

Claims

1. A method of clearing obsolete entries from a first one of a plurality of mapping caches, each of the plurality of mapping caches being associated with a corresponding one of a plurality of processing units of a computing device, each of the caches being used to translate virtual addresses to physical addresses and storing mappings based on an address translation map, the method comprising:

   maintaining a counter;
   updating said counter each time the first one of the plurality of mapping caches is flushed;
   recording said counter's value in response to a change in the address translation map;
   determining, based on a comparison of said counter's value with the recorded counter value, that the first one of the plurality of mapping caches has not definitely been flushed since said change in the address translation map occurred; and
   flushing the first one of the plurality of mapping caches.

2. The method of claim 1, wherein said counter is one of a plurality of counters, each counter being associated with a corresponding one of the mapping caches.

3. The method of claim 2, wherein each of the plurality of counters is updated when a counter's corresponding mapping cache is flushed.

4. The method of claim 1, wherein the address translation map comprises links to pages of memory including a first page, and wherein said change comprises placing the address translation map in a state in which the address translation map does not contain any links to said first page.

5. The method of claim 1, wherein the address translation map defines portions of a memory that are readable by an entity, said portions of said memory including a first portion, and wherein said change comprises placing the address translation map in a state in which said first portion is not readable by said entity.

6. The method of claim 1, wherein the address translation map defines portions of a memory that are writeable by an entity, said portions of said memory including a first portion, and wherein said change comprises placing the address translation map in a state in which said first portion is not writeable by said entity.

7. The method of claim 1, wherein a policy defines permissible access to a memory, and wherein the method further comprises:

controlling the content of the address translation map such that the address translation map does not expose to an entity virtual address mappings that would permit said entity to access said memory in violation of said policy;

wherein said change comprises either a modification to the map that places or maintains the map in conformance with said policy, or a modification to the map that limits said entity's write access to the map.

8. The method of claim 7, wherein the address translation map comprises a link to a portion of said memory, wherein controlling the content of the address translation map comprises making said portion of said memory inaccessible to said entity, and wherein said change comprises removing all links to said portion of said memory from the address translation map.

9. The method of claim 1, further comprising:

determining that an event has arisen that potentially makes use of the state of the address translation map prior to said change;

wherein flushing the first one of the plurality of map caches is performed in response to the determination that said event has arisen.

10. The method of claim 9, wherein said event comprises a translation of a virtual address to a physical address that is contained in a portion of memory that was de-linked from the address translation map by said change.

11. A system for managing the use of address mapping caches, the system comprising:

a plurality of processors, each of the processors having a mapping cache and a counter associated therewith;

a memory that stores an address translation map, each of the mapping caches storing mappings based on said address translation map, there being a policy that governs access to said memory, the contents of the address translation map being controlled to prevent exposure of mappings that permit access to said memory in violation of said policy;

first logic that flushes a first one of the mapping caches and that increments a first one of the counters when said first one of the mapping caches is flushed;

second logic that records the current value of said first counter in response to a change in said address translation map or in a property regarding said address translation map, the recorded counter value being stored in association with said change;

third logic that compares the recorded counter value with the current value of the first counter and that causes said first one of the mapping caches to be flushed if the comparison indicates that said first one of the mapping caches has not been flushed since the change.

12. The system of claim 11, wherein each of the processors is associated with said first one of the counters, and wherein said first logic increments said first one of the counters when any of the mapping caches is flushed.

13. The system of claim 11, wherein each of the mapping caches is associated with a different one of a plurality of counters, and wherein said first logic increments the counter corresponding to a given mapping cache when the given mapping cache is flushed.

14. The system of claim 11, wherein said change comprises placing said address translation map in a state in which all links to a first page of said memory are removed from said address translation map.

15. The system of claim 11, wherein said change comprises placing said address translation map in a state in which there is no writeable link to a first page of said memory.

16. The system of claim 11, wherein said third logic is invoked in response to a detection that a result of said change is to be used.

17. The system of claim 16, wherein said change com-

prises placing said address translation map in a state in which all links to a first page of said memory are removed from said address translation map, and wherein said detection is based on a virtual address having been translated to a location on said first page.

18. A computer-readable medium having encoded thereon computer-executable instructions to perform a method of managing the flushing of a translation lookaside buffer that caches address mappings, the method comprising:

receiving an access request that indicates a target location by virtual address;
translating said virtual address to obtain a physical address of said target location;
comparing (1) a stored counter value associated with a page that comprises said target address with (2) a current counter value;
determining based on the comparison between said stored counter value and said current counter value that the translation lookaside buffer has not been flushed since an event affecting a mapping of said page has been modified; and
flushing the translation lookaside buffer.

19. The computer-readable medium of claim 18, wherein the address mappings are defined by an address translation map stored in a memory, the translation lookaside buffer caching mappings that are based on said address translation map.

20. The computer-readable medium of claim 18, wherein a policy defines accessibility of a memory to a software entity, and wherein said event comprises removing links to a page of said memory from an address translation map on which said address mappings are based, said page being inaccessible to said software entity under said policy.

21. The computer-readable medium of claim 18, wherein a policy defines accessibility of a memory to a software entity, and wherein said event comprises adjusting an address translation map to make mappings to a page non-writeable, wherein either: (1) said policy defines said page as being non-writeable by said software entity, or (2) said page stores a portion of said address translation map.

22. The computer-readable medium of claim 18, wherein an address translation control mechanism determines the membership of a set of pages of a memory that may be used to store portions of an address translation map, and wherein said event comprises a change in membership of said set.

23. A method of clearing obsolete entries from a first one of a plurality of mapping caches, each of the plurality of mapping caches being associated with a corresponding one of a plurality of processing units of a computing device, each of the caches being used to translate virtual addresses to physical addresses and storing mappings based on an address translation map, the method comprising:

maintaining an object based on which the sequential progression of events can be determined;
recording said object's value in response to a change to the address translation map that removes one or more of, (1) read access, or (2) read/write access, from a page;
in response to a request to access said page, determining based on a comparison of said object's current value with the recorded value that the first one of the plurality of mapping caches has been flushed since said change occurred; and
if it cannot be determined with certainty based on said comparison that the first one of the plurality of mapping caches has been flushed since said change occurred, then flushing the first one of the plurality of mapping caches.

24. The method of claim 23, wherein said object comprises a clock.

25. The method of claim 23, wherein said object comprises a counter that is incremented each time the first one of the plurality of mapping caches is flushed.

26. The method of claim 23, wherein said object comprises a plurality of counters, each of the counters corresponding to one of the plurality of mapping caches, wherein each of the counters is incremented when the counter's corresponding mapping cache is flushed.

27. The method of claim 23, wherein said change comprises removing read access to said page, and wherein said access request comprises a request to read said page.

28. The method of claim 23, wherein said change comprises removing read/write access to said page, and wherein said access request comprises a request to write said page.

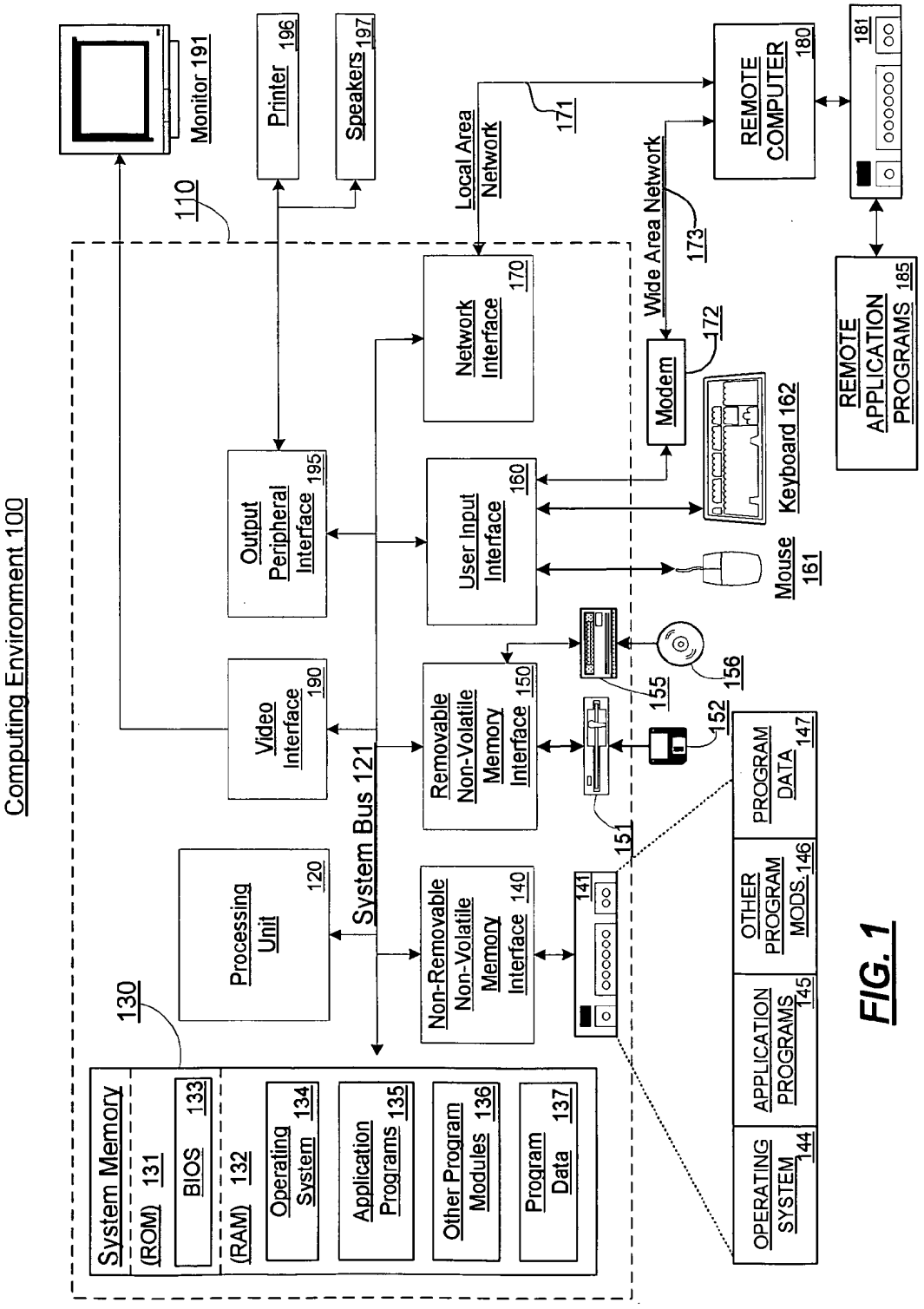

Computing Environment 100
110
Monitor 191
Printer 196
Speakers 197
Local Area Network
171
REMOTE COMPUTER 180
181
Wide Area Network
173
Network Interface 170
Modem 172
REMOTE APPLICATION PROGRAMS 185
Output Peripheral Interface 195
User Input Interface 160
Keyboard 162
Mouse 161
Video Interface 190
Removable Non-Volatile Memory Interface 150
155
156
152
151
System Bus 121
Processing Unit 120
Non-Removable Non-Volatile Memory Interface 140
141
130
System Memory
(ROM) 131
BIOS 133
(RAM) 132
Operating System 134
Application Programs 135
Other Program Modules 136
Program Data 137
OPERATING SYSTEM 144
APPLICATION PROGRAMS 145
OTHER PROGRAM MODS. 146
PROGRAM DATA 147

FIG. 1

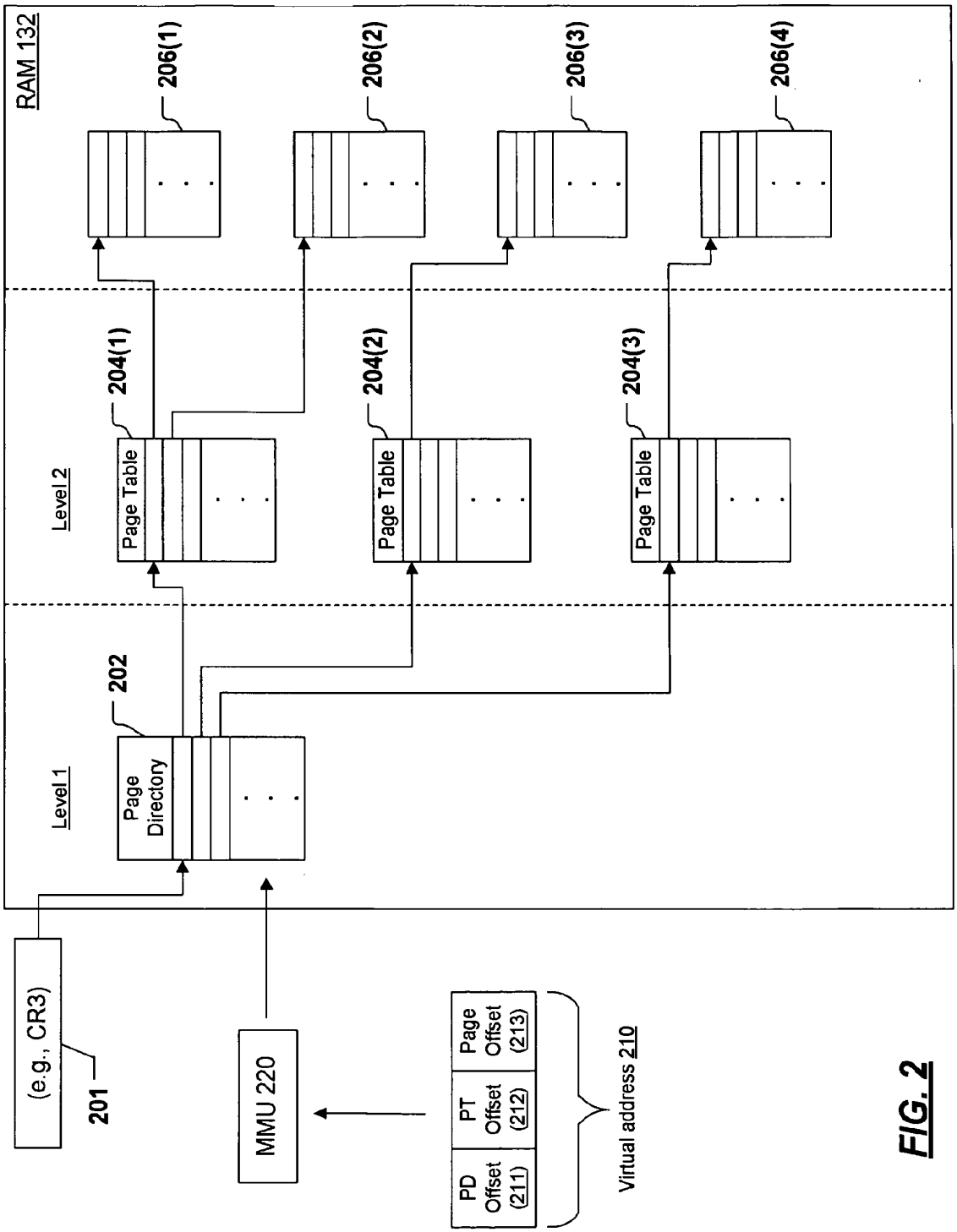

RAM 132
206(1)
206(2)
206(3)
206(4)
204(1)
204(2)
204(3)
Page Table
Page Table
Page Table
Level 2
202
Page Directory
Level 1
(e.g., CR3)
201
MMU 220
PD Offset (211)
PT Offset (212)
Page Offset (213)
Virtual address 210

FIG. 2

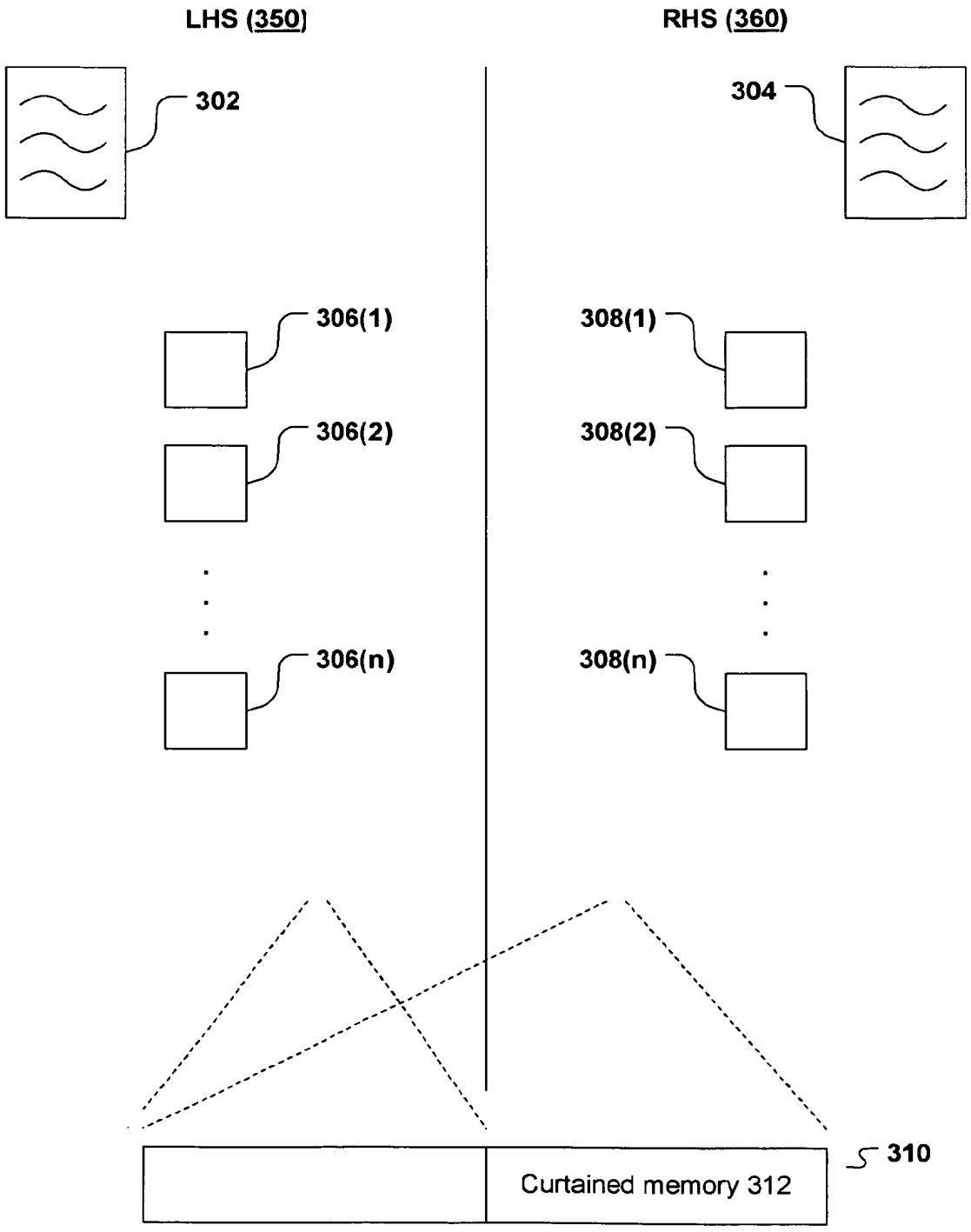

LHS (350)
RHS (360)
302
304
306(1)
306(2)
306(n)
308(1)
308(2)
308(n)
Curtained memory 312
310

FIG. 3

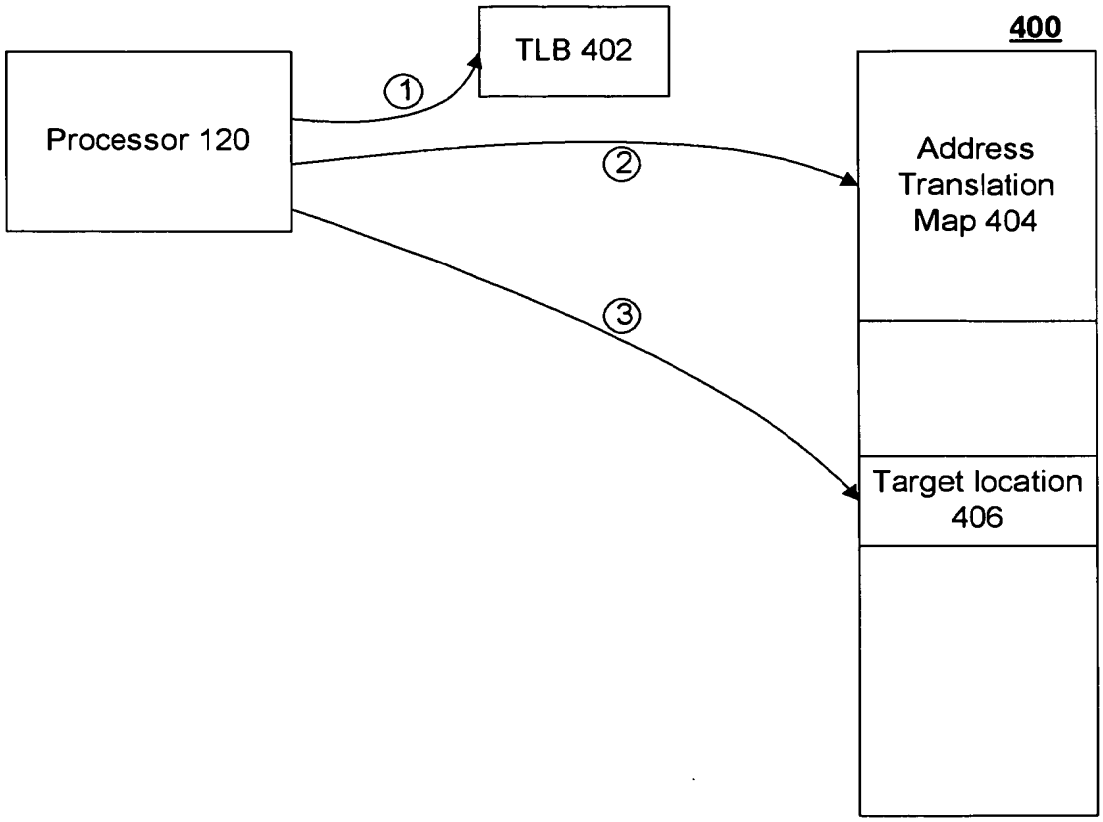

TLB 402
400
Processor 120
1
2
Address Translation Map 404
3
Target location 406

FIG. 4

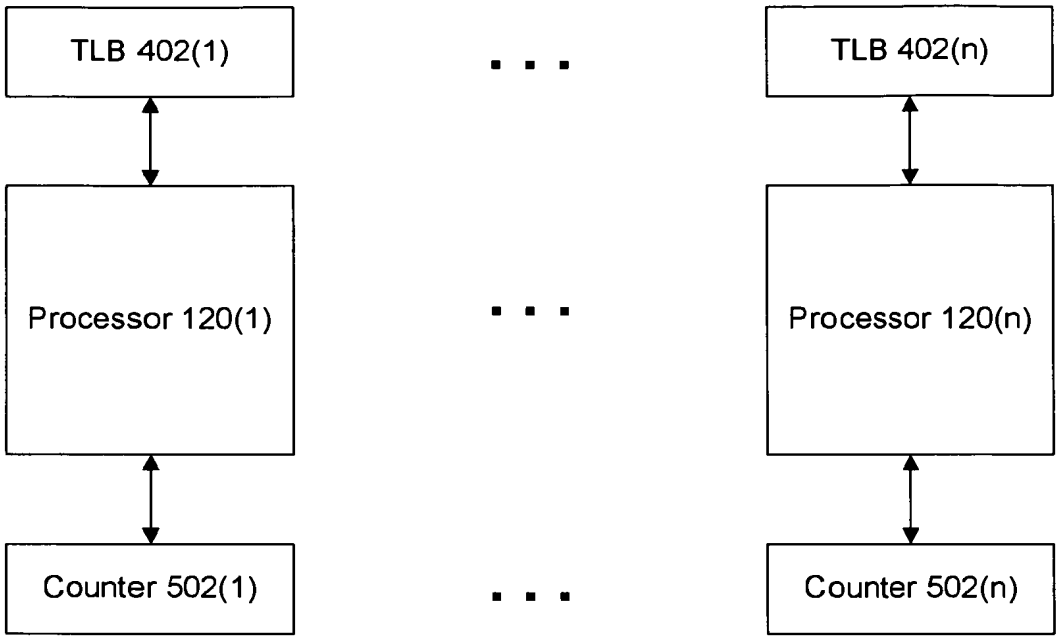

TLB 402(1)
TLB 402(n)
Processor 120(1)
Processor 120(n)
Counter 502(1)
Counter 502(n)

FIG. 5

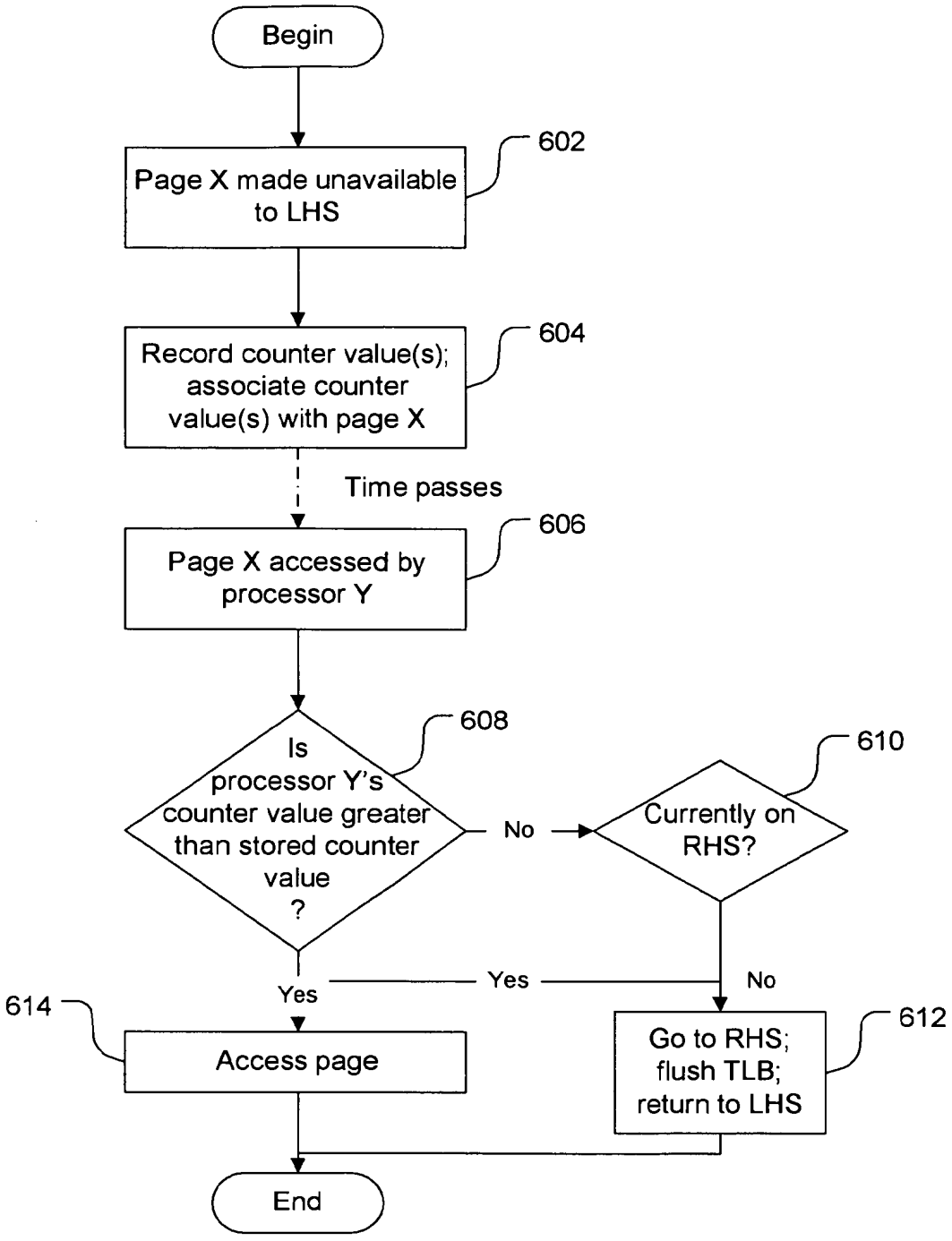

Begin
Page X made unavailable to LHS
602
Record counter value(s); associate counter value(s) with page X
604
Time passes
Page X accessed by processor Y
606
Is processor Y's counter value greater than stored counter value ?
608
No
Currently on RHS?
610
Yes
Yes
No
614
Access page
Go to RHS; flush TLB; return to LHS
612
End

FIG. 6

PREVX1

GhostBusters 3

by Mihai G. Chiriac

Stealth malware has been around for ages: even the first known virus, Brain, used techniques to hide itself from the unsuspecting eye. The number of stealth viruses decreased dramatically from the days of DOS, where virtually every successful virus used some form of hiding technique, to Windows. However, malware with stealth capabilities is becoming increasingly common and dangerously enough it starts to climb the WildCore charts. This paper will quickly describe some of the techniques used by stealth malware, as well as the current detection approaches. After that, a new detection method is being introduced, described, analyzed, and, of course, put to the test.

Classification

Common malware use stealth technology to hide processes, files, registry entries, open ports, or a combination of the above. Technically speaking, malware programs manage to cover their tracks in a variety of ways - but depending on the detection difficulty, we make the following classification:

1. *Pointer modification* is one of the most common ways for hooking. APISPY32 by Matt Pietrek patches the import table of the host program (error-prone, mostly because programs can use dynamically resolved function addresses), while other hooking programs might patch the export table of the dynamic link library containing the specific function to be hooked (but this method will not hook already loaded programs). Pointer modification is used extensively in kernel mode, where the vast majority of rootkits simply patch the service dispatch table.

2. *Startup Code Patching* is also widely used. Most application-level malware insert a branching instruction, replacing the first bytes of the hijacked function. Since the x86 architecture supports variable size instructions, the hijacking program might contain a tiny disassembler (the method used by *Microsoft*'s Detours). Entry-level hooking programs simply save the first bytes and enable/disable the hook as they need (many problems regarding threads/synchronization may appear using this technique).

3. *Inline Code Patching* is used, so far, only in high-level rootkit programs. It involves complex code analysis of the target function and code injection.

4. *Filter Drivers* attach to a specific device using one of the IoAttachDevice* functions and "filter" any access to the device (and to all the underlying filter drivers).

5. *Direct Kernel Object Manipulation* (also known as DKOM). Programs using DKOM are among the most difficult to detect; for example, the function `ZwQuerySystemInformation` may walk the double-linked list of EPROCESS structures starting at `PsActiveProcessHead`, if called with the `SystemProcessesAndThreadsInformation` parameter. The *Fu* rootkit simply unlinks its process object from the list; it still gets CPU time because its thread objects are still in place (and scheduling is thread-based). As you noticed, I did not mention some hooking techniques, like patching directly the host's program code or the creation of a "proxy dll" both are too specific to be widely used.

Detecting Stealth Programs

Most detection algorithms involve string searching and pattern matching. This is not entirely a bad approach, since exact identification is important, but, of course, there are many disadvantages (new versions will be missed and security vendors must supply updates).

VICE, developed by James Butler and Greg Hoglund detects "hooks" by analyzing function addresses. If a function points outside the module, it is considered a suspected hook. If the function begins with a branching instruction, and the branch destination is outside the module, again, the function is marked as hooked. *VICE* detects many rootkits, but shows many false positives and, unfortunately, it is helpless against powerful malware.

Jan K. Rutkowski presented the *Execution Path Analysis* (EPA) technique. The EPA makes use of the single-stepping capabilities of IA-32 processors; by

©2005 Prevx.

setting the TF bit in the Eflags register and hooking Int 1, an analyzing program receives control after every instruction. If a function is hooked, of course, the execution path will be changed to the malware program's code. The PatchFinder program uses the EPA technique to count the CPU instructions needed to execute a function; then, a decision is made, upon comparing saved and actual values.

Therefore, we have on one hand a basic address/instruction analyzer; on the other, we have a single-stepping engine that counts instructions. The next step is obvious.

Function Emulation

In the autumn of 2003, I started working on a code emulation project, codenamed *Ktulu*, mainly to help me in my unpacking tasks. Since the beginning, I knew that I would use the emulator in more than just one project, so I tried to make it as modular as possible; the virtual CPU emulation library is based on the classics fetch / decode / emulate and imports only two functions from the memory manager: `MEM_Read()` and `MEM_Write()`. The memory manager is responsible of dividing the 4GB address space into blocks managed by virtual address descriptors; these descriptors, depending on an internal flag, may contain either a pointer to a memory area (filled with data) or a pointer to a specific function responsible of the whole "paging in" process. In *Ktulu*, I had three different "page in" functions, one for loading pages from the file, another for loading pages from my virtual kernel image and the third, a "zero page" provider. For function emulation, it is enough to write only one function (I called it `MEM_LoadPage`) responsible of reading a block of memory of a specific size, starting at a specific address, from the process we want to inspect.

MEM_LoadPage()

Since many stealth programs hook `ReadProcessMemory` (some of them hook `NtReadVirtualMemory`), our `MEM_LoadPage()` must work at the lowest possible level. This way, it should be virtually unhookable. At first, I wanted to go as close to the hardware as possible, but imagine the following scenario: one finds a specific page and attempts to read it; if the page is invalid, accessing it generates a page fault. If a malware program hooks the page fault handler, we may receive a modified "version" of the page. Therefore, I decided to use a simpler approach: I simply find the source process (using `PsLookupProcessByProcessId`) and attach to it (using `KeAttachProcess`), then, after some checks, use plain `RtlCopyMemory`.

The Call of Ktulu...

...is very easy: we simply initialize our virtual CPU, by setting up the instruction and stack pointers (the general registers are not really important) and, of course, the machine specific registers (we only need the `SYSENTER_EIP_MSR`, since the ring 0 stack pointer is initialized elsewhere). If we really want to be accurate, we initialize the debug registers too; this way we can emulate correctly the hardware breakpoints. At this point, our virtual memory is empty; when the processor needs a page, we will simply "read" it using `MEM_LoadPage()` and mark it as "present" the next accesses to that page will reference only our copy.

What Should We Emulate?

Most stealth programs hide their process; enumerating processes is a trivial task for applications and can be achieved in many ways: - The *toolhelp* functions:

- `CreateToolhelp32Snapshot`, `Process32First` and `Process32Next` represent a very popular way of "walking" the process list; `CreateToolhelp32Snapshot` simply allocates a chunk of memory and calls `NtQuerySystemInformation` (NtQSI); the `Process32First/Next` functions work only with that memory area.
- The *psapi* functions: `EnumProcesses`, `EnumProcessModules`, etc, finally call NtQSI
- NtQSI itself
- etc.

Basically, it all sums up to NtQSI (the *Task Manager* uses this function directly, so, under *NT* based systems, even if some program hooks `Process32Next`, it will still be visible in the task list).

©2005 Prevx

Hiding files is again very common in stealth programs; `NtVdmControl`, `NtQueryDirectoryFile` and `FindNextFile` seem to be targets for some malware authors. We just add emulation for these functions too. Another function we need is `DeleteFileA` (it will eventually call `DeleteFileW`, `NtOpenFile`, `NtQueryInformationFile`, `NtSetInformationFile`, etc.). Hiding registry keys seems a popular sport, so we should emulate functions like `NtEnumerateKey`.Gathering information

Let us try a practical approach: emulation of *CreateFileA*. We push the parameters on our virtual stack for this example, we try to open our own memory device driver, \\.\LLRead. We see how the control flow passes from kernel32 (section .text) to ntdll, function `RtlInitAnsiString`, then `RtlAnsiStringToUnicodeString`, `RtlMultiByteToUnicodeN`, and `CreateFileW`. Now we go through `RtlInitUnicodeString`, `RtlIsDosDeviceName_U`, then back to `RtlInitUnicodeString`. After 1745 emulated instructions and about a dozen of other functions, we arrive to ntdll.NtCreateFile, then to the "system call stub" in the "shared user data", then to "sysenter".

We enter ring 0, we see how the kernel copies the parameters to the kernel mode stack and how it initializes the EBX register (the instruction is a `MOV EBX, [EDI][EAX*04]`). The EDI register points to the service dispatch table and EAX is the function index. After a "`CALL EBX`" we end up inside NTOSKRNL, section "`PAGE`", inside the function `NtCreateFile` that simply calls `IoCreateFile`. This is where the fun begins. The execution continues in NTOSKRNL (and sometimes in HAL.DLL, for some low-level functions) until it reaches `IofCallDriver` with a simple `CALL [ESI][EAX*04][38]`, our own driver receives control (it receives the `IRP_MJ_CREATE` control code) and processes the request (see Table 1).

Table 1:

```
ntoskrnl.exe      [ .text]    | 0x804EC04B     CALL [ESI][EAX*04][38]
llread.sys        [ .text]    | 0xFCA4D2A0     PUSH ESI
llread.sys        [ .text]    | 0xFCA4D2A1     MOV ESI, [ESP][0C]
llread.sys        [ .text]    | 0xFCA4D2A5     XOR DL, DL
llread.sys        [ .text]    | 0xFCA4D2A7     MOV ECX, ESI
llread.sys        [ .text]    | 0xFCA4D2A9     AND [ESI][18], 00
llread.sys        [ .text]    | 0xFCA4D2AD     CALL [FCA4DC20]

// > Library ntoskrnl.exe, function IofCompleteRequest(),

ntoskrnl.exe      [ .text]    | 0x804EC051     JMP [8053E404]
ntoskrnl.exe      [ .text]    | 0x804EB3A5     PUSH EBP
```

After 8208 emulated instructions, 118 allocated memory pages, 18974 cache hits and 137633 cache misses, we have in EAX an open handle for our driver.

Obviously, we have an immense amount of data to process. The problem is what to do with all this data; we can store the number of emulated instructions, the number of memory accesses, even the number of ModR/M decodings. Of course, we store the code block information; for every code block, I decided to store the starting/ending addresses, owning module information (name, base, size of image), section name, number of parses, number of "children" and links to the children. When a code block gets executed for the first time, I try to do a "reverse GetProcAddress" and store this information too.

©2005 Prevx.

Depending on internal flags, the code block may be the "entry" block, a normal block, or the "exit" block (very helpful when analyzing the output in Wingraph32).

Filtering information

Even for relatively small functions, we can get huge amounts of information; not all of it is relevant. A simple call to `DbgPrint()` generates a 1.7 MB log file with completely useless information, so we can simply "return" when we reach `DbgPrint` (fortunately the function is `cdecl`). This is not an 100% safe optimization approach, if we are really paranoid, since some rootkits may find `DbgPrints` in legitimate drivers and hook `DbgPrint`; this way they will receive control and evade detection. Walking the "shortest" path in function emulation seems to be a good optimization approach; passing invalid parameters to some functions (like too small memory buffers, or invalid file handles) shortens the execution path considerably (but the price to pay is relatively high: an *inline patcher* may evade detection).

Heuristic Detection

Even if we do not have a "clean PC state" to compare with, we can attempt heuristic detection. Most stealth programs hijack normal function flow to their own memory areas: some of them allocate heap memory (it is very suspicious to have the control flow redirected to heap memory areas) others even make use of the wasted space (due to the section alignment) between two sections of a PE module. In application mode, the execution flow goes to kernel32 to ntdll (sometimes back to kernel32) and then to the system call stub. In kernel mode, most of the time, the execution flow goes from NTOSKRNL to HAL (the addresses vary, but generally they are between 0x80400000 and 0x80800000). Therefore, of course, if the emulation of `NtQuerySystemInformation` leads us to some high addresses (higher than 0xF0000000) we have reasons to worry.

Control Flow Graphs (CFG) and Graph Isomorphism

Most of the time, we have a saved, "clean PC state". Comparing the two graphs and spotting differences is not a trivial task; basically, given graphs A (the saved, clean CFG) and graph B (the CFG generated on a possibly infected computer) we need to detect subgraph C (or, possibly, subgraphs C0, C1, ...) so that B C = A (or B C0 C1 - ... = A). The simplest way is to check every code block in graph B for its "corresponding" in graph A; then, using the relations between these code blocks, heuristically build the (possibly malware) subgraph(s). Of course, we can use classic algorithms for subgraph detection.

A bigger problem arises when the saved CFG is created on a different computer; this way, we cannot use direct code block comparisons; I attempted to do as much code block recognition as possible. For example, if subgraph A1 gives control to subgraph A2 (let's say A1->A2), and A2->A3; B1 -> B2, and B2 -> B3; if we have a positive match between A1 and B1, and between A3 and B3, and only a partial match between A2 and B2, assume A2 = B2. This way, A1->A2->A3 = B1->B2->B3 (positive subgraph match) and we can search again for positive matches.

Removing Malware Subgraphs

For a vast majority of hooking programs it is very easy to isolate the "injected" subgraphs. For detection purposes, this is enough; for every injected subgraph, we write two log files, one containing the graph description in nice GDL format and another with the disassembly, register informations, etc. Very important (and interesting) for forensic analysis, *but*, we can do more. We can remove the hooks, at least in some cases.

For *pointer modification* malware, the disinfection is very easy; we can easily analyse the injected subgraph's "parent" code and patch the corresponding pointer table.

For *startup code patching* malware, the disinfection is a bit difficult; we need a (very complex) code analyzer or, we can simply store the first X bytes (0x10 bytes seems a reasonable size) and a checksum value for the original code. If by overwriting the first X bytes of the possibly infected code block with the saved bytes we get the same checksum, we re-check the subgraphs and hope (or pray?) they are identical.

Automated disinfection of *inline patching* or *filtering* stealth programs is very difficult; of course, it can be

©2005 Prevx.

achieved, up to some point, but maybe booting from a clean disk is a better approach.

Real-World Testing

Backdoor.Rebbew, also known as Webber, was one of the first wide-spread malware programs with stealth capabilities. It is capable of hiding its process under NT-based systems by hooking the function NtQuerySysteminformation from ntdll.dll. It injects a "JMP" at the beginning of NtQSI; the jump destination is an "empty" area, somewhere inside the memory image of ntdll; this empty area may be between the header (module base) and the first section or between two consecutive sections (wherever there is enough space). The "payload" is a 128-byte block of code that checks if NtQSI was called with the *SystemProcessesAndThreadsInformation* parameter and the *OutputBufferSize* parameter is larger than 0x1F40, if this is the case, the payload simply walks the list returned by NtQSI and removes its process block from the list. Since all of this is done in user mode (ntdll.dll is private for every process) *Rebbew* uses a trick to inject its payload in all the processes: it finds the physical pages where ntdll.dll resides (all unmodified instances of ntdll point to the same physical pages) and writes its code there.

The detection is very easy, even if we do not have a saved PC state. We see how the execution flows in the (presumably) empty space between the sections of ntdll, then a jump back to NtQSI's code, then back in the empty space; the removal is easy, if we have a saved state, we just re-write the first bytes, saved in our code block structure.

The same *startup code patching* technique is used in the "Hacker Defender" rootkit (execution is redirected to an allocated memory buffer) and Win32.Maddis.A (execution is redirected to a hidden module, "helper.dll"). Both malware programs pose no problems to *Ktulu*.

In kernel mode, since most rootkits use the *pointer modification* technique, we have no problems in detecting and removing them. For filter drivers the detection is easy; let us see an example: I installed on my test computer a copy of BitDefender, then *FileMon*, and I started the emulation of *DeleteFileA*. Of course, the emulation will go through *DeleteFileW*, etc. but also through all the registered filters. From *IofCallDriver* in ntoskrnl we end up in the first (actually, last) registered driver (see Table 2) then,

Table 2:

ntoskrnl.exe	[.text]	0x804EC049	PUSH EDX
ntoskrnl.exe	[.text]	0x804EC04A	PUSH ECX
ntoskrnl.exe	[.text]	0x804EC04B	PUSH ECX CALL [ESI][EAX*04][38]
FILEM.SYS	[.text]	0xF75DCCB6	MOV EAX, [ESP][04]
FILEM.SYS	[.text]	0xF75DCCBA	PUSH [ESP][08]
FILEM.SYS	[.text]	0xF75DCCBE	ECX, [EAX][28]
FILEM.SYS	[.text]	0xF75DCCC1	PUSH EAX

©2005 Prevx.

after the *FileMon* driver has finished its work, it will issue a call to *IofCallDriver* that is responsible of calling the next filter in list (see Table 3).

Table 3:

```
ntoskrnl.exe    [ .text]    | 0x804EC04B    CALL [ESI][EAX*04][38]
filespy.sys     [ .text]    | 0xF7972602    PUSH EBP
filespy.sys     [ .text]    | 0xF7972603    MOV EBP, ESP
filespy.sys     [ .text]    | 0xF7972605    SUB ESP, 44
filespy.sys     [ .text]    | 0xF7972608    MOV EAX, [EBP][08]
```

When BitDefender's filter driver (filespy) finishes its work, the system gives control to *sr.sys* (the System Restore filter driver) and then to the last filter in the chain (in our case, *FastFAT.sys*). Detecting malware filters is extremely easy, since the code block containing the *IofCallDriver* dispatcher will have more parses and more children. Disinfection *should* be possible; let's see a bit the code in *IofCallDriver* (see Table 4).

Table 4:

```
// > Library ntoskrnl.exe, function IofCallDriver(),
ntoskrnl.exe    [ .text]    | 0x804EC022    JMP [8053E400]
ntoskrnl.exe    [ .text]    | 0x804EC028    DEC [EDX][23]
ntoskrnl.exe    [ .text]    | 0x804EC02B    MOV AL, [EDX][23]
ntoskrnl.exe    [ .text]    | 0x804EC02E    TEST AL, AL
ntoskrnl.exe    [ .text]    | 0x804EC030    JLE $+00027144
...
ntoskrnl.exe    [ .text]    | 0x804EC049    PUSH EDX
ntoskrnl.exe    [ .text]    | 0x804EC04A    PUSH ECX
ntoskrnl.exe    [ .text]    | 0x804EC04B    CALL [ESI][EAX*04][38]
```

This code, of course, needs a deeper analysis, but we see that a driver is responsible of passing forward the pointers to all the "lower level" filters. Therefore, having valid "parent" and "child" drivers *might* help in the disinfection process: simply make the pointers "parent" filter jump directly to the "child". Of course, this is a very dangerous and error-prone technique, but it's interesting enough to worth mentioning.

Win32.*Mimail.R* (also known as W32/Opasa@mm or Win32.Mimail.V) opens the "PhysicalMemory" device and creates a call-gate; this way, it can execute code at the highest privilege level, code that removes Mimail's process object from the process list. It is very interesting to see how the virus gets some internal kernel variables, by disassembling NTOSKRNL's memory mapped image. Unfortunately, since Mimail.R does not hook functions and uses DKOM technology, our tool cannot detect it. The positive side is that enumerating processes is done easily (walking the internal structures used by the scheduler), and DKOM has a very limited usage.

©2005 Prevx.

Conclusions

Function emulation is a very powerful forensic analysis technique: Using it is of tremendous help in detecting any type of hooks (e.g. it is extremely interesting to emulate the CreateFileA function when the "DAEMON Tools" package is installed), not only malware. Even if the removal is impossible, the logs provide invaluable information for security analysts. The advantages over the classic EPA are obvious; besides the unending tricks usable for disabling int 1 tracing (remember *TbClean*?) one can imagine the following very probable- scenario: a heavy loaded computer might need extra-instructions for paging-in some of its code or data, so false alarms are likely to appear. In *Ktulu*, calling (for example) *NtQuerySystemInformation* produces the same CFG even when a different number of processes are currently running (this is true for most functions, especially when using the "shortest path" technique), though some code blocks may have a different number of parses; this is not the case for the classic EPA.

Even if this paper was aimed at *Windows NT/2000/XP* rootkit analysis and detection, function emulation works equally well on *Linux*. We only need to re-write the MEM_LoadPage() function to make it work; then, to gather data, we need to write a "reverse `GetProcAddress`" for *Linux* modules. A *Linux* version of *Ktulu* is currently under development and I expect very good results.

September 2005

References:

Peter Szor, "Memory Scanning Under Windows NT" (1999)
Jamie Butler, "Direct Kernel Object Manipulation" (2003)
Halvar Flake, "More fun with graphs" (2003)
Jan K. Rutkowski, "Advanced Windows 2000 Rootkit Detection" (2003)
Gergely Erdelyi, "Anatomy of stealth malware" (2004)
BitDefender Virus Encyclopedia, "Backdoor.Rebbew" description (2004)
Galen Hunt, Doug Brubacher, "Detours: Binary Interception of Win32 Functions (1999)
James Butler, Greg Hoglund, "VICE" help files (2004)

©2005 Prevx.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/694,261	03/30/2007	MIHAI CHIRIAC	339000-P0004

CONFIRMATION NO. 1684

47604
DLA PIPER US LLP
P. O. BOX 9271
RESTON, VA20195

Date Mailed. 02/14/2008

NOTICE OF NEW OR REVISED PROJECTED PUBLICATION DATE

The above-identified application has a new or revised projected publication date. The current projected publication date for this application is CPUB. If this is a new projected publication date (there was no previous projected publication date), the application has been cleared by Licensing & Review or a secrecy order has been rescinded and the application is now in the publication queue.

If this is a revised projected publication date (one that is different from a previously communicated projected publication date), the publication date has been revised due to processing delays in the USPTO or the abandonment and subsequent revival of an application. The application is anticipated to be published on a date that is more than six weeks different from the originally-projected publication date.

More detailed publication information is available through the private side of Patent Application Information Retrieval (PAIR) System. The direct link to access PAIR is currently http://pair.uspto.gov. Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Questions relating to this Notice should be directed to the Office of Patent Publication at 1-888-786-0101.

PART 1 - ATTORNEY/APPLICANT COPY

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/694,261	03/30/2007	MIHAI CHIRIAC	339000-P0004

CONFIRMATION NO. 1684

47604
DLA PIPER US LLP
P. O. BOX 9271
RESTON, VA20195

Title: METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

Publication No. US-2008-0040710-A1
Publication Date: 02/14/2008

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Pre-Grant Publication Division, 703-605-4283

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
11/694,261	03/30/2007	2193	2750	339000-P0004	47	5

CONFIRMATION NO. 1684

47604
DLA PIPER US LLP
P. O. BOX 9271
RESTON, VA20195

FILING RECEIPT

Date Mailed: 04/12/2007

Receipt is acknowledged of this regular Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please mail to the Commissioner for Patents P.O. Box 1450 Alexandria Va 22313-1450. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).**

Applicant(s)

MIHAI CHIRIAC, Bucharest, ROMANIA;

Assignment For Published Patent Application

PREVX LIMITED, PRIDE PARK, UNITED KINGDOM

Power of Attorney: The patent practitioners associated with Customer Number 47604

Domestic Priority data as claimed by applicant

This appln claims benefit of 60/789,156 04/05/2006

Foreign Applications

If Required, Foreign Filing License Granted: 04/11/2007

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US11/694,261**

Projected Publication Date: 10/11/2007

Non-Publication Request: No

Early Publication Request: No

Title

METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

Preliminary Class

717

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the

subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

Electronic Patent Application Fee Transmittal

Application Number:	11694261
Filing Date:	
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Filer:	Dale S. Lazar
Attorney Docket Number:	339000-P0004

Filed as Large Entity

Utility Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Independent claims in excess of 3	1201	1	200	200
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**200**

Electronic Acknowledgement Receipt

EFS ID:	1660024
Application Number:	11694261
International Application Number:	
Confirmation Number:	1684
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Customer Number:	47604
Filer:	Dale S. Lazar
Filer Authorized By:	
Attorney Docket Number:	339000-P0004
Receipt Date:	06-APR-2007
Filing Date:	
Time Stamp:	09:30:29
Application Type:	Utility

Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	$200
RAM confirmation Number	1482
Deposit Account	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)

1	Fee Worksheet (PTO-06)	fee-info.pdf	8167	no	2
Warnings:					
Information:					
			Total Files Size (in bytes):	8167	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UTILITY PATENT APPLICATION TRANSMITTAL (Only for new nonprovisional applications under 37 CFR 1.53(b))	Attorney Docket No.	339000-P0004
	First Inventor or Application Identifier	Mihai CHIRIAC
	Title	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
PO Box 1450
Alexandria, VA 22313-1450

1. ☐ Fee Transmittal Form (e.g. PTO/SB/17)
(Submit an original and a duplicate for fee processing)
☐ Applicant claims small entity status.
2. ☒ Specification — Total Pages 44
3. ☒ Drawing(s) *(35 U.S.C. 113)* — Total Sheets 2
4. ☒ Oath or Declaration — Total Pages 1
   a. ☒ Newly executed (original or copy)
   b. ☐ Copy from a prior application *(37 C.F.R. §1.63(d))*
   (for continuation/divisional with box 16 completed)
5. ☐ Incorporation By Reference (*usable if box 4B is checked*)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4B, is considered to be part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

ACCOMPANYING DOCUMENTS
6. ☐ Assignment Papers (cover sheet & documents(s))
7. ☐ 37 C.F.R. §3.73(b) Statement ☐ Power of Attorney
(when there is an assignee)
8. ☐ English Translation Document *(if applicable)*
9. ☒ Information Disclosure Statement (IDS)/PTO-1449 ☒ Copies of IDS Citations
10. ☐ Preliminary Amendment
11. ☐ White Advance Serial No. Postcard
12. ☐ Certified Copy of Priority Document(s)
(if foreign priority is claimed)
13. ☐ Request for Priority
14. ☒ Application Data Sheet
15. ☐ Other:

16. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below:
☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application no.:
Prior application information: Examiner: Group Art Unit:

17.
☐ This application is a ☐ Continuation ☐ Division ☐ Continuation-in-part (CIP)
of application Serial No. Filed on
☒ This application claims priority of provisional application Serial No. 60/789,156, Filed April 5, 2006

18. CORRESPONDENCE ADDRESS

Customer No. 47604

Name	Dale S. Lazar	Registration No.	28,872		
Signature		Date	March 30, 2007	Telephone	703.773.4149

REST1\593685.1

APPLICATION DATA SHEET

Application Information

Application Number::	NEW
Filing Date::	3/30/07
Application Type::	REGULAR
Subject Matter::	UTILITY
Suggested Classification::	
Suggested Group Art Unit::	
CD-ROM or CD-R::	NONE
Number of Disks::	
Number of copies of CDs::	
Sequence Submission::	
Computer Readable Form (CRF)::	NO
Number of Copies of CRF::	
Title::	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
Attorney Docket Number::	339000-P0004
Request for Early Publication::	NO
Request for Non-Publication::	NO
Suggested Drawing Figure::	
Total Drawing Sheets::	2
Small Entity::	NO
Latin Name::	
Variety Denomination Name::	
Petition Included::	NO
Petition Type::	
Licensed US Govt. Agency::	
Contract of Grant Numbers::	
Secrecy Order in Parent Appln.::	NO

INITIAL
3/30/2007

APPLICANT INFORMATION

Applicant Authority Type::	INVENTOR
Primary Citizenship Country::	ROMANIA
Status::	FULL CAPACITY
Given Name::	MIHAI
Middle Name::	
Family Name::	CHIRIAC
Name Suffix::	
City of Residence::	BUCHAREST
State or Province of Residence::	
Country of Residence::	ROMANIA
Street of mailing address::	STR. ALEEA ZORELELOR, NR.3, BI 44B, AP. 122
City of mailing address::	BUCHAREST
State or Province of mailing address::	
Country of mailing address::	ROMANIA
Postal or Zip Code of mailing address::	061184

CORRESPONDENCE INFORMATION

CORRESPONDENCE NUMBER::	47604
PHONE NUMBER::	703-773-4149
FAX NUMBER::	703-773-5200
E-MAIL ADDRESS::	DALE.LAZAR@DLAPIPER.COM

REPRESENTATIVE INFORMATION

INITIAL
3/30/2007

Representative Customer Number:: 47604

DOMESTIC PRIORITY INFORMATION

Application::	Continuity Type::	Parent Application::	Parent Filing Date::
This Application	An application claiming the benefit under 35 USC 119(e)	60/789,156	April 5,2006

FOREIGN PRIORITY INFORMATION

Country::	Application Number::	Filing Date::	Priority Claimed

ASSIGNEE INFORMATION

Assignee Name::	PREVX LIMITED
Street mailing address::	3 NEW PARK PLACE
City mailing address::	PRIDE PARK
State or Province of mailing address::	DERBY
Country of mailing address::	UNITED KINGDOM
Postal or Zip Code of mailing address::	DE24 8DZ

INITIAL
3/30/2007

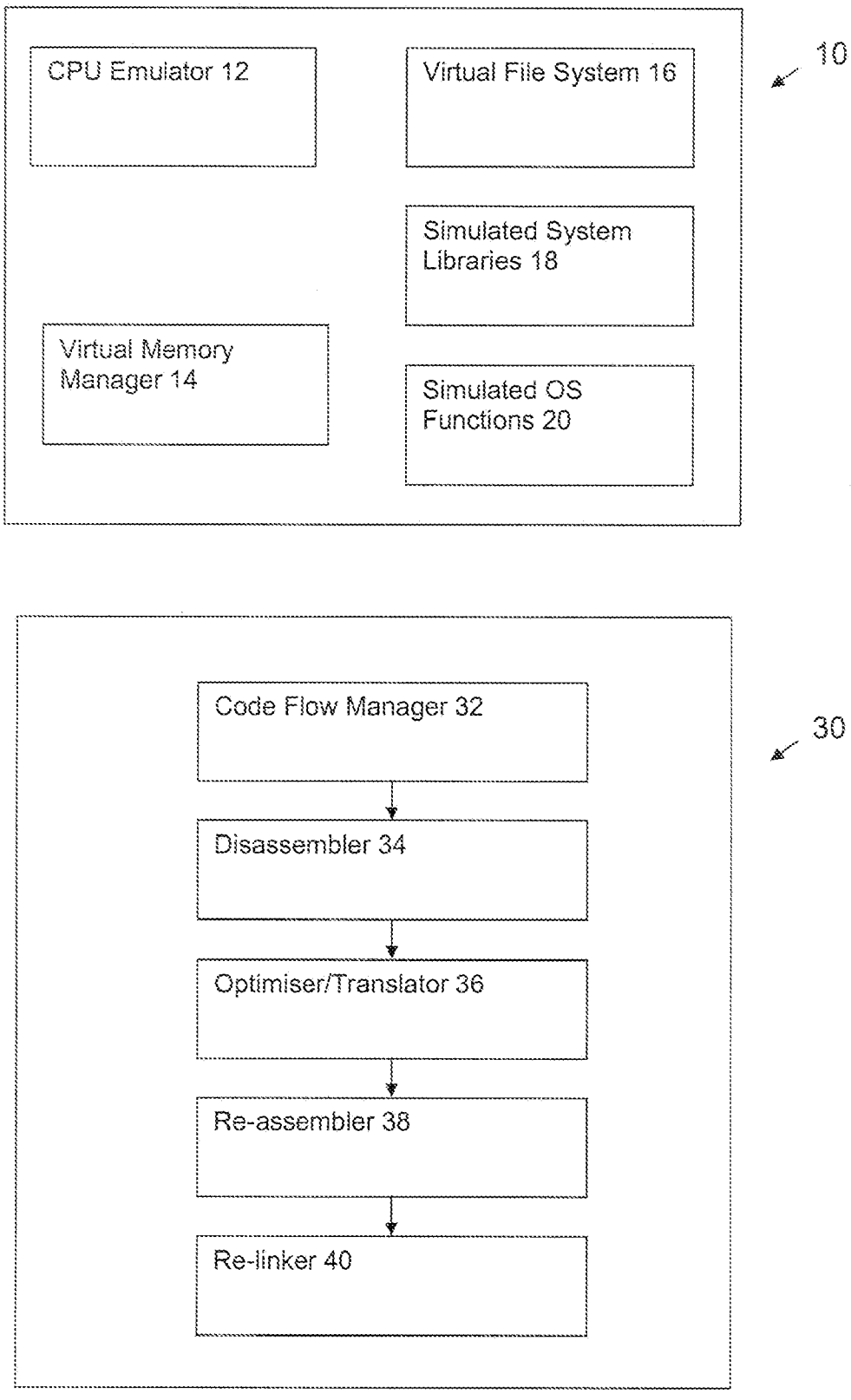

CPU Emulator 12
Virtual File System 16
10
Simulated System Libraries 18
Virtual Memory Manager 14
Simulated OS Functions 20
Code Flow Manager 32
30
Disassembler 34
Optimiser/Translator 36
Re-assembler 38
Re-linker 40

Fig. 1

2/2

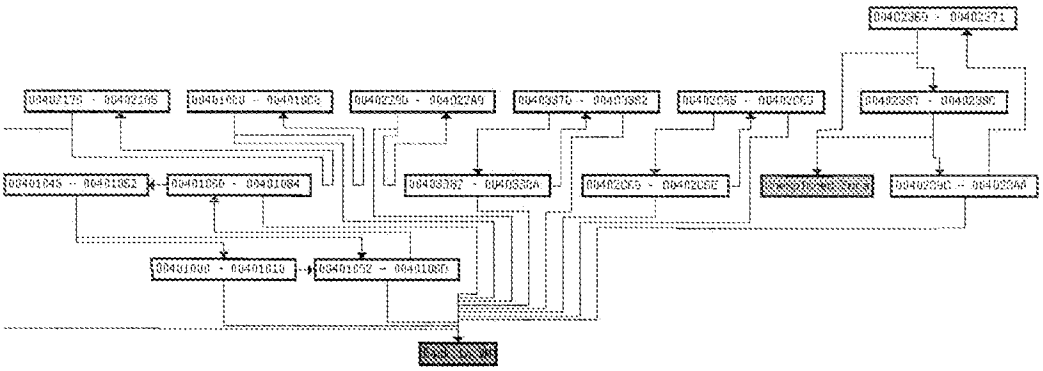


Fig. 2

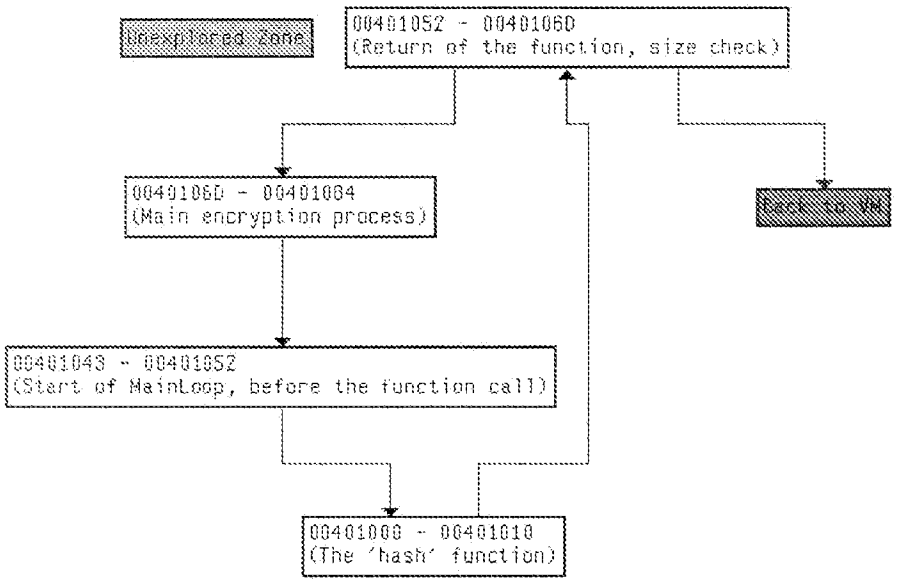

Unexplored zone
00401052 - 0040106D
(Return of the function, size check)
0040106D - 00401084
(Main encryption process)
00401043 - 00401052
(Start of MainLoop, before the function call)
00401000 - 00401010
(The 'hash' function)

Fig. 3

ALL PATENTS, INCLUDING DESIGN FOR APPLICATION BASED ON PCT, PARIS CONVENTION, NON PRIORITY, OR PROVISIONAL APPLICATIONS

COMBINED DECLARATION AND POWER OF ATTORNEY U.S.A.

FOR ATTORNEYS' USE ONLY

ATTORNEYS' DOCKET NO. 339000-P0004

As a below named inventor, I declare that my residence, post office address and citizenship are stated below next to my name, the information given herein is true, that I believe that I am the original, first and sole inventor (if only one name is listed at 101 below), or an original, first and joint inventor (if plural inventors are named below at 101-103, or on additional sheets attached hereto) of the subject matter which is claimed and for which patent is sought on the invention entitled:

Title DYNAMIC CODE TRANSLATOR

which is described and claimed in:

- ☐ PCT International Application No. , filed , (if applicable) and amended on
- ☒ the attached specification
- ☐ the specification in application Serial No. , filed

I (We) hereby state that I (We) have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I (We) acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I (We) hereby claim foreign priority benefits under Title 35, United States Code, §119 (a)-(d) or §365 (b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT international application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate or §365(b) having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s)			Priority Claimed	
(Number)	(Country)	(Day/Month/Year Filed)	☐ Yes	☐ No
(Number)	(Country)	(Day/Month/Year Filed)	☐ Yes	☐ No
(Number)	(Country)	(Day/Month/Year Filed)	☐ Yes	☐ No

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Application No. 60/789,156	Filing Date April 5, 2006	Application No.	Filing Date

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s), or §365 (c) of any PCT international application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT international application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR §1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application.

(Application Serial No.)	(Filing Date)	(Status: patented, pending, abandoned)

POWER OF ATTORNEY: I (We) hereby appoint DLA Piper US LLP, Intellectual Property Group, telephone number (720) 773-4000 (to whom all communications are to be directed), and persons of that firm who as associated with USPTO Customer No.

47604

individually and collectively my attorneys, with full powers of substitution and revocation, to prosecute this application and to transact all business in the USPTO connected therewith.

Inventor(s) name must include at least one unabbreviated first or middle name.

101 FULL NAME OF INVENTOR	FAMILY NAME: CHIRIAC	GIVEN NAME: Mihai	MIDDLE NAME:	
RESIDENCE & CITIZENSHIP	CITY: Bucharest	STATE OR FOREIGN COUNTRY: Romania	COUNTRY OF CITIZENSHIP:	
POST OFFICE ADDRESS	POST OFFICE ADDRESS: Str. Aleea Zorelelor, nr. 3, Bl 44B, Ap. 122	CITY: Bucharest	STATE OR COUNTRY: Romania	ZIP CODE: 061184
102 FULL NAME OF INVENTOR	FAMILY NAME:	GIVEN NAME:	MIDDLE NAME:	
RESIDENCE & CITIZENSHIP	CITY:	STATE OR FOREIGN COUNTRY:	COUNTRY OF CITIZENSHIP:	
POST OFFICE ADDRESS	POST OFFICE ADDRESS:	CITY:	STATE OR COUNTRY:	ZIP CODE:
103 FULL NAME OF INVENTOR	FAMILY NAME:	GIVEN NAME:	MIDDLE NAME:	
RESIDENCE & CITIZENSHIP	CITY:	STATE OR FOREIGN COUNTRY:	COUNTRY OF CITIZENSHIP:	
POST OFFICE ADDRESS	POST OFFICE ADDRESS:	CITY:	STATE OR COUNTRY:	ZIP CODE:

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

SIGNATURE OF INVENTOR 101*	SIGNATURE OF INVENTOR 102*	SIGNATURE OF INVENTOR 103*
DATE 29.03.2007	DATE	DATE

☐ Additional inventors are named on separately numbered sheets attached hereto

DLA Piper US LLP
P.O. Box 9271, Reston, VA 20195

Docket No.: 339000-P0004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Mihai CHIRIAC	Conf. No.	Unknown
SERIAL NUMBER:	New Application	Group:	Unknown
FILED:	March 30, 2007	Examiner:	Unknown

FOR: METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

March 30, 2007

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry)not more than three months prior to the filing of this statement.

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application or, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement.

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

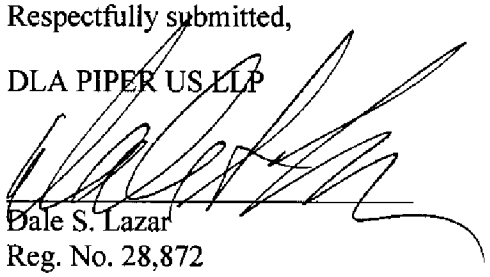
- ☒ without fee and within three months of the filing date of the application.
- ☐ without fee and within three months of the date of entry of the U.S. national stage.
- ☐ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
- ☐ without fee and with the appropriate certification above.
- ☐ without fee and with a new CPA application.
- ☐ without fee and with a Request for Continued Examination.
- ☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
- ☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith and for which no check is enclosed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER US LLP



Dale S. Lazar
Reg. No. 28,872
Attorney of Record

P.O. Box 9271
Reston, VA 20195
Tel. (703) 773-4000
Fax (703) 773-5200

DSL/jsk

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	1	of	2

Complete if Known	
Application Number	New Application
Filing Date	March 30, 2007
First Named Inventor	Mihai CHIRIAC
Art Unit	Unknown
Examiner Name	Unknown
Attorney Docket Number	339000-P0004

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -7013483	03-14-2006	ALADDIN KNOWLEDGE SYSTEMS LTD	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 96/30829	10-03-1996	APPLE COMPUTER, INC.		
		WO 99/15966	04-01-1999	SYMANTEC CORPORATION		
		EP 1536341 A2	06-01-2005	MICROSOFT CORPORATION		

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional). 2 See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. 3 Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). 4 For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. 5 Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. 6 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer,

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.
U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND **TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Substitute for form 1449A/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*				***Complete if Known***	
				Application Number	New Application
				Filing Date	March 30, 2007
				First Named Inventor	Mihai CHIRIAC
				Art Unit	Unknown
				Examiner Name	Unknown
Sheet	2	of	2	*Attorney Docket Number*	339000-P0004

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		"DEFEATING POLYMORPHISM: BEYOND EMULATION, Adrian E. STEPAN	
		"GHOSTBUSTER 3", Mihai CHIRIAC	

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
[1] Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

US007013483B2

(12) **United States Patent**
Cohen et al.

(10) **Patent No.: US 7,013,483 B2**
(45) **Date of Patent: Mar. 14, 2006**

(54) **METHOD FOR EMULATING AN EXECUTABLE CODE IN ORDER TO DETECT MALICIOUSNESS**

(75) Inventors: **Oded Cohen**, Tivon (IL); **Inbal Meir**, Haifa (IL); **Yanki Margalit**, Ramat-Gan (IL); **Dany Margalit**, Ramat-Gan (IL)

(73) Assignee: **Aladdin Knowledge Systems Ltd.**, Tel Aviv (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 286 days.

(21) Appl. No.: **10/335,871**

(22) Filed: **Jan. 3, 2003**

(65) **Prior Publication Data**

US 2004/0133796 A1 Jul. 8, 2004

(51) **Int. Cl.**
H04L 713/20 (2006.01)

(52) **U.S. Cl.** **726/25**; 726/23; 726/24; 713/165; 713/167; 713/188

(58) **Field of Classification Search** 713/188, 713/200–201, 167, 165; 380/4, 3
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,398,196	A	* 3/1995	Chambers	714/28
5,842,002	A	* 11/1998	Schnurer et al.	703/21
5,978,917	A	* 11/1999	Chi	713/201
6,035,423	A	3/2000	Hodges et al.	
6,067,410	A	* 5/2000	Nachenberg	703/28
6,269,456	B1	7/2001	Hodges et al.	
6,357,008	B1	* 3/2002	Nachenberg	713/200
2002/0013910	A1	1/2002	Edery et al.	
2002/0073330	A1	6/2002	Chandnani et al.	

FOREIGN PATENT DOCUMENTS

WO	WO01/88673	11/2001
WO	WO02/37740	5/2002

OTHER PUBLICATIONS

http://livedocs.macromedia.com/colfusion/6/Developing_coldFusion_MX_Applications_with_CFML/queryDB6.htm.*
Pleszkoch et al., "Improving Network System with Function Extraction Technology for Automated Calculation of Program Behavior", Proceeding of the 37th Hawaii International Conference on System Science, 2004.*
http://www.cgisecurity.com/owasp/html/ch11s04.*

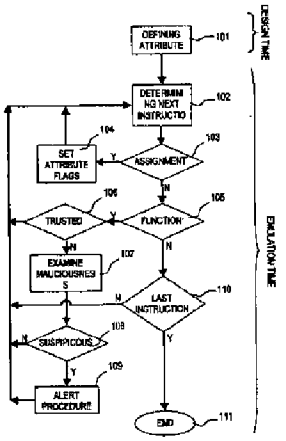
* cited by examiner

Primary Examiner—Gregory Morse
Assistant Examiner—Tongoc Tran
(74) *Attorney, Agent, or Firm*—Mark M. Friedman

(57) **ABSTRACT**

The present invention is directed to a method for emulating an executable code, whether it is a human-readable code (e.g., macro and script) or a compiled code (e.g. Windows executable). At the design time, one or more content attributes are defined for the variables of the code. A content attribute indicates a property with relevance to maliciousness, e.g. Windows directory, a random value, ".EXE" at the right of a string, etc. A content attribute may be implemented, for example, by a flag. Also defined at the design time, is a list of malicious states, where a malicious state comprises at least the combination of a call to a certain system function with certain content, as the calling parameter(s). When emulating an assignment instruction, the attributes of the assigned variable are set according to the assigned content. When emulating a mathematical operator, a content mathematics is also applied. When emulating a function call, the current state (i.e. the function identity and the calling content and values) is compared with the predefined malicious states, and if at least one malicious state corresponds, then the maliciousness of the code is determined.

24 Claims, 1 Drawing Sheet



www.freepatentsonline.com

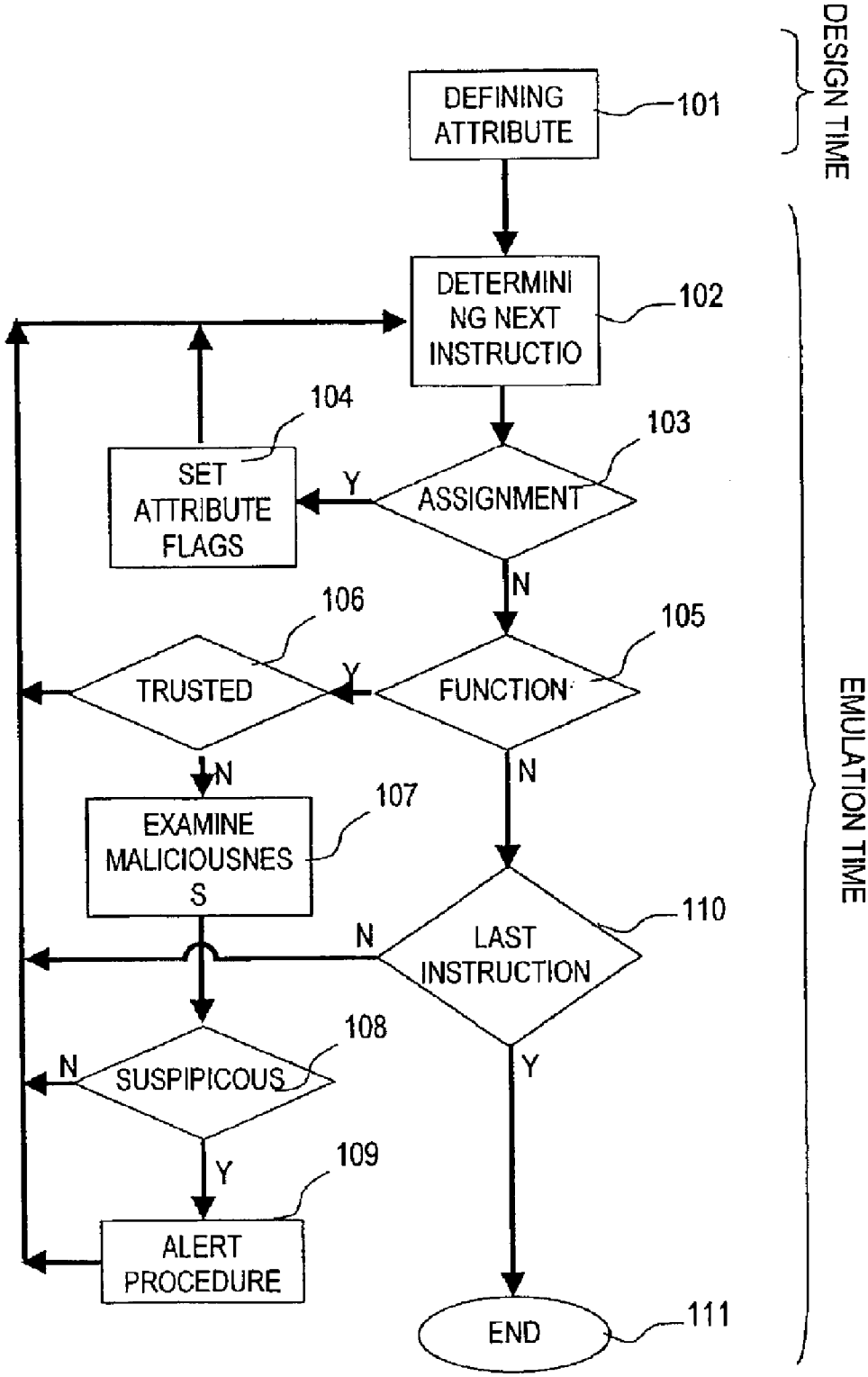


Fig. 1

www.freepatentsonline.com

METHOD FOR EMULATING AN EXECUTABLE CODE IN ORDER TO DETECT MALICIOUSNESS

FIELD OF THE INVENTION

The present invention relates to the field of detecting presence of virus and other kinds of malicious code by emulation.

BACKGROUND OF THE INVENTION

Emulation is one of the methods of detecting viral activity. The emulator creates a virtual environment under which the code of the tested sample is examined instruction-by-instruction. The viral activity is detected by correlating the changes with a viral behavior.

Emulation has several advantages in comparison to other methods of detecting and/or preventing viral damages, since it helps to detect polymorphic viruses, encrypted viruses and unknown viruses, which cannot be detected by other methods.

A programmer can indicate the maliciousness of a source code just by reviewing the code. For example, if he identifies a loop that gets the name of the files of a directory and deletes them, he can assume that the code is malicious. Contrary to a human programmer, an emulator, which is a machine, cannot indicate viral code just by reviewing the code, but it has to execute the code while comparing the caused changes with known patterns. In other words, the emulator, being a machine, is absent of human intelligence.

It is an object of the present invention to provide a method for emulating an executable code in order to detect maliciousness, which is more "intelligent" than known in the prior art.

Other objects and advantages of the invention will become apparent as the description proceeds.

SUMMARY OF THE INVENTION

The present invention is directed to a method for emulating an executable code, whether it is a human-readable code (e.g., macro and script) or a compiled code (e.g. Windows executable). At the design time, one or more content attributes are defined for the variables of the code. A content attribute indicates a property with relevance to maliciousness, e.g. Windows directory, a random value, ".EXE" at the right of a string, etc. A content attribute may be implemented, for example, by a flag. Also defined at the design time, is a list of malicious states, where a malicious state comprises at least the combination of a call to a certain system function with certain content, as the calling parameter(s). When emulating an assignment instruction, the attributes of the assigned variable are set according to the assigned content. When emulating a mathematical operator, a content mathematics is also applied. When emulating a function call, the current state (i.e. the function identity and the calling content and values) is compared with the predefined malicious states, and if at least one malicious state corresponds, then the maliciousness of the code is determined.

The system function can be, for example, a part of the programming language of the executable, a function provided by the operating system under which the code is to be executed, a function provided by a linked package of routines.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood in conjunction with the following figure:

FIG. 1 schematically illustrates a method for emulating an executable, in order to detect malicious code.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The term Macro in conjunction with computers, refers herein to an executable code embedded within another entity, like a document or executable code.

The term Script in conjunction with computers, refers in the art to a program built into the HTML code of a Web page that work on almost all Internet applications. Scripts can also be used with a variety of applications that support their scripting syntax. VBScript and JavaScript are examples of scripting languages.

Macros and scripts are "non-compiled executables", i.e. programs wherein their executable code is the source code, contrary to "compiled executables" which are converted to machine-code before execution. ASP and XML are other examples of non-compiled executables.

Due to their human readability, non-compiled executables are exploited by vandals (malicious executables written into scripts and macros). They can do almost anything—infect a system with Trojans, modify files, cause DoS attacks, or spread malicious code using Microsoft Networks or Microsoft Outlook. Macro-viruses that attack Microsoft Word work by copying themselves into the global environment—usually by modifying NORMAL.DOT. Unfortunately, the enhanced power of software applications also gives virus writers the potential to cause more damage.

It should be noted that the examples herein are presented in a human-readable programming code, although they apply to both compiled and non-compiled code. Moreover, the examples are presented in a schematic programming language, rather than an existing programming language.

There are some programming tools and computer resources commonly used by malicious code. For example, the random generator, which is a programming tool provided by almost any operating system and almost any programming language, is commonly used by viruses to prevent the repetition of the same code when duplicating itself, thereby hardening the detection of the malicious code by bit-pattern (also known as "virus signature") scanning. A human-readable code, as a program written in a programming language, makes use of variables. Since it is plain text, the mechanism that duplicates the malicious code can generate random names for its variables, to add non-effective code comments between its effective code lines, etc. The non-effective code may also be generated randomly, thus using the random generator.

The real time clock, which is a programming-tool provided by any operating system and almost any programming language, is another example of a virus-commonly used programming tool, since it enables a late-trigger of the virus code, and also because it can be used as a substitute to the random generator. Moreover, some of the viruses have a late trigger, thereby enabling them to be widely propagated before causing the damage, and consequently their discovery. The late trigger is tuned to a future date, which involves use of the real time clock of the host machine.

Another example of a content used by malicious code in Windows operating system is the Windows folder, which stores Windows system files. A virus may try to change one of the existing files in the Windows folder, create a file with a random name, etc.

A combination of some of the contents described above may also indicate the presence of malicious code within an

www.freepatentsonline.com

executable. For example, creating a new file having a random name in the Windows directory can be due to a virus. However, a more accurate indication of a malicious code can be obtained if this is an executable file. Thus, if the created file has an EXE extension, which is the extension of Windows executables, or VBS, which is the extension of Visual Basic Script, the executable can be classified as malicious.

According to a preferred embodiment of the present invention, a "malicious state" comprises at least executing a certain system function with certain content as its parameter (s). Thus, at least two elements are involved—a certain system function and certain content as the parameter(s).

It should be noted that the term "content" of a variable refers herein also to its value. Nevertheless, in some cases the value has been also mentioned.

A state can be indicated as malicious if it corresponds to at least one pre-defined malicious state. Two states "correspond" if their elements (i.e. function identity and the calling content) are equal or at least "similar (e.g., at least certain percentage of their content/value is equal). Maliciousness can be defined also by tracking the history of the called functions, e.g., if certain functions have been called with certain content, or even without regard-ness to their content.

Identifying a call to a certain function is quite obvious, and almost any compiler parser performs it. However, the concept of dealing with the content of a parameter rather than its value is novel.

For example, the parameter when calling the following function is a random number:

```
MyVariable=Random( )

MyVariable=MyVariable+1

Call MyFunction(My_Variable)
```

Where Random( ) is a random number generator.

The first time MyFunction is performed, its parameter may be, e.g., 0.443213, and the next time it can be, e.g., 0.332455. Thus, from analyzing the text it is not possible to indicate that the parameter has a random nature.

According to a preferred embodiment of the invention, the content of a variable can be indicated as follows:

Each variable of the executable is characterized by several attributes (e.g. flags), which points out on its content (referred herein as content attributes). For example:

- a flag to indicate if the variable has been assigned with the Windows folder name;
- a flag to indicate if the variable has been assigned with a random value;
- a flag to indicate if the variable has been assigned with the real time;
- a flag to indicate if the variable has been assigned with the value ".EXE", which is the extension of a type of executable file under the Windows operating system;

and so forth.

While emulating the execution of an executable, the following instructions (for example) are performed:

```
File_name=Windows_directory( )+Random( )+0.2+".EXE"

Create_file(File_name)
```

Where: Windows_directory( ) is a system function that returns a string of the Windows directory, and Random returns a numeric random integer.

In this example, the Windows folder flag, the Random flag and the EXE flag of the variable File_name are turned on.

It should be noted that

```
File_name=Windows_directory( )+Random( )+0.2+".EXE"
```

actually comprises several instructions:

Executing the Random function;

Executing the Windows_directory function;

Mathematical addition;

String addition.

It should also be noted that while some programming languages convert a numeric value to a string value and vice versa when it is required, some compilers have no such ability. Thus, assuming the function Str converts a numeric value to a string, in this case the example should look like:

```
File_name=Windows_directory( )+Str(Random( )+0.2)+".EXE"
```

There is no point in considering assigning a value to variables as malicious operation; however without tracking the history of the assignments to a variable, its content cannot be determined. Thus, the examining stage should be at the point where specific system functions which may cause damage to the computer are called, e.g. a function that creates a file, a function that changes the content of a file, and so forth.

The following instruction also creates an executable file with a random name in the Windows directory:

```
Create_file(Windows_directory( )+Random( )+".EXE")
```

However, instead of using a variable as the parameter of the function Create_file, the name of the file is the parameter. Actually, this is equivalent to creating a temporary variable, and assigning a value to the temporary variable just before the execution of said function:

```
Temp_parameter=Windows_directory( )+Random( )+".EXE"

Call Create_file(Temp_parameter)
```

As mentioned above, according to the present invention, the variables comprise content attributes in addition to the value. Thus, there is a question as to how the content is influenced by mathematical operations. For example:

```
MyValue=123+First_var+Second_var
```

Where, for example, the Real time flag of the variable First_var is turned on and the EXE flag of the variable Second_var is turned on. It is obvious that in this case the content result of the assignment is turning on the Real time flag and the EXE flag of the variable MyValue. Thus, the emulator should be provided with "content mathematics" rules. Such rules, which for example comprise a commutative law of addition, an associative law of addition and so forth, should also take into consideration comparison statements and other features supported by the emulated programming language. Of course it is desired to emulate the values as well, in addition to emulating the content.

Another example:

```
Var_A=GetAddressBook( )

Var_B=Var_A->Count( )
```

Where Var_A has the content type "address book object".

When the operator "->Count( )" is operated on an "address book object", it returns not just a numeric value,

www.freepatentsonline.com

but a numeric value with the content attribute "number of entries in address book", thus adding the "number of entries in address book" to the variable Var_B.

Another example:

```
Var_A=GetAddressBook( )

Var_C=Var_A[2]
```

After the assignment to Var_C, its value will be the second entry of the address book which has been assigned to Var_A. The content attributes of Var_A may be something like "address book database", and the content attribute of Var_C may be something like "address book entry".

FIG. **1** schematically illustrates a method for emulating an executable, in order to detect malicious code.

At step **101**, which is the design step, a group of malicious states is defined. A malicious state is the combination of a call to a certain system function with certain content. This is a preliminary stage, which is performed at the anti-virus firm, as a part of the anti-virus emulator design. Of course the list of the content attributes and/or functions can reside on a file, and modified later (whether at the anti-virus firm residence or at the installation site).

At the next step **102**, which is performed at the emulation time (i.e. run time), the next instruction to be performed is determined.

At the next step **103**, the type of the instruction is examined. If it is an assignment instruction, then the flow continues with step **104**, otherwise the flow continues with step **105**.

At step **104**, the instruction is analyzed in order to determine the content attribute(s) of the assigned variable to be set, and the corresponding values are set, and from there, the flow returns to step **102**.

At step **105**, if the type of the instruction is a function call, then the flow continues with step **106**; otherwise the flow continues with step **110**.

From step **106**, if the function is pre-determined as trusted, then the flow continues to step **102**; otherwise flow continues to step **107**.

At step **107** the maliciousness is examined by comparing the current state (i.e. the calling function with the content attributes of the parameters) with the malicious states that have been pre-defined in step **101**. Determining the maliciousness of a code can be indicated if the current state corresponds to at least one of the pre-defined malicious states.

From step **108**, if the instruction seems to be suspicious according to the examination carried out at step **107**, then the flow continues with step **109**, where an alert procedure is performed, and from there to step **102**.

From step **110**, if the last tested instruction is the last instruction (e.g. a Stop instruction), then the emulation stops, otherwise the flow continues with step **102**.

Usually, the emulation continues until the last instruction is performed, and then the results of the emulation are examined. This way, more details are gathered and the maliciousness of the code can be determined more precisely. However, the alert procedure can suspend further execution, thereby shortening the emulation time.

As known to a person of ordinary skill in the art, a function can be declared such that a parameter is transferred by its address or by its value. In case of transferring a value, a temporary parameter is created and its address is transferred as a parameter, like in the other case. In both cases when emulating the code, the variables, whether permanent or temporary, also comprise the content attributes.

It should be noted that the term "function" also refers herein to procedures, software interrupts and so forth.

The emulation can be executed at the user's machine, at the gateway to a network and so forth. For example, while a user may install an emulator to check every incoming HTML file while browsing the Internet/Intranet, an organization may install such an emulator at the gateway to its network, at the mail server and so forth.

Those skilled in the art will appreciate that the invention can be embodied by other forms and ways, without losing the scope of the invention. The embodiments described herein should be considered as illustrative and not restrictive.

What is claimed is:

1. A method for emulating an executable code, said method comprising:

pre-defining at least one content attribute having relevance to maliciousness, for at least one variable of said executable code;

pre-defining at least one malicious state, being a call to a certain system function with certain content and/or value;

upon emulating an assignment instruction, setting the content attributes of the assigned variable according to the assigned content, and the value of the assigned variable;

upon emulating a mathematical operator, employing content mathematics; and upon emulating a function call, checking that the current state, being the combination of the function identity with the content and/or value of the calling parameters, corresponds to at least one of the pre-defined malicious states, thereby indicating the maliciousness of said code.

2. The method according to claim **1**, wherein said at least one content attribute is selected from a group comprising: a programming flag, a Boolean value, a numeric value, and a string value.

3. The method according to claim **1**, wherein said executable includes compiled code.

4. The method according to claim **1**, wherein said executable includes human-readable code.

5. The method according to claim **3**, wherein said compiled code is selected from the group consisting of Windows EXE, Windows DLL, Windows OCX, Linux executables, Solaris executables.

6. The method according to claim **4**, wherein said human-readable code is selected from the group consisting of macro, script, VBScript, JavaScript, BAT, PHP, and ASP.

7. The method according to claim **1**, wherein at least one said checking includes testing an equality of at least one element selected from the group consisting of an identity of said function said content of said calling parameters, and said value of said calling parameters.

8. The method according to claim **1**, wherein at least one said checking includes testing a similarity of at least one element selected from the group consisting of an identity of said function, said content of said calling parameters, and said value of said calling parameters.

9. The method according to claim **1**, wherein at least one said checking includes testing if certain functions have already been called, testing if certain functions have been already called with certain parameters.

10. The method according to claim **1** wherein said system function is a random value.

www.freepatentsonline.com

11. The method according to claim 1 wherein said system function is a tine value.

12. The method according to claim 1 wherein said system function is a certain folder name.

13. The method according to claim 1 wherein said system function is a number of files in a folder.

14. The method according to claim 1 wherein said system function is a certain folder extension.

15. The method according to claim 1, wherein said system function is a function written in a programming language of the executable.

16. The method according to claim 1, wherein said system function is an operating system function of an operating system under which said code is to be executed.

17. The method according to claim 1, wherein at least one said parameter is a permanent parameter.

18. The method according to claim 1, wherein at least one said parameter is a temporary parameter.

19. The method according to claim 1, wherein at least one said parameter is passed by its content.

20. The method according to claim 1, wherein at least one said parameter is passed by its address.

21. A method for emulating, said method comprising:

providing non-compiled code;

pre-defining at least one content attribute having relevance to maliciousness, for at least one variable of said executable code;

pre-defining at least one malicious state, at least one said malicious state being a call to a certain system function with at least one said content attribute;

upon emulating an assignment instruction within said non-compiled code, setting the content attributes of the assigned variable according to the assigned content, and the value of the assigned variable;

upon emulating a mathematical operator within said non-compiled code, employing content mathematics; and

upon emulating a function call within said non-compiled code, checking that the current state, being the combination of the function identity with the content and/or value of the calling parameters, corresponds to at least one of the pre-defined malicious states, thereby indicating the maliciousness of said code.

22. A method for emulating, said method comprising:

providing non-compiled executable code;

pre-defining at least one content attribute having relevance to maliciousness, for at least one variable of said executable code;

pre-defining at least one malicious state, being a call to a certain system function with certain content and/or value;

upon emulating an assignment instruction within said non-compiled executable code, setting the content attributes of the assigned variable according to the assigned content, and the value of the assigned variable;

upon emulating a mathematical operator within said non-compiled executable code, employing content mathematics; and

upon emulating a function call within said non-compiled executable code, checking that the current state, being the combination of the function identity with the content and/or value of the calling parameters, corresponds to at least one of the pre-defined malicious states, thereby indicating the maliciousness of said code.

23. A method for emulating, said method comprising:

pre-defining at least one content attribute having relevance to maliciousness, for at least one variable of said executable code;

pre-defining at least one malicious state, said at least one said malicious state being a call to a certain system function with a variable having a certain content attribute;

upon emulating an assignment instruction, setting the content attributes of the assigned variable according to the assigned content, and the value of the assigned variable;

upon emulating a mathematical operator, employing content mathematics; and

upon emulating a function call, checking that a first current state, being the combination of the function identity with the content attributes of the calling parameters, corresponds to at least one of the pre-defined malicious states, thereby indicating the maliciousness of said code.

24. The method of claim 23 wherein upon emulating a function call a second current state being the combination of the function identity with the content and/or value of the calling parameters is checked.

* * * * *

www.freepatentsonline.com

Electronic Patent Application Fee Transmittal

Application Number:	
Filing Date:	
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Filer:	Dale S. Lazar/Jacqueline Kimvilakani
Attorney Docket Number:	339000-P0004

Filed as Large Entity

Utility Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Utility application filing	1011	1	300	300
Utility Search Fee	1111	1	500	500
Utility Examination Fee	1311	1	200	200
Pages:				
Claims:				
Claims in excess of 20	1202	27	50	1350
Independent claims in excess of 3	1201	1	200	200
Miscellaneous-Filing:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	**2550**

Electronic Acknowledgement Receipt

EFS ID:	1641321
Application Number:	11694261
International Application Number:	
Confirmation Number:	1684
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Customer Number:	47604
Filer:	Dale S. Lazar/Jacqueline Kimvilakani
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-P0004
Receipt Date:	30-MAR-2007
Filing Date:	
Time Stamp:	15:58:52
Application Type:	Utility

Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	$2550
RAM confirmation Number	489
Deposit Account	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)

1	Transmittal of New Application	339000P0004TRANSMITTAL.pdf	609932	no	1
Warnings:					
Information:					
2	Application Data Sheet	339000P0004ADS.pdf	1002466	no	3
Warnings:					
Information:					
This is not an USPTO supplied ADS fillable form					
3	Specification	339000P0004SPECIFICATION.pdf	14949142	no	31
Warnings:					
Information:					
4	Claims	339000P0004CLAIMS.pdf	5496127	no	12
Warnings:					
Information:					
5	Abstract	339000P0004ABSTRACT.pdf	318408	no	1
Warnings:					
Information:					
6	Drawings	339000P0004FIGURES.pdf	573954	no	2
Warnings:					
Information:					
7	Oath or Declaration filed	339000P0004DECLARATION.pdf	2181285	no	1
Warnings:					
Information:					
8	Information Disclosure Statement (IDS) Filed	339000P0004IDS.pdf	950724	no	2
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					

9	Information Disclosure Statement (IDS) Filed	339000P0004PTO1449.pdf	1480893	no	2
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
10	Information Disclosure Statement (IDS) Filed	339000P0004USP7013483.pdf	4589770	no	6
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
11	Foreign Reference	339000P0004WO96-30829.pdf	10090622	no	18
Warnings:					
Information:					
12	Foreign Reference	339000P0004WO99-15966.pdf	21175150	no	46
Warnings:					
Information:					
13	Foreign Reference	339000P0004EP1536341A2.pdf	11848475	no	18
Warnings:					
Information:					
14	NPL Documents	339000P0004NPL.pdf	7463683	no	9
Warnings:					
Information:					
15	NPL Documents	339000P0004NPL2.pdf	4576288	no	7
Warnings:					
Information:					
16	Fee Worksheet (PTO-06)	fee-info.pdf	8593	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	87315512		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (12-04)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

11/694,261

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)		OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))							300
SEARCH FEE (37 CFR 1.16(k), (i), or (m))							500
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))							200
TOTAL CLAIMS (37 CFR 1.16(i))	47 minus 20 =	27	X$25=		OR	X$50	1350
INDEPENDENT CLAIMS (37 CFR 1.16(h))	5 minus 3 =	2	X$100=			X$200=	400
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR						
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))			N/A			N/A	
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	2750

APPLICATION AS AMENDED – PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDI-TIONAL FEE ($)		OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDI-TIONAL FEE ($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))							OR		
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDI-TIONAL FEE ($)		RATE ($)	ADDI-TIONAL FEE ($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))							OR		
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

3-30-07

Best Available Copy

METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

This application claims the benefit of priority to US application serial no. 60/789156, filed April 5th, 2006, the content of which is hereby incorporated by reference.

The present invention relates to a method, a computer program and a computer for analysing an executable co file.

The term "malware" is used herein to refer generally to any executable computer file that is or contains malicious code, and thus includes viruses, Trojans, etc. For convenience, however, and as is conventional, the term "virus" will also be used generically herein to refer to any of these types of malware.

A typical anti-virus product, such as anti-virus scanning software, scans files to look for signatures in the file that are known to be indicative of the presence of a virus. In order to try to defeat anti-virus products, the malware writers use obfuscation techniques in order to attempt to hide the signature of the virus code from detection. Typically, the obfuscation involves encrypting or packing the viral code. Anti-virus products typically incorporate fixed decryption algorithms to decrypt or unpack files that are encrypted or packed using known encryption algorithms, but this means that the anti-virus product has to be continually updated to include new decryption algorithms as new encryption algorithms appear. Moreover, the same or similar basic viral codes are encrypted using different encryption algorithms, which make it even harder for the anti-virus products to detect the

Best Available Copy

viruses. Furthermore, modern so-called polymorphic viruses make the job of anti-virus products even more difficult because a polymorphic virus changes its encryption algorithm and the corresponding encryption keys each time it replicates from one computer to another. Accordingly, it is now very difficult or almost impossible for the makers of the anti-virus products to use fixed algorithms to decrypt polymorphic viruses.

Accordingly, different techniques were developed for decrypting polymorphic viruses in order to expose the underlying viral code. These include for example emulation and cryptanalysis. However, each of these techniques is rather slow, relatively speaking, and cannot successfully be run in real time on for example a user's personal computer. For example, in the early days of personal computers, say from about 1986 onwards, viruses were typically very small files of a few kilobytes (KB) and such small files can easily be emulated or analyzed in a short period of time (of less than 1 second say). On the other hand, polymorphic viruses are now often written in a high level language and can have a size of 2 megabytes (MB) or more and can contain upwards of 200 million instructions. Known emulation techniques can take 20 seconds to 1 minute or more to emulate such files, which clearly makes such techniques inappropriate and unacceptably slow for real-time "on the fly" use. Indeed, some current anti-virus products simply ignore files having large sizes (say over 1 MB) and/or those files that contain more than a certain number of instructions (say more than 10 or 20 million instructions), which clearly is not safe for the user.

Best Available Copy

Reference is now made to the paper "Defeating Polymorphism: Beyond Emulation" by Adrian Stepan of Microsoft Corporation published in "Virus Bulletin Conference October 2005". The entire content of the paper "Defeating Polymorphism: Beyond Emulation" is hereby incorporated by reference. This paper provides a good overview of current problems with polymorphic viruses. As discussed in the paper by Stepan, conventional emulation is relatively slow. In four examples given in the paper, decryption of a number of files using emulation took around one minute or more. This is clearly unacceptable to users who will expect any virus checking to be carried out almost invisibly, and within a handful of seconds at the most.

A method for decrypting malware to try to overcome this problem is disclosed in the paper by Stepan. In simple terms, the method operates as follows. First, the type of file is checked (whether for example the file is a Portable Executable file as defined by Microsoft or the Executable and Linking Format used in Linux). Then a virtual environment that depends on the file type is created, including for example virtual memory, virtual registers, etc. For example, a Windows-like environment is created for a PE file, and a Linux-like environment is created for an Elf file. The file is then mapped to the virtual environment. Then an instruction is fetched from the Instruction Pointer, the instruction is translated, and emulated. The Instruction Pointer is then updated. Depending on conditions at the time, the virtual environment is exited or the next instruction is translated and emulated.

Best Available Copy

To overcome the speed problems of emulation, the paper by Stepan proposes a technique in which the malware code is partitioned or disassembled into basic blocks of code. Then, a table is created having an entry for each block of code. Searching is carried out in the proposed method for each block of code rather than for each instruction. The use of blocks of code means that the table has significantly fewer entries than if it were to store an entry for each instruction, such that the search process is quicker. Once the basic blocks have been created from the original malware code, the decryption process in the method disclosed by Stepan proceeds as follows. In the virtual environment, a check is first made as to whether the current basic block has already been translated. If yes, then the translated code for that basic block is then executed; the Instruction Pointer is then updated and either the process exited or returned to the next basic block. If the current basic block has not already been translated, then it is translated and the translated code is executed in order to emulate the instructions associated with that basic block. The Instruction Pointer is then updated and either the process exited or returned to the next basic block.

As disclosed in the test examples in the paper by Stepan, this so-called dynamic translation of the malware code speeds up the decryption process by a factor of four to five or so, taking between seven and fifteen seconds to decrypt the test files. This process can handle on average about twenty million instructions per second, though in specific optimized cases, perhaps up to fifty million instructions per second can be handled. This processing time, especially the average processing time, is still

Best Available Copy

nevertheless too slow to be acceptable for real time use, especially given the large file sizes of modern polymorphic viruses.

According to a first aspect of the present invention, there is provided a method of unpacking/decrypting an executable computer file using a host computer, the method comprising: partitioning the executable computer file into plural basic blocks of code; translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer; linking at least some of the translated basic blocks of code in memory of the host computer; and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

In the present context, a basic block is a "straight-line" piece of code that has no jumps (or "branches") or jump targets within its body. If a jump target is present in the block, it is only at the start of the block (and in that case defines the start of the block). A jump is only at the end of the block, and thus defines the end of the block.

The translated code that is obtained in the preferred embodiment of this aspect is functionally equivalent to the corresponding original code but is safe to execute on the host computer. The memory is typically real memory of the host computer.

Best Available Copy

Linking the translated basic blocks makes the execution of those translated basic blocks faster. This is especially the case where the basic blocks are found in loops in the unpacking/decryption process. For example, when unpacking or decrypting a file, some blocks may execute 100,000 times or more, meaning that the unpacking/decryption loops can take more than 90% of the time needed to unpack/decrypt the file in the prior art methods. Linking provides a direct jump between the linked blocks, which can increase the speed of operation by up to a factor of 5 or more allowing on average around 100 million instructions per second to be processed, making the method viable in real-time with minimal interruption to the user even in the case of large and/or polymorphic viruses.

In a preferred embodiment, the method comprises, when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory.

In a preferred embodiment, a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the

Best Available Copy

memory address that follows the first translated basic block.

Preferably, the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated. This allows for the fact that during the translation, the size of some already-translated blocks and their addresses in memory may be changed and therefore linked basic blocks need to be re-linked to their new addresses. It should be borne in mind that a translated basic block may branch to a non-translated basic block. Accordingly, it is preferred that indirect jumps are used and that blocks are re-linked dynamically as they are translated.

In a preferred embodiment, the method comprises passing flow of the method to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks. In such cases, the virtual machine can be used to overcome the cache miss or CPU exception, which might otherwise halt the execution on the host computer, for example by providing updated cache addresses or information to overcome the CPU exception. Flow can then return to the execution on the host computer.

In a preferred embodiment, the method comprises emulating the executable file by executing the basic blocks of code, wherein the basic blocks of code that are translated are only those that execute more than a predetermined number of times during the emulation. Preferably, the value for said predetermined number of times is dynamically variable during the emulation.

Best Available Copy

In a preferred embodiment, the method comprises creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block, and comprising emulating the executable file by executing the basic blocks of code, and, during the execution of a basic block, checking at least one of the read page and the write page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

In practice, during the emulation, if there is a cached real address corresponding to the virtual address that is being accessed, that cached real address is used, else a memory mapper can be used to update the address(es) in the cache(s). The use of caches for basic blocks can significantly increase the speed of operation during emulation because, for example, provided there is present in a cache a cached real address that corresponds to a virtual address to be accessed, a translation between virtual memory and real memory addresses or other similar time-consuming operation can be avoided. In some cases it has been found that there is a high number of cache "hits", i.e. where there is present in a cache a cached real address that corresponds to a virtual address to be accessed, and such cases are particularly quick to analyze and unpack/decrypt. Such cases include those where the

Best Available Copy

file is decrypted using linear decryption, common decompression techniques, etc.

In a preferred embodiment, the method comprises creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprises, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

Some embodiments may use a set of cache pages for every instruction.

According to a second aspect of the present invention, there is provided a method of unpacking/decrypting an executable computer file using a host computer, the method comprising: partitioning the executable computer file into plural basic blocks of code; creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and, emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be

Best Available Copy

analyzed to determine whether the executable computer file is or should be classed as malware; wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

As discussed above, the use of caches for basic blocks can significantly increase the speed of operation during emulation because, for example, provided there is present in a cache a cached real address that corresponds to a virtual address to be accessed, a translation between virtual memory and real memory addresses or other similar time-consuming operation can be avoided.

In a preferred embodiment, the method comprises creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprises, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

The emulation is preferably carried out in virtual memory on the host computer. This keeps the real components of the host computer, including for example the real memory, real input/output devices, real storage devices, etc., safe from attack by the file that is being unpacked.

Best Available Copy

The partitioning is preferably carried out dynamically. For example, new blocks might be discovered or existing blocks might be modified as a result of processing blocks that were previously discovered. This means that already-discovered basic blocks may have to be repartitioned and data about them restored as necessary.

In an embodiment, the executable computer file is a Portable Executable (PE) file. This file type is a file type specified by Microsoft Corporation as part of the original WIN32 specifications. Alternatively, the file might be of the Executable and Linking Format (ELF) used in Linux.

According to a third aspect of the present invention, there is provided a computer program for unpacking/decrypting an executable computer file using a host computer, the computer program comprising program instructions for causing a said host computer to carry out a method of: partitioning the executable computer file into plural basic blocks of code; translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer; linking at least some of the translated basic blocks of code in memory of the host computer; and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

Best Available Copy

According to a fourth aspect of the present invention, there is provided a computer program for unpacking/decrypting an executable computer file using a host computer, the computer program comprising program instructions for causing a said host computer to carry out a method of: partitioning the executable computer file into plural basic blocks of code; creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and, emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware; wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

The preferred embodiments of the computer programs provide computer programs that are arranged to carry out the preferred embodiments of the method described above.

According to a fifth aspect of the present invention, there is provided a method of analysing an executable computer file using a host computer, the method comprising:

Best Available Copy

partitioning the executable computer file into plural basic blocks of code; creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; emulating the executable file by executing the basic blocks of code, wherein, during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block; translating those basic blocks of code that execute more than a predetermined number of times during the emulation into translated basic blocks of code that can be executed by the host computer; linking at least some of the translated basic blocks of code in memory of the host computer; and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

This most preferred method provides the maximum potential speed of operation by using emulation where possible and reasonable, using at least a read cache and a write cache for at least some of the basic blocks to minimise the number of times that a virtual address has to be mapped to a real address and vice versa during the

Best Available Copy

execution of the basic blocks in the emulation, using execution of translated basic blocks when the basic blocks are found to be executed more than a certain number of times during emulation, and using linking of at least some of the translated basic blocks where possible to speed the execution of the translated basic blocks.

Preferred embodiments of this method correspond to the preferred embodiments of the methods described above. Similarly, there is also provided a computer program for causing a computer to carry out the method and a computer programmed to carry out the method.

Embodiments of the present invention will now be described by way of example with reference to the accompanying drawings, in which:

Fig. 1 shows schematically a functional block diagram of the operation of an example of a computer program according to an embodiment of the present invention;

Fig. 2 shows schematically a block diagram of a first example of linked basic blocks of code; and,

Fig. 3 shows schematically a block diagram of a second example of linked basic blocks of code.

The preferred embodiment of an aspect of the present invention is implemented in a computer program which in use is installed and runs on a user's personal computer. The preferred embodiment operates generally as follows. In summary, the file is first inspected for type and "obvious" or known indicators that it is malware. Then, the running

Best Available Copy

of the file on the host computer is emulated in a safe, virtual environment created on the host computer so as to allow the file to be unpacked/decrypted and to collect execution-level flags. Then, the image of the decrypted file is analyzed in combination with the API (application program interface) data obtained during the emulation to determine whether or not it is or should be classed as malware. These three main stages will be discussed in some detail below. Importantly, during the emulation stage, if it is found that any particular basic blocks of code are emulated more than a certain number of times, as can happen for example in loops that arise during the decryption process that occurs as part of the emulation of a packed/encrypted file, then those blocks of code are subject to special treatment as will be discussed below. Also importantly, during the emulation stage, cache memories are used for basic block addresses, which can significantly increase the speed of operation.

In the first main stage, the executable computer file that is to be analyzed to determine whether or not it is malware is firstly inspected to determine its format, including whether for example the file is of the Microsoft Portable Executable type or the Linux Executable and Linking Format type. (Much of the present specification assumes that the file is of the PE type, although the principles described herein are directly applicable and easily converted to be applicable to ELF and files.) As is known per se, the file can then be quickly searched to look for any obvious oddities that are known to be present in malware. Then the file is inspected to see if one or more of the following flags are present:

Best Available Copy

(i) the file is packed/compressed/encrypted, which can be determined by for example entropy analysis of the file. Malware is often packed or compressed or encrypted whereas non-malware files are usually not packed or compressed or encrypted;

(ii) the Entry Point is in the last section of the file;

(iii) the code section is writable, which is an indicator for self-modifying code;

(iv) there is a mismatch in the DataDirectory[x].VirtualAddress/Size and the [Last]SectionAdress/Size. Because for example the Microsoft PE loader does not check for these abnormalities, this is the place where many viruses reside;

(v) the file has undeclared internal overlays; and,

(vi) differences in the section flags/name and the DataDirectory flags (e.g. a file with a .reloc section, but no relocations).

Some of the flags are only found in malware. However, some flags may also be present in legitimate files, though such cases are very, very low in number.

In the second main stage, after these flags have been checked, the potentially malicious file is run in a virtual environment that emulates the real host computer. Details of the setting up and operation of the virtual environment for this emulation will be discussed further below. The emulation of the running of the file allows the detection

Best Available Copy

of suspicious instructions, memory accesses, RTC (real time clock) accesses, internal windows structures accesses, exceptions, etc. A list of some of the execution-level flags that can be obtained is as follows:

(i) during the decryption/decompression, it is detected if the file is packed twice, which is common in malware but never used in real applications;

(ii) specific packers (such as Morphine, MoleBox, Mew) used by malware writers are detected;

(iii) common packers (such as UPX, Aspack) with slight modifications are detected. The slight modifications are used by malware writers as an anti-antivirus trick so that standard decompression does not work;

(iv) anti-debugging/anti-tracing tricks are detected;

(v) APIs (application program interfaces) are monitored. Packers/cryptors compress/encrypt the Import Table, so in a normal analysis session it is very hard to tell which functions are being used by a program. Some packers spend a great amount of code doing what is called "API redirection", that is they hijack the original API functions to a set of stubs that are created at runtime by the packer. Also, even if a memory dump of the program is available, the list of APIs may not be available since some packers use hashes and patch the ".idata" section directly. Since the preferred embodiment controls the entire decompression process, it can be detected when the packer's code parses the export table of the DLLs in the virtual environment; and,

Best Available Copy

(vi) exported functions are monitored. Since many Trojans/backdoors use plug-ins (a famous example being Back Orifice), it is possible to say whether the file is a backdoor plug-in or not by simply analyzing the export data, as plug-ins need specific names/structures to work together correctly with the host.

After the (emulated) decompression has finished, not only is the above list of flags and APIs used by the program available, but also the real, unpacked/decrypted data is available, which is used in the subsequent analysis of the file.

In the third main stage, a detailed scan is made in the file's original, decrypted image, and the results combined with the API data to allow a determination to be made as to whether or not the file is or should be treated as malware.

Referring now to Figure 1, the preferred computer program when running can be thought of as providing two main components, a virtual machine engine 10 and an optimisation engine 30. The virtual machine engine 10 is responsible for setting up the virtual machine environment which is used for the emulation in the second main stage discussed above. The optimisation engine 30 is responsible for ensuring that the process can be carried out sufficiently quickly.

The virtual machine engine 10 provides inter alia a CPU emulator 12, a virtual memory manager 14, a virtual file system 16, a set of simulated system libraries 18, and

Best Available Copy

a set of simulated Windows-like operating system functions 20 (such as a memory mapper, a thread scheduler, a structured exception handler, etc.), which together make up an IA32 (32 bit Intel Architecture) emulator. The virtual machine engine 10 provides a module loader function which is responsible for checking the consistency of the main module ("module" being the name given to the in-memory version of a PE file when loaded into memory), i.e. for example whether the file is a valid PE file; for initialising the CPU/virtual memory; mapping the main module inside the virtual memory; setting pointers to the specific "Swap-In" routines; setting up the necessary structures (Process Environment Block, Thread Environment Block, Structured Exception Handler); and setting up a default stack and a default register set.

The CPU emulator 12 provides the necessary emulation of the host CPU, memory, input/output ports, etc. such that the virtual machine is, preferably, a complete clone of the host computer. The virtual CPU works the same as a normal, real CPU, using for example fetch/decode/emulate functions.

The virtual memory manager 14 is capable of virtualising a full 4GB address space (even if the real, host memory is less than 4GB). The virtual memory manager 14 enables analysis of any kind of memory access, thus allowing the detection of potentially malicious files (by, for example, detecting an illegal access to system variables).

The virtual file system 16 enables monitoring of all accesses to the (virtual) file system by a potentially malicious file, which enables analysis through heuristics

Best Available Copy

(i.e. patterns of behaviour). The virtual file system 16 may provide a number of "goat" files, which are files that are intended to lure malware into attacking or otherwise compromising them. Each goat file may have a "dirty" bit to indicate its status, the "dirty" bit initially being cleared and then being set whenever it is detected that the program file being analyzed is attempting to modify or overwrite the goat file.

The simulated system libraries 18, which are loaded on demand during emulation, are set depending on the import table or specific requirements of the program file being analyzed. Each time a simulated system library function gets called during the emulation, the virtual machine engine 10 checks its parameters and logs any unusual activity or results.

The simulated operating system functions 20 enable control over any low-level system functions, thus enhancing the detection capabilities.

The virtual machine engine 10 may also provide a set of virtual processes with antiviral names. These do not have any function other than to try to lure malware into attempting to attack them, typically by ending the process, which is again evidence that the program file being analyzed is malware.

The optimisation engine 30 provides inter alia a code flow manager 32, a disassembler 34, an optimiser/translator 36, a re-assembler 38 and a re-linker 40.

Best Available Copy

During the emulation, the code flow manager 32 creates "basic blocks" of code from the program file instructions that are being emulated by appropriately partitioning the program file instructions. As mentioned above, in the present context, a basic block is a "straight-line" piece of code that has no jumps (or "branches") or jump targets within its body. If a jump target is present in the block, it is only at the start of the block (and in that case defines the start of the block). A jump is only at the end of the block, and thus defines the end of the block. The code flow manager 32 also detects "hot spots", which will be discussed further below.

The code flow manager 32 also detects unusual changes in the code flow. For example, a buffer overflow is considered an unusual change in the code flow. By way of another example, the code flow manager 32 verifies whether "ret"-like instructions link correctly with "call"-like instructions. A return to an exported function, a return to an existing code block, generated by a CALL instruction, and a return to the heap or stack can certainly be regarded as suspicious operations, and are reported accordingly.

Some basic blocks execute more often than others. In an encrypted program file for example, the decryption loops can take more than 90% of the time needed to decrypt the file. Those basic blocks and loops are often called "hot spots". What makes a basic block a "hot spot" depends on the virtual machine implementation. If N is taken as a threshold, then it can be said that "every basic block that executes for N times or more is a hot spot". By way of example only, N may be 40 or 400 or 1,000 or 10,000 or 20,000, or any integer in between. N may be varied

Best Available Copy

dynamically during the emulation, depending on for example the presence and number of the flags present as found during the inspection of the file in the first main stage briefly mentioned above. In general, the value of N can be safely decreased, and the maximum number of emulated instructions safely increased, to provide a large improvement in the speed of analysis of malicious files whilst having only a small impact on the speed of analysis of files that are not malicious. The code flow manager 32 maintains a flag for every basic block and a "basic block execution count" to help in detecting time-consuming, possibly infinite loops, and to allow detection of hot spots by determining whenever a basic block of code is executed more than N times during the emulation.

Returning now to the description of the optimisation engine 30, the disassembler 34 takes the basic blocks of code provided by the code flow manager 32 and translates them into a set of internal instructions for the subsequent stages in the optimisation engine 30.

The optimiser/translator 36 removes unnecessary instructions from the translated instructions provided by the disassembler 34. Most polymorphic decryptors add garbage code to confuse analysis tools, and this is removed by the optimiser/translator 36. In practice, given that most of the variations between polymorphic or metamorphic viruses is in the garbage code and the translated/normalized code for the different generations of a particular virus are often very similar, this removal of unnecessary instructions makes detection of the virus much easier. The optimiser/translator 36 also changes the instruction(s) provided by the disassembler 34 by adapting

Best Available Copy

them to the requirements of the virtual machine in which the emulation is run (e.g. by ensuring that memory accesses by the program file being analyzed are rerouted to the virtual memory), whilst preserving the logic of the program file.

Whenever the code flow manager 32 detects that a particular basic block of code has been emulated more than N times during the emulation process, the basic block is "translated". In other words, functionally equivalent code that is completely safe to execute directly on the real host processor is generated from the basic block. This translation of the basic blocks of code is done by the re-assembler 38.

The "hot spot" basic blocks of code are passed to the real host processor and memory for execution. Not only are the "hot spot" basic blocks executed on the real processor rather than in the virtual machine, but also the "hot spot" basic blocks are "linked" (or, in a sense, "re-linked") in the real memory by the re-linker 40. Linking is a significant contributor to the speed of operation of the preferred embodiment compared to the prior art. Linking provides a direct jump between the linked blocks and thus avoids the need to look up jump addresses with the associated laborious memory accesses, etc. during execution on the real host processor.

Linking of blocks can be achieved in a number of ways. For example, a first translated basic block can be linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an

Best Available Copy

unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block. One or more of these techniques can be used to link any particular pair of blocks.

When a basic block of code is newly translated by the re-assembler 38, a check is made whether an already translated basic block of code has a jump instruction to the newly translated basic block of code. If so, the newly translated basic block of code is linked to the already translated basic block of code in memory.

It should be borne in mind that during the translation, the size of some already-translated blocks and their addresses in memory may be changed and therefore linked basic blocks need to be re-linked to their new addresses. However, a translated basic block may branch to a non-translated basic block. Accordingly, it is preferred that indirect jumps are used and that blocks are re-linked dynamically as they are translated.

This linking of basic blocks for execution in the real memory of the real host processor speeds the processing of the basic blocks. In the case especially of long decryption loops that span across many basic blocks, there can be a speed boost of a factor of 5 or more to 100 million instructions per second, which compares to the typical maximum of 20 million instructions per second that can be achieved in the prior art. In other words, a

Best Available Copy

process that took say 5 seconds in the prior art, which is probably unacceptably long to most users, can be carried out in 1 second, which is more likely to be acceptable to most users. Practically speaking, a potentially malicious program file can be translated and re-linked and then allowed to decrypt practically at near real-speed. This makes the method viable in real time with minimal interruption to the user even for large and/or polymorphic viruses.

Figure 2 shows schematically translated code blocks for a simple example when N (the threshold) is 40. Figure 3 shows schematically the case where only the basic blocks that get emulated 400 times or more are linked. This provides a much clearer clear view of the decryption loop.

Whilst the translated basic blocks of code are executed on the real host processor in real memory, the virtual machine can be used when needed, for example in the case of a cache miss, CPU exception, etc.

An example of a "cache miss" is as follows. When a basic block is being translated prior to execution on the real host processor, information about the memory access(es) made by the basic block during the emulation is gathered by the virtual machine engine 10 and used to create cached pages of memory addresses, which are used during execution of the translated basic block on the real host processor. It can happen during this execution, for example after a number of iterations of the translated basic block, that an address being accessed is outside the range of these cached pages of memory addresses. In that case, the virtual machine engine 10 can take control of the

Best Available Copy

process, check the type of memory access being attempted by the execution of the translated basic block, and load a new cache page with a new address for the address that is being accessed.

An example of a CPU exception such that the virtual machine engine 10 can take control of the process is as follows. Assume that a decompression loop consists of four basic blocks X, Y, Z and T. Assume that block X jumps to block Y, block Y jumps to block Z, block Z jumps to block T and block T jumps back to block X. Further assume that the re-assembler 38 cannot translate block T. In that case, during execution of the translated basic blocks on the real host processor, the flow would be as follows:

(i) execute X, which is linked to Y and so jumps to Y;
(ii) execute Y, which is linked to Z and so jumps to Z;
(iii) execute Z, which is linked to T and so would normally jump to T. However, T has not been translated. Accordingly, flow returns to the virtual machine;
(iv) in the virtual machine, emulate T, notice that it would jump to X, and then return to (i).

In this way, if there is a CPU exception (on the real host processor), such as flow passing to a basic block that has not been translated, the virtual machine can take control to deal with the exception.

A separate important contributor to improving the speed of operation of the preferred embodiment is the use of cache pages of memory for each basic block of code for use during emulation. The use of cache pages during emulation and the linking method for the translated basic

Best Available Copy

blocks for execution on the real host processor discussed above can be used individually or together.

Consider the following example code:

```
mov ecx, 10; the ECX register is set to 10
_begin: mov eax, [0x401000 + ecx * 4]; the 32 bit EAX register gets the value from the address 0x401000 + ECX * 4
mov [0x402000 + ecx * 4], eax; the value of EAX is written to the address 0x402000 + ECX * 4
loop _begin; decrement ECX and jump to _begin if ECX is not 0
```

For the two memory accesses, the method disclosed in the paper by Stepan discussed above calls a function called a "memory mapper". Since the virtual memory is, of course, virtual, a translation is required between the virtual and the real memory in order to be able to access the real memory where data is to be stored or fetched. Since in this method the memory mapper is invoked for every instruction that accesses memory, its simple presence is a problem for the execution speed of the emulator.

The preferred embodiment of the present method tries to minimise the number of times that the memory mapper is invoked during emulation by using at least two separate pages of cache for every basic block, these being a "read" page and a "write" page. More preferably, there is a third page of cache for every basic block, a "stack" page. The cache pages are preferably created as the blocks are translated. This set up of the cache pages is based on the assumption that a single basic block accesses ONE "read" page, ONE "write" page and, sometimes, ONE stack page.

Best Available Copy

This is true for most decryption engines. Some embodiments, however, may use a set of cache pages for every instruction.

In simple terms, the cache pages provide pairs of correspondences between real and virtual addresses, i.e. (a) a "read" cached virtual address - a "read" cached real address; (b) a "write" cached virtual address - a "write" cached real address; and (c) a "stack" cached virtual address - a "stack" cached real address.

Consider the case where a virus wants to access address 0xBAADF00D, with an instruction like mov eax, [0xBAADF00D]. This instruction reads a 32 bit value from the address 0xBAADF00D into the EAX register. Assume that the virtual address 0xBAADF00D is mapped to the real address 0xDEADBABE.

When emulating, the process is like this:

a) decode the instruction, see that it is necessary to read a 32 bit word from the virtual address 0xBAADF00D;

b) parse the internal structures stored by the virtual machine engine 10; discover that virtual address 0xBAADF00D is mapped to the real address 0xDEADBABE;

c) read a 32 bit word from 0xDEADBABE, and put it into the EAX register.

Now imagine that the mov eax, [0xBAADF00D] instruction is inside a loop such that it is a hot spot and has to be translated.

As mentioned above, the method proposed by Stepan generates code like this:

Best Available Copy

a) call a Memory_Mapper function, for every memory access, "translate" the virtual address 0xBAADF00D to 0xDEADBABE;

b) access the memory from 0xDEADBABE.

This "translation", which implies parsing the internal structures of the virtual machine, takes time.

The caches of the present method are used like this:

a) see if address 0xBAADF00D is in the cache. If not, call a Memory_Mapper, and update the caches.

b) access the memory from 0xDEADBABE.

In the best case (linear decryption, common decompression, etc.), this provides a very good rate of cache hits and this greatly increases the speed of emulation of the basic blocks. In a very worst case, it is necessary to invoke the memory mapper, for example when an out-of-bound page access is requested, but then in that eventuality the present method is the same as the method proposed by Stefan.

For the example code given above, the pseudo-code for the generated code obtained in the present method is like this:

```
mov ecx, 10
_begin:

if (is_address_in_read_cache (0x401000 + ecx * 4) =
FALSE)  // happens rarely {
call Memory Mapper
put_address_in_read_cache (0x401000 + ecx * 4)
```

Best Available Copy


```
}

    mov eax, [read_cache_address + offset]

    if (is_address_in_write_cache (0x402000 + ecx * 4) =
FALSE) // happens rarely {
    call Memory Mapper
    put_address_in_write_cache (0x402000 + ecx * 4)
}

    mov [write_cache_address + offset], eax

    loop _begin
```

One problem when dealing with memory accesses is inter-page-boundary accesses. Since the present method preferably uses an "on-demand" paging scheme, some addresses that are linear in the virtual memory may actually not be linear in the real memory manager. This is just a consequence of memory pages being loaded on demand and unloaded. For example, some pages that were not accessed for a long period of time may be unmapped as a consequence of memory needs. In extreme cases, the performance penalty is too high when it is necessary to generate two memory accesses (read and write) over two consecutive pages. So, preferably consecutive pages are kept linear in memory if needed (e.g. because an application does cross-page memory accesses).

Although the embodiments of the invention described with reference to the drawings comprise computer processes performed in computer apparatus and computer apparatus

Best Available Copy

itself, the invention also extends to computer programs, particularly computer programs on or in a carrier, adapted for putting the invention into practice. The program may be in the form of source code, object code, a code intermediate source and object code such as in partially compiled form, or in any other form suitable for use in the implementation of the processes according to the invention. The carrier be any entity or device capable of carrying the program. For example, the carrier may comprise a storage medium, such as a ROM, for example a CD ROM or a semiconductor ROM, or a magnetic recording medium, for example a floppy disk or hard disk. Further, the carrier may be a transmissible carrier such as an electrical or optical signal which may be conveyed via electrical or optical cable or by radio or other means.

Embodiments of the present invention have been described with particular reference to the examples illustrated. However, it will be appreciated that variations and modifications may be made to the examples described within the scope of the present invention.

Best Available Copy

CLAIMS

1. A method of unpacking/decrypting an executable computer file using a host computer, the method comprising:

partitioning the executable computer file into plural basic blocks of code;

translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer;

linking at least some of the translated basic blocks of code in memory of the host computer; and,

executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

2. A method according to claim 1, comprising, when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory.

3. A method according to claim 1, wherein a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any

Best Available Copy

existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block.

4. A method according to claim 1, wherein the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated.

5. A method according to claim 1, comprising passing flow of the method to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks.

6. A method according to claim 1, comprising emulating the executable file by executing the basic blocks of code, wherein the basic blocks of code that are translated are only those that execute more than a predetermined number of times during the emulation.

7. A method according to claim 6, wherein the value for said predetermined number of times is dynamically variable during the emulation.

8. A method according to claim 6, wherein the emulation is carried out in virtual memory on the host computer.

9. A method according to claim 1, comprising creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating

Best Available Copy

at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block, and comprising emulating the executable file by executing the basic blocks of code, and, during the execution of a basic block, checking at least one of the read page and the write page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

10. A method according to claim 9, comprising creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprising, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

11. A method according to claim 1, wherein the partitioning is carried out dynamically.

12. A method according to claim 1, wherein the executable file is a Portable Executable File.

13. A method of unpacking/decrypting an executable computer file, the method comprising:

partitioning the executable computer file into plural basic blocks of code;

creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache

Best Available Copy

memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and,

emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware;

wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

14. A method according to claim 13, comprising creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprising, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

15. A method according to claim 13, wherein the emulation is carried out in virtual memory on the host computer.

Best Available Copy

16. A method according to claim 13, wherein the partitioning is carried out dynamically.

17. A method according to claim 13, wherein the executable computer file is a Portable Executable file.

18. A computer program for unpacking/decrypting an executable computer file using a host computer, the computer program comprising program instructions for causing a said host computer to carry out a method of:

partitioning the executable computer file into plural basic blocks of code;

translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer;

linking at least some of the translated basic blocks of code in memory of the host computer; and,

executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

19. A computer program according to claim 18, wherein the computer program is arranged so that the method comprises, when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory.

Best Available Copy

20. A computer program according to claim 18, wherein the computer program is arranged so that a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block.

21. A computer program according to claim 18, wherein the computer program is arranged so that the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated.

22. A computer program according to claim 18, wherein the computer program is arranged so that flow of the method passes to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks.

23. A computer program according to claim 18, wherein the computer program is arranged so that the method comprises emulating the executable file by executing the basic blocks of code, wherein the basic blocks of code that are translated are only those that execute more than a predetermined number of times during the emulation.

Best Available Copy

24. A computer program according to claim 23, wherein the computer program is arranged so that the value for said predetermined number of times is dynamically variable during the emulation.

25. A computer program according to claim 23, wherein the computer program is arranged such that the emulation is carried out in virtual memory on the host computer.

26. A computer program according to claim 18, wherein the computer program is arranged so that the method comprises creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block, and comprises emulating the executable file by executing the basic blocks of code, and, during the execution of a basic block, checking at least one of the read page and the write page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

27. A computer program according to claim 26, wherein the computer program is arranged so that the method comprises creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprises, during the execution of a basic

Best Available Copy

block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

28. A computer program according to claim 18, wherein the computer program is arranged so that the partitioning is carried out dynamically.

29. A computer program according to claim 18, wherein the computer program is arranged so that the executable computer file is a Portable Executable File.

30. A computer program for unpacking/decrypting an executable computer file, the computer program comprising program instructions for causing a said host computer to carry out a method of:

partitioning the executable computer file into plural basic blocks of code;

creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and,

emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware;

Best Available Copy

wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

31. A computer program according to claim 30, wherein the computer program is arranged so that the method comprises creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprises, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

32. A computer program according to claim 30, wherein the computer program is arranged so that the emulation is carried out in virtual memory on the host computer.

33. A computer program according to claim 30, wherein the computer program is arranged so that the partitioning is carried out dynamically.

34. A computer program according to claim 30, wherein the computer program is arranged so that the executable computer file is a Portable Executable file.

35. A computer programmed to carry out a method according to claim 1.

Best Available Copy

36. A computer programmed to carry out a method according to claim 13.

37. A method of analysing an executable computer file using a host computer, the method comprising:

partitioning the executable computer file into plural basic blocks of code;

creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block;

emulating the executable file by executing the basic blocks of code, wherein, during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block;

translating those basic blocks of code that execute more than a predetermined number of times during the emulation into translated basic blocks of code that can be executed by the host computer;

linking at least some of the translated basic blocks of code in memory of the host computer; and,

executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be

Best Available Copy

analyzed to determine whether the executable computer file is or should be classed as malware.

38. A method according to claim 37, comprising, when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory.

39. A method according to claim 37, wherein a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block.

40. A method according to claim 37, wherein the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated.

41. A method according to claim 37, comprising passing flow of the method to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks.

Best Available Copy

42. A method according to claim 37, comprising creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprising, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

43. A method according to claim 37, wherein the value for said predetermined number of times is dynamically variable during the emulation.

44. A method according to claim 37, wherein the emulation is carried out in virtual memory on the host computer.

45. A method according to claim 37, wherein the partitioning is carried out dynamically.

46. A method according to claim 37, wherein the executable computer file is a Portable Executable file.

47. A computer programmed to carry out a method according to claim 37.

Best Available Copy

ABSTRACT

In one aspect, an executable computer file is partitioned into basic blocks of code. At least some basic blocks are translated into translated basic blocks. At least some translated basic blocks are linked in memory of a computer. At least some translated basic blocks on the computer are executed so as to enable the file to be unpacked or decrypted. In this way, the file can be analyzed to determine whether the file is or should be classed as malware. In another aspect, at least a read page of cache memory is created for at least some basic blocks and at least a write page of cache memory is created for at least some basic blocks. During the execution of a basic block, at least one of the read page and the write page is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☒ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

DEFEATING POLYMORPHISM: BEYOND EMULATION

Adrian E. Stepan
Microsoft Corp., One Microsoft Way, Redmond, WA 90852, USA

Tel +1 425 706 9498 • Fax +1 425 936 7329

ABSTRACT

The most common method of detecting malware relies on signatures extracted from the malware body. Attempting to defeat this method and evade detection, malware writers have resorted to code obfuscation techniques, thus creating polymorphic viruses.

There are several well-known methods of decrypting polymorphic viruses, such as emulation, cryptanalysis (x-ray) and dedicated decryption routines. Each of these methods has some limitations: x-ray can handle only simple decryptions; dedicated routines require significant development effort and neither scales well with the number of detected viruses. Emulation doesn't have these weaknesses, but emulating code is significantly slower than executing it on a real CPU. Therefore a very complex polymorphic virus would take an unreasonable length of time to emulate until it is decrypted.

This paper proposes a new method of dealing with polymorphic malware. The method relies on disassembling the analysed code dynamically and performing just-in-time compilation targeted for the host CPU. The code obtained as a result can be executed safely on the host CPU, with little degradation in execution speed, compared to the original code. This provides the same flexibility as emulation but performance, in terms of speed, is dramatically improved. Additionally, the method could be used for other purposes, such as generic unpacking of packed executables, and behaviour-based analysis of complex code.

1. DETECTING MALWARE, AN ONGOING BATTLE

Malware has a long history of evolution. During the past two decades, malware has evolved with regard to replication and spreading mechanisms, as well as techniques used to prevent analysis and/or detection. Such techniques include anti-debugging, encryption, packing, entry point obscuring, etc.

One of the first methods used to detect malware relied on signatures extracted from the malware body. Despite the significant evolution of malware, using signatures is still the most common detection method used today. However, there are some things that have changed, such as the types of signature data and the methods used to search for such signatures. A modern AV engine may use lots of different types of data extracted from the malware body:

- Patterns, with or without wildcards, also known as 'strings'
- Checksums (CRC, MD5, SHA1)
- Behaviour patterns
- File geometry, execution flow geometry
- Statistic distribution of code instructions

Of course, any combination of the above could be used as a malware signature, and the list is not exhaustive. In an attempt to defeat detection by signatures, malware writers started to use code obfuscation techniques, such as encryption. In the beginning, viruses used fairly simple encryption schemes and only the keys changed from one generation to another, while the encryption algorithm remained constant; these are known as oligomorphic viruses. Later, more sophisticated polymorphic techniques were developed. Such viruses were able to change both the encryption algorithm and the keys used to encrypt themselves upon each replication; some were able to generate multiple encryption layers.

It is still possible to detect a polymorphic virus using signatures, but the virus body must be decrypted first. There are several methods that are widely used in the AV Industry for the purpose of decrypting polymorphic viruses: cryptanalysis (also known as x-ray), dedicated decryption routines, emulation, etc.

2. TECHNIQUES CURRENTLY USED TO DEFEAT POLYMORPHISM

X-ray works by attempting to find the decryption key by using the known decryption algorithm and a fragment of decrypted code, which is part of the signature. For each key, an equation is written, that expresses this key as a function of the encrypted code, decrypted code and the other keys. Solving the system will produce the correct set of keys required to decrypt the virus. The method is fairly simple to implement and has good performance for a single given decryption algorithm. In some cases, it is also able to detect fragments of the virus code, even if the entire virus is not present or is not functional. However, the method does not scale well with the number of detected viruses, because it needs to be run for each different encryption algorithm. Also, it can only handle simple algorithms, as the equation system becomes impossible to solve for more complex ones. Its usefulness is very limited in the case of viruses having multiple encryption layers.

Dedicated decryption routines can be developed to detect any virus, and performance for any given virus is usually better compared to the x-ray method. Unfortunately, writing such a routine requires that the virus is analysed completely and the developer understands completely all the possible variants of encryption that the virus can generate. A thorough analysis of the malware and then developing and testing a specific detection routine could take a lot of work and a lot of time to accomplish. Therefore, the response time when using this solution can often be quite long. Additionally, this method does not scale well with the number of detected viruses, as each file must be checked with all the available routines.

Executing the decryption code from the virus itself would decrypt the virus body with excellent speed performance, but this is not a very good idea, for a number of reasons:

- There isn't any simple, reliable way of stopping the execution when the decryption is complete; therefore the virus could replicate, do some damage to the host system and re-encrypt itself and without being detected
- The virus could do some damage even during decryption
- The code could be buggy and crash or enter an infinite loop
- The virus code could be written for a different hardware/software platform, so it might not run at all

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

BNSDOCID: <XP 2409481A_I_>

Of course, the code could be executed in a controlled environment, such as a virtual machine, but a general-purpose virtual machine is very complex software. Including one in an AV product would require software emulation for the full system environment, from device drivers to system APIs, which is simply too much overhead.

None of the methods described above is able to detect new malware in a generic way. Their purpose is to just decrypt polymorphic viruses, so that signature-based detection can be used. While it's possible to develop dedicated routines that are able to detect entire malware families (and often new variants as well), writing a routine that is able to analyse an arbitrary program and determine malicious behaviour is not feasible.

Use of emulation solves all the above problems. Potential malicious code runs in a controlled, simulated environment. Hardware resources such as CPU registers are modelled using software data structures. The behaviour of each instruction is reproduced by a set of software routines designed to update the corresponding data structures, in the same way the instructions would update the hardware resources when executed on a real CPU. Each instruction is first decoded, in order to find the instruction type, length, operands that need to be updated, etc. After this, the appropriate emulation routine is called to update the data structure describing hardware resources. The address of the next instruction is obtained either as a result of instruction decoding or computed by the emulation routine (in the case of branch instructions).

The emulation process usually starts at the program's entry point and instructions are emulated sequentially until a malware signature is found, the emulator is able to conclude that the program is not malicious or emulator resources are exhausted. In addition, the emulator has to decide when to scan for malware signatures and what data to scan, call the scanner, collect and analyse data obtained during emulation for behaviour-based heuristic detection or for the purpose of deciding that the program is not malicious.

Emulation can be used to decrypt any encrypted code, regardless of the complexity of the encryption algorithm, given that the decryption code is available and the emulator is provided with enough resources to complete the decryption. Providing resources such as memory is not very difficult, as the requirement is comparable with that of the analysed program (the overhead caused by the emulator's internal data structures is typically negligible). However, emulating code is significantly slower than running the code on a CPU that can execute it natively. This limitation is impossible to overcome, because for each emulated instruction an emulator has to execute hundreds of instructions to decode it, update the internal data structures, decide if scanning is needed, decide if more instructions should be emulated or not, find the address of the next instruction, etc. Typically, emulation is hundreds of times slower than execution.

An emulator in an AV engine is required to analyse any given file in a finite time, during which it must determine if the file is malicious or not. When the maximum allowed time for a file has elapsed and no malware signatures have been detected, the emulator must stop the analysis and conclude the given file is not malicious. It is always possible that the maximum time limit was set to be too short for a particular malware to be detected. On the other hand, increasing this time limit will deteriorate the emulator's average speed. This happens because there will always be a small percentage of clean files for which the emulator will never be able to determine that they are clean, no matter how much it will analyse. Of course, the time limit can be adjusted dynamically: the emulator could increase it if suspicious behaviour is detected or decrease it otherwise. However, there are instances in which legitimate programs are encrypted with the same encryption engines as certain viruses, or viruses use code that looks benign, etc. Even with adjusting the time limit dynamically, there will have to be a hard limit, to avoid having the emulator analysing a file in an infinite loop. This means that it will always be possible for a virus writer to determine what this limit is for a particular AV engine and write a virus that would need to be emulated longer than that in order to be detected. Viruses such as Win32://Coke, KME-based, etc. would take unreasonably long to emulate, but they would still decrypt and replicate on the real machine in a reasonable time, because native execution is typically hundreds of times faster than emulation.

3. DYNAMIC TRANSLATION

The 'dynamic translation' method described below offers the same flexibility as emulation, while improving performance significantly. It relies on disassembling the code to be analysed dynamically and translating it into functionally equivalent code that is safe to execute on the host machine. The executable code obtained as a result of the translation is persisted; if the code is executed inside a loop, the persisted code can, in most cases, be executed directly, without requiring retranslation.

Disassembling and translating an instruction requires a computational effort that is comparable to emulating an instruction; executing the obtained code is typically slower than executing the original instruction, but much less so than emulating the instruction. If a code sequence is executed in a loop, the code will be translated and executed at the first loop iteration, and for all subsequent iterations the persisted code obtained at the first iteration will be executed. Thus, the method eliminates redundant analysis of repeating code sequences. Compared to emulation, the time required to complete the first loop iteration would be approximately the same, while the subsequent iterations will take considerably less time.

3.1. Partitioning the code into blocks

One of the problems that the implementation of the DT engine has to address is determining whether translated code is available for any given instruction, and if so, locating the corresponding code. One possible solution is maintaining a table with virtual addresses of translated instructions and addresses of corresponding executable code. However, searching a virtual address in this table for each processed instruction is computationally very expensive, negating the speed advantage of executing translated code. A much more efficient way of solving this issue would be to partition the original code into blocks of instructions and only store a table entry for each block. This way, the table would have significantly fewer entries and searching would need to be performed for each block, as opposed to each instruction. Dividing the original code into blocks cannot be done in an arbitrary way; blocks need to have some specific properties, as described below, that limit the size of each block. On the

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

BNSDOCID: <XP 2409481A I >

other hand, the bigger the blocks are, the more efficient the storage and searching will be.

For the rest of this paper, a 'basic block' (BB) will be defined as a contiguous block of code having a single entry point at the beginning of the block and a single exit point at the end of the block. If the code within such a block is executed via a call instruction to the beginning address of the block, all the instructions in the block will be executed. A single instruction is needed, at the end of the block, to return control to the caller. As a consequence, any basic block of original code will contain at most one jump instruction. Basic blocks that don't contain any jump instructions are valid, but suboptimal. They could be created as a result of splitting different blocks or for other practical reasons such as the DT engine having insufficient resources to translate a bigger block, etc. Such blocks don't need to be treated differently, as we can consider that they end with a virtual unconditional jump to the following instruction.

Each discovered BB will be described by a set of properties, such as:

- Block boundaries; the address of the first instruction in the BB and the address immediately following the last instruction in the BB will be used to delimit the block; these are linear addresses in the address space of the original code.
- Executable code obtained as a result of translating the original code in the block.
- Miscellaneous flags, indicating if the block was translated or not, if it was scanned for malware signatures, if the block has any known successor blocks, etc.

Discovering and delimiting basic blocks is a dynamic process, meaning that new blocks may be discovered or existing blocks could be modified as a result of processing previously discovered blocks. After translating a block and executing the resulted code, the beginning address of the next block to be processed will be the destination address of the jump instruction at the end of the block that was just executed. Let's call this address the 'current address'. The next block to be analysed has to be a BB starting at the current address. This block will be determined by searching the current address in the list of block address ranges for previously discovered blocks; the following situations are possible:

i An existing BB is found that begins at the current address; this becomes the current BB and will be processed in the same way as the previous BB.

ii The current address is found inside the address range of an existing BB. According to our definition, a BB must have a single entry point; therefore this block must be split. The existing block will be modified such as its address range will end at the current address and a new BB will be created, starting at the current address. If the split block was already translated, the existing code will not be truncated, but the code starting at the current address will be retranslated for the newly created block.

iii The current address is not found inside the address range of any existing BB. In this case, a new BB will be created, starting at the current address.

If the previous BB ended with an immediate jump instruction – for which the destination address is constant – this means that the current BB will always be a successor of the previous BB. This information can be stored, in order to avoid unnecessary searching in the BB address ranges in case these blocks are inside a loop. If the previous jump instruction was unconditional, the current BB is the only possible successor; in the case of a conditional jump instruction, there can be at most two different successor blocks. If a block ends with a computed jump instruction, the destination address could be different each time the block is executed, so in this case determining the successor block requires searching, as described above, in the list of existing block address ranges. A list of successor blocks determined at previous iterations could be also stored and used to speed up searching, in case the number of different possible successor blocks is reasonably small.

Delimiting the original code into basic blocks as described above and maintaining a data structure describing the blocks and the relations between them has several advantages:

- If several blocks are executed inside a loop, searching for a successor block needs to be done only once for each successor; after all the successors of a particular block have been determined, there is no need to search for a successor of that block at any subsequent loop iteration.
- The list of beginning addresses for the discovered blocks can be used as a list of 'entry points' to scan for malware signatures; each discovered BB needs to be scanned only once.
- It provides data for applying dynamic code optimizations. Since optimizing code is computationally expensive, blocks that are executed more frequently are better candidates for optimizations.

3.2. Scanning for malware signatures

After the virus body has been decrypted, the virus can be detected by finding a signature in the decrypted body. If the virus is not metamorphic, any fragment of the virus body could be used as a signature, providing that it is specific enough to provide accurate identification and not cause false positives. Finding a signature in the decrypted virus body raises the problem of deciding when and where to search for signatures; the search method must guarantee that no signature could go undetected, but it also has to be as fast as possible. In the case of a polymorphic virus, the scanner can guarantee detection only if at least one signature search is performed after the virus body has been completely decrypted, but before it starts encrypting itself again.

Given an arbitrary virus, it is extremely difficult to determine the exact moment when the above condition is met during the analysis. This is true for both emulation and the dynamic translation methods. One possible approach would be to detect decryptor loops and scan the code decrypted by each such loop at the end of the loop. However, this is not very easy to implement, as determining decryptor loops, loop exit criteria and range of decrypted code are fairly complex tasks. Also, this method does not guarantee a minimum number of scans; if the virus has multiple decryption layers, or the same layer is executed multiple times (i.e. in brute-force decryption), the code will be scanned redundantly for each decryption layer. There are also polymorphic viruses for which the entire decrypted body cannot be found in memory at any time (i.e. Dark_Paranoid). In such cases, an instruction

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

BNSDOCID: <XP 2409481A I>

or a small code sequence is decrypted, executed and then immediately encrypted again. For these viruses, the decrypted body can be obtained by logging each instruction the first time it is encountered and sorting the logged instructions by address. Scanning the resultant log guarantees that the entire virus code will be scanned for signatures, and no redundant search operations are performed, because no code sequence will be scanned more than once.

It is generally preferable to extract signatures from the constant part of a polymorphic virus and not from the decryptor code; therefore the code that is used for decryption doesn't necessarily need to be scanned. In practice, however, it is typically more computationally expensive to identify the code used for decryption precisely and exclude it from scanning, than it is to scan it.

Another difficulty comes from the fact that one cannot predict the location in the decrypted code where a signature might be present. Therefore, scanning a chunk of code without any knowledge about the code structure requires searching for signatures starting from up to M - N offsets in the given code chunk, where N is the size of the chunk in bytes and M is the size of the shortest signature searched.

Dividing the code into basic blocks, as described in section 3.1., provides a means to scan the entire code that was analysed, in a reliable way and with minimum number of signature searches. Having the data structure describing the basic blocks, it is easy to determine, for any particular block, whether it is the first time we're analysing it (in which case we also need to scan it) or whether we have analysed it before and no scanning is needed. As part of maintaining the BB data structure, we are also required to detect when any block that has previously been analysed is being overwritten (because if this happened, the block would need to be retranslated). A block can be scanned at any time after it was discovered and before it is overwritten, and it needs to be scanned only once. If we choose, for instance, to use only signatures starting at the BB boundary, the maximum number of scans needed will be the number of unique basic blocks in the code. Of course, a signature could span multiple blocks, in which case all of these blocks need to be discovered and decrypted, so that the signature can be detected. If some of the blocks are overwritten before a signature is detected, the code must be logged as described above.

3.3. Translating code

Given an arbitrary program code to be analysed by an AV engine not only needs the code to be considered unsafe, but it could also be the case that the code is compiled to run on a different hardware platform than the AV engine. Even if the code to be analysed executes natively on the same CPU as the AV engine, it might need to run in a different CPU mode (i.e. x86 real mode vs. protected mode), have different memory mapping mode, execute under a different operating system, etc. Therefore, even if the code was safe or we could somehow make sure the host machine cannot be damaged, it still wouldn't be possible to execute correctly any arbitrary part of the given code, from the AV engine process. It is, however, possible to translate the given code into another code sequence that is functionally equivalent with the original one and that can be safely and correctly executed on the host machine. In our case, the host CPU will be used to execute the translated code directly, while in the case of emulation a virtual CPU is used instead of the real one. Other hardware resources (I/O ports, IRQ controllers, disk drives, etc.) are typically virtualized to protect the host machine from any damage.

There are multiple ways in which a code translation that meets the above criteria can be achieved:

i Translating directly from the original code to target code: each original instruction will be decoded and then an equivalent instruction or instruction sequence will be generated for the target code. This is the simplest translation method to implement, for any source and target binary languages.

ii Translating using an intermediate language (IL): each original instruction will first be translated into an intermediate code sequence and then the intermediate code will be translated to target native code. This method is preferable when we have multiple sources (S) and multiple target (T) languages. With direct translation, we would need in this case S * T translators, while using an intermediate language we only need S + T translators (S translators from a source language to IL and T translators from IL to a target language). There are other advantages as well, such as the possibility to perform code optimizations using the IL form. The IL should be platform-independent, but could be designed in a way that favours translation speed for some particular translators (those that are more frequently used). As a drawback, the IL would need to support all the possible operators, operand types and combinations of these, from all the source languages, raising its degree of complexity. This could prove to be a problem, because translating to or from languages with a reduced instruction set is generally faster, per translated instruction, than in the case of complex languages.

iii Combining the above methods could be achieved in a way that preserves the advantages of both, without any of their disadvantages. Most instructions could be translated using a fairly simple intermediate language, while the most exotic and complex ones, that would also require a complex IL, will be translated directly.

Typically, the code obtained as a result of translation will not be as efficient as the original code. This may happen for various reasons: some instructions or operand encodings from the source language might not have a 1:1 correspondence in either the IL or the target language, some hardware resources used in the source language need to be preserved or not used at all in the target language because of specific restrictions, memory mapping has to be virtualized because the source and target languages might use different mappings, etc.

It is possible to perform some optimizations at basic block level, at translation time, to improve the efficiency of the translated code. If the translation uses an IL, the best idea would be to perform the optimizations on the IL, because the algorithms involved will only need to support this language, as opposed to all the source and target languages. The IL could also be designed to facilitate optimizations such as define-usage chains, copy propagation, etc., while the other languages might not be very suitable for performing such optimizations. Also, the IL may be used to pass translation 'hints' from the source translator to the target translator.

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

BNSDOCID: <XP 2409481A >

If the analysed code is linear, the code will be translated once and executed once. As the execution time is negligible compared to translation time, in this case translation would account for most of the analysis time. For clean files that don't unpack or decrypt themselves and don't have any suspicious behaviour, there is usually no need to analyse lots of looping code; in this case, the translators must be optimized for speed. On the other hand, for polymorphic malware and occasional clean files that contain unpacking or decryption code, most time will be spent analysing loops and executing code that is already translated. In this case, the speed of the code generated by the translators is more important than the translation speed. As improving the efficiency of the translated code is done at the cost of translation speed, a compromise between these two must be obtained.

An example of code translation is given in Appendix A.

3.4. DT execution flow

For each file that needs to be scanned for malware, analysis consists of sequentially identifying and processing of basic blocks, as defined in section 3.1. At the beginning of the analysis, the current address is initialized to the entry point of the program to be scanned. After each BB is processed, the current address is updated to the destination address of the jump instruction at the end of this block. The analysis continues with the BB starting at the new current address, until a malware signature is detected or the program is determined to be clean.

In a simplified description, after each BB is analysed, the following processing algorithm has to be performed for the next one:

1 Look in the data structure describing relationships between blocks for a known successor of the last analysed block; if a known successor exists, make this block the current BB and continue from step 6.

2 Search the block address range table for a previously discovered block starting at the current address. If such a block is found, make it the current BB and continue from step 5.

3 If the current address is found inside the address range of an existing BB, split this BB to end at the current address. Create a new BB starting at the current address, make the new block the current BB and continue from step 5.

4 If no existing block was found, that either starts or includes the current address in its address range, create a new BB starting at the current address; this will be the current BB.

5 Update the data structure describing relationships between blocks: store the information that the current BB is a successor of the previously processed BB.

6 If the current BB was not scanned for signatures, scan for signatures starting at the current address. If a signature is found, stop the analysis and report the file as malicious, otherwise mark the block as scanned.

7 If the current BB is already translated and the code was not overwritten since last translation, continue from step 10.

8 If the current BB was previously translated but the original code was overwritten, discard the translated code, as it is no longer valid.

9 Translate the current BB.

10 Prepare for executing the translated code for the current BB: save hardware resources that are used by both this algorithm and the translated code and need to be preserved, such as CPU registers, etc., depending on implementation.

11 Execute the translated code for the current BB via a call instruction; after execution is complete, control will be returned by the executed code to the caller.

12 Restore resources saved at step 10.

13 Handle any errors that might have happened during execution, decide whether to continue analysing the next block or stop.

During execution, the translated code has to check, before each write to memory operation, whether an existing block would be overwritten by the data being written. The checking may be skipped only if it can be determined, at translation time, that this particular write operation could never overwrite an existing block. If one or more blocks are overwritten, the corresponding translated code will be marked as 'dirty', meaning that it will be discarded at step 8, when those blocks will be processed again. If one of the blocks that were overwritten is the current BB and code beyond the current execution point was overwritten, the execution must not be allowed to continue to the end of the current BB, as this would mean executing code that is no longer valid. In this case, the write operation is allowed to happen and then execution is allowed to continue until the entire translated code sequence corresponding to the current original instruction is executed. This is done because execution cannot be interrupted in the middle of the code sequence for an original instruction, otherwise resuming the execution would not be possible. After all, overwritten blocks are marked as dirty, execution is interrupted and control is returned to the caller, with the current address set to the address where execution was interrupted.

Handling of exceptions such as division by 0, page fault, etc. is done in a similar way: execution is interrupted at the address of the instruction that generated the exception; if an exception handler is present, execution may continue with the exception handler code. Information is passed to the exception handler, enabling it to resume execution at the point where it was interrupted.

Obtaining the address of the original instruction corresponding to a given instruction in the translating code requires some computational effort. For the original code, the real CPU keeps an instruction pointer register, updating it after each instruction is executed. Reproducing this behaviour in the translated code would mean generating and executing code for an extra

```
<add instruction_pointer, instruction_code_size>
```

operation, for each translated instruction. For simple arithmetic instructions, this would mean doubling the translated code size and execution time, which is unacceptable. Therefore, computing the value of the current instruction pointer will only be done when needed, if an exception happens or the current block is being overwritten.

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

ENSDOCID: <XP 2409481A I >

For this purpose, a table is used, that relates offsets of instructions within the current BB to addresses of corresponding instructions, in the original code.

Special care has to be taken when calling high-level language compiled functions from the translated code. CPU registers that might be used by the compiler in global optimizations must be saved at step 10 and restored before calling a high-level language function, in case they were changed by the translated code. The stack pointer and frames must also be preserved. The translated code must also save and restore any registers computed before the high-level language function is called, whose values are still needed after the function returns.

3.5. Environment

In order to obtain a correct behaviour while analysing programs, the DT engine must provide access to various hardware devices (disk drives, keyboard, mouse, network interface, video card, real-time clock, etc.), as well as software resources, such as BIOS data structures and routines and operating system APIs. If the code provided as input to the DT engine could be malicious, most of these resources need to be virtualized. This requires a lot of development effort, given the large number of devices and system APIs that need to be supported. However, virtualizing the system environment for a dynamic translation engine is done in almost the same way as in the case of an emulator, so code reusing is possible if an emulator is available.

Accessing virtualized devices, as opposed to real ones, offers improved speed performance. A virtual device is in fact a data structure in memory, which can be accessed much faster than a disk drive, for instance. Often there's no need to fully implement all the functionality of a device, because no existing malware would require it. Therefore, a virtual device will be less complex than a real one, making it even faster in operation.

Real devices may be used in a few cases – for instance, the current time could be obtained by using the real-time clock. However, this may cause the analysed code to behave incorrectly. As the code actually runs slower inside the DT engine than it would run natively, time inside the DT engine should also pass proportionally slower. Otherwise, it would be possible for the analysed code to determine, based on this inconsistency, that it's running in a virtual environment. This is used by malware as an anti-emulation technique.

Some malware would function correctly only if the environment is configured in a particular way. For instance, they might require a specific OS version, work only on a certain file system type, check for the presence of a specific file in a known place or work only within a certain calendar date range, etc. In these cases, it is difficult to configure the environment so that the code will run correctly, as different malware may have different conflicting requirements.

4. DEVELOPING FURTHER APPLICATIONS

Decrypting code in order to provide signature-based detection for polymorphic executable file infectors is just one possible application of dynamic translation. The code to be analysed doesn't need to be a CPU instruction code, it could also be a platform-independent byte code, such as MSIL byte code or a script. Translating from a non-binary language to native executable code for the host CPU can be done in a similar fashion as translating from a binary language. Using an intermediate language might be more suitable for translating from a non-binary language, because it could provide support for variable number of operands and operand allocation, which are required by virtually all script languages.

Using DT to analyse scripts, however, has to deal with some specifics. For instance, a script might not be able to modify itself while it is being interpreted, but it could generate other script files dynamically and call the interpreter to run these. In this case, each particular file can be translated statically, but the analysis is still a dynamic process, because not all of the component files are available at the beginning of the analysis. Providing an environment for analysing scripts is more difficult than in the case of executable files. Some of the major challenges are accurately reproducing the behaviour of the script interpreter and of OS commands and utilities that might be called from the script. Different versions of operating system or script interpreter might behave differently.

Dynamic translation can also be used to translate malware detection routines from a platform-independent byte code into executable code for the host CPU. This way, an AV engine can easily be updated with new detection capabilities, without the need to actually change the engine code. The benefits provided are smaller engine code, smaller (and faster) updates and less testing effort required for the new routines. The new detection routines will be sent as 'data', the same way as signature/pattern database updates. They will be loaded and translated to native code when the engine is loaded in memory and then executed, as needed, with little speed penalty compared to native code generated by a compiler.

The approach used to translate such routines will be slightly different from the one used to translate files that are scanned for malware, because in this case it is already known that the detection routine code is not malicious. This allows the use of a real environment as opposed to an emulated one. Also, there is no need to scan any of the translated code and there is no need to check for overwritten blocks, because the code doesn't decrypt or otherwise modify itself. All the successors for all basic blocks can be determined when the engine is loaded, which means that we don't have to search for any successor block at execution time. For these reasons, the translated code obtained for detection routines will be a lot more efficient than the code typically obtained by translating files for the purpose of scanning.

In the case of packed executable malware, unpacking is typically needed before a signature can be detected. Using signatures extracted from packed code is not always practical, even for non-polymorphic malware. Some strings or other data might change inside the binary, upon each replication, causing the packed file to change in such a way that the signature won't match any more. Detection based on packed code is not possible if the packer is polymorphic. Heuristic or generic detection can be achieved only by using unpacked code.

Writing unpacking routines for all the packers publicly available takes a lot of development and test effort. In some cases, writing an unpacker routine would require reverse engineering the packer and there might be some legal restrictions preventing this. In the absence of a dedicated unpacking routine, a packed executable could be emulated until the unpacked code is obtained. However, unpacking with

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

BNSDOCID: <XP 2409481A I >

an emulator could be very slow, especially for large packed files that would typically require emulating several millions of instructions. Using dynamic translation, a file could be unpacked significantly faster, compared to emulation, providing detection for malware packed with new packers, with reasonable speed performance, before a dedicated routine is developed. In some cases, the generic unpacking using DT could prove fast enough that dedicated routines won't even be needed.

APPENDIX A

The table below shows an example of translation of a 16-bit x86 code sequence sample to 32-bit x86 target code, as generated by the prototype implementation of the Dynamic Translation method.

Original code	Intermediate language	Translated code	Comments
L_decrypt: mov eax, [bx]	mov -, d[(reg.DS<<4)+reg.BX], reg.EAX	movzx edx, w[ebp+offset_reg_BX] movzx eax, w[ebp+offset_reg_DS] shl eax, 4 add edx, eax call [ebp+offset_memory_mapper] mov ecx, [eax] mov [ebp+offset_reg_EAX], ecx	The memory mapper receives the virtual address in edx and returns the real address in eax
add ax, 1234h	add reg.AX, 1234h, reg.AX	lea eax, [ebp+offset_reg_AX] add w[eax], 1234h	Possible optimization: add w[ebp+offset_reg_AX], 1234h
xor al, ah	xor reg.AL, reg.AH, reg.AL Saveflags reg.Flags	mov eax, [ebp+offset_reg_AH] xor [ebp+offset_reg_AL], al lahf seto al mov [ebp+offset_reg_Flags], ax	The layout of the Flags register image in memory may be different than the actual layout of the native Flags register, for speed reasons
mov dx, ax	mov -, reg.AX, reg.DX	lea eax, [ebp+offset_reg_DX] movzx ecx, w[ebp+offset_reg_AX] mov w[eax], cx	Possible optimization: movzx ecx, w[ebp+offset_reg_AX] mov w[ebp+offset_reg_DX], cx
shr eax, 16	shr reg.EAX, 10h, reg.EAX	lea eax, [ebp+offset_reg_EAX] shr d[eax], 10h	Possible optimization: shr d[ebp+offset_reg_EAX],10h Flags don't need to be stored, because they will be overwritten by the next instruction.
ror ax, cl	ror reg.AX, reg.CL, reg.AX Saveflags reg.Flags	mov cl, [ebp+offset_reg_CL] ror w[ebp+offset_reg_AX], cl lahf seto al mov [ebp+offset_reg_Flags], ax movzx edx, w[ebp+offset_reg_BX] movzx eax, w[ebp+offset_reg_DS]	
mov [bx], ax	mov -, reg.AX, w[(reg.DS<<4)+reg.BX]	shl eax, 4 add edx, eax call [ebp+offset_memory_mapper] mov cx, [ebp+offset_reg_AX] mov [eax], cx	
mov [bx+2],dx	mov -, reg.DX, w[(reg.DS<<4)+reg.BX+2]	mov edx, 2 add dx, [ebp+offset_reg_BX] movzx eax, w[ebp+offset_reg_DS] shl eax, 4 add edx, eax call [ebp+offset_memory_mapper] mov cx, [ebp+offset_reg_DX] mov [eax], cx	The virtual address could be obtained by adding 2 to the virtual address computed for the previous instruction. However, implementing such optimizations in a generic way is computationally expensive.

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

BNSDOCID: <XP 2409481A I >

Original code	Intermediate language	Translated code	Comments
inc bx	Loadflags reg.Flags inc reg.BX, -, reg.BX Saveflags reg.Flags	mov ax, [ebp+ offset_reg_Flags] add al, 7Fh sahf lea eax, [ebp+offset_reg_BX] inc w[eax] lahf seto al mov [ebp+ offset_reg_Flags], ax	The previous 'mov' instructions in the original code do not affect CF, but the corresponding translated code sequences do so. The IL only allows saving all flags and not individual ones, so we must load the native flags prior to executing the 'inc' instruction that doesn't affect CF, in order to preserve the correct value of the CF flag.
loop L_decrypt L_endloop:	sub addr_L_endloop, (reg.CS<<4), reg.IP dec reg.CX, -, reg.CX setz -, -, reg.T32 jopz reg.T32, (reg.CS<<4)+reg.IP–24h	lea eax, [ebp+offset_reg_IP] mov ebx, addr_L_decrypt movzx ecx, w[ebp+offset_reg_CS] shl ecx, 4 sub ebx, ecx mov [eax], bx lea eax, [ebp+offset_reg_CX] dec w[eax] lea eax, [ebp+offset_reg_T32] mov ebx, eax mov eax, 0 setz al mov [ebx], eax mov ecx, [ebp+offset_reg_T32] jecxz L_jump_taken mov b[ebp+offset_jump_info], 2 jmp L_return L_jump_taken: mov edx, -24h add dx, [ebp+offset_reg_IP] movzx eax, w[ebp+offset_reg_CS] shl eax, 4 add edx, eax mov [ebp+offset_crt_address], edx mov b[ebp+offset_jump_info], 3 L_return: ret	The code sequence for a jump instruction must update the instruction pointer register, compute the virtual address of the next instruction and provide the DT engine information about the jump instruction, such as type of jump and whether the jump was taken or not. It is possible to replace 'jmp L_return' with a 'ret' instruction, but this would negate the possibility of calling a debug trace function from the generated code after each instruction sequence, making debugging of generated code more difficult. The final 'ret' instruction is not generated while translating the jump instruction; it is generated after the entire block has been translated. This ensures that any translated block will be able to return control to the caller, even if the original code in that block doesn't end with a jump instruction.

The following format was used for IL instructions: <opcode source_operand_1, source_operand_2, destination_operand>. For example, 'add x, y, d' means 'd = x + y'. Loadflags and Saveflags are not separate IL instructions; they are encoded in the binary form of affected IL instructions. In this particular implementation, the IL language accepts simple operands, such as registers and constants, as well as more complex operands, like registers shifted with a constant, a sum of register, constant or shifted operands, etc. It is possible to design the IL to only support simple operands, in which case each IL instruction would be simpler and faster to generate, but more IL instructions would be needed, in average, to translate an original instruction.

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

BNSDOCID: <XP 2409481A I >

APPENDIX B

The following table shows comparative speed test results, obtained by benchmarking a prototype implementation of the dynamic translation method presented in this paper, versus the emulator in the last version of *RAV* anti-virus engine. Within each test, the DT prototype and the emulator analysed the exact same instructions and scanned for malware with the same signature set. The best time of three consecutive runs was selected, for each test.

The files used for the first three tests were infected with polymorphic file infectors; both the emulator and the DT prototype had 100% detection rate on these test sets. The test set for the last test consisted of 100 copies of an executable file containing a nested decryptor loop – interior loop having 1,020 iterations, exterior loop having 256 iterations.

The benchmark results indicate that:

- The DT method provides a significant speed improvement over emulation, in all tests.
- In the case of emulation, speed performance is not affected, in most cases, by the complexity of the analysed files. The time required to emulate a given code sequence is proportional with the type and number of instructions emulated. An emulator takes little or even no advantage of repeating code sequences (a slight improvement is noticed for the last test set).
- The speed performance provided by the DT method, relative to native execution, improves with the average number of analysed instructions per file, as the probability of finding repeating code sequences increases. Thus, detection of heavily polymorphic viruses, requiring millions of instructions to be analysed, can be accomplished in a reasonable time by using dynamic translation.

test number	method	analyzed instructions x 1,000,000	analyzed files	average instructions per file x 1000	total time [seconds]	MIPS P4@3GHz	average slowdown rate*
1	emulation DT	394	7070	58	63 15	6.3 26	317 77
2	emulation DT	422	2000	211	65 11	6.5 38.3	307 52
3	emulation DT	427	738	579	68 8	6.3 53.4	317 37
4	emulation DT	418	100	4180	54 7	7.7 60	260 33

* the average slowdown rate is defined as the time needed by emulation / DT to analyse a given test code divided by the time needed by a real CPU to natively execute the same code. An average of 1.5 clock cycles per instruction was used for the purpose of estimating the total time required for native execution of the files in the test sets. Actual execution time would be difficult to measure given that infected files were used.

 ©2005 Virus Bulletin Ltd. No part of this reprint may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publishers.

BNSDOCID: <XP 2409481A I >

Electronic Patent Application Fee Transmittal

Application Number:	11477807
Filing Date:	30-Jun-2006
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Filer:	Dale S. Lazar
Attorney Docket Number:	339000-000003

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Claims in excess of 20	1202	2	52	104
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Extension - 1 month with $0 paid	1251	1	130	130
Miscellaneous:				
Request for continued examination	1801	1	810	810
		Total in USD ($)		**1044**

Electronic Acknowledgement Receipt

EFS ID:	9331912
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Dale S. Lazar
Filer Authorized By:	
Attorney Docket Number:	339000-000003
Receipt Date:	28-JAN-2011
Filing Date:	30-JUN-2006
Time Stamp:	16:03:35
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$1044
RAM confirmation Number	3055
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Request for Continued Examination (RCE)	339000-3RCE.pdf	544852 bea52256f4a50dd4ce968991f3c77ea0081719ad	no	1
Warnings:					
This is not a USPTO supplied RCE SB30 form.					
Information:					
2	Extension of Time	339000-3EOT-012811.pdf	611539 726bc7140dfda5eddb5fecf7beb32a98b99b603d	no	1
Warnings:					
Information:					
3		339000-3RCE-Amendment.pdf	7409624 5e8f0c91a8d8793379e2494a44fcac85f7c5577f	yes	23

Multipart Description/PDF files in .zip description

Document Description	Start	End
Amendment After Final	1	1
Claims	2	17
Applicant Arguments/Remarks Made in an Amendment	18	23

Warnings:					
Information:					
4	Transmittal Letter	339000-3IDS.pdf	856616 ea9d56621c20f8c9cf9147110a86fc5aa90d84dc	no	3
Warnings:					
Information:					
5	Information Disclosure Statement (IDS) Filed (SB/08)	339000-3PTO1449.pdf	1209776 7e9d23048938a888b14eea3a029123827a610dbd	no	2
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
6	Miscellaneous Incoming Letter	339000-3RCS.pdf	316375 2b5544a5bfe85917e8992f9111a0be98434a0b99	no	1
Warnings:					
Information:					
7	Foreign Reference	WO9630829.pdf	1311198 fe61e40e076296b2584f628377bfc1fc9f60d3f1	no	18

Warnings:					
Information:					
8	Foreign Reference	WO9915966.pdf	1845982 814b96b350d4534a120c23fcd5b28d9a73995ef0	no	46
Warnings:					
Information:					
9	Foreign Reference	EP1536341.pdf	1142936 bd6124042603baa487aeeea0ea4aabe51de59e68	no	18
Warnings:					
Information:					
10	NPL Documents	Ghostbusters.pdf	622897 c5b2b0ef218736bd79797395ec4719032c653934	no	7
Warnings:					
Information:					
11	NPL Documents	IFW-11694261-012811.pdf	3490060 12458a1c13a8aa782ab2d22c60445d347a07be3e	no	79
Warnings:					
Information:					
12	NPL Documents	DefeatingPolymorphism.pdf	897468 a8f900c27f37df22f5ce25ab246ca401fa6ee324	no	10
Warnings:					
Information:					
13	Fee Worksheet (PTO-875)	fee-info.pdf	33347 908fbd563f2536ff43844b0233061da78803c52e	no	2
Warnings:					
Information:					
			Total Files Size (in bytes):	20292670	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
11/477,807	06/30/2006	

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY ☐ RATE ($)	SMALL ENTITY ☐ FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT	01/28/2011	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	* 49	Minus	** 47	= 2	X $ =		OR	X $52=	104
	Independent (37 CFR 1.16(h))	* 8	Minus	***8	= 0	X $ =		OR	X $220=	0
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	**104**

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/VIOLA ROGERS/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application. Currently amended claims are shown with additions underlined and deletions in ~~strikethrough text~~ except double brackets may be placed before and after the deleted characters to show deletion of five or fewer characters.

1. (CURRENTLY AMENDED) A method of classifying a computer object as malware, the method comprising:

at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored;

~~comparing~~ determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware;

classifying the computer object as malware when the data indicates that the computer object is malware ~~on the basis of said comparison~~;

when the determining does not indicate that the computer object is ~~not known~~ malware, initially classifying [[an]] the computer object as not malware;

generating a mask for [[said]] the computer object that defines acceptable behaviour for the computer object; ~~and~~

running the computer object on at least one of the remote computers;

automatically monitoring operation of the computer object on the at least one of the remote computers;

allowing the computer object to continue to run when behavior of the computer object is permitted by the mask;

disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask; and

reclassifying the computer object as malware [[if]] when the actual monitored behaviour extends beyond that permitted by the mask.

2. (CURRENTLY AMENDED) A method according to claim 1, wherein the data about the computer object that is sent from the plural remote computers to the base computer ~~and that is used in the comparison~~ includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

3. (CURRENTLY AMENDED) A method according to claim 1, wherein the ~~comparing~~ determining identifies relationships between the object and other objects.

4. (ORIGINAL) A method according to claim 3, wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware.

5. (ORIGINAL) A method according to claim 3, wherein said other objects include the object or similar objects stored on at least some of the remote computers.

6. (ORIGINAL) A method according to claim 3, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

7. (ORIGINAL) A method according to claim 1, wherein the data is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on the respective remote computers.

8. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

9. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents data about said object.

10. (ORIGINAL) A method according to claim 9, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object;

vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

11. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents the physical size of the object.

12. (CANCELLED)

13. (CURRENTLY AMENDED) Apparatus for classifying a computer object as malware, the apparatus comprising:

a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored;

the base computer being constructed and arranged to ~~compare~~ determine whether the data about the computer object received from said plural computers indicates that the computer object is malware; and,

the base computer being constructed and arranged to classify the computer object as malware ~~on the basis of said comparison~~ when the base computer determines that the date indicates that the computer object is malware,

wherein the base computer is constructed and arranged to generate a mask for an object that is initially classed not as malware, said mask defining acceptable behaviour for the object, wherein ~~and the base computer or each of the plural remote computers is constructed and~~

~~arranged to monitor~~ operation of the object is automatically monitored when the object is running on at least one of the remote computers, the object being allowed to continue to run when behavior of the object is permitted by the mask, the object not being permitted to run when the actual monitored behaviour of the object extends beyond that permitted by the mask, and ~~to~~ ~~reclassify~~ the object being reclassified as malware [[if]] when the actual monitored behaviour extends beyond that permitted by the mask.

14. (CURRENTLY AMENDED) Apparatus according to claim 13, ~~wherein the base computer is constructed and arranged so as to compare the data where~~ the data includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

15. (CURRENTLY AMENDED) Apparatus according to claim 13, wherein the base computer is constructed and arranged so that the ~~comparing~~ determining identifies relationships between the object and other objects.

16. (ORIGINAL) Apparatus according to claim 15, the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.

17. (ORIGINAL) Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least some of said remote computers.

18. (ORIGINAL) Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

19. (ORIGINAL) Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on said respective remote computers.

20. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

21. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents data about said object.

22. (ORIGINAL) Apparatus according to claim 21, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

23. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents the physical size of the object.

24. (CANCELLED)

25. (CURRENTLY AMENDED) A method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers, the method comprising:

providing from a remote computer to a base computer data about a computer object that is stored on the remote computer;

the data including one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the

object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers;

the data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer,

generating ~~receiving at the remote computer~~ a mask for the [[a]] computer object ~~stored at the remote computer~~, [[said]] the mask defining acceptable behaviour for the object;

executing the computer object at the remote computer; [[and]]

automatically monitoring operation of the computer object at the remote computer compared to the behaviour permitted by the mask;

allowing the object to continue to run when behavior of the computer object is permitted by the mask;

disallowing the object to run when the actual monitored behaviour of the object extends beyond that permitted by the mask; and

reclassifying the object as malware when the actual monitored behaviour extends beyond that permitted by the mask.

~~and, in the event that the behaviour of the object extends beyond that permitted by the mask, sending data to the base computer that the behaviour of the object has exceeded that permitted by the mask.~~

26. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

27. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents data about said object.

28. (ORIGINAL) A method according to claim 27, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

29. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents the physical size of the object.

30. (CURRENTLY AMENDED) A storage medium ~~carrier~~ storing a computer program comprising program instructions for causing a computer to perform the method of claim 25.

31. (WITHDRAWN) A method of determining the protection that a remote computer has from malware, the method comprising:

receiving at a base computer details of all or selected security products operating at a point in time on said remote computer;

receiving similar information from other remote computers connected to the base computer; and,

identifying any malware processes that were not identified by said other remote computers having that particular combination of security products.

32. (WITHDRAWN) A method according to claim 31, comprising providing information to the user of said remote computer that said remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

33. (WITHDRAWN) A method according to claim 31, wherein the details of all or selected security products includes the name of the security products, versions, and loaded signature files.

34. (WITHDRAWN) Apparatus for determining the protection that a remote computer has from malware, the apparatus comprising:

a base computer constructed and arranged to receive computer details of all or selected security products operating at a point in time on said remote computer;

the base computer being constructed and arranged to receive similar information from other remote computers connected to the base computer; and,

the base computer being constructed and arranged to identify any malware processes that were not identified by said other remote computers having that particular combination of security products.

35. (WITHDRAWN) Apparatus according to claim 34, wherein the base computer is constructed and arranged to provide information to the user of said remote computer that said

remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

36. (WITHDRAWN) Apparatus according to claim 34, wherein the details of all or selected security products includes the name of the security products, versions, and loaded signature files.

37. (WITHDRAWN) A method of classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the method comprising:

classifying a first computer object as malware or as safe;

identifying in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and,

classifying said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

38. (WITHDRAWN) A method according to claim 37, comprising monitoring activities of said first computer object and reclassifying the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the method further comprising automatically classifying as malware any computer object that has a key in which said unique identifier component is present.

39. (WITHDRAWN) A computer program comprising program instructions for causing a computer to perform the method of claim 38.

40. (WITHDRAWN) Apparatus for classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the apparatus comprising:

a computer constructed and arranged to classify a first computer object as malware or as safe;

the computer being constructed and arranged to identify in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and,

the computer being constructed and arranged to classify said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

41. (WITHDRAWN) Apparatus according to claim 40, wherein the computer is constructed and arranged to monitor activities of said first computer object and reclassify the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the computer further being constructed and arranged to automatically classify as malware any computer object that has a key in which said unique identifier component is present.

42. (WITHDRAWN) A method of installing software on a computer, the method comprising:

on initiation of installation of software on a computer, providing a computer-generated prompt on the computer to a user to ascertain whether the user authorises the installation; and,

ceasing the installation if a user authorisation is not received, else:

receiving at the computer the user's authorisation to proceed with the installation;

proceeding with the installation;

obtaining data about computer objects that are created or used during the installation;

storing said data at the local computer.

43. (WITHDRAWN) A method according to claim 42, wherein the locally stored data is referred to during the installation to ensure that all objects created or used during the installation are part of the installation process, and, if it is found that objects created or used during the installation are not part of the installation process, either or both of: (i) ceasing the installation and (ii) providing a computer-generated prompt on the computer to the user accordingly.

44. (WITHDRAWN) A method according to claim 42, comprising connecting the computer to a community database that is connectable to a plurality of computers, and uploading the stored data to the community database for comparison with similar data provided by other computers.

45. (WITHDRAWN) A method according to claim 42, comprising downloading data about trusted installers to the computer, said data about trusted installers being referred to during the

installation such that any objects relating to or created by the trusted installer are automatically authorised to proceed.

46. (WITHDRAWN) A method according to claim 45, wherein said data about trusted installers is referred to only for a predetermined time period following receipt at the computer of the user's authorisation to proceed with the installation.

47. (WITHDRAWN) A computer program comprising program instructions for causing a computer to perform the method of claim 42.

48. (NEW) A method according to claim 1, further comprising:

receiving information at the base computer about the behaviour of the object running on at least one of the remote computers;

wherein the generating of the mask for the object comprises building the mask with the base computer in accordance with normal behaviour of the object determined from the received information.

49. (NEW) A method according to claim 1, wherein:

if the monitored operation of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, comparing at the base computer the non-permitted behaviour with the behaviour of at least one other object;

if the comparison indicates that the non-permitted behaviour is known to be not malicious for the at least one other object, allowing the non-permitted behaviour for the object; and

if the comparison indicates that the non-permitted behaviour is known to be not malicious for the at least one other object, modifying the mask to allow the non-permitted behaviour for the object in the future.

50. (NEW) An apparatus according to claim 13, wherein:

the base computer is constructed and arranged to receive information about the behaviour of the object running on at least one of the remote computers; and

the base computer is constructed and arranged to generate the mask in accordance with normal behaviour of the object determined from the received information.

51. (NEW) An apparatus according to claim 13, wherein:

the base computer is constructed and arranged to compare the non-permitted behaviour with the behaviour of at least one other object if the monitored operation of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask;

the base computer is constructed and arranged to allow the non-permitted behaviour for the object if the comparison indicates that the non-permitted behaviour is known to be not malicious for the at least one other object; and

the base computer is constructed and arranged to modify the mask to allow the non-permitted behaviour for the object in the future if the non-permitted behaviour is known to be not malicious for the at least one other object.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	339000-000003	4785

47604 7590 02/15/2011

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

EXAMINER	
ZECHER, CORDELIA P K	
ART UNIT	PAPER NUMBER
2432	

MAIL DATE	DELIVERY MODE
02/15/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/477,807	MORRIS ET AL.	
	Examiner	Art Unit	
	Cordelia Zecher	2432	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 January 2011*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-11,13-23,25-51* is/are pending in the application.

4a) Of the above claim(s) *31-47* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11,13-23,25-30 and 48-51* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on January 28, 2011 has been entered.

2. Claims 31 – 47 are withdrawn from further consideration pursuant to 37 CFR 1.142(b) as being drawn to nonelected Species 2 and 3, there being no allowable generic or linking claim. Election was made **without** traverse in the reply filed on April 9, 2010.

Response to Arguments

3. Applicant's arguments, see Remarks, filed January 28, 2011, with respect to 101 have been fully considered and are persuasive. Based on applicant's definition of a storage medium in the specification, and applicant's statement that a storage medium is a manufacture, the storage medium is considered statutory since it does not include transitory embodiments. The rejection of claim 30 under 101 has been withdrawn.

4. Applicant's arguments, see Remarks, filed January 28, 2011, with respect to the objection to the specification have been fully considered and are persuasive. The objection to the specification has been withdrawn.

5. Applicant's arguments with respect to generating the mask, and executing the object on the remote computer as defined by the mask have been considered but are moot in view of the new ground(s) of rejection.

6. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

Claim Rejections - 35 USC § 103

7. Claims 1 – 10, 13 – 22, 25 – 28, and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev's US Publication 2003/0177394 A1 in view of Lambert et al's US Publication 2002/0099952 A1.

8. Referring to claims 1 and 13, Dozortsev teaches:

a. At a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored (page 5, paragraph 45). While Dozortsev does not explicitly state that the central computer is connected to plural remote computers, Dozortsev does teach that the central computer is connected to at least one client computer (page 4, paragraph 34). The computer being connected to at least one computer fairly suggests that it can be connected to a plurality of remote computers.

b. Determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware (page 5, paragraph 45).

c. Classifying the computer object as malware when the data indicates that the computer object is malware (page 5, paragraph 45).

d. When the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware (page 5, paragraphs 46-47).

e. Running the computer object on at least one of the remote computers (page 5, paragraph 47).

f. Monitoring operation of said object and reclassifying the object as malware when the actual monitored behavior is malicious (pages 5, paragraph 49, page 6, paragraph 50).

2. Dozortsev does not explicitly teach generating a mask that defines acceptable behavior for the computer object, wherein only allowing the computer object to continue to run when the behavior of the computer object is permitted by the mask, and disallowing the computer object to run when the actual monitored behavior extends beyond that permitted by the mask. However, Lambert teaches that various software or suitable data structures are identified and classified with rule sets that automatically and transparently determine the ability of the software to execute, including a default rule that applies to any software file not classified under a specific rule (page 6, paragraph 52). Lambert goes on to teaches that the enforcement mechanism uses the security information to compute a restricted token (page 11, paragraph 83). The restricted token is equivalent to the claimed mask. The restricted token is then used to determine is a request is allowed or denied (page 6, paragraph 48-49).

3. Dozortsev and Lambert are from the same field of endeavor, malware detection. At the time of invention, it would have been obvious to one of ordinary skill in the art to modify the investigation of Dozortsev to include the restricted token of Lambert. The suggestion for doing so would have been to identify/classify software along with a highly flexible policy based infrastructure to enforce decisions about whether software can run, and if so, how much the software will be restricted when it runs (page 2, paragraph 9).

4. Referring to claims 2 and 14, Dozortsev teaches that the data about the computer object that is sent from the remote computer to the base computer and that is used in the comparison includes executable instructions contained within or constituted by the subject (page 3, paragraph 21).

5. Referring to claims 3 and 15, Dozortsev teaches comparing identifies relationships between the object and other objects (page 3, paragraph 22).

6. Referring to claims 4 and 16, Dozortsev teaches wherein if at least one other object to which said object is related is classified as malware, then classifying said object as malware (page 3, paragraph 25).

7. Referring to claims 5 and 17, Dozortsev teaches the other objects include objects stored on at least some of the remote computers (page 5, paragraphs 43-44).

8. Referring to claims 6 and 18, Dozortsev teaches that the other objects include otherwise process related objects (page 5, paragraph 44).

9. Referring to claims 7 and 19, Dozortsev teaches that the data is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on the remote computer (page 3, paragraph 21).

10. Referring to claims 8 and 20, Dozortsev teaches that the key has one component that represents executable instructions contained within or constituted by the object (page 3, paragraph 21).

11. Referring to claims 9 and 21, Dozortsev teaches that the key has at least one component that represents data about said object (page 3, paragraph 21).

12. Referring to claims 10, 22 and 28, Lambert teaches the data about the object includes the physical and folder location of the object on disk (page 7, paragraph 58). Dozortsev and Lambert are from the same field of endeavor, malware detection. At the time of invention, it would have been obvious to one of ordinary skill in the art to modify the signature of Dozortsev to include the full path name of Lambert. The suggestion for doing so would have been so the administrator can generally restrict user files, yet set the system files to fun normally, while being assured that the user or another program cannot simply defeat this setup by placing unknown files in the system directory (page 7, paragraph 58)..

13. Referring to claims 25 and 30, Dozortsev teaches:

g. Providing from a remote computer to a base computer data about a computer object that is stored on the remote computer (page 5, paragraph 45).

h. The data including executable instructions contained within or constituted by the object (page 3, paragraph 21).

i. The data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer (page 3, paragraph 21).

j. Executing the computer object at the remote computer (page 5, paragraph 47).

k. Monitoring operation of said object and reclassifying the object as malware when the actual monitored behavior is malicious (pages 5, paragraph 49, page 6, paragraph 50).

14. Dozortsev does not explicitly teach generating a mask that defines acceptable behavior for the computer object, wherein only allowing the computer object to continue to run when the behavior of the computer object is permitted by the mask, and disallowing the computer object to run when the actual monitored behavior extends beyond that permitted by the mask. However, Lambert teaches that various software or suitable data structures are identified and classified with rule sets that automatically and transparently determine the ability of the software to execute, including a default rule that applies to any software file not classified under a specific rule (page 6, paragraph 52). Lambert goes on to teaches that the enforcement mechanism uses the security information to compute a restricted token (page 11, paragraph 83). The restricted token is equivalent to the claimed mask. The restricted token is then used to determine is a request is allowed or denied (page 6, paragraph 48-49).

15. Dozortsev and Lambert are from the same field of endeavor, malware detection. At the time of invention, it would have been obvious to one of ordinary skill in the art to modify the investigation of Dozortsev to include the restricted token of Lambert. The suggestion for doing so would have been to identify/classify software along with a highly

flexible policy based infrastructure to enforce decisions about whether software can run, and if so, how much the software will be restricted when it runs (page 2, paragraph 9).

16. Referring to claim 26, Dozortsev teaches that the key has one component that represents executable instructions contained within or constituted by the object (page 3, paragraph 21).

17. Referring to claim 27, Dozortsev teaches that the key has at least one component that represents data about said object (page 3, paragraph 21).

18. Referring to claims 48 and 50, Lambert teaches

l. Receiving information about the behavior of the object running on at least one of the remote computers (page 11, paragraph 83). The enforcement mechanism receives the file information (page 11, paragraph 83). The file information is equivalent to the information about the behavior of the object.

m. Wherein the generating of the mask for the object comprises building the mask with the base computer in accordance with normal behavior of the object determined from the received information (page 11, paragraph 83). The security information is used to compute the restricted token (page 11, paragraph 83).

19. Claims 11, 23, and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Lambert as applied above, and further in view of Barton et al's US Publication 2005/0021994 A1. Dozortsev in view of Lambert discloses all the limitations of the parent claims. Dozortsev in view of Lambert does not explicitly disclose the data about the object including the size of the object. However, Barton

discloses indicating for the object the file name, size and location (pages 2-3, paragraph 39). Dozortsev in view of Lambert and Barton are analogous art because they are from the same field of endeavor, malware detection. At the time of the invention, it would have been obvious to one of ordinary skill in the art, having the teachings of Dozortsev in view of Lambert and Barton before him or her, to modify the signature of Dozortsev in view of Lambert to include the data size of Barton. The suggestion/motivation for doing so would have been the characteristics are strongly indicative of a particular file being one of the candidates (page 1, paragraph 14).

20. Claims 48 – 51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Lambert as applied above, and further in view of Gales US Publication 2003/0084323 A1.

21. Referring to claims 48 and 50, Dozortsev in view of Lambert discloses all the limitations of the parent claim. Dozortsev in view of Lambert does not explicitly disclose receiving information about the behavior of the object running on the remote computer, and generating the mask comprises building the mask with the base computer in accordance with normal behavior of the object. However, Gales discloses monitoring the network traffic and usage patters for a predetermined period of time, and then automatically generating a activity profile based on the monitored activity (page 3, paragraph 22). Dozortsev in view of Lambert and Gales are analogous art because they are from the same field of endeavor, computer security. At the time of the invention, it would have been obvious to one of ordinary skill in the art, having the teachings of

Dozortsev in view of Lambert and Gales before him or her, to modify the system of Dozortsev in view of Lambert to include the automatic profile generation of Gales. The suggestion for doing so would have been to be able to determine whether future network activity is acceptable or unacceptable (page 3, paragraph 18).

22. Referring to claims 49, and 51, Dozortsev in view of Lambert does not explicitly disclose if the object permits non-permitted behavior, comparing the non-permitted behavior with the behavior of another object, and if the comparison indicates the non-permitted behavior is known to be not malicious, allowing the non-permitted behavior and updating the mask to allow the non-permitted behavior. However, Gales discloses determining if a network event exceeds the activity profile (page 4, paragraph 29, step 214), if the event exceeds the profile then determining if the event is categorized in the event library as acceptable or authorized (page 4, paragraph 30, step 218), and when the event is acceptable or authorized, the activity profile is updated and the event is executed (page 4, paragraph 31).

23. Dozortsev in view of Lambert and Gales are analogous art because they are from the same field of endeavor, computer security. At the time of the invention, it would have been obvious to one of ordinary skill in the art, having the teachings of Dozortsev in view of Lambert and Gales before him or her, to modify the system of Dozortsev in view of Lambert to include the updating the profile of Gales. The suggestion/motivation for doing so would have been to continuously monitor and update networks usage and activity patterns to determine whether network events may constitute unauthorized usage or intrusion (page 4, paragraph 25).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Cordelia Zecher whose telephone number is (571)272-7771. The examiner can normally be reached on Monday - Thursday 8:00 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/C. Z./
Examiner, Art Unit 2432

/Gilberto Barron Jr./
Supervisory Patent Examiner, Art Unit 2432

Notice of References Cited	Application/Control No. 11/477,807	Applicant(s)/Patent Under Reexamination MORRIS ET AL.	
	Examiner Cordelia Zecher	Art Unit 2432	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2002/0099952	07-2002	Lambert et al.	713/200
*	B	US-2003/0084323	05-2003	Gales, George S.	713/200
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010	02/09/2011						
	1	✓	✓	✓						
	2	✓	✓	✓						
	3	✓	✓	✓						
	4	✓	✓	✓						
	5	✓	✓	✓						
	6	✓	✓	✓						
	7	✓	✓	✓						
	8	✓	✓	✓						
	9	✓	✓	✓						
	10	✓	✓	✓						
	11	✓	✓	✓						
	12	✓	-	-						
	13	✓	✓	✓						
	14	✓	✓	✓						
	15	✓	✓	✓						
	16	✓	✓	✓						
	17	✓	✓	✓						
	18	✓	✓	✓						
	19	✓	✓	✓						
	20	✓	✓	✓						
	21	✓	✓	✓						
	22	✓	✓	✓						
	23	✓	✓	✓						
	24	✓	-	-						
	25	✓	✓	✓						
	26	✓	✓	✓						
	27	✓	✓	✓						
	28	✓	✓	✓						
	29	✓	✓	✓						
	30	✓	✓	✓						
	31	N		N						
	32	N		N						
	33	N		N						
	34	N		N						
	35	N		N						
	36	N		N						

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010	02/09/2011						
	37	N		N						
	38	N		N						
	39	N		N						
	40	N		N						
	41	N		N						
	42	N		N						
	43	N		N						
	44	N		N						
	45	N		N						
	46	N		N						
	47	N		N						
	48			✓						
	49			✓						
	50			✓						
	51			✓						

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	1	of	2

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	MORRIS, Melvyn et al.
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -60/789,156			
		US -7,013,483	3/14/2006		
		US -2003/0023857	1/30/2003	Hinchlife	
		US -2005/0027686	2/3/2005	Shipp	
		US -6,338,141	1/8/2002	Wells	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 96/30829	10/3/1996	APPLE COMPUTER, INC.		
		WO 99/15966	4/1/1999	SYMANTEC CORPORATION		
		EP 1536341 A2	6/1/2005	MICROSOFT CORPORATION		

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional). 2 See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. 3 Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). 4 For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. 5 Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. 6 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

 ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	2	of	2

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	MORRIS, Melvyn et al.
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Image File Wrapper of U.S. Appl 11/694,261 as of January 28, 2011	
		"GHOSTBUSTER 3", Mihai CHIRIAC, pages 1-7, Prevx1 September 2005	
		A. Stephan; "Defeating Polymorphism: Beyond Emuliation"; Virus Bulletin Conference; October 2005	
Examiner Signature	/Cordelia Zecher/	Date Considered	02/09/2011

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

 ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L2	4	("20030023857" | "20050027686" | "6338141" | "7013483").PN.	US-PGPUB; USPAT	OR	ON	2011/02/09 08:21
L3	3433	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
L4	282	3 and mask$3	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
L5	35	3 and mask$3 and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
L6	328	3 and control$3 with environment	US-PGPUB; USPAT	OR	ON	2011/02/09 08:43
L7	65	3 and control$3 with environment and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:43
L8	18	3 and defin$3 with boundary and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:45
L9	5	3 and access$3 with rights with object and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:46
L10	2	"20020099952"	US-PGPUB; USPAT	OR	ON	2011/02/09 09:11
L11	24	(updat$3 or modif$4) with token and 3	US-PGPUB; USPAT	OR	ON	2011/02/09 10:54
L12	37	("20010049793" | "20030225698" | "20040068721" | "20050203819" | "20060031686" | "4713753" | "4805222" | "4926476" | "5367624" | "5481700" | "5819091" | "5915025" | "5958051" | "5978484" | "5983349" | "5991519" | "6295605" | "6378069" | "6405202" | "6775779" | "6931540" | "6981281" | "7028180" | "7051366" | "7076557").PN. OR ("7350204").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:23
L13	32	12 and (modif$4 or updat$3)	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:23

L14	6232	automatic$4 with (updat $3 or chang$3 or modif$4) with access	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:35
L15	29	3 and 14	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:35
S1	1	"20070016953"	US-PGPUB; USPAT	OR	ON	2010/01/07 08:58
S2	1	("20060026684").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 08:32
S3	1	S2 and malware	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
S4	1164	726/24.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
S5	2821	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:34
S6	6	(("6772346") or ("6944772") or ("20040006704") or ("20040073810") or ("20050021994") or ("20040153644")).PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 09:18
S7	1	("20030177394").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 10:48
S8	0	central near database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
S9	0	database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
S10	115859	database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
S11	2632	central near database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
S12	30	central near database with network with separate	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
S13	17	central near database with distribute with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
S14	199	central near database with distribut$3 with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54
S15	0	central near database with distribute with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54
S16	4	central near database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55
S17	240	database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55

S18	24	database with distribut$3 with network with known and (virus or malware)	US-PGPUB; USPAT	OR	ON	2010/04/28 12:56

EAST Search History (Interference)

< This search history is empty>

2/9/2011 12:35:56 PM
C:\ Documents and Settings\ ckane\ My Documents\ EAST\ Workspaces\ 11477807.wsp

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	Art Unit
	Cordelia Zecher	2432

SEARCHED			
Class	Subclass	Date	Examiner
726	24	4/28/10	cz

SEARCH NOTES		
Search Notes	Date	Examiner
Searched EAST (see attached)	2/9/11	cz
Inventor Search	4/28/10	cz

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

DOCKET NO. 339000-000003 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:	Melvyn Morris	Confirmation No.	4785
Application No.	11/477,807	Group Art Unit:	2432
Filed:	June 30, 2006	Examiner:	C. Zecher
For:	METHODS AND APPARATUS FOR DEALING WITH MALWARE		

AMENDMENT UNDER 37 C.F.R. 1.111

Assistant Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

June 15, 2011

Dear Sir:

This is in response to the Office Action dated February 15, 2011, and having a period for response set to expire on May 15, 2011. A petition for a one-month extension of time, together with the requisite fee for same, is submitted herewith, thereby extending the period for response to June 15, 2011.

The following amendments and remarks are respectfully submitted. Reconsideration of the claims is respectfully requested.

Amendments to the Claims begin on page 2 of this paper.

Remarks/Arguments begin on page 14 of this paper.

Remarks/Arguments

On page 3 of the Office Action, claims 1-10, 13-22, 25-28, and 30 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev (U.S. Pub. 2003/0177394) in view of Lambert et al. (U.S. Pub. 2002/0099952). Claims 1, 13, and 25 are independent claims, and claims 2-10, 14-22, 26-28, and 30 depend directly or indirectly on either claim 1, 13, or 25. Claims 1, 13, and 25 have been amended. The applicants respectfully traverse the rejection for at least the reason that Dozortsev in view of Lambert does not teach or suggest every limitation of amended claims 1, 13, and 25.

Claim 1 has been amended to include, *inter alia*, "receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;... automatically generating a mask for the computer object that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the object determined from said received data ". Claims 13 and 25 have been amended to include similar limitations. Dozortsev and Lambert fail to disclose at least these limitations.

The claims now require the received data to include information about the behaviour of the object, and the mask to be generated based on normal behaviour of the object determined from the received data (including behaviour information). For example, data may be accumulated about the behavour of objects as they run on one or more remote computers. This data may suggest that some or all of the monitored behaviour is normal behaviour, and a mask may be automatically generated based on the normal behaviour.

The Examiner argues that Dozortsev teaches receiving data about a computer object on page 5 par. [0045]. See Office Action page 3. This information is an executable code signature. Dozortsev teaches that this signature is a hash of the executable code itself, which may be compared to a database of hashes. "(T)he monitoring application will create a suspect executable code signature (using MD5 or another algorithm) and then compare the suspect executable code signature with entries in the local database(.)" Dozortsev page 5 par. [0044]. One of ordinary skill in the relevant art would know MD5 to be a common hashing algorithm. Dozortsev does not teach or suggest that the signature includes behaviour information.

The Examiner notes that Dozortsev does not teach generating a mask that defines acceptable behaviour for the computer object, allowing the object to run when its behaviour is permitted by the mask, and disallowing it to run when the actual monitored behaviour extends beyond that permitted by the mask. See Office Action page 4. The Examiner argues that Lambert discloses this by teaching defining rule sets for software and computing a restricted token that is equivalent to the claimed mask. See Office Action page 4. Lambert teaches that the criteria on which the rules are based include hash values, whether the file is digitally signed, file system or network path, origin or source, etc. See Lambert page 6 par. [0053]. Examples of policies and rules also appear in the table on page 10 par. [0082]. While Lambert discloses that other criteria are feasible (see Lambert page 6 par. [0082]), all examples and discussions of criteria in Lambert are directed to attributes of a program, rather than information about how the program behaves when it is run. There is no teaching or suggestion of receiving behaviour information and using normal behaviour to define a mask. The Examiner also notes that this is the case on page 9 of the Office Action. Additionally, Lambert teaches that an administrator classifies the software using

the criteria and sets the rules for the token. See Lambert page 6 par. [0052]-[0054]. This is clearly manual, and not automatic generation of a mask.

Dozortsev in view of Lambert does not teach or suggest receiving behaviour information about an object or automatically generating a mask in accordance with normal behaviour of the object determined from the received data. Therefore, claims 1, 13, and 25 should be allowed. Claims 2-10, 14-22, 26-28, and 30 should be allowed for at least the reasons given with respect to claims 1, 13, and 25.

On page 8 of the Office Action, claims 11, 23, and 29 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Lambert and Barton et al. (U.S. Pub. 2005/0021994). Claim 11 depends on claim 7, claim 23 depends on claim 19, and claim 29 depends on claim 25. Therefore, claims 11, 23, and 29 should be allowed for at least the same reasons as claims 7, 19, and 25. Furthermore, Barton fails to compensate for the deficiencies of Dozortsev and Lambert with respect to claims 1-10, 13-22, 25-28, and 30. Barton is directed towards malware scans of subsets of files known to be more likely to contain malware. See Barton page 1 par. [0013]. Barton does not teach or suggest receiving behaviour information about an object or automatically generating a mask in accordance with normal behaviour of the object determined from the received data.

On page 9 of the Office Action, claims 48-51 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Lambert and Gales (U.S. Pub. 2003/0084323). Claims 48 and 50 have been cancelled. Claim 49 depends on claim 1 and claim 51 depends on claim 13, so they should be allowed for at least the same reasons as claims 1 and 13. Furthermore, Gales fails to compensate for the deficiencies of Dozortsev and Lambert with

repect to claims 1-10, 13-22, 25-28, and 30. The Examiner notes that Gales discloses monitoring network traffic and usage patterns and automatically generating an activity profile based on the monitored activity on page 3 par. [0022]. However, Gales is monitoring network activity, not the behaviour of an object. Monitoring network activity could involve monitoring traffic levels and/or contents of packets, but it does not involve receiving information about the underlying behaviour causing or caused by the traffic levels and/or packet contents. Therefore, Gales does not teach or suggest suggest receiving behaviour information about an object or automatically generating a mask in accordance with normal behaviour of the object determined from the received data.

The Examiner argues that it would have been obvious to combine Dozortsev, Lambert, and Gales. See Office Action page 10. The applicants disagree, particularly with respect to Gales. Computer security is a wide ranging field, and Gales deals with a fundamentally different type of security from Dozortsev and Lambert. Dozortsev and Lambert are directed towards identifying and/or blocking malware running on remote computers (i.e. objects already present on the computers). Gales is directed towards preventing network intrusion from outside sources such as hackers, and monitors network traffic. See Gales page 2 par. [0013] ("intrusion detection system"). The applicants suggest that while it may seem reasonable to combine Gales with Dozortsev and Lambert in hindsight (given the present application), one of ordinary skill in the relevant art would not have done so on the basis of the prior art. Furthermore, Gales teaches against analysis of computer objects in favor of its own techniques of analyzing network traffic. See Gales page 1 par. [0004]. Therefore, one of ordinary skill in the relevant art would not be

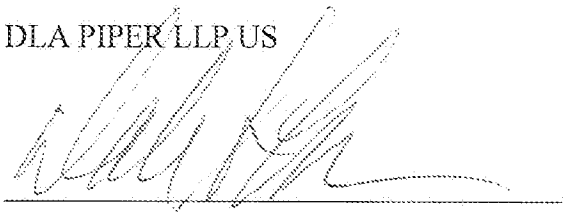
motivated to combine the teachings of Gales with those of Dozortsev and Lambert, which discuss analyzing various non-behavioural aspects of computer objects.

The application has been amended to place the application condition for allowance. All amendments are supported by the specification. An early notice of allowance is earnestly solicited.

Please charge any shortage in the fees or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US



Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/NP/jcc
PO Box 2758
Reston, VA 20195
(703) 773.4000 Telephone
(703) 773.5200 Fax

Electronic Patent Application Fee Transmittal

Application Number:	11477807
Filing Date:	30-Jun-2006
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Filer:	Dale S. Lazar/Josefina Cheng
Attorney Docket Number:	339000-000003

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 1 month with $0 paid	1251	1	130	130

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**130**

Electronic Acknowledgement Receipt

EFS ID:	10312429
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Dale S. Lazar/Josefina Cheng
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000003
Receipt Date:	15-JUN-2011
Filing Date:	30-JUN-2006
Time Stamp:	16:46:53
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$130
RAM confirmation Number	3649
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Extension of Time	339000-3EOT-061511.pdf	522120 2beb7c79f9c4f6ac9cce8090edae52c533b54417	no	1
Warnings:					
Information:					
2		339000-3Amendment-061511.pdf	6407626 2b69663b5400205dea88eb29bb874dc1d320b217	yes	18

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Amendment/Req. Reconsideration-After Non-Final Reject	1	1
	Claims	2	13
	Applicant Arguments/Remarks Made in an Amendment	14	18

Warnings:					
Information:					
3	Fee Worksheet (SB06)	fee-info.pdf	29849 0c9ada369e8e9989ac9aea227989b7b630f9bc19	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	6959595		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) **FY 2005** *(Fees pursuant to me Consolidated Appropriations Act, 2005 (H.R. 4818).)*	Docket Number 339000-000003
Application Number 11/477,807	Filed June 30, 2006
For METHODS AND APPARATUS FOR DEALING WITH MALWARE	
Art Unit 2432	Examiner C. Zecher

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Fee Code	
☒	One Month (37 CFR 1.1 7(a)(1))	$130	1251	130.00
☐	Two Month (37 CFR 1.1 7(a)(2))	$490	1252	
☐	Three Month (37 CFR 1.1 7(a)(3))	$1,110	1253	
☐	Four Month (37 CFR 1.1 7(a)(4))	$1730	1254	
☐	Five Month (37 CFR 1.1 7(a)(5))	$2,350	1255	

☐ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check in the amount of the fee is enclosed.

☒ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☐ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 50-3266. I have enclosed a duplicate copy of this sheet.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

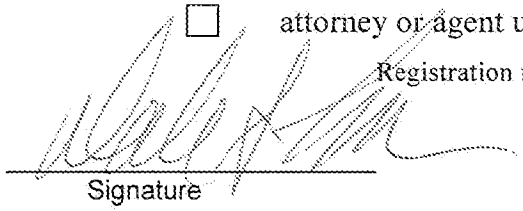
I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed (Form PTO/SB/96).

☒ attorney or agent of record. Registration Number 28,872

☐ attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 ___.



Signature

June 15, 2011
Date

Dale S. Lazar
Typed or printed name

703-773-4149
Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the more interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

☒ Total of 1 forms are submitted

This collection of information is required by 37 CFR 1.136(a). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S. C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 6 minutes to complete, including gathering, mating, and submitting the completed application form to the USPTO. Time will vary depending upon the Individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, US Patent and Trademark Office. U.S. Department OF Commerce, P.O. Box 1450, Alexandria, V 4 22313-4150. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS.
SEND TO: Commissioner for Patent. P.O. Box 1450, Alexandria, DA 22233-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2

American LegalNet, Inc
www.USCourtForms.com

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
11/477,807	06/30/2006	

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY ☐ RATE ($)	SMALL ENTITY ☐ FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT	06/15/2011	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	* 47	Minus	** 49	= 0	X $ =		OR	X $52=	0
	Independent (37 CFR 1.16(h))	* 8	Minus	***8	= 0	X $ =		OR	X $220=	0
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	0

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/ALLYSON PURNELL/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application. Currently amended claims are shown with additions underlined and deletions in ~~strikethrough text~~ except double brackets may be placed before and after the deleted characters to show deletion of five or fewer characters.

1. (CURRENTLY AMENDED) A method of classifying a computer object as malware, the method comprising:

at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;

determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware;

classifying the computer object as malware when the data indicates that the computer object is malware;

when the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware;

automatically generating a mask for the computer object that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the object determined from said received data;

running said object on at least one of the remote computers;

automatically monitoring operation of the object on the at least one of the remote computers;

allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask;

disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask; and,

reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask.

2. (PREVIOUSLY PRESENTED) A method according to claim 1, wherein the data about the computer object that is sent from the plural remote computers to the base computer includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

3. (PREVIOUSLY PRESENTED) A method according to claim 1, wherein determining identifies relationships between the object and other objects.

4. (ORIGINAL) A method according to claim 3, wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware.

5. (ORIGINAL) A method according to claim 3, wherein said other objects include the object or similar objects stored on at least some of the remote computers.

6. (ORIGINAL) A method according to claim 3, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

7. (ORIGINAL) A method according to claim 1, wherein the data is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on the respective remote computers.

8. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

9. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents data about said object.

10. (ORIGINAL) A method according to claim 9, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

11. (ORIGINAL) A method according to claim 7, wherein the key has at least one component that represents the physical size of the object.

12. (CANCELLED)

13. (CURRENTLY AMENDED) Apparatus for classifying a computer object as malware, the apparatus comprising:

a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;

the base computer being constructed and arranged to determine whether the data about the computer object received from said plural computers indicates that the computer object is malware; and,

the base computer being constructed and arranged to classify the computer object as malware when the base computer determines that the [[date]] data indicates that the computer object is malware.

wherein the base computer is constructed and arranged to automatically generate a mask for an object that is initially classed not as malware, said mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data, wherein operation of the object is automatically monitored when the object is running on at least one of the remote computers, the object being allowed to continue to run when behaviour of the object is permitted by the mask, the object not being permitted to run when the actual monitored behaviour of the object extends beyond that permitted by the mask, and the object is reclassified as malware when the actual monitored behaviour extends beyond that permitted by the mask.

14. (PREVIOUSLY PRESENTED) Apparatus according to claim 13, the data includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

15. (PREVIOUSLY PRESENTED) Apparatus according to claim 13, wherein the base computer is constructed and arranged so that the determining identifies relationships between the object and other objects.

16. (ORIGINAL) Apparatus according to claim 15, the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.

17. (ORIGINAL) Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least some of said remote computers.

18. (ORIGINAL) Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

19. (ORIGINAL) Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on said respective remote computers.

20. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

21. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents data about said object.

22. (ORIGINAL) Apparatus according to claim 21, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

23. (ORIGINAL) Apparatus according to claim 19, wherein the key has at least one component that represents the physical size of the object.

24. (CANCELLED)

25. (CURRENTLY AMENDED) A method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers, the method comprising:

providing from a remote computer to a base computer data about a computer object that is stored on the remote computer;

the data including information about the behaviour of the object running on one or more remote computers and including one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers;

the data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer,

automatically generating a mask for the computer object, the mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data;

executing the computer object at the remote computer;

automatically monitoring operation of the computer object at the remote computer compared to the behaviour permitted by the mask;

allowing the object to continue to run when behaviour of the computer object is permitted by the mask;

disallowing the object to run when the actual monitored behaviour of the object extends beyond that permitted by the mask; and,

reclassifying the object as malware when the actual monitored behaviour extends beyond hat permitted by the mask. wherein the base computer is arranged to receive information about the behaviour of the object running on one or more remote computers, wherein the step of generating a mask for that object comprises building the mask with the base computer in accordance with normal behaviour of the object determined from said received information.

26. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

27. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents data about said object.

28. (ORIGINAL) A method according to claim 27, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

29. (ORIGINAL) A method according to claim 25, wherein the key has at least one component that represents the physical size of the object.

30. (PREVIOUSLY PRESENTED) A storage medium storing a computer program comprising program instructions for causing a computer to perform the method of claim 25.

31. (WITHDRAWN) A method of determining the protection that a remote computer has from malware, the method comprising:

receiving at a base computer details of all or selected security products operating at a point in time on said remote computer;

receiving similar information from other remote computers connected to the base computer; and,

identifying any malware processes that were not identified by said other remote computers having that particular combination of security products.

32. (WITHDRAWN) A method according to claim 31, comprising providing information to the user of said remote computer that said remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

33. (WITHDRAWN) A method according to claim 31, wherein the details of all or selected security products includes the name of the security products, versions, and loaded signature files.

34. (WITHDRAWN) Apparatus for determining the protection that a remote computer has from malware, the apparatus comprising:

a base computer constructed and arranged to receive computer details of all or selected security products operating at a point in time on said remote computer;

the base computer being constructed and arranged to receive similar information from other remote computers connected to the base computer; and,

the base computer being constructed and arranged to identify any malware processes that were not identified by said other remote computers having that particular combination of security products.

35. (WITHDRAWN) Apparatus according to claim 34, wherein the base computer is constructed and arranged to provide information to the user of said remote computer that said remote computer may be susceptible to attack by said malware processes on the basis of said identifying.

36. (WITHDRAWN) Apparatus according to claim 34, wherein the details of all or selected security products includes the name of the security products, versions, and loaded signature files.

37. (WITHDRAWN) A method of classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the method comprising:

classifying a first computer object as malware or as safe;

identifying in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and,

classifying said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

38. (WITHDRAWN) A method according to claim 37, comprising monitoring activities of said first computer object and reclassifying the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the method further comprising automatically classifying as malware any computer object that has a key in which said unique identifier component is present.

39. (WITHDRAWN) A computer program comprising program instructions for causing a computer to perform the method of claim 38.

40. (WITHDRAWN) Apparatus for classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the apparatus comprising:

a computer constructed and arranged to classify a first computer object as malware or as safe;

the computer being constructed and arranged to identify in a key relating to said first computer object a component that uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and,

the computer being constructed and arranged to classify said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

41. (WITHDRAWN) Apparatus according to claim 40, wherein the computer is constructed and arranged to monitor activities of said first computer object and reclassify the first computer object as malware in the case that it was initially classified as safe and subsequently determined to be malware, the computer further being constructed and arranged to automatically classify as malware any computer object that has a key in which said unique identifier component is present.

42. (WITHDRAWN) A method of installing software on a computer, the method comprising:

• on initiation of installation of software on a computer, providing a computer-generated prompt on the computer to a user to ascertain whether the user authorises the installation; and,

ceasing the installation if a user authorisation is not received, else:

receiving at the computer the user's authorisation to proceed with the installation;

proceeding with the installation;

obtaining data about computer objects that are created or used during the installation;

storing said data at the local computer.

43. (WITHDRAWN) A method according to claim 42, wherein the locally stored data is referred to during the installation to ensure that all objects created or used during the installation are part of the installation process, and, if it is found that objects created or used during the installation are not part of the installation process, either or both of: (i) ceasing the installation and (ii) providing a computer-generated prompt on the computer to the user accordingly.

44. (WITHDRAWN) A method according to claim 42, comprising connecting the computer to a community database that is connectable to a plurality of computers, and uploading the stored data to the community database for comparison with similar data provided by other computers.

45. (WITHDRAWN) A method according to claim 42, comprising downloading data about trusted installers to the computer, said data about trusted installers being referred to during the installation such that any objects relating to or created by the trusted installer are automatically authorised to proceed.

46. (WITHDRAWN) A method according to claim 45, wherein said data about trusted installers is referred to only for a predetermined time period following receipt at the computer of the user's authorisation to proceed with the installation.

47. (WITHDRAWN) A computer program comprising program instructions for causing a computer to perform the method of claim 42.

48. (CANCELLED)

49. (PREVIOUSLY PRESENTED) A method according to claim 1, comprising if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, comparing at the base computer the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects, allowing said behaviour for the object and modifying the mask to allow that behaviour for that object in future.

50. (CANCELLED)

51. (PREVIOUSLY PRESENTED) A method according to claim 13, wherein, if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, the base computer is arranged to compare the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects the base computer is arranged to allow said behaviour for the object and to modify the mask to allow that behaviour for that object in future.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	339000-000003	4785

47604 7590 07/25/2011

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

EXAMINER	
ZECHER, CORDELIA P K	
ART UNIT	PAPER NUMBER
2432	

MAIL DATE	DELIVERY MODE
07/25/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/477,807	MORRIS ET AL.	
	Examiner	Art Unit	
	CORDELIA ZECHER	2432	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *15 June 2011*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-11,13-23,25-30,49 and 51* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11,13-23,25-30,49 and 51* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Response to Arguments

1. Applicant's arguments with respect to the Gales reference have been fully considered but they are not persuasive.

2. In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

3. In response to applicant's argument that there is no teaching, suggestion, or motivation to combine the references, the examiner recognizes that obviousness may be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988), *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992), and *KSR International Co. v. Teleflex, Inc.*, 550 U.S. 398, 82 USPQ2d 1385 (2007). In this case, the suggestion for doing so would have been to determine if future activity is acceptable or unacceptable (paragraph 18).

4. Applicant's arguments with respect to claims 1 – 11, 13 – 23, 25 – 30, 49 and 51 have been considered but are moot in view of the new ground(s) of rejection.

5. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

Claim Rejections - 35 USC § 103

6. Claims 1 – 10, 13 – 22, 25 – 28, and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev's US Publication 2003/0177394 A1 in view of Kester et al's US Publication 2005/0210035 A1.

7. Referring to claims 1 and 13, Dozortsev teaches:

a. At a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored (page 5, paragraph 45). While Dozortsev does not explicitly state that the central computer is connected to plural remote computers, Dozortsev does teach that the central computer is connected to at least one client computer (page 4, paragraph 34). The computer being connected to at least one computer fairly suggests that it can be connected to a plurality of remote computers.

b. Determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware (page 5, paragraph 45).

c. Classifying the computer object as malware when the data indicates that the computer object is malware (page 5, paragraph 45).

d. When the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware (page 5, paragraphs 46-47).

e. Running the computer object on at least one of the remote computers (page 5, paragraph 47).

f. Monitoring operation of said object and reclassifying the object as malware when the actual monitored behavior is malicious (pages 5, paragraph 49, page 6, paragraph 50).

2. Dozortsev does not explicitly teach receiving data about the behavior of the object running on the remote computer, automatically generating a mask in accordance with normal behavior of the object determined from the received data that defines acceptable behavior for the computer object, and only allowing the computer object to continue to run when the behavior of the computer object is permitted by the mask, and disallowing the computer object to run when the actual monitored behavior extends beyond that permitted by the mask. However, Kester the workstation management module monitoring the behavior of an application (paragraph 41), wherein the behavior of the application is used to determine the access privileges for the application (paragraphs 43-44). Kester goes on to teach that the privileges are used to determine what actions are permitted (paragraphs 55 and 56).

3. Dozortsev and Kester are from the same field of endeavor, monitoring for malware. At the time of invention, it would have been obvious to one of ordinary skill in the art to modify the investigation of Dozortsev to include the privileges based on behavior of Kester. The suggestion for doing so would have been for the system to be able to keep or change a previous access privilege based on the subsequent activity of an application (paragraph 41).

4. Referring to claims 2 and 14, Dozortsev teaches that the data about the computer object that is sent from the remote computer to the base computer and that is used in the comparison includes executable instructions contained within or constituted by the subject (page 3, paragraph 21).

5. Referring to claims 3 and 15, Dozortsev teaches comparing identifies relationships between the object and other objects (page 3, paragraph 22).

6. Referring to claims 4 and 16, Dozortsev teaches wherein if at least one other object to which said object is related is classified as malware, then classifying said object as malware (page 3, paragraph 25).

7. Referring to claims 5 and 17, Dozortsev teaches the other objects include objects stored on at least some of the remote computers (page 5, paragraphs 43-44).

8. Referring to claims 6 and 18, Dozortsev teaches that the other objects include otherwise process related objects (page 5, paragraph 44).

9. Referring to claims 7 and 19, Dozortsev teaches that the data is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on the remote computer (page 3, paragraph 21).

10. Referring to claims 8 and 20, Dozortsev teaches that the key has one component that represents executable instructions contained within or constituted by the object (page 3, paragraph 21).

11. Referring to claims 9 and 21, Dozortsev teaches that the key has at least one component that represents data about said object (page 3, paragraph 21).

12. Referring to claims 10, 22 and 28, Kester teaches the data about the object includes the events initiated or involving the object when the object is created, configured or runs on the respective remote computers (paragraph 43). Dozortsev and Kester are from the same field of endeavor, monitoring for malware. At the time of invention, it would have been obvious to one of ordinary skill in the art to modify the investigation of Dozortsev to include the privileges based on behavior of Kester. The suggestion for doing so would have been for the system to be able to keep or change a previous access privilege based on the subsequent activity of an application (paragraph 41).

13. Referring to claims 25 and 30, Dozortsev teaches:

g. Providing from a remote computer to a base computer data about a computer object that is stored on the remote computer (page 5, paragraph 45).

h. The data including executable instructions contained within or constituted by the object (page 3, paragraph 21).

i. The data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer (page 3, paragraph 21).

j. Executing the computer object at the remote computer (page 5, paragraph 47).

k. Monitoring operation of said object and reclassifying the object as malware when the actual monitored behavior is malicious (pages 5, paragraph 49, page 6, paragraph 50).

14. Dozortsev does not explicitly teach receiving data about the behavior of the object running on the remote computer, automatically generating a mask in accordance with normal behavior of the object determined from the received data that defines acceptable behavior for the computer object, and only allowing the computer object to continue to run when the behavior of the computer object is permitted by the mask, and disallowing the computer object to run when the actual monitored behavior extends beyond that permitted by the mask. However, Kester the workstation management module monitoring the behavior of an application (paragraph 41), wherein the behavior of the application is used to determine the access privileges for the application (paragraphs 43-44). Kester goes on to teach that the privileges are used to determine what actions are permitted (paragraphs 55 and 56).

15. Dozortsev and Kester are from the same field of endeavor, monitoring for malware. At the time of invention, it would have been obvious to one of ordinary skill in the art to modify the investigation of Dozortsev to include the privileges based on behavior of Kester. The suggestion for doing so would have been for the system to be able to keep or change a previous access privilege based on the subsequent activity of an application (paragraph 41).

16. Referring to claim 26, Dozortsev teaches that the key has one component that represents executable instructions contained within or constituted by the object (page 3, paragraph 21).

17. Referring to claim 27, Dozortsev teaches that the key has at least one component that represents data about said object (page 3, paragraph 21).

18. Claims 11, 23, and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Kester as applied above, and further in view of Barton et al's US Publication 2005/0021994 A1. Dozortsev in view of Kester discloses all the limitations of the parent claims. Dozortsev in view of Kester does not explicitly disclose the data about the object including the size of the object. However, Barton discloses indicating for the object the file name, size and location (pages 2-3, paragraph 39). Dozortsev in view of Kester and Barton are analogous art because they are from the same field of endeavor, malware detection. At the time of the invention, it would have been obvious to one of ordinary skill in the art, having the teachings of Dozortsev in view of Kester and Barton before him or her, to modify the signature of Dozortsev in view of Kester to include the data size of Barton. The suggestion/motivation for doing so would have been the characteristics are strongly indicative of a particular file being one of the candidates (page 1, paragraph 14).

19. Claims 49 and 51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Kester as applied above, and further in view of Gales US Publication 2003/0084323 A1.

20. Referring to claims 49, and 51, Dozortsev in view of Kester does not explicitly disclose if the object permits non-permitted behavior, comparing the non-permitted behavior with the behavior of another object, and if the comparison indicates the non-permitted behavior is known to be not malicious, allowing the non-permitted behavior and updating the mask to allow the non-permitted behavior. However, Gales discloses determining if a network event exceeds the activity profile (page 4, paragraph 29, step 214), if the event exceeds the profile then determining if the event is categorized in the event library as acceptable or authorized (page 4, paragraph 30, step 218), and when the event is acceptable or authorized, the activity profile is updated and the event is executed (page 4, paragraph 31).

21. Dozortsev in view of Kester and Gales are analogous art because they are from the same field of endeavor, computer security. At the time of the invention, it would have been obvious to one of ordinary skill in the art, having the teachings of Dozortsev in view of Kester and Gales before him or her, to modify the system of Dozortsev in view of Kester to include the updating the profile of Gales. The suggestion/motivation for doing so would have been to continuously monitor and update networks usage and activity patterns to determine whether network events may constitute unauthorized usage or intrusion (page 4, paragraph 25).

Conclusion

22. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to CORDELIA ZECHER whose telephone number is (571)272-7771. The examiner can normally be reached on Monday - Thursday 8:00 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/CORDELIA ZECHER/
Examiner, Art Unit 2432

/Benjamin E Lanier/
Primary Examiner, Art Unit 2432

Notice of References Cited	Application/Control No. 11/477,807	Applicant(s)/Patent Under Reexamination MORRIS ET AL.	
	Examiner CORDELIA ZECHER	Art Unit 2432	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2005/0210035	09-2005	Kester et al.	707/010
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L4	18	"7350204"	US-PGPUB; USPAT	OR	ON	2011/07/21 09:19
L5	1	"7350204".pn.	US-PGPUB; USPAT	OR	ON	2011/07/21 09:19
L8	42	("20010049793" | "20030225698" | "20040068721" | "20050203819" | "20060031686" | "4713753" | "4805222" | "4926476" | "5367624" | "5481700" | "5819091" | "5915025" | "5958051" | "5978484" | "5983349" | "5991519" | "6295605" | "6378069" | "6405202" | "6775779" | "6931540" | "6981281" | "7028180" | "7051366" | "7076557").PN. OR ("7350204").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:21
L9	106	("2002/0099952").URPN.	USPAT	OR	ON	2011/07/21 09:46
L10	16	("20020026578" | "20020099952" | "20020124170" | "20030033535" | "20040064708" | "20040078604" | "20040117615" | "20050138388" | "5941947" | "6151590" | "6397261" | "6571335" | "6810525" | "7010600" | "7194664").PN. OR ("7509497").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:49
L11	1	(8 or 9 or 10) and behaviour	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:50
L12	1	"7698713".pn.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:51

L13	2	"20030131152"	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:52
L14	57	(8 or 9 or 10) and behavior	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:52
L15	2	"20050210035"	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 10:01
L16	2	15 and behav$5	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 10:04
S1	1	"20070016953"	US-PGPUB; USPAT	OR	ON	2010/01/07 08:58
S2	1	("20060026684").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 08:32
S3	1	S2 and malware	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
S4	1164	726/24.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
S5	2821	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:34
S6	6	(("6772346") or ("6944772") or ("20040006704") or ("20040073810") or ("20050021994") or ("20040153644")).PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 09:18
S7	1	("20030177394").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 10:48
S8	0	central near database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
S9	0	database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
S10	115859	database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
S11	2632	central near database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
S12	30	central near database with network with separate	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
S13	17	central near database with distribute with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
S14	199	central near database with distribut$3 with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54
S15	0	central near database with distribute with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54

S16	4	central near database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55
S17	240	database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55
S18	24	database with distribut$3 with network with known and (virus or malware)	US-PGPUB; USPAT	OR	ON	2010/04/28 12:56
S19	4	("20030023857" | "20050027686" | "6338141" | "7013483").PN.	US-PGPUB; USPAT	OR	ON	2011/02/09 08:21
S20	3433	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S21	282	S20 and mask$3	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S22	35	S20 and mask$3 and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S23	328	S20 and control$3 with environment	US-PGPUB; USPAT	OR	ON	2011/02/09 08:43
S24	65	S20 and control$3 with environment and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:43
S25	18	S20 and defin$3 with boundary and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:45
S26	5	S20 and access$3 with rights with object and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:46
S27	2	"20020099952"	US-PGPUB; USPAT	OR	ON	2011/02/09 09:11
S28	24	(updat$3 or modif$4) with token and S20	US-PGPUB; USPAT	OR	ON	2011/02/09 10:54
S29	37	("20010049793" | "20030225698" | "20040068721" | "20050203819" | "20060031686" | "4713753" | "4805222" | "4926476" | "5367624" | "5481700" | "5819091" | "5915025" | "5958051" | "5978484" | "5983349" | "5991519" | "6295605" | "6378069" | "6405202" | "6775779" | "6931540" | "6981281" | "7028180" | "7051366" | "7076557").PN. OR ("7350204").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:23

S30	32	S29 and (modif$4 or updat $3)	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:23
S31	6232	automatic$4 with (updat $3 or chang$3 or modif$4) with access	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:35
S32	29	S20 and S31	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:35

EAST Search History (Interference)

< This search history is empty>

7/ 21/ 2011 11:11:01 AM
C:\ Documents and Settings\ ckane\ My Documents\ EAST\ Workspaces\ 11477807.wsp

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010	02/09/2011	07/21/2011					
	1	✓	✓	✓	✓					
	2	✓	✓	✓	✓					
	3	✓	✓	✓	✓					
	4	✓	✓	✓	✓					
	5	✓	✓	✓	✓					
	6	✓	✓	✓	✓					
	7	✓	✓	✓	✓					
	8	✓	✓	✓	✓					
	9	✓	✓	✓	✓					
	10	✓	✓	✓	✓					
	11	✓	✓	✓	✓					
	12	✓	-	-	-					
	13	✓	✓	✓	✓					
	14	✓	✓	✓	✓					
	15	✓	✓	✓	✓					
	16	✓	✓	✓	✓					
	17	✓	✓	✓	✓					
	18	✓	✓	✓	✓					
	19	✓	✓	✓	✓					
	20	✓	✓	✓	✓					
	21	✓	✓	✓	✓					
	22	✓	✓	✓	✓					
	23	✓	✓	✓	✓					
	24	✓	-	-	-					
	25	✓	✓	✓	✓					
	26	✓	✓	✓	✓					
	27	✓	✓	✓	✓					
	28	✓	✓	✓	✓					
	29	✓	✓	✓	✓					
	30	✓	✓	✓	✓					
	31	N		N	N					
	32	N		N	N					
	33	N		N	N					
	34	N		N	N					
	35	N		N	N					
	36	N		N	N					

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010	02/09/2011	07/21/2011					
	37	N		N	N					
	38	N		N	N					
	39	N		N	N					
	40	N		N	N					
	41	N		N	N					
	42	N		N	N					
	43	N		N	N					
	44	N		N	N					
	45	N		N	N					
	46	N		N	N					
	47	N		N	N					
	48			✓	-					
	49			✓	✓					
	50			✓	-					
	51			✓	✓					

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

SEARCHED			
Class	**Subclass**	**Date**	**Examiner**
726	24	4/28/10	cz

SEARCH NOTES		
Search Notes	**Date**	**Examiner**
Searched EAST (see attached)	7/21/11	cz
Inventor Search	4/28/10	cz

INTERFERENCE SEARCH			
Class	**Subclass**	**Date**	**Examiner**

PTO/SB/30 (04-05)
Approved for use through 02/29/2008. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number

REQUEST FOR CONTINUED EXAMINATION (RCE) TRANSMITTAL

Address to:
Mail Stop RCE
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

This is a Request for Continued Examination (RCE) under 37 CFR 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. See Instruction Sheet for RCEs (not to be submitted to the USPTO) on page 2.

1. **Submission required under 37 CFR 1.114** Note: If the RCE is proper, any previously filed unentered amendments and amendments enclosed with the RCE will be entered in the order in which they were filed unless applicant instructs otherwise. If applicant does not wish to have any previously filed unentered amendment(s) entered, applicant must request non-entry of such amendment(s).

   a ☐ Previously submitted. If a final Office action is outstanding, any amendments filed after the final Office action may be considered as a submission even if this box is not checked.
   - (i) ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on
   - (ii) ☐ Other

   b ☒ Enclosed
   - (i) ☒ Amendment/Reply
   - (ii) ☒ Information Disclosure Statement (IDS)
   - (iii) ☐ Affidavit(s)/Declarations
   - (iv) ☐ Other _____

2. **Miscellaneous**

   a ☐ Suspension of action on the above-identified application is requested under 37 CFR 1.103(c) for a period of ___ months. (Period of suspension shall not exceed 3 months; Fee under 37 CFR 1.17(i) required)

   b ☐ Other

3. **Fees** The RCE fee under 37 CFR 1.17(e) is required by 37 CFR 1.114 when the RCE is filed.

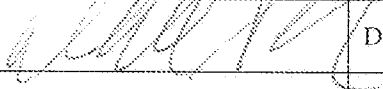
   a ☐ The Director is hereby authorized to charge the following fees, any underpayment of fees, or credit any overpayments, to Deposit Account No. 50-3266. I have enclosed a duplicate copy of this sheet.
   - (i) ☐ RCE fee required under 37 CFR 1.17(e)
   - (ii) ☐ Extension of time fee (37 CFR 1.136 and 1.17)
   - (iii) ☐ Other

   b ☐ Check in the amount of ___ enclosed

   c ☒ Payment by credit card (Form PTO-2038 enclosed)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT, REQUIRED

Signature		Date	December 23, 2011
Name (Print/Type)	Dale S. Lazar	Registration No.	28,872

This collection of information is required by 37 CFR 1.114. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a) **FY 2005** *(Fees pursuant to me Consolidated Appropriations Act, 2005 (H.R. 4818).)*	Docket Number 339000-000003
Application Number 11/477,807	Filed June 30, 2006
For METHODS AND APPARATUS FOR DEALING WITH MALWARE	
Art Unit 2432	Examiner C. Zecher

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

		Fee	Fee Code	
☐	One Month (37 CFR 1.1 7(a)(1))	$150	1251	
☒	Two Month (37 CFR 1.1 7(a)(2))	$560	1252	$560.00
☐	Three Month (37 CFR 1.1 7(a)(3))	$1270	1253	
☐	Four Month (37 CFR 1.1 7(a)(4))	$1980	1254	
☐	Five Month (37 CFR 1.1 7(a)(5))	$2690	1255	

☐ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check in the amount of the fee is enclosed.

☒ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 50-3266. I have enclosed a duplicate copy of this sheet.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

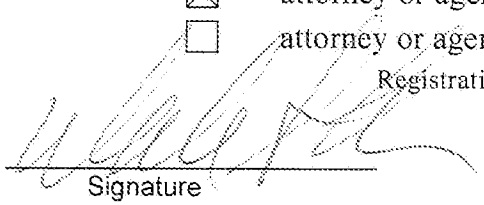
I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) is enclosed (Form PTO/SB/96).

☒ attorney or agent of record. Registration Number 28,872

☐ attorney or agent under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34 ___.



Signature

December 23, 2011
Date

Dale S. Lazar
Typed or printed name

703-773-4149
Telephone Number

NOTE: Signatures of all the inventors or assignees of record of the more interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below.

☒ Total of 1 forms are submitted

This collection of information is required by 37 CFR 1.136(a). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S. C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 6 minutes to complete, including gathering, mating, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, US Patent and Trademark Office. U.S. Department OF Commerce, P.O. Box 1450, Alexandria, V.4 22313-4150. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patent. P.O. Box 1450, Alexandria, DA 22233-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2

American LegalNet, inc
www.USCourtForms.com

EAST\47673867.1

DOCKET NO. 339000-000003 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:	Melvyn Morris	Confirmation No.	4785
Application No.	11/477,807	Group Art Unit:	2432
Filed:	June 30, 2006	Examiner:	C. Zecher

For: METHODS AND APPARATUS FOR DEALING WITH MALWARE

RESPONSE UNDER 37 CFR 1.114

Assistant Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

December 23, 2011

Dear Sir:

This is in response to the Final Office Action dated July 25, 2011, and having a period for response set to expire on October 25, 2011. A petition for a two-month extension of time, together with the requisite fee for same, is submitted herewith, thereby extending the period for response to December 25, 2011 .

The following amendments and remarks are respectfully submitted. Reconsideration of the claims is respectfully requested.

Remarks/Arguments begin on page 2 of this paper.

Remarks/Arguments

On page 3 of the Office Action, claims 1-10, 13-22, 25-28, and 30 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev (U.S. Pub. 2003/0177394) in view of Kester et al. (U.S. Pub. 2005/0210035). Claims 1, 13, and 25 are independent claims, and claims 2-10, 14-22, 26-28, and 30 depend directly or indirectly on either claim 1, 13, or 25. The applicants respectfully traverse the rejection for at least the reason that Dozortsev in view of Kester does not teach or suggest every limitation of amended claims 1, 13, and 25.

Claim 1 includes, *inter alia*, "receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behavior of the object running on one or more remote computers;... automatically generating a mask for the computer object that defines acceptable behavior for the computer object, wherein the mask is generated in accordance with normal behavior of the object determined from said received data; running said object on at least one of the remote computers; automatically monitoring operation of the object on the at least one of the remote computers; allowing the computer object to continue to run when behavior of the computer object is permitted by the mask; disallowing the computer object to run when the actual monitored behavior of the computer object extends beyond that permitted by the mask; and reclassifying the computer object as malware when the actual monitored behavior extends beyond that permitted by the mask." (emphasis added). Claims 13 and 25 include similar limitations. Dozortsev and Kester fail to disclose at least these limitations.

The claims require the received data to include information about the behavior of the object, and the mask to be generated based on normal behavior of the object determined from the received data (including behavior information). For example, data may be accumulated about the behavior of objects as they run on one or more remote computers. This data may suggest that some or all of the monitored behavior is normal behavior, and a mask may be automatically generated based on the normal behavior.

The Examiner notes that Dozortsev does not teach receiving data about the behavior of an object running on a remote computer, automatically generating a mask that defines acceptable behavior for the computer object, allowing the object to run when its behavior is permitted by the

mask, and disallowing it to run when the actual monitored behavior extends beyond that permitted by the mask. See Office Action page 4. The Examiner argues that Kester teaches these features. The applicants disagree.

Kester teaches a computer implemented system which allows a company to control what programs employees are running on their workstations and how they access the network. See Kester par. [0007]-[0011]. Each user workstation 101 has a "Workstation Management Module" 200 which monitors requests for programs to be run and/or requests for already running programs to access the network. The Workstation Management Module 200 looks up in a hash/policy database 204 to find the policy or rule to apply for that program/network access request. Monitored behavior and/or activity data is stored in the hash/policy database 204 as a "hash." See Kester par. [0051]-[0052]. A policy may be associated with a hash, but it is distinct from the hash. See Kester par. [0054]. According to the policy (not the hash), the Workstation Management Module 200 allows or disallows the program to run/access the network. See Kester par. [0039]-[0040] and [0055]-[0056].

The policy decisions about what programs/network access is allowed are made at the "Application Server Module" 102. Information about programs running on workstations and network requests is logged at the Workstation Management Module 200 and sent to the Application Server Module. See Kester par. [0067]. The Application Server Module 102 provides a user interface 106 by which the company's Network Administrator can view this data and then classify the programs/network access requests, and create policies for the programs/network access requests or for the classes of programs/network access requests. See Kester par. [0071]-[0072]. These classifications and policies are then propagated back to the Workstation Management Module 200, stored in the hash/policy table, and are applied subsequently by the Workstation Management Module 200 to determine whether to allow or disallow user attempts to run the program or access the network. See Kester par. [0077]. Additionally, the programs/network access requests can be classified by a human reviewer at an Application Database Factory 110 which communicates with the Application Server Modules 102 to receive information about the programs and return their classifications once made. See Kester par. [0076]. Importantly, whether at the Application Server Module 102 or at the Application Database Factory 110, a human Network Administrator/reviewer makes all of the

policy decisions regarding whether programs and/or network access should be allowed. See Kester par. [0085] and [0089].

In more detail, Kester par. [0071] teaches how a Network Administrator can classify applications at the Application Server Module 102 (e.g. Internet Explorer could be classified as a web browser). Kester par. [0072] teaches how the Network Administrator creates access privileges/policies/rules for users, either for individual programs or for classes of programs (e.g. a particular user may or may not be allowed to run a web browser at his workstation). This allows a program to be associated with a policy at the application server module 201 according to the classification given to it by the Network Administrator. A set of hashes/policies for the programs on workstation is then downloaded to the workstation and populates the hash/policy database. See Kester par. [0077]. When the program is run, the appropriate policy will be used by the Workstation Management Module 200 to allow or disallow the program to run.

A similar system is employed for monitoring and determining network access by a program. The policy/hash table 204 at the user's workstation 101 contains entries for "expected behavior" of the programs, i.e. previously encountered network access requests with which a policy has been associated by the Network Administrator. The policy can be to allow the access or disallow the access, etc. See Kester par. [0143]. Details of network access requests are sent from the Workstation Management Module 200 to the Application Server Module 102. A Network Administrator at the Application Server Module 102 makes a decision as to what network access is allowed. See Kester par. [0163]-[0164]. Additionally, a human analyst at the Application Database Factory 110 can decide what network access is allowed. The human reviewer can analyze the network access data to determine a category for the application, e.g. expected or unexpected network access. These can then be associated with policies or rules. See Kester par. [0178]. As part of this process, the user interface 302 can display to the reviewer the "expected network activity" for a program, which can be derived from previously seen behavior for the program. This information helps the human reviewer analyze the application and determine its expected behavior See Kester par. [0179]. The human reviewer then decides what behavior is allowed for the application. See Kester par. [0187]. These determinations are then fed back to the hash/policy table 204.

Thus, in summary, in Kester the hash/policy table 204 for each workstation stores details of previously seen programs (categorized programs) and previously seen network access requests (expected network access behavior) and policies for those programs and requests that have been set by the Network Administrator/human reviewer. The policies, not the hash data about programs and network access requests, determine whether a program can be run or whether a network access request is allowable. These policies may be set by the Network Administrator/human reviewer at the level of the individual program/network access request or at the level of the classification given to the program/network access request. This provides a way for a Network Administrator of a company to control what programs employees can run and what network access the programs can make and implement appropriate policies for that company. Kester also does not teach or suggest reclassifying an object as malware if its monitored behavior extends beyond that permitted by the mask. Kester has no teachings relating to classifying an object as malware at all.

This is clearly different from the claims of the present invention. The present invention is concerned with a method of classifying a computer object as malware. As part of this process, a mask is automatically generated defining acceptable behavior for the object based on the normal behavior of the object. The automatically generated mask is used to automatically monitor the operation of the object and to reclassify the object as malware if the behavior extends beyond the mask. A human decision is not necessary to generate the mask or monitor the operation.

Furthermore, one of ordinary skill in the relevant art would not combine Dozortsev and Kester. The arrangement of Kester is impractical for identifying malware for several reasons. First, the involvement of a human analyst means that policy decisions take time to be made. It also takes a large amount of time and resources for a human analyst to analyze the large amounts of data and research the programs and make policy decisions. It is also difficult to scale the system to a large number of users, since of course this increases the burden on the network administrator.

Kester is concerned with a system to allow an employee to control what their employees can do on their computers. This is in essence a matter of individual policy for each company and may vary between companies or even between users in a company. It is therefore essential to have the Network Administrator review each program for each user to decide what policy is

appropriate for their company. Thus, a skilled person would have no motivation to adapt Kester to result in the present invention to automatically generate a mask, as to do so would mean that the system of Kester would no longer be able to perform its function of allowing a company to implement appropriate policies governing what their employees can do on their computers.

Dozortsev is concerned with detecting malware. Kester is concerned with allowing a company to implement policies to control what programs and network access their employees can make on their computers. These are fundamentally different problems. One of ordinary skill in the relevant art attempting to cure the deficiencies in Dozortsev would have no motivation to look to Kester for relevant teachings as the two documents are in different fields and are concerned with different problems.

Dozortsev in view of Kester fails to teach or suggest at least "automatically generating a mask for the computer object that defines acceptable behavior for the computer object, wherein the mask is generated in accordance with normal behavior of the object determined from said received data… allowing the computer object to continue to run when behavior of the computer object is permitted by the mask; disallowing the computer object to run when the actual monitored behavior of the computer object extends beyond that permitted by the mask; and reclassifying the computer object as malware when the actual monitored behavior extends beyond that permitted by the mask." Also, no one of ordinary skill in the relevant art would combine Dozortsev and Kester. For at least these reasons, claims 1, 13, and 25 should be allowed. Claims 2-10, 14-22, 26-28, and 30 depend directly or indirectly on either claim 1, 13, or 25 and should be allowed for at least the same reasons.

On page 8 of the Office Action, claims 11, 23, and 29 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Kester and Barton et al. (U.S. Pub. 2005/0021994). Claim 11 depends on claim 7, claim 23 depends on claim 19, and claim 29 depends on claim 25. Therefore, claims 11, 23, and 29 should be allowed for at least the same reasons as claims 7, 19, and 25. Furthermore, Barton fails to compensate for the deficiencies of Dozortsev and Kester with respect to claims 1-10, 13-22, 25-28, and 30. Barton is directed towards malware scans of subsets of files known to be more likely to contain malware. See Barton page 1 par. [0013]. For example, Barton does not teach or suggest automatically

generating a mask in accordance with normal behavior of the object determined from the received data.

On page 9 of the Office Action, claims 48-51 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Dozortsev in view of Kester and Gales (U.S. Pub. 2003/0084323). Claims 48 and 50 have been cancelled. Claim 49 depends on claim 1 and claim 51 depends on claim 13, so they should be allowed for at least the same reasons as claims 1 and 13. Furthermore, Gales fails to compensate for the deficiencies of Dozortsev and Kester with respect to claims 1-10, 13-22, 25-28, and 30. The Examiner notes that Gales discloses monitoring network traffic and usage patterns and automatically generating an activity profile based on the monitored activity on page 3 par. [0022]. However, Gales is monitoring network activity, not the behavior of an object. Monitoring network activity could involve monitoring traffic levels and/or contents of packets, but it does not involve receiving information about the underlying behavior causing or caused by the traffic levels and/or packet contents. Therefore, Gales does not teach or suggest receiving behavior information about an object or automatically generating a mask in accordance with normal behavior of the object determined from the received data.

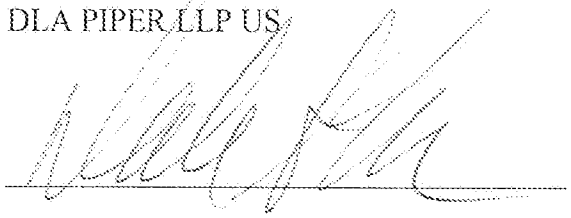
The Examiner argues that it would have been obvious to combine Dozortsev, Kester, and Gales. See Office Action page 9. The applicants disagree. Dozortsev and Kester would not be combined by those of ordinary skill for the reasons given above. Computer security is a wide ranging field, and Gales deals with a fundamentally different type of security from Dozortsev. Dozortsev is directed towards identifying and/or blocking malware running on remote computers (i.e. objects already present on the computers). Gales is directed towards preventing network intrusion from outside sources such as hackers, and monitors network traffic. See Gales page 2 par. [0013] ("intrusion detection system"). Furthermore, Gales teaches against analysis of computer objects in favor of its own techniques of analyzing network traffic. See Gales page 1 par. [0004]. The Examiner argues that monitoring usage as taught by Gales is similar to monitoring usage and activity as taught by Dozortsev. See Office Action page 9. The applicants disagree because the techniques of monitoring objects present on a computer and the techniques of monitoring external network traffic attempting to access a computer remotely are fundamentally distinct. Therefore, one of ordinary skill in the relevant art would not be

motivated to combine the teachings of Gales with those of Dozortsev, which discuss analyzing various non-behavioral aspects of computer objects.

Please charge any shortage in the fees or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US



Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/NP/maf
PO Box 2758
Reston, VA 20195
(703) 773.4000 Telephone
(703) 773.5200 Fax

Docket No.: 339000-000003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Melvyn Morris	Conf. No.	4785
SERIAL NUMBER:	11/477,807	Group:	2432
FILED:	June 30, 2006	Examiner:	C. Zecher
FOR:	METHODS AND APPARATUS FOR DEALING WITH MALWARE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

December 23, 2011

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

U.S. 2010/0101381 is the English Language equivalent of JP 2003-196112

WO 95/12162 is the English Language Equivalent of JP9-504395

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☐ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☒ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/maf
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2003/101381	5/29/2003	Mateev	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 95/12162	5/4/1995	Sophos PLC		
		EP 1 549 012	6/29/2005	DataCenter Technologies		
		JP 2003-196112	7/11/2003			
		JP 3-233629	10/17/1991	NEC Corp		
		JP 9-504395	4/28/1997			
		JP 6-110718	4/22/1994	Toshiba Corp		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Japanese Office Action issued in JP 2008-518975 on September 13, 2011	
		English Language Translation of Japanese Office Action issued in JP 2008-518975 on September 13, 2011	
		English Language Abstract and Translation of JP 6-110718 published April 22, 1994	
		English Language Abstract of JP 3-233629 published October 17, 1991	
		English Language Abstract of JP 2003-196112 published July 11, 2003	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Electronic Patent Application Fee Transmittal

Application Number:	11477807
Filing Date:	30-Jun-2006
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Filer:	Dale S. Lazar/Bert Lee
Attorney Docket Number:	339000-000003

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Extension - 2 months with $0 paid	1252	1	560	560

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Request for continued examination	1801	1	930	930
		Total in USD ($)		**1490**

Electronic Acknowledgement Receipt

EFS ID:	11700133
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Dale S. Lazar/Bert Lee
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000003
Receipt Date:	23-DEC-2011
Filing Date:	30-JUN-2006
Time Stamp:	14:34:18
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$1490
RAM confirmation Number	1475
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Foreign Reference	D1equivWO9512162A1.pdf	814560 76e316fef9eeb967a09c80cc4f734475d9b52b3a	no	23
Warnings:					
Information:					
2	Foreign Reference	D2EP1549012A1.pdf	802684 74b0525be8de52789acb2ed2d2f609f448dd8629	no	19
Warnings:					
Information:					
3	Foreign Reference	D5JP2003-196112.pdf	1173541 5a8ca3fd9a79aaf502814d6b79f0bdf0c1062db4	no	21
Warnings:					
Information:					
4	Foreign Reference	D4JP3-233629.pdf	192169 7b0ad4827f6e41c20365e32263838db5aede1641	no	7
Warnings:					
Information:					
5	Foreign Reference	D1JP9-504395.pdf	486616 5dfc9c398879c28144989c4750759c01b64e16e0	no	20
Warnings:					
Information:					
6	Foreign Reference	D3JP6-110718.pdf	544000 6dc0e22988c4bc748897eed21088efd321852f8d	no	15
Warnings:					
Information:					
7	Non Patent Literature	P10717JP-WJapaneseobjections.pdf	125377 65f00b8dd4484264c751bf13a8455dbd2f0b70af	no	5
Warnings:					
Information:					
8	Non Patent Literature	P10717JP-WOATranslation.pdf	155462 83ac10d3bad7687552b4a56310e6b890399c4b5b	no	7
Warnings:					
Information:					
9	Non Patent Literature	D3JP6-110718AbstractandTranslation.pdf	629386 d50a2d361c6a625b95290e2665b48615d0df6501	no	12
Warnings:					
Information:					

10	Non Patent Literature	D4JP3-233629Abstract.pdf	36645 5e68cefeb1686e0cd5f4bac66081c7948677eb8a	no	1
Warnings:					
Information:					
11	Non Patent Literature	D5JP2003-196112Abstract.pdf	372434 347a9e0ad4d0f769e12b1897b6cda674a1d6f935	no	1
Warnings:					
Information:					
12		339000-000003-RCE-Amendment-IDS.pdf	5754586 34bb4f6f643e8092d0d975fa9d51d1b8fca654ac	yes	14

Multipart Description/PDF files in .zip description

Document Description	Start	End
Request for Continued Examination (RCE)	1	1
Extension of Time	2	2
Amendment Submitted/Entered with Filing of CPA/RCE	3	3
Applicant Arguments/Remarks Made in an Amendment	4	10
Transmittal Letter	11	13
Information Disclosure Statement (IDS) Form (SB08)	14	14

Warnings:					
Information:					
13	Fee Worksheet (SB06)	fee-info.pdf	31949 9d925192a94e1ab8b0cd404befadb55490928131	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	11119409		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/00	A1	(11) International Publication Number: WO 95/12162
		(43) International Publication Date: 4 May 1995 (04.05.95)

(21) International Application Number: PCT/GB94/02378

(22) International Filing Date: 28 October 1994 (28.10.94)

(30) Priority Data:
9322292.5 29 October 1993 (29.10.93) GB

(71) Applicant *(for all designated States except US)*: SOPHOS PLC [GB/GB]; 21 The Quadrant, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YS (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: HRUSKA, Jan [GB/GB]; 1 Thatch Farm Cottages, Tubney, Oxfordshire OX13 5QQ (GB). LAMMER, Peter [GB/GB]; Manor Cottage, Dry Sandford, Abingdon, Oxfordshire OX13 6JW (GB).

(74) Agent: NEWSTEAD, Michael, John; Page Hargrave, Temple Gate House, Temple Gate, Bristol BS1 6PL (GB).

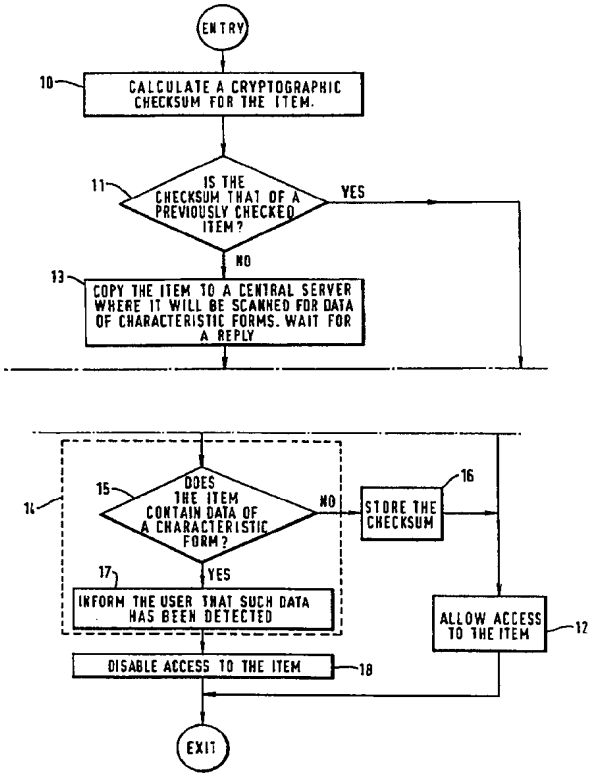
(81) Designated States: AM, AT, AT (Utility model), AU, BB, BG, BR, BY, CA, CH, CN, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), EE, ES, FI, FI (Utility model), GB, GE, HU, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SI, SK, SK (Utility model), TJ, TT, UA, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD, SZ).

Published
With international search report.

(54) Title: VALIDITY CHECKING

(57) Abstract

The present invention relates to the checking of the validity of data in a data processing system.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Australia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

VALIDITY CHECKING

The present invention relates to a method and apparatus for checking the validity of data in a data processing network, for example for checking whether the data contains viruses or other unwanted data or whether it has been authorised for or barred from use in the network or a part of it.

In general, data of a computer file or disk sector (such as a computer program) can be checked for unwanted data, or information indicating whether the file has been authorised for or barred from use, by the file being searched for data of a predetermined form. This form may comprise predetermined characteristics such as the presence of certain information anywhere in the file, possibly in any order, or at a certain location in the file, possibly in combination with other such data. For instance, computer viruses are stored in the data of a computer file as a set of virus data which can serve as instructions for the virus to operate. A file can be checked for known viruses by a virus detection procedure which searches the file for characteristics that are known to be indicative of each virus. As the number of known viruses to be checked for increases (around 3000 are currently known) the amount of storage capacity needed to store information defining the characteristics of all the known viruses increases too.

In a computer network of workstations and a file server it is conventional for each workstation to itself check on the validity of the data held by it. However, this means that every workstation must use a portion of its storage capacity to store information defining all the characteristic forms to be searched for. In total this requires a large amount of storage capacity, and as more characteristic forms come to be searched for, for example as new viruses are identified, it may become infeasible

for workstations to carry out searching themselves because of the limitations of their operating systems. Also, each workstation must be updated individually to include new characteristics. This is inconvenient where there is a large number of workstations.

WO 93/01550 discloses a system for controlling the use of a licensed product, in which in order to determine whether access can be made to a licensed product stored for access by a licensee's data processor a licence datagram is copied to a licensor's data processor which returns a reply message. The licence datagram contains different data from the licensed product.

According to a first aspect of the present invention there is provided a method for checking the validity of an item of data stored for access by a first data processor of a data processing network comprising at least two interconnected data processors, the method comprising the steps of: the first data processor causing the item of data to be copied to the second data processor; and the second data processor determining whether data of a characteristic form indicative of validity or invalidity of the item is present in the item and reporting to the first data processor on the validity of the item. Preferably, the first data processor is a workstation of the network. Preferably, the second data processor is a file server of the network. The data of a characteristic form is suitably indicative of the invalidity of the item of data, for example indicating a virus or other unwanted data or indicating that the item has been barred from use.

In a system of this type information defining the characteristic form(s) to be tested for needs to be stored for access only by the second data processor. When new characteristics are to be added only a single storage means (to which the second data processor has access) needs to be updated. Where the network includes

further data processors equivalent to the first data processor these can preferably also cause any item of data stored for access by them whose validity is to be checked to be copied to the second data processor. The first data processor and any data processors equivalent to it preferably do not store or normally have access to a list of information defining any characteristic forms of data to be tested for.

Preferably, the second data processor reports to the first data processor on the validity (or invalidity) of the item of data.

Preferably, information defining a plurality of characteristic forms of data to be tested for is stored by, or for access by, the second data processor, and the second data processor tests for the presence of such characteristic forms in an item of data by testing for the presence of data of any of the characteristic forms in the item.

The item of data may suitably be a file or program to be accessed, for example by being loaded or executed, by the first data processor. The item of data preferably comprises a sequence of executable instructions.

Preferably, the first data processor intercepts commands to access an item of data and in response to such a command being detected causes the validity of the item of data to be checked. Preferably, the first data processor prevents access to the item of data, for example by a user of the first data processor, unless or until the item has been found to be valid, i.e. free of unwanted data of the characteristic form(s) or of data indicating that the item has been barred from use. To achieve this, the first data processor suitably includes means for detecting a command to access an item of data, to allow it to intercept that command and ensure that the item is valid before it is accessed. Preferably the first data

processor may allow a user of the first data processor to force the system to check the validity of any or all items of data stored for access by the first data processor. Preferably, the first data processor is configured to, on receipt of a report from the second data processor on whether data of the characteristic form(s) has been found in the item, prevent or deny access to an item of data that has been found to contain data of the characteristic forms, and/or to allow access to an item of data that has been found not to contain such data. Thus access may be prevented to items that contain unwanted data such as viruses or that have been barred from use.

Preferably, the characteristic forms of data may include forms of data indicative of the validity of the item of data, for example indicating whether the item has been authorised for use. The first data processor may then prevent or deny access to any item that does not include such data and/or allow access only to items that do include such data.

Preferably, the first data processor stores or has access to a set of records, each characteristic of an item of data that has been found to be valid, and the method comprises the steps of: generating a record characteristic of an item of data whose validity is to be checked; searching for that record in the set of records; and causing the item of data to be copied to the second data processor only if the record is not found in the set of records. Preferably, the first data processor includes storage means for storing the set of records and/or processing means for generating records and comparing them with the contents of the set of records. Preferably, in response to the second data processor reporting that an item of data is valid the record that is characteristic of that item of data is added to the set of records. Each record is preferably a checksum calculated for the corresponding item of data.

According to the present invention from a second aspect there is provided a data processing system comprising a plurality of data processors interconnected as a network, and comprising: means in a first data processor of the network (preferably a workstation) for causing an item of data to be copied to a second data processor of the network (preferably a file server); means in the second data processor for testing for the presence, in the item, of data indicative of the validity or invalidity of the item and on the basis of that test generating a validity signal indicative of the validity of the item; and means for transmitting the validity signal to the first data processor.

The second data processor preferably reports or transmits the validity signal to the first data processor in the form of a report message, file or packet. The second data processor may suitably scan periodically to determine whether it has received an item of data for testing; alternatively the first data processor may transmit a signal, for example as a packet, to the second data processor informing it that it has sent an item of data for testing.

The present invention will now be described by way of example with reference to the accompanying drawings, in which:

Figure 1 shows schematically a typical data processing system for use with the present invention;

Figure 2 is a flow diagram illustrating one embodiment of the present invention; and

Figure 3 is a flow diagram illustrating another embodiment of the present invention.

Figure 1 shows a data processing system in the form of a network including data processors configured as a file

server 1 and workstations 2a, 2b and 2c. The general architecture of the network is conventional, for example with IBM PC-type or Apple Macintosh workstations and a VAX/VMS, Novell or OS2 file server. Each workstation can store data files and execute programs. The file server 1 includes storage means 3 for storing data, data processing means 4 and communication means 5 for communication with the workstation. The workstation 2a includes storage means 6, data processing means 7 and communication means 8 for communication with the file server. Each other workstation includes equivalent components. The storage means of each workstation may be located remotely of the rest of the workstation, for example at the file server.

When the validity of a file, for example a sequence of executable instructions such as a program, or in general an item of data, that is stored for access by or about to be executed at a workstation needs to be checked the file is copied to the file server, which tests for the presence of data of a characteristic form in the file and returns a report message, or in general a signal, indicating whether the file contains such data or whether the file in valid or invalid.

In more detail, each workstation is configured to detect when there is a need for the validity of a file to be checked, by intercepting commands to access, for example by loading or executing, any file on the workstation and immediately preventing access to that file until its validity has been checked and the file found to be valid. The workstation may be configured only to intercept commands to access certain "protected" items, such as programs and boot sectors.

The procedure shown in Figure 2 is executed to check the validity of a file. First, the workstation carries out a preliminary procedure to find whether the file has previously been checked and found to be valid, to avoid

a need to carry out the full validity-checking procedure more than once for each file. The workstation calculates (box 10 in Figure 2) a cryptographic checksum that is characteristic of the file that is to be checked. This may suitably be done using a standard ANSI X9.9 or ISO 8731 part 2 procedure to calculate a 32 or 64 bit checksum. This checksum is searched for in a list to which the workstation has access of checksums of files that have already been checked and have been found to be valid (box 11). This list may be stored in the storage means of the workstation or by the file server as a network service. If the checksum of the file under test is found in the list then it is assumed that the file is valid, and access to the file is allowed (box 12). If the checksum of the file under test is not found in the list then the file is copied to the file server (box 13), to be tested directly for validity.

The steps carried out by the file server are indicated generally by box 14 in Figure 2. Information defining the characteristic forms of data indicative of the file's validity or invalidity is stored at the file server. These characteristic forms may indicate whether the file contains unwanted data, such as a virus, or whether it has been authorised for or barred from use. For a virus, for example, the characteristics may indicate the form of data characteristic of the virus such as instructions found at the start of the file (typically "jump" instructions) or elsewhere in the file, which for some viruses may appear in any sequence. When the file server receives a copy of a file that it is to test for the presence of data of the characteristic forms it scans the file (box 15) to find whether any data of the characteristic forms is present in the file and returns a report message to the workstation that sent the file for checking, indicating whether such data was found. If the file server is to test for data in the file indicating that the file has been authorised for use then (not illustrated in Figure 2) its report message to the

workstation must also define whether the data that was found is indicative of validity or invalidity. Alternatively, the report message may report directly on whether the file is valid or invalid.

If the report message indicates that the file is free of unwanted data or data indicative of barring and/or (where implemented) that the file contains data indicative of authorisation (i.e. the message indicates that the file is valid) then the workstation adds the checksum of the file to its list of checksums of valid files (box 16), and it allows access to the file. Otherwise, if the report message indicates that the file is invalid then the workstation informs the user (box 17), for example by displaying a message, and prevents access to the file (box 18).

Alternatively, an operator of the workstation can instruct the validity of any or all files stored for access by the workstation to be checked, to authenticate the stored files. This authentication may be carried out omitting the step of testing the file's checksum against the stored list (box 11), so as to ensure that each file is tested directly by the file server for the presence of data of the characteristic forms.

The system may also be configured to require that in addition to being checked for data indicating that the file is valid or invalid any file that is introduced to the system must be known to the system as having been authorised by a network supervisor before it can be accessed. One way of achieving this is the procedure shown in Figure 3, which may be followed when a file has been copied to the file server for testing. The file server calculates a checksum for the file (box 20) and searches for it in a list stored by the file server of checksums of files that have already been authorised by the supervisor for use (box 21). If the checksum is found in the list then a report message is returned to

the relevant workstation, indicating that the file can be used (box 22) - this might happen if another workstation has previously passed the file to the file server for testing. If the checksum is not found then the file is tested for the presence of data of the characteristic forms in the way described above (boxes 23 to 25) and is sent to the network supervisor (box 26) together with a message reporting on the file's validity. If the file is then authorised by the supervisor its checksum as calculated by the file server is added to the file server's list of checksums of authorised files (box 27) and a report message is returned to the relevant workstation indicating that access to the file can be allowed (box 22). If the file is not authorised by the supervisor then its checksum is not added to the list and a report message is returned to the relevant workstation indicating that the file is not to be accessed (box 28). This procedure may be used in addition to the inclusion in files of data indicating whether the file has been authorised or barred from use (not illustrated in Figure 3).

Since according to the system described above only one list of information defining the characteristics to be tested for needs to be stored - by the file server - only one copy of the list needs to be altered when the system is to be updated. This is more convenient than prior systems in which copies held by every workstation must be altered. As more characteristic forms come to be searched for, for example as new viruses are identified, file servers (typically having more powerful operating systems than workstations) will remain capable of testing for characteristic forms. Also, if a single list of checksums of valid files is stored for access by all workstations then action to check a file for characteristic data is needed only when the file is first accessed by any workstation, not each time each workstation accesses it for the first time.

Two methods by which a file may be transferred to the file server and the report message returned to the workstation will now be described. According to the first method the workstation copies the file to be tested (in an encrypted form), together with data identifying the workstation, to the file server as a file of a randomly-chosen name having a predetermined format (for example, having a predetermined file extension). The file server is configured to scan periodically for such files and when one is found it is decrypted by the file server and tested for the presence of data of the characteristic forms. The file server returns the response message to the workstation identified in the received file by generating a response file, containing the response message, for transmission to the workstation. The name of the response file is generated as a function of the name of the corresponding file transmitted by the workstation, so that where several workstations have sent files for checking each can identify the file containing the response to its request for checking.

According to the second method the transmission of files may rely on network packets. The file to be tested is copied to the file server as described above but instead of the file server scanning periodically for files to be tested the transmitting workstation sends a packet message to the file server informing it that it has sent a file to be tested. When this is received the file server tests the file. The report message is returned to the transmitting workstation as a packet.

The system may operate by workstations communicating with each other or the file server via intermediate networks.

Instead of the file server testing for the presence of data of the characteristic forms this function may be performed by a selected workstation of the network.

While the invention has been particularly shown and described with reference to a preferred embodiment thereof it will be understood by those skilled in the art that various changes in form and details may be made without departing from the spirit and scope of the invention.

CLAIMS

1. A method for checking the validity of an item of data stored for access by a first data processor of a data processing network comprising at least two interconnected data processors, the method comprising the steps of:

the first data processor causing the item of data to be copied to a second data processor; and

the second data processor determining whether data of a characteristic form indicative of validity or invalidity of the item is present in the item and reporting to the first data processor on the validity of the item.

2. A method as claimed in claim 1, wherein a set of records, each characteristic of an item of data that has been found to be valid, is stored for access by the first data processor and the method comprises the steps of:

generating a record that is characteristic of the item of data whose validity is to be checked;

searching for that record in the set of records; and

causing the item of data to be copied to the second data processor only if the record is not found in the set of records.

3. A method as claimed in claim 2, wherein in response to the second data processor reporting that an item of data is valid the record that is characteristic of that item of data is added to the set of records stored for access by the first data processor.

4. A method as claimed in claim 1, wherein the first data processor intercepts commands to access an item of data and in response to such a command being detected causes the item of data to be checked for the presence of data of the characteristic form.

5. A method as claimed in claim 1, wherein the first data processor prevents access to an item of data unless or until it has been found to be valid.

6. A method as claimed in claim 1, wherein the item of data comprises a sequence of executable instructions.

7. A method as claimed in claim 1, wherein the first data processor is a workstation.

8. A method as claimed in claim 1, wherein the second data processor is a file server.

9. A data processing system comprising a plurality of data processors interconnected as a network, and comprising:

means in a first data processor of the network for causing an item of data to be copied to a second data processor of the network;

means in the second data processor for testing for the presence, in the item, of data of a characteristic form indicative of validity or invalidity of the item and on the basis of that test generating a validity signal indicative of whether such data has been detected in the item; and

means for transmitting the validity signal to the first data processor.

10. A data processing system as claimed in claim 9, wherein the second data processor includes storage means for storing a set of information defining data of a plurality of characteristic forms indicative of invalidity.

11. A data processing system as claimed in claim 9, wherein the second data processor includes storage means for storing a set of information defining data of a plurality of characteristic forms indicative of validity.

12. A data processing system as claimed in claim 9, wherein the first data processor has no access to a set of information defining data of characteristic forms.

13. A data processing system as claimed in claim 9, wherein the first data processor includes means for accepting a command to check for the presence of data of the characteristic form(s) in an item of data and, in response to such a command, checking for the presence of such data in the item.

14. A data processing system as claimed in claim 9, comprising at least three data processors connected as a network and means for causing an item of data to be copied to the said second data processor from any other data processor of the network for testing for the presence of data of the characteristic form(s) in the item.

15. A data processing system as claimed in claim 9, wherein the first data processor is a workstation.

16. A data processing system as claimed in claim 9, wherein the second data processor is a file server.

17. A data processing system as claimed in claim 9, for carrying out the method of any of claims 1 to 6.

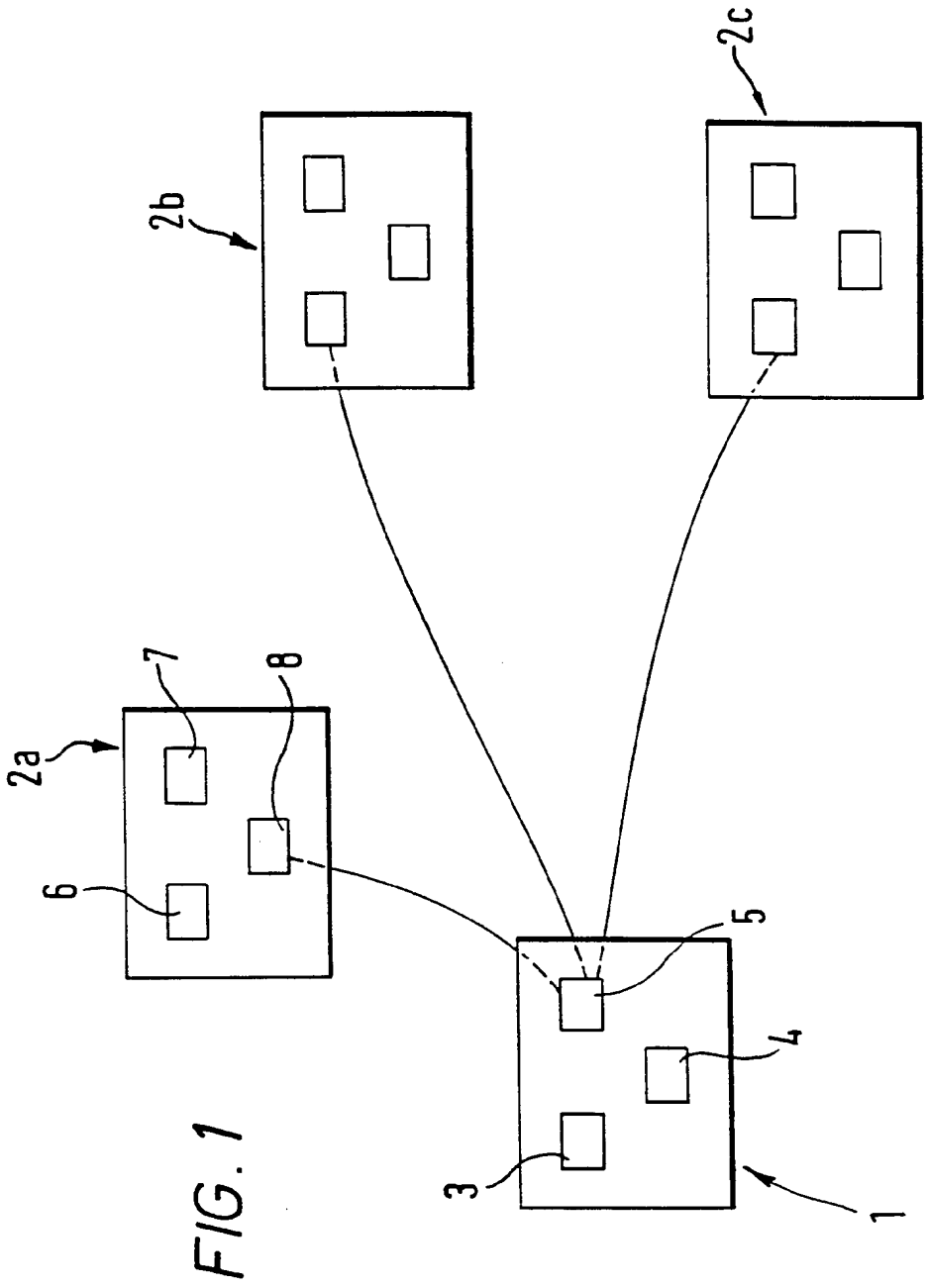
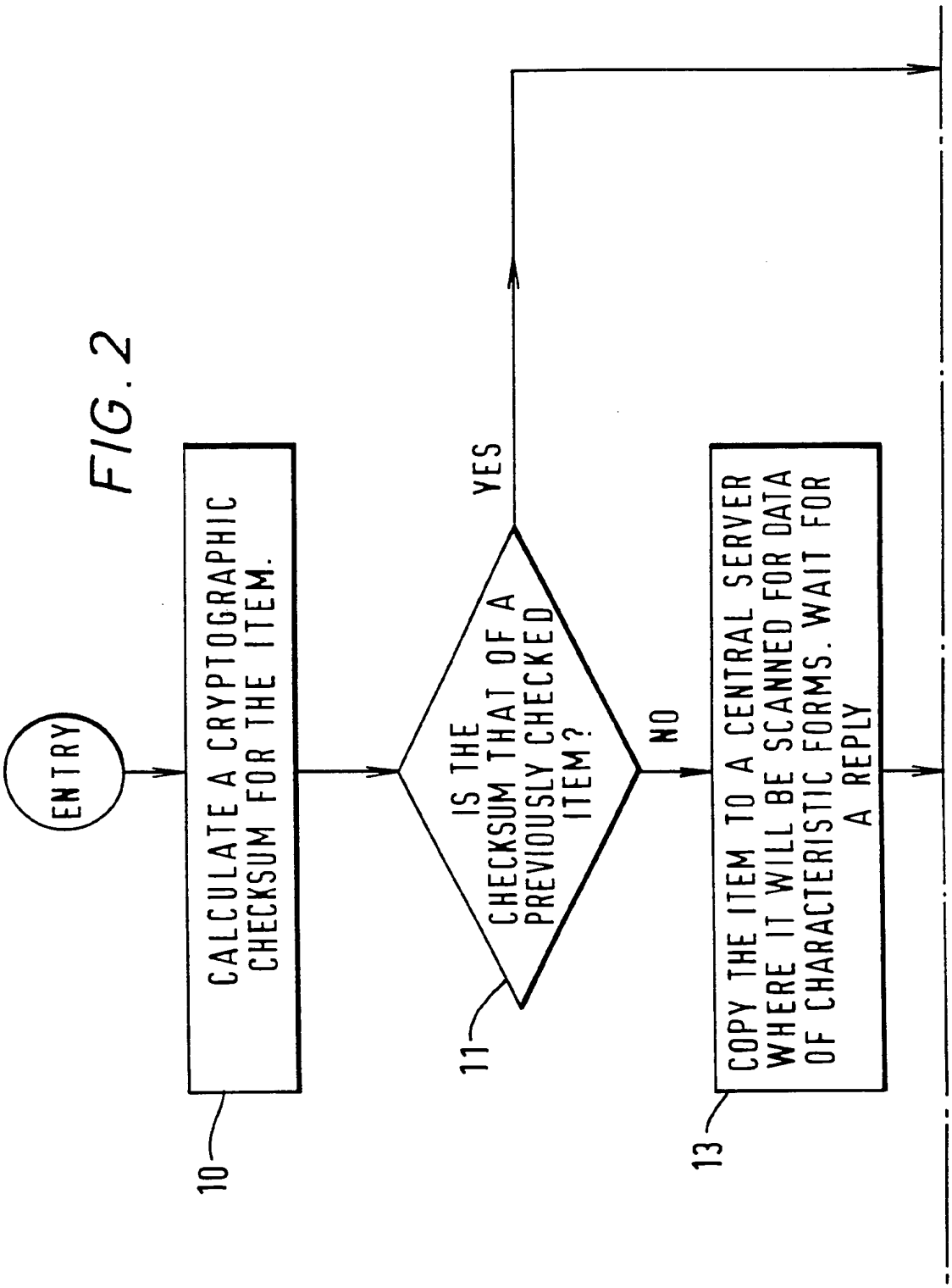


FIG. 1

FIG. 2


ENTRY
CALCULATE A CRYPTOGRAPHIC CHECKSUM FOR THE ITEM.
10
IS THE CHECKSUM THAT OF A PREVIOUSLY CHECKED ITEM?
11
YES
NO
COPY THE ITEM TO A CENTRAL SERVER WHERE IT WILL BE SCANNED FOR DATA OF CHARACTERISTIC FORMS. WAIT FOR A REPLY
13

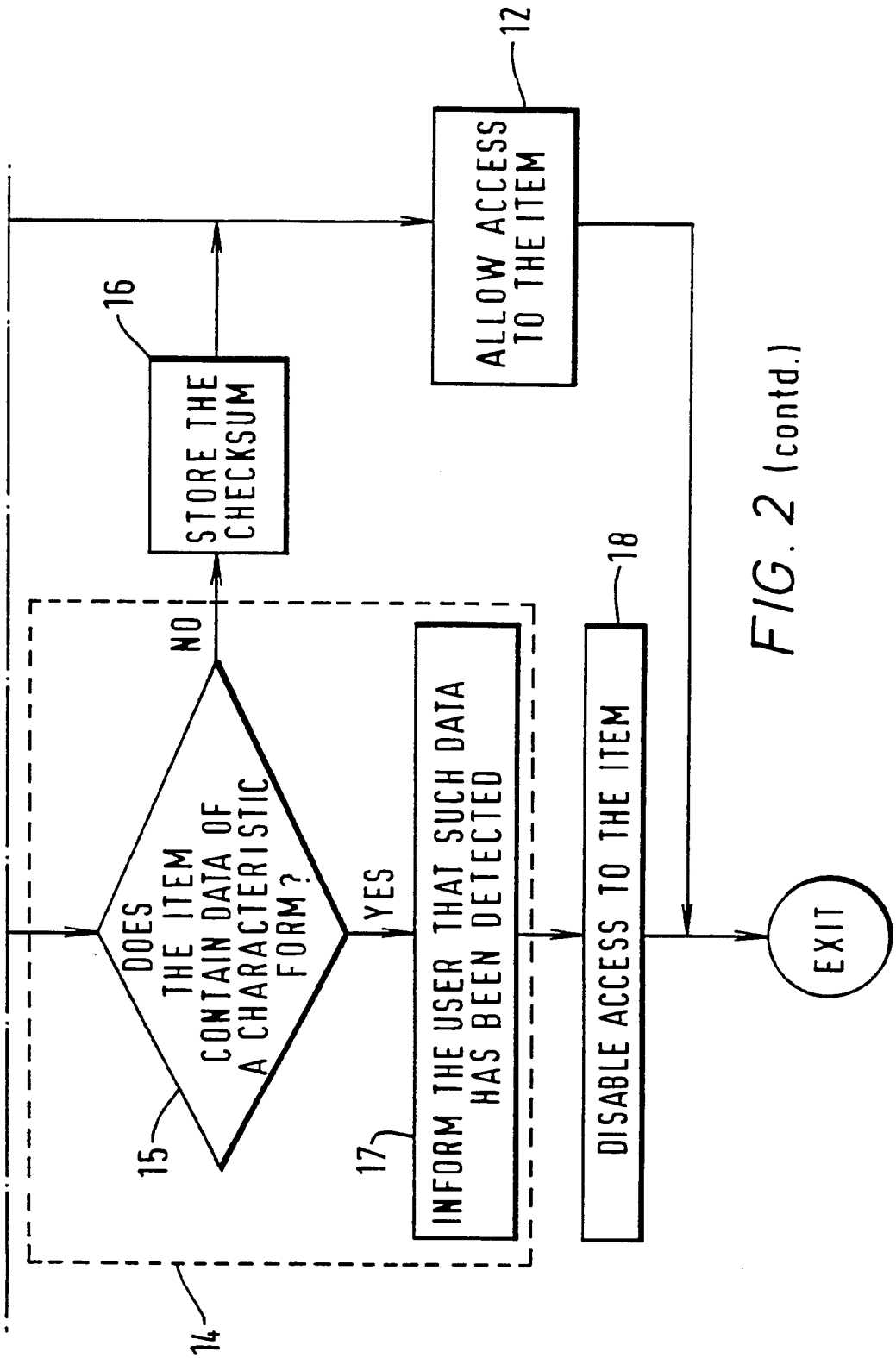

12
ALLOW ACCESS TO THE ITEM
16
STORE THE CHECKSUM
NO
DOES THE ITEM CONTAIN DATA OF A CHARACTERISTIC FORM?
YES
15
17
INFORM THE USER THAT SUCH DATA HAS BEEN DETECTED
18
DISABLE ACCESS TO THE ITEM
EXIT
14

FIG. 2 (contd.)

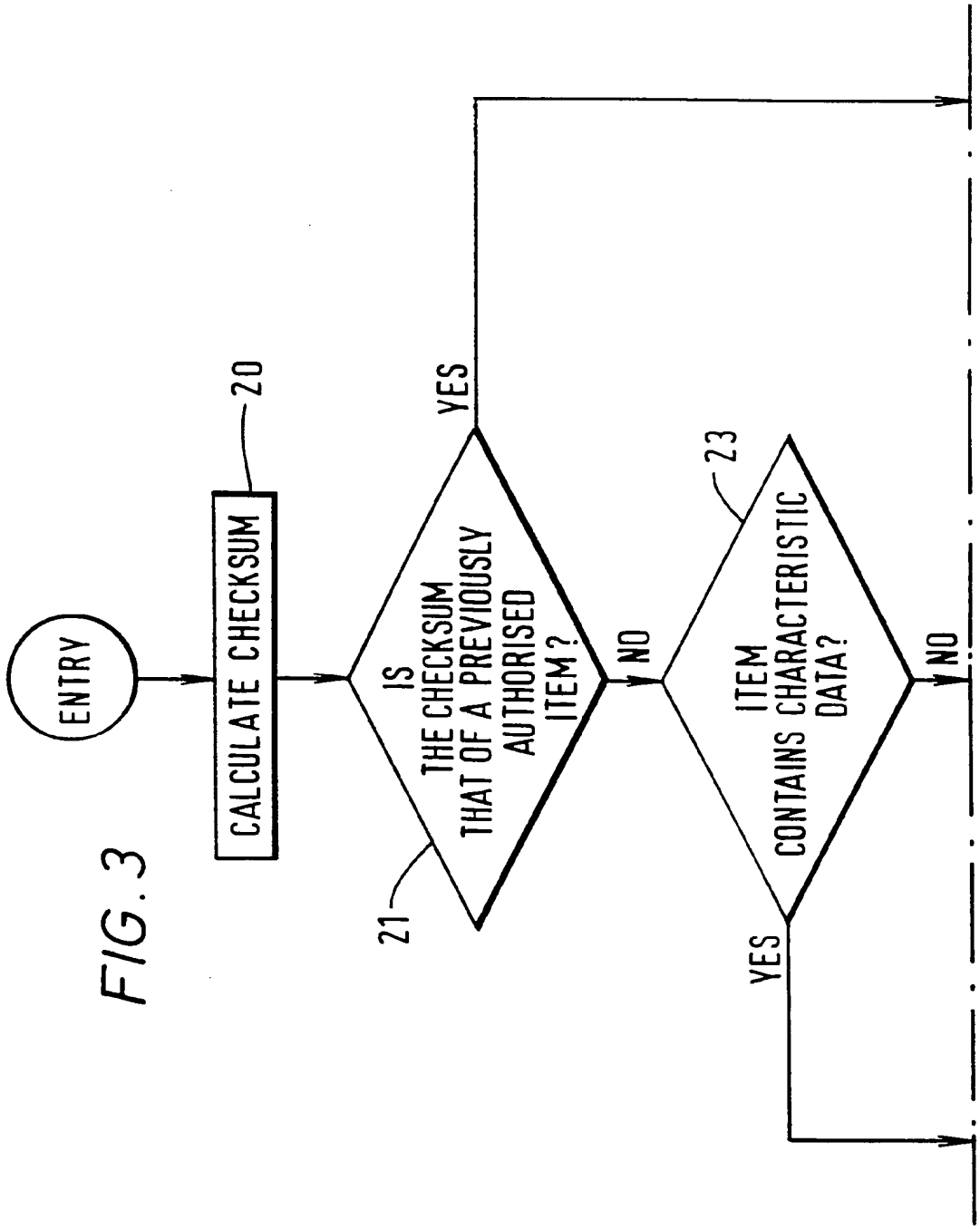

ENTRY
CALCULATE CHECKSUM
20
IS THE CHECKSUM THAT OF A PREVIOUSLY AUTHORISED ITEM?
21
YES
NO
ITEM CONTAINS CHARACTERISTIC DATA?
23
YES
NO

FIG. 3

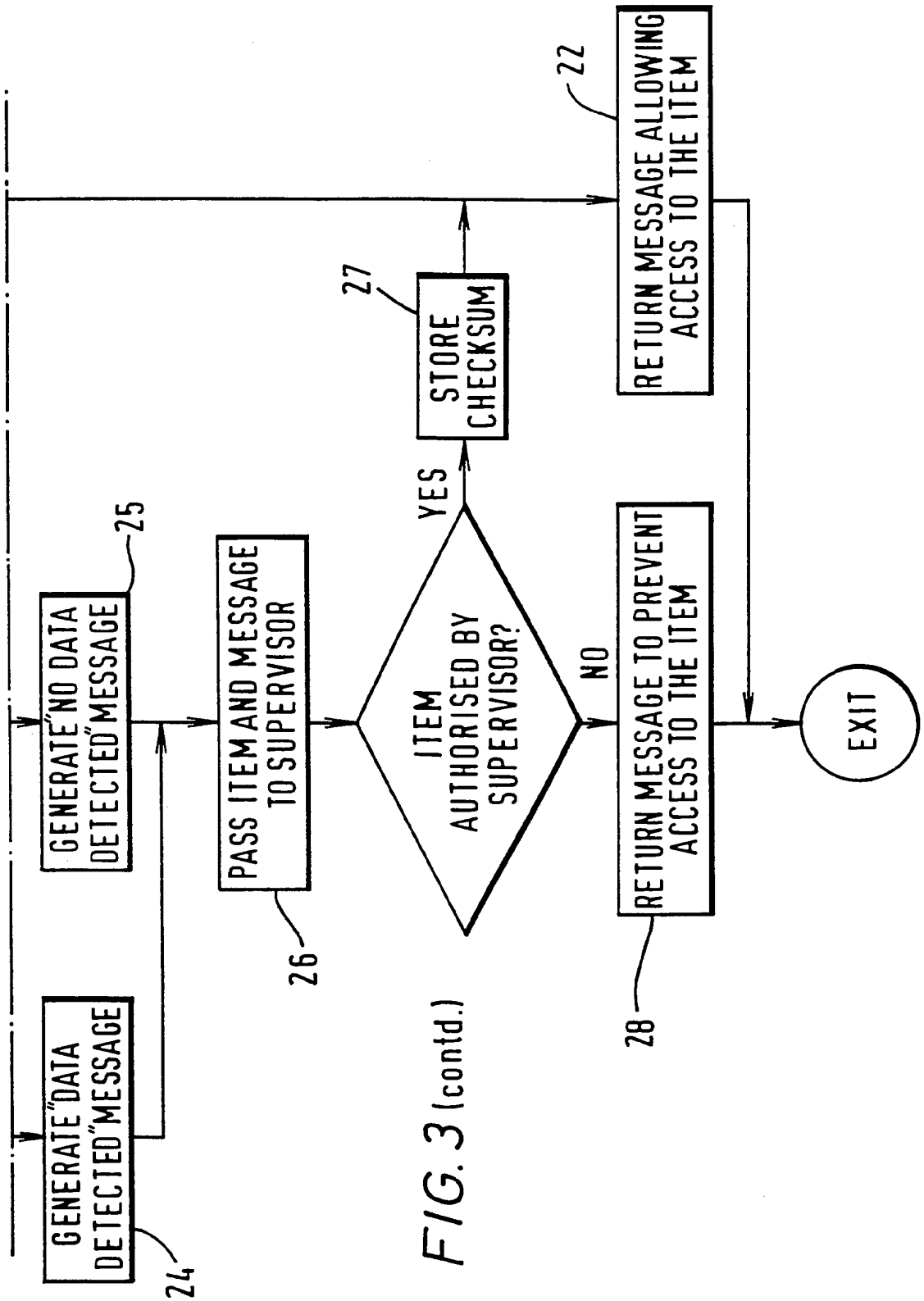

GENERATE "DATA DETECTED" MESSAGE
24
GENERATE "NO DATA DETECTED" MESSAGE
25
PASS ITEM AND MESSAGE TO SUPERVISOR
26
ITEM AUTHORISED BY SUPERVISOR?
YES
NO
STORE CHECKSUM
27
RETURN MESSAGE TO PREVENT ACCESS TO THE ITEM
28
RETURN MESSAGE ALLOWING ACCESS TO THE ITEM
22
EXIT

FIG. 3 (contd.)

INTERNATIONAL SEARCH REPORT

Intern. al Application No
PCT/GB 94/02378

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F11/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	EP,A,0 449 242 (NATIONAL SEMICONDUCTOR CORPORATION) 2 October 1991 see claims 1,11 ---	1-17
Y	EP,A,0 400 233 (MICHAEL H. FRANCISCO) 5 December 1990 see claim 1 -----	1-17

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 January 1995	31.01.95

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Corremans, G

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Intern al Application No
PCT/GB 94/02378

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0449242	02-10-91	NONE		
EP-A-0400233	05-12-90	US-A-	4845715	04-07-89
		GB-A-	2232279	05-12-90

Form PCT/ISA/210 (patent family annex) (July 1992)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 549 012 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.06.2005 Bulletin 2005/26

(51) Int Cl.[7]: **H04L 29/06**, G06F 1/00

(21) Application number: **03447310.8**

(22) Date of filing: **24.12.2003**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PT RO SE SI SK TR
Designated Extension States:
AL LT LV MK

(71) Applicant: **DataCenterTechnologies N.V.**
9080 Lochristi (BE)

(72) Inventor: **De Spiegeleer, Kristof**
8301 Knokke-Heist (BE)

(74) Representative: **Bird, William Edward et al**
Bird Goen & Co.,
Klein Dalenstraat 42A
3020 Winksele (BE)

(54) Method and system for identifying the content of files in a network

(57) A method and system for performing securing and controlling of a network using content identification of files in a network having a central infrastructure and local computing devices is presented. The method comprises calculating a hash value of a new file created or received on a local computing device, transmitting the hash value to the central infrastructure, comparing the hash value with a previously determined hash value stored in a database on the central infrastructure to determine whether the file is new to the network and if the file is new to the network, checking the file content with a content identifying engine, installed and updated on the central infrastructure. Content attributes are determined for the files which allow to perform appropriate actions on the local computing devices according to policy rules.

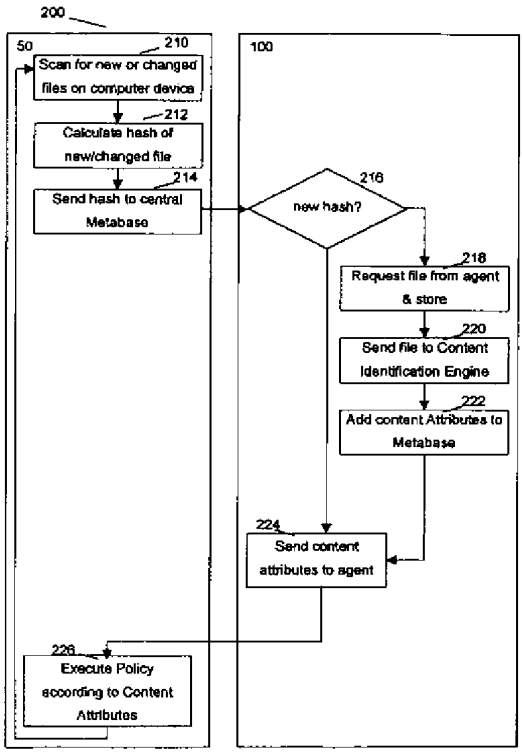


Fig. 3

EP 1 549 012 A1

Printed by Jouve, 75001 PARIS (FR)

Description

Technical field of the invention

[0001] The invention relates to a method and system to control the content of computer files, e.g. containing text or graphical data. More specifically, a method and system is described for checking and managing the security status and the content of computer files on a local computing device in a network environment.

Background of the invention

[0002] In today's world, computers are widely spread. Very often, especially in business environment, they are interconnected in small or larger networks. As software and data often are an important part of the investment goods of both private persons and firms, it is important to protect single computing devices and complete networks and their workstations against attacks from viruses, trojan horses, worms and malicious software. Another problem is related to the amount of files containing undesirable content such as explicit adult content. These files are often received on local computing devices uninvited and unwanted.

[0003] To solve the security problems associated with viruses, virus protection systems, also called virus checkers, have been developed. Some examples of conventional virus checkers are Norton AntiVirus, McAfee VirusScan, PC-cillin, Kaspersky Anti-Virus. Most of these conventional virus protection software packages can be configured so that they are continuously running in the background of the computing device and providing continuous protection. These virus protection systems compare codes of new or amended software with fingerprints (e.g. parts of code introduced in files by the viruses) of well known viruses. Other virus protection systems compare codes of all data available on the computing device. This leads to the use of a significant amount of central processing unit (CPU) time, which limits the capacity of the computing device for performing other tasks. Furthermore, the working principle of these virus checkers makes these software packages work rather reactive than proactive, as the fingerprint of the virus needs to be known in order for a virus scanning program to recognise it. This implies that the database of fingerprints needs to be updated very regularly in order to be secured against relatively new viruses. Consequently, the secure state of the computer is not only depending on external factors like the accurateness with which fingerprints of new viruses are made available by the suppliers of virus protection software packages, but also on the sense of duty of the user regarding performing updates regularly. If updates are provided centrally from a server automatically, then network capacity is reduced as these virus updates must be sent to each workstation.

[0004] In a network environment, the problem of updating such a database of fingerprints becomes significantly more important, as it implies that the responsibility is put to all users, who all have to update their virus checker database. Alternatively, the virus scanning could be performed by a central server, thus limiting the updating for new fingerprints to the central server. Nevertheless this implies that a large amount of data needs to be transferred over the network on a regular basis thereby utilising large amounts of expensive network bandwidth and possibly (depending on the number of clients for the server) overloading the network or server capacity for other activities.

[0005] In order to limit the amount of CPU time used, additional techniques have been developed to speed up the virus scanning process. These very often include hashing of the content of files. Hashing is one example of application of a "one-way-function". A one-way-function is an algorithm which when applied in one direction makes the reverse direction almost impossible to perform. A one-way-function generates a value such as a hash value by a calculation on the content of a file and can uniquely fingerprint this file if the one-way-function is complex enough to avoid duplicate values from different files. The uniqueness of a hashing function depends on the type of hashing function that is used, i.e. the size of the digest that is formed and the quality of function. Good hashing functions have the fewest collisions in a table, i.e. the chance of providing the same hash value for different files is the smallest. As mentioned, this is also determined by the size of the digest, i.e. hash value, that is calculated. If e.g. a 128-bit digest is used, the number of possible different values that can be obtained is 2^{128}.

[0006] It is known to use hashing for virus checking, possibly in a network environment. Typically, a hash of an application selected to run on a local computer is calculated, a stored hash from a database on a secured computer is retrieved on the local computer, whereby the secured computer can be a secured part of the local computer or a network server, and both values are compared. If there is a match, the application is executed, if there is no match, a security action is performed. This security action comprises loading a virus scanner on the local computer. It may also comprise alerting the network administrator. Furthermore it is also known to use this for differentiating accessibility to software from different workstations and as a way of checking whether software is licensed.

[0007] It is also known to use hashing in a method of identifying rogue software on a computer system or device. The method typically is applicable in a network environment. A hash value of a software application to be executed is calculated, this hash value is transferred to a server and compared with previously stored values. One of the essential features is that the method uses a database on a server, the server being a server with a large number of clients. The database on the server thereby is built up by adding information by different cli-

ents so that most software applications and their corresponding fingerprints are already stored in the database. The database is built up by checking software applications on authenticity with the owners of the application. If this is not possible, the system is also able to give a heuristic result, evaluating the occurrence of this application on local computers from other clients.

[0008] Methods for sending an electronic file by electronic mail, i.e. e-mail, including a file content and message content identifier are known. Depending on the message content identification, the message is delivered to a customer or not. The method may be used to organise e-mail delivery, but it has the disadvantage of being focussed on e-mail delivery and it does not allow to secure all files in a network.

[0009] Monitoring of electronic mail messages, to protect a computer system for protection against virus attacks and unsolicited commercial e-mail (UCE) is also known. Such a system is preferably installed on a mail server or an Internet Service Provider and checks specific parts of e-mails by calculating a digest and comparing it with stored digest values of e-mails previously received. In this way it is determined whether the e-mail has an approved digest or whether the e-mail is UCE or contains a e-mail worm. The system has the disadvantage that it is focussed on e-mail viruses and SPAM and that it does not allow to check all data files or executable files which are possibly infected, e.g. by files copied from external memory storage means like floppy disks or CD-ROMs or by e.g. Trojan horses.

[0010] Controlling the execution of software on different workstations according to certain policy rules by a network server is known, whereby an improved computer security system is obtained, by classifying software. It is suggested that this classification can be based on several forms of data one of which is e.g. the hash values of software data. This typically is performed by the calculation of hash values of a program if it is selected for loading and execution, and comparison of the hash value with a trusted value to determine the rule of execution. The classification also may be based on a hash of the content, a digital signature, the file system or network path or the URL zone.

[0011] The above mentioned methods and systems describe the use of hashing functions to check whether software applications are authentic or to regulate the execution of software applications. Nevertheless the problem of virus scanning all new files in a network using a conventional virus scanner whereby the necessity of updating the database of fingerprints of a conventional virus scanner on every local computer is limited is not discussed.

Summary of the invention

[0012] It is an object of the present invention to provide a system and a method for identifying the content of files new on local computing devices in a network while limiting the necessity of updating the information needed by a content identifier on every local computing device.

[0013] It is an advantage of the present invention that, if the invention is used as a virus checker, the security level is further increased as the database of fingerprints of a conventional virus scanner does not have to be updated on every local computing device.

[0014] It is a specific advantage of the present invention that the content of a file new to a network is only identified once for the whole network.

[0015] It is furthermore a specific advantage of the present invention that the total processor (CPU) processing time in the network and the amount of network traffic is reduced.

[0016] The above described object and advantages are obtained with a method and system of content identification in a network according to the present invention.

[0017] The method for identifying the content of a data file in a network environment is used for a network having at least one local computing device linked to a remaining part of the network environment including a central infrastructure. The method and system comprises calculating a reference value for a new file on one of said at least one local computing devices using a one-way-function, transmitting said calculated reference value to said central infrastructure, comparing said calculated reference value with reference values previously stored within the remaining part of the network environment.

[0018] The method further comprises, after comparing, deciding that the content of the new file is already identified if a match between said calculated reference value and a previously stored reference value is found and retrieving the corresponding content attributes; or deciding that the content of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure identifying the content of said new file and determining content attributes corresponding with the content of the new file and storing a copy of said content attributes, after deciding, triggering an action on said local computing device in accordance with said content attributes.

[0019] In the method for identifying the content of a data file in a network environment, the reference value may be a hash value. The reference values previously stored may be stored within the central infrastructure. In the method and system for identifying the content of a data file in a network environment, identifying the content of the new file may comprise scanning the new file for viruses using an anti-virus checker means on a central infrastructure.

[0020] The method may furthermore comprise transferring the new file from the local computing device to the central infrastructure before said identifying the content of said new file is performed. Furthermore it may comprise storing a copy of the new file on the central

infrastructure after transferring the new file from the local computing device to the central infrastructure. An address of where the file is stored may be stored together with the hash value, as to be able to quickly track copies of the files stored on the central infrastructure.

[0021] In the method of the present invention, triggering an action on the local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with a copy of a previous version of said new file. Furthermore, triggering an action on the local computing device in accordance with said content attributes may also comprise replacement of the new file on the local computing device with another version of said new file restored from the remaining part of the network environment.

[0022] The method of the present invention furthermore may comprise sharing the new file on the local computing device to the central infrastructure before said identifying the content of said new file is performed and whereby said identifying the content of said new file is performed by remotely identifying the content over the network environment. The method may comprise checking the functioning of the local agent on the local computing device.

[0023] Furthermore, triggering an action on the local computing device may be performed after transmitting the content attributes corresponding to the new file to the local computing device.

[0024] In the method for identifying the content of a data file in a network environment, identifying the content of the new file may comprise one or more of the group of scanning for adult content, scanning for Self Promotional Advertising Messages or Unsolicited Commercial E-mail (UCE) and scanning for copyrighted information. The method may further relate to a method and system for providing a content firewall, whereby one local computing device is connected to the external network, which may e.g. be the internet, and the one local computing device is also connected to the network environment formed by the remaining local computing devices. The one local computing device thus links the network environment with an external network and is the only computing device that is directly connected to sources external from the network environment. The local computing device thus acts as a content firewall as to protect the network environment from attacks originating from places in the external network. The local computing device may act as a content firewall working in a promiscuous way, i.e. whereby the local computing device acts as a content firewall that sees all traffic passing by, executes the hashing and comparing functions and contacts the agents to enforce a policy.

[0025] The method may be specifically related to a method for checking the security status of a network and its components. In this embodiment, a method for determining the security status of a data file in a network environment is used in a network having at least one local computing device linked to a remaining part of the network environment including a central infrastructure. The method comprises calculating a reference value for a new file on one of the at least one local computing devices using a one-way-function, transmitting said calculated reference value to said central infrastructure, comparing said calculated reference value with reference values previously stored within the remaining part of the network environment and after comparing, deciding that the security status of the file has already been checked if a match between the calculated reference value and a previously stored reference value is found and retrieving the corresponding security status; or deciding that the security status of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure checking the security status of the new file and determining the security status corresponding with the new file and storing a copy of the security status, followed by after deciding, triggering an action on said local computing device in accordance with the security status of the new file. This action may be e.g. making the file inaccessible for the user of the local computing device and for other users in the network or restoring the infected file.

[0026] The methods described above may be triggered by an action performed on the local agent. The triggering by an action performed on the local agent may be e.g. running an application or opening a file.

[0027] The invention also relates to a method for altering a system for identifying the content of a file in a network environment according to the systems described above, the network environment comprising means for calculating a one-way function, at least one local computing device linked to a remaining part of the network environment including a central infrastructure and means for identifying the content and, the method comprising altering said means for identifying the content or said means for calculating a one-way function, scanning the remaining part of the network environment for reference values calculated with a one-way function and for each of the reference values, requesting a file that corresponds with said reference value from said network environment, sending the file to means for identifying the content, identifying the content of said file and determining content attributes corresponding with the content of the file and storing a copy of said content attributes, sending the content attributes to every local computing device containing the file and after sending; triggering an action on said local computing device in accordance with said content attributes. Altering of a system for identifying the content of a file in a network environment may be triggered by the introduction of a new one-way function to calculate reference values or may be also triggered by the updating of the means for identifying the content of the files. The method may be used to scan only part of the hashing keys in the remain-

ing part of the network environment, e.g. hashing keys of files of which the content is identified after a certain date, as to minimise the actions to be performed. The date of the previous content identification may be retrieved from the content attributes.

[0028] The reference values may be hashing values. The means for identifying the content may be an anti-virus checker means, a means for scanning for adult content, a means for scanning for Self Promotional Advertising Messages or a means for scanning for copyrighted information. Triggering an action on the local computing device in accordance with said content attributes may comprise replacement of the file on the local computing device with another version of the file restored from the remaining part of the network environment or may comprise replacement of a file with a copy of a previous version of the file or may comprise putting the file in quarantine or removing the file.

[0029] The invention is also related to a computer program product for executing any of the above described methods, when executed on a network. The invention furthermore relates to a system for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part the network environment which includes a central infrastructure and, said remaining part including a stored database, whereby the system comprises means for calculating a reference value for a new file on said local computing device using a one-way-function, means for transmitting said calculated reference value to said central infrastructure and means for comparing said calculated reference value with previously stored reference values from the database. The system furthermore comprises means for deciding whether the content of the new file is already identified based on comparison of said calculated reference value and reference values previously stored within the remaining part, means located on the central infrastructure, for identifying the content of the new file and as to assign content attributes if the new file has not been identified yet and means for storing said content attributes within the remaining part, and means for triggering an action on said local computing device in accordance with content attributes for said new file.

[0030] In the system according to the present invention, the means for identifying the content of a file may comprise an anti-virus checker means on said central infrastructure. Furthermore, means for storing a copy of the new file within the remaining part. The means for identifying the content of a file may comprise one or more of the group of means for scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

[0031] The invention may also relate to a machine readable data storage device, storing the computer program product for executing any of the above described methods, when executed on a network. Furthermore, the invention may also relate to the transmission of the computer program product for executing any of the above described methods.

[0032] Although there has been constant improvement, change and evolution of methods of virus scanning and content identification of data files, the present concepts are believed to represent substantial new and novel improvements, including departures from prior practices, resulting in the provision of more efficient, stable and reliable methods of this nature.

[0033] These and other characteristics, features and advantages of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, which illustrate, by way of example, the principles of the invention. This description is given for the sake of example only, without limiting the scope of the invention. The reference figures quoted below refer to the attached drawings.

Brief description of the drawings

[0034]

Fig. 1 is a schematic representation of a computer network

Fig. 2 is a schematic representation of a central infrastructure and its basic software components

Fig. 3 is a schematic representation of a local agent-driven content identification process.

Fig. 4 is a schematic representation of a metabase-driven content identification process.

Fig. 5 is a schematic representation of a computer network to which the content firewall system and method can be applied.

[0035] In the different figures, the same reference figures refer to the same or analogous elements.

Description of illustrative embodiments

[0036] The present invention will be described with respect to particular embodiments and with reference to certain drawings but the invention is not limited thereto but only by the claims. The drawings described are only schematic and are non-limiting. In the drawings, the size of some of the elements may be exaggerated and not drawn on scale for illustrative purposes. Where the term "comprising" is used in the present description and claims, it does not exclude other elements or steps.

[0037] Furthermore, the terms first, second, third and the like in the description and in the claims, are used for distinguishing between similar elements and not necessarily for describing a sequential or chronological order. It is to be understood that the terms so used are interchangeable under appropriate circumstances and that the embodiments of the invention described herein are capable of operation in other sequences than described or illustrated herein.

[0038] In this description, the terms "file", "program",

"computer file", "computer program", "data file" and "data" are used interchangeably, and any one use may imply the other terms, according to the context used. The terms "hash" and "hashing" will be used as examples of the application of one-way-functions but the present invention is not limited to a particular form of one-way-function.

[0039] The term "computing device" should be interpreted widely to include any device capable of carrying out computations and/or executing algorithms. A computing device may be any of a laptop, workstation, personal computer, PDA, smart phone, router, network printer or any other device which has a processor and can be connected to a network such as e.g. faxing devices or copiers or any dedicated electronic device such as a so-called "hardware firewall" or a modem.

[0040] The method and system to secure and control a network by identifying the content of each new file in the network can be used on any type of network. This may be a private network which may be a virtual private network, a local area network (LAN) or a wide area network (WAN). This may also be within a part of a public wide area network such as the internet. If a part of a public wide area network is used, this may be performed by remotely providing the method and system for identifying the content of each file by a service provider using an ASP or XSP business model, wherein the central infrastructure is provided to a paying client operating a local computing device. An exemplary network 10 is shown in figure **1**, showing several local computing devices 50a, 50b, ... , 50i and a central infrastructure 100, also called a server. The number of local computing devices 50 connected to the network 10 is not limiting for the method of securing and controlling a network 10 according to the current invention. In business environment this number of local computing devices 50 typically ranges from a few to a few thousands. The method and system for identifying the content of each new file present in the network 10 may be used with many different operating systems such as Microsoft DOS, Apple Macintosh OS, OS/2, Unix, DataCenter-Technologies' Operating Systems, ...

[0041] In order to provide a quick method of securing and determining content identification of files, the method and system according to the present invention will determine hash values of new files present on the local computing devices 50, compare them with previously stored hash values and file information on a central server and determine the content of files new to the network 10 using a content identifying engine on the central infrastructure 100. The content attributes describing the content of a new file are then send to the local computing device 50 where an appropriate action is performed. It is also possible that the content attributes are not sent to the local computing device 50 but that the appropriate action is triggered from the central infrastructure 100. New files typically are files wherein new content has been generated on a local computing device 50 or when an external file has been received. The wording "file" may refer to data as well as to software applications, also called software.

[0042] The central infrastructure 100 contains a database, also called metabase 110, which contains a record for every hash value that is calculated for a file that already exists on one of the local computing devices 50. Besides the hash value, this record also contains a number of other fields. In these fields, file source information is stored. The file source information corresponding with a specific hash value includes the file name, a list of local computing devices 50 where the files that correspond to this hash value are residing on, including the path to the file on the file system of the local computing devices 50 and the date of last modification. An example of file source information for a specific file is given in Table 1.

Table 1

Filename	Myexampleword.doc
Path	c:\data\
Assetname	PCmarketing001
ModDate	23/4/2002

[0043] In a further field, a list of content attributes that identifies the type of content that is enclosed by the file is stored. The content attributes can e.g. refer to a file that contains a virus, a file that is a copyrighted MP3 audio file, a file that is a copyrighted video file, a file that is a picture, a file that is a picture that might contain adult content, a file that is a Self Promotional Advertising Message (SPAM), a file that is a HOAX, a file containing explicit lyrics or a file containing pieces of executable code. This list is not limiting.

[0044] The central infrastructure 100 furthermore contains a content identification engine 120. This can be a software application 130 or a set of software applications 130a, 130b, 130c, 130d, ... that use the content of a file to determine which type of content the file contains. These software applications may be various :

- a virus scanner : this is a piece of software that scans the content of the presented file and compares it with a database of known fingerprints of viruses. This can be any conventional virus scanning software like e.g. Norton anti-virus by Symantec Corporation, McAfee by Network Associates Technologies Inc., PC-cillin by Trend Micro, Kapersky Anti-Virus by Kaspersky Lab, F-secure Anti-Virus by F-Secure Corporation, ...
- an adult content in pictures scanner : this is a piece of software that scans the content of the presented file for the presence of shading, colors, textures that might represent adult content. Scanning pictures for adult content is already known. Content adult can e.g. be determined by the amount of nude that is

shown. Skin tones have hue-saturation values that are in a specific range. Therefore, if an image is scanned, it is possible to determine the amount of pixels having a skin tone character and to compare it with the total number of pixels. The ratio of skin tone pixels to the total number of pixels allows to determine a ratio of possible adult content in an image. Thresholds often are introduced so that images can be classified according to their possible adult content. In similar way, video images can be categorised, whereby the video is split into its different frames and wherein the images are categorised according to the above method.

- A scanner for internet content ratings : A piece of software that scans objects for adult content based on the PICS, i.e. the Platform for Internet Content Selection, label system. On voluntary basis, internet content providers can provide internet objects with a PICS rating determining the adult content in the internet object. This PICS rating is stored in the meta data of the object. This data normally is not visible to the viewer of an internet object. The rating systems is well known and an example of a scanner for internet content ratings is provided in the Netscape web browser for scanning the content of web pages.
- A scanner for scanning an object for explicit lyrics which may indicate adult content. This is known for both text files and audio files. Audio files are first transferred to text files. Subsequently, the text files are scanned and compared with databases which contain explicit lyrics.
- a SPAM-engine: A piece of software that scans the content of e-mail messages for the presence of unalleged SPAM. Algorithms to recognize SPAM area already known. These are typically based on decomposing the text in an electronic mail message, associating statistics with the text using a statistical analyzer and coupling a neural network engine to the statistical analyzer to recognize unwanted messages based on statistical indicators.

Other examples of software applications that could be used in the content identification engine 120 are e.g. engines that scan for copyrighted content and that compares the content of the file to a database of copyrighted information, etc. In some adoptions, a human operator can pursue the role of content identification engine 120, where he manually tags a file with a content identification attribute. When the content identification engine 120 is activated, it will take a file from the local agent as input and produce a set of attributes that represent the detected content.

[0045] The content Identification engine 120 will thus be constructed as a piece of software aggregating the functionality of a set of third party engines.

[0046] In a further embodiment of the invention, a system and method in accordance with the above embodiment is described whereby the record corresponding with a specific hash value stored in the metabase 110 also comprises a field wherein the location of the file on the central infrastructure 100 corresponding with the hash value is stored. In this embodiment, a copy of all different files present on the local computing devices 50 in the network 10 is stored on the central infrastructure 100. So, the central infrastructure 100 of this embodiment also comprises a large amount of storing space. This preferably is a secured part of the central infrastructure 100, not directly connected to the network 10 so that these identical copies of the files present on the local computing devices 50 can be used in case the files on the local computing devices 50 are corrupted e.g. by a virus.

[0047] The hash value of the files are calculated using a hashing function. A hashing function typically is a one way function, i.e. given the digest, it is at least computationaly prohibitive to reconstruct the original data. Different types of hashing functions could be used : MD5, SHA-1 or ripemd all available from RSA Data Security Inc., haval which is designed at the University of Wollongong, snefru which is a Xerox secure hash function, etc.. The hashing functions most often used are MD5 and SHA-1. The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be 'compressed' in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem. The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. An alternative hashing function SHA-1, i.e. Secure Hashing Algorithm - 1, is a hashing algorithm generating a 160-bit hash. Newer versions of this algorithm also provide bit lengths of 256 and 512.

[0048] In the above mentioned embodiments describing a method and system to secure and/or control the network 10, a local agent is installed on the local computing device 50. The local agent is a piece of software that is running on a local computing device 50 and that performs certain algorithms and procedures. The local agent on the local computing device 50 is triggered typically in situations where new content is being generated on local computing devices 50. In order to avoid unnecessary hash value calculations and data transfer, a policy is setup to determine which actions will trigger the local agent and which actions do not trigger it. If e.g. a text document is being created, it is not necessary to check the file every time the document is saved. The policy for such a type of documents would preferably be that the document is checked e.g. if the file is both saved

and closed. Some examples of actions which could trigger the local agent and thus start the content identification process are opening or receiving e-mail messages, opening or receiving e-mail attachments, running executable files, running files with dll or piff extension ... Applying this policy thus allows to prevent from continuously checking and scanning of documents leading to a limitation of the number of unnecessary hash calculations and content identification operations and thus limiting the unnecessary use of CPU time and load on the network traffic. The method and system of content identification is not limited due to the type of application in which the file is made.

[0049] The content identification process can be either triggered by the local agent on the local computing device 50 or it can be triggered by the central infrastructure 100. The latter process typically occurs in situations wherein new algorithms or tools are being used for content identification. Such new algorithms or tools can either be optimised algorithms and tools or previously uninstalled tools. Some examples of these tools, without restricting to these functions, could be virus checking, checking whether a file is a copyrighted MP3 Audio File, checking whether a file is a copyrighted Video File, checking whether a file is a picture that might contain adult content, checking whether a file is tagged as being SPAM or HOAX, checking whether a file contains explicit lyrics or checking whether a file contains copyrighted pieces of executable code. Updating of these tools may influence the status of the files and thus may in principle influence the corresponding records in the metabase 110. Therefore, depending on the type of update of the content identification means 120, it may be interesting to update the corresponding records.

[0050] In a specific embodiment, the method relates to a virus checker for a network environment. The networks 10 on which this method can be applied are the same as those described for the previous embodiments. The local agent calculates the hash value of a new file on the local computing device 50. This new file may comprise new content generated on the local computing device 50 or an external file which is received on the local computing device 50. The hash value of the new file and the corresponding file information then is sent to a central infrastructure 100, also called server, where it is compared to previously stored hash values corresponding with files that are already present on the different local computing devices 50 of the network 10. This comparison allows to check whether the file is new or not in the entire network 10. Alternatively, the hash value may also be first compared with a local database of hash values and file information corresponding with the files present on that particular local computing device 50 and subsequently, if the file has been found not yet present on the local computing device 50, the hash value and the corresponding file information may be interchanged with the central infrastructure 100 so it can be checked whether the file is new or not in the entire network 10. Although transferring the file information and the hash value of every new file only corresponds with a very small fraction of the network traffic for a conventional central virus checker, this alternative could reduce the network traffic used for virus checking even further. If a hash value has been identified as new on the network 10, the metabase agent triggers the local agent to transfer the file corresponding with the new hash value from the local computing device 50 to the central infrastructure 100. The transferring of the file may be performed in a secured way, i.e. the file may be transferred such that it cannot be influenced by a virus present at a network connection or such that, it if contains a virus, this cannot be spread over the whole network 10. To obtain this, a known secure transmission route, a tunnel and/or known session ecryption/decryption techniques may be used. A conventional virus checker, installed and updated on the central infrastructure 100 then checks the file for viruses. This can be any conventional virus checker like e.g. Norton anti-virus by Symantec Corporation, McAfee by Network Associates Technologies Inc., PC-cillin by Trend Micro, Kapersky Anti-Virus by Kaspersky Lab, F-secure Anti-Virus by F-Secure Corporation, ...

[0051] A specific advantage of the above described embodiments in the current invention is that the virus scanning software does not need to be updated on every local agent but that this is restricted to updating of the virus scanning software of the central infrastructure 100. In this way the security level of the network 10 is increased significantly as the security does not depend on the punctuality of the different users of the network 10 to update their virus scanning software. If the scanned file has no virus it will be marked in the metabase 110 as being a virus free file. If there is a virus found in a file the file will be marked as dangerous. A query will happen to the metabase 110 to find all files over the network 10 having the same corrupted hashing key. The result is a list of files with path, and assetname where the file is located. This information can be used to do actions to eleminate the danger of found viruses on all local computing devices 50, i.e. all workstations, from the complete network 10. In this way proactive virus scanning can be performed on other local computing devices 50, based on a virus detection on a first local computing device 50. Depending the policy defined for virus checking, the virus engine will inform an agent installed on the affected system to remove the file and if possible replace with either a recovered version delivered by the virus engine located on the central infrastructure 100, or a previous version of the file which didn't have the virus yet. The latter can be done easily by searching the metabase for a previous version of that file, or it can be performed by searching an uninfected version on another local computing device 50. If an uninfected version cannot be retrieved from either another local computing device 50 or the metabase residing on 100, the virus scanner should have a feature which allows it to save a

new disinfected copy of the file on the central infrastructure 100. These advantages are also present for other content identifying packages.

[0052] In an alternative embodiment, if a file having a new hash value has been identified in the network 10, instead of transferring the file to the central infrastructure, the file may be automatically shared locally and a remote checker then may transfer a file-system which allows to check the file across the network 10 using the file sharing. The content tagging still is performed by the server. In order to improve security, the accessibility to the shared file is restricted to the server. Furthermore, a java applet could be transferred to the local agent to allow checking other files.

[0053] The previous embodiments are an improvement over a central virus checker which scans local computing devices 50 through the network 10. This is only possible if the local drives, e.g. C:\, D:\, ..., are shared. Besides the dangers of sharing with respect to security, the local user also easily can change the local sharing properties thereby preventing the remote checker from checking the files. This is at least partly avoided with the current invention as changing the network 10 sharing properties does not influence the operation of calculating the hash value of new files and sending it to the central infrastructure 100.

[0054] Another advantage is that it saves CPU time on the local computing device 50 as the CPU does not have to keep doing virus checking, it only has to calculate a one way function. It also saves network time : the administrating server does not have to update the virus checkers on the local computing devices 50 with virus updates, as a single central virus checker only is used and updated.

[0055] Fig. 3 shows a method 200 of the content identification process triggered by the local agent on the local computing device 50 according to the above mentioned embodiments. The different steps that occur during the process, both on a local computing device 50 and on the central infrastructure 100 are discussed.

[0056] The content identification process is based on continuously scanning for new data or applications on the local computing device 50 by the local agent. This scanning for data and applications is limited by the policy rule for determining when the local agent should be triggered, as described above. If a "new" file has been detected the method for securing and controlling the network 10 by content identification of new files is initiated. This is step 210. Method 200 then proceeds to step 212.

[0057] In step 212, a hash value of the "new" file is calculated using a hashing function like MD5 or SHA-1. This calculation is performed by using some CPU time of the local computing device. Nevertheless, the amount of CPU time used is drastically smaller than the CPU time that would be necessary if e.g. a conventional virus checker was used to check the file on the local computing device 50. Method 200 then proceeds to step 214.

[0058] In step 214, the hash value and the file source information is transferred from the local agent to the central infrastructure 100 of the network 10. If necessary, this transfer can be a secured transfer, whereby it is avoided that a virus which is positioned on a network connection changes both the file source information or the hashing key during transport of this data. Such a secured transmission can be made over a known secure transmission route, via a tunnel, or using known session ecryption/decryption techniques.

[0059] In step 216, the hash value is compared with the data already present in the metabase 110. As in the metabase 110, the hash values and file source information of all old files - i.e. every file that has been present on the network 10 and that is not "new" as described above - present in the network 10 are stored, it is possible to check whether the file already is present in the network 10. Therefore, if the hash value has been identified as new, this implies that the file is "new" for the whole network 10. If the file is new, method 200 proceeds to step 218. If the hash value is not new, this means that somewhere on a local computing device 50 in the network 10, the file does already exist. In this case, there already exists content attributes describing the content of the file. Method 200 then proceeds to step 224.

[0060] In step 218 the metabase agent triggers the local agent to transfer the file corresponding with the new hash value from the local computing device 50 to the central infrastructure 100. The transferring of the file may be performed in a secured way, i.e. the file may be transferred such that it cannot be influenced by a virus present at a network connection or such that, it if contains a virus, this cannot be spread over the whole network 10. To obtain this a known secure transmission route, a tunnel and/or known session ecryption/decryption techniques may be used. Method 200 further proceeds to step 220.

[0061] In step 220 the file is loaded in the content identification engine 120 and the file is processed. For this processing CPU time of the central infrastructure 100 is used. The content identification engine 120 can comprise, as described above, a conventional virus checker, a means for checking picture information, a means for checking SPAM, etc. This can be a repetitive action where multiple content identification engines are called in turn. Method 200 then proceeds to step 222.

[0062] In step 222 content attributes, which identify the content of the file, are determined for the file. These content attributes are then stored in the metabase 110, thus allowing to identify the status of the file if, in future operations, the file is found 'new' on another local computing device 50. Method 200 then proceeds to step 224. Depending on the embodiment used, a following step may include the storing of the file on the central infrastructure 100 and adding the path to this file to the metabase 110. This step is not shown in figure 3.

[0063] In step 224 the content attributes are sent to

the local agent. Based on this content attributes, the local agent performs an appropriate action in agreement with the policy rule set for these content attributes. This is performed in step 226. This can be e.g. deleting the file if it was infected, replacing the file with a previous version which was not infected, ... In a specific embodiment, the execution of appropriate actions based on the policy rules are triggered by the agent of the metabase 110, so that step 224 can be avoided.

[0064] The content policy is a policy that determines what should be done with a file depending on the content attributes determined by the content identification engine 120. The content policy can comprise actions such as deleting the file, deleting the file and replacing it with a previous version, copying the file onto another computing device while leaving a copy on the originating computing device, moving the file onto another computing device while deleting the original file on the originating computing device, logging the presence of the file, changing the attributes of the file like hiding it or making it read-only, making the file unreadable, making the file un-executable, etc. The content policy will be executed by the local agent, e.g. when the content attributes are received from the central infrastructure 100. The content policy for that agent will be downloaded to the local computing device 50 by the agent from a central policy infrastructure.

[0065] Fig. 4 shows a method 300 of the content identification process triggered by the content identification engine 120 according to the above mentioned embodiments . The different steps that occur during the process, both on a local computing device 50 and on the central infrastructure 100 are discussed.

[0066] This process typically is used in situations where new algorithms or tools are being used for content identification. Such new algorithms or tools can either be optimised algorithms and tools or previously uninstalled tools. As mentioned earlier this may be regulated by a policy : the triggering of the content identification process may be determined by the type of new algorithms and tools that are being used for content identification.

[0067] Method 300 is initiated by change of the content identification engine 120, e.g. by providing new algorithms or tools for the content identification engine 120.

[0068] In the first step 302 of method 300 upon triggering, the metabase 110 is scanned for hash values corresponding with hashing keys. Method 300 then proceeds to step 304.

[0069] In step 304, a file that corresponds with the hashing key is requested. This file can be either requested from the central storage on the central infrastructure 100 or it can be requested from a local computing device 50. The local computing device 50 then gives permission to the central infrastructure 100 to upload the corresponding file. The path to the file corresponding with the hash value is available from the record corresponding with each hash value. If the record stores different paths all corresponding with a copy of the corresponding file, the agent on the central infrastructure 100 retrieves one copy of the file, e.g. by scanning the paths listed in the record until a local computing device 50 has been found that is at that time connected to the network 10 and that allows uploading of the file. Method 300 then proceeds to step 306.

[0070] Once the file has been retrieved, the file is sent to the content identification engine 120. This is performed in step 306. The upgraded content identification engine 120 then scans the content of the file and produces content attributes corresponding with the file. Method 300 further proceeds to step 308.

[0071] In step 308, the content attributes are stored in the metabase 110, to allow in future security steps to immediately identify the content of the files. Method 300 further proceeds to step 310.

[0072] In step 310, the content attributes are sent to every local agent that resides on a local computing device 50 whereon the corresponding file is stored. The paths can be found in the record of the corresponding hashing key stored in the metabase 110. In this step, content attributes are send to every file for which a path is mentioned in the record of the corresponding hashing key. Method 300 proceeds to step 312.

[0073] In step 312, the local agent on the corresponding local computing devices 50 executes the policy according to the content attributes and the according to the local computing device 50.

[0074] One of the major advantages of the embodiments of the invention is that a file new to the entire network 10 only needs to be scanned once. If on another local computing device 50, an identical copy of this file is used, installed, opened or saved and closed, the file will be recognised by the central infrastructure 100 as being known to the network 10, in this way avoiding the need to re-check the content of the file. This especially is advantageous if the invention is used for networks 10 having a large number of local computing devices 50.

[0075] When operating, the method and system for identifying the content of new files optionally can comprise checking 'the heartbeat' of the local agent at regular times, i.e. it can be checked whether the local agent is still running on the local computing device 50. This can avoid that a user locally shuts down the agent, thus making the local computing device 50 vulnerable. If the local agent has been shut down, the network administrator can be warned. Furthermore a warning message could be send to the local computing device 50 thereby warning the user of the local computing device 50. The network administrator could also put the local computing device 50 in quarantine so that it can not damage other local computing devices 50 in the network 10. Furthermore, the central agent can also try to rerun the local agent.

[0076] In a similar way, the method and system for identifying the content of new files optionally can check

at regular times whether the local computing device 50 is still connected to the network 10. If the local computing device 50 is not connected to the network 10 anymore, the local agent may further operate, storing hashing keys of new files in a waiting list to be checked once the network connection is restored. In the mean time, the corresponding files may be put in quarantine or depending on the type of file e.g. may be prevented from being executed.

[0077] The above described embodiments may be used as a content firewall for the different computing devices connected to the external network. For every incoming/outgoing file, incoming/outgoing message or incoming/outgoing data frame, the content firewall calculates the hash, checks whether this is new, checks whether it is tagged for specific content and enforces the policy associated with the specific content.

[0078] In a further embodiment, another configuration for using the present invention as a content firewall is described. A schematic overview of a computer network wherein this method and system may be used, is shown in Fig. 5. Only one reconfigurable firewall electronic device 50, such as a local computing device which may be in the form of a dedicated reconfigurable firewall electronic device, is directly connected to an external network 400 such as e.g. the internet, and the remaining local computing devices 410, are not directly connected to the external network 400, but grouped in a network environment and only connected to the external network 400 by their connection to the electronic reconfigurable firewall device.. The external network may be any possible network available. It is a goal of the content firewall as represented by the reconfigurable electronic firewall device 50 to protect the network environment comprising the remaining local computing devices 410 from attacks originating from places and/or devices in the external network. The reconfigurable electronic firewall device 50 50 either contains a local copy of the metabase or it can use a high speed secured network to a central infrastructure 100 which is part of the internal network. This allows for fast queries through the metabase. During operation, the reconfigurable electronic firewall device 50 functioning as a content firewall performs the following actions : the hash value of incoming files or incoming messages or incoming data frames are calculated. Subsequently, the calculated hash values are compared with the metabase, which is either stored locally or by using a high speed secured network, and it is determined whether the incoming file, incoming message or incoming data frame is new. Furthermore, it is checked whether this file, this message or this data frame is tagged for specific content. Depending on the specific content, a policy is enforced which is associated with the specific content. This policy may be to let it pass through to its final destination, to drop it, to log it, to put it in quarantine, etc. This system requires sufficient CPU power, in order not to slow down the network speed noticeably.

[0079] In the case where none of the local computing devices connected to the network is equipped with a removable device, i.e. allowing for non-scanned content to be opened or executed on that device, this is a very secure and managable setup.

[0080] In another embodiment of the invention, a similar configuration for use of the present invention as a content firewall in promiscuous mode is provided. The content firewall thereby sees all traffic passing by, executes the hashing and comparing functions and contacts the agents to enforce a policy. The advantage of this approach is that there is no single point of failure and no bottleneck anymore and furthermore that still no resources are used on the local computing devices for calculating hashes. Furthermore, no bandwidth is used for contacting the central metabase. The disadvantage is that local agents need to be installed on all computing devices of the internal network.

In accordance with the above described embodiments, the present invention includes a computer program product which provides the functionality of any of the methods according to the present invention when executed on a computing device. Further, the present invention includes a data carrier such as a CD-ROM or a diskette which stores the computer product in a machine readable form and which executes at least one of the methods of the invention when executed on a computing device. Nowadays, such software is often offered on the Internet, hence the present invention includes transmitting the printing computer product according to the present invention over a local or wide area network.

Claims

1. A method for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part of the network environment including a central infrastructure and, the method comprising

    - calculating a reference value for a new file on one of said at least one local computing devices using a one-way-function,
    - transmitting said calculated reference value to said central infrastructure,
    - comparing said calculated reference value with reference values previously stored within the remaining part of the network environment,
    - after comparing,

        - deciding that the content of the new file is already identified if a match between said calculated reference value and a previously stored reference value is found and retrieving the corresponding content attributes; or

- deciding that the content of the new file is not yet identified if no match between said calculated reference value and any of the previously stored reference values is found, followed by said central infrastructure identifying the content of said new file and determining content attributes corresponding with the content of the new file and storing a copy of said content attributes,
- after deciding, triggering an action on said local computing device in accordance with said content attributes.

2. A method according to claim 1, wherein the reference value is a hash value.

3. A method according to claim 1 or 2, wherein reference values previously stored are stored within the central infrastructure.

4. A method according to the previous claim wherein said identifying the content of said new file comprises scanning the new file for viruses using a anti-virus checker means on said central infrastructure.

5. A method according to any of the previous claims furthermore comprising transferring the new file from the local computing device to the central infrastructure before said identifying the content of said new file is performed.

6. A method according to claim 5, furthermore comprising storing a copy of the new file on the central infrastructure after said transferring the new file from the local computing device to the central infrastructure

7. A method according to any previous claim wherein said triggering an action on said local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with a copy of a previous version of said new file.

8. A method according to any previous claim, wherein said triggering an action on said local computing device in accordance with said content attributes may comprise replacement of the new file on the local computing device with another version of said new file restored from the remaining part of the network environment.

9. A method according to claim 1, 2, 3 or 4 furthermore comprising sharing the new file on the local computing device to the central infrastructure before said identifying the content of said new file is performed and whereby said identifying the content of said new file is performed by remotely identifying the content over the network environment.

10. A method according to any of the previous claims, wherein said triggering an action on said local computing device in accordance with said content attributes is performed after transmitting the content attributes corresponding to the new file to the local computing device.

11. A method according to the previous claims wherein said identifying the content of said new file comprises one or more of the group of scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

12. A method according to any of claims 1 to 3 or 5, whereby one local computing device links said network environment with an external network, said one local computing device acting as a content firewall as to protect the network environment.

13. A computer program product for executing any of the methods as claimed in claims 1 to 12 when executed on a network.

14. A system for identifying the content of a file in a network environment, said network environment comprising at least one local computing device linked to a remaining part the network environment which includes a central infrastructure and, said remaining part including a stored database, whereby the system comprises :

- means for calculating a reference value for a new file on said local computing device using a one-way-function,
- means for transmitting said calculated reference value to said central infrastructure,
- means for comparing said calculated reference value with previously stored reference values from the database,

whereby the system further comprises:

- means for deciding whether the content of the new file is already identified based on comparison of said calculated reference value and reference values previously stored within the remaining part,
- means located on the central infrastructure, for identifying the content of the new file and as to assign content attributes if the new file has not been identified yet and means for storing said content attributes within the remaining part, and
- means for triggering an action on said local

computing device in accordance with content attributes for said new file.

15. A system according to claim 14 wherein said means for identifying the content of a file comprise an anti-virus checker means on said central infrastructure.

16. A system according to claim 14 or 15 furthermore comprising means for storing a copy of the new file within the remaining part.

17. A system according to any of claims 14 to 16 wherein said means for identifying the content of a file comprises one or more of the group of means for scanning for adult content, scanning for Self Promotional Advertising Messages and scanning for copyrighted information.

18. A method for altering a system for identifying the content of a file in a network environment according to claims 14 to 17, said network environment comprising means for calculating a one-way function, at least one local computing device linked to a remaining part of the network environment including a central infrastructure and means for identifying the content and, the method comprising

- altering said means for identifying the content or said means for calculating a one-way function
- scanning the remaining part of the network environment for reference values calculated with a one-way function
- for each of said reference values,
    - requesting a file that corresponds with said reference value from said network environment
    - sending the file to means for identifying the content
    - identifying the content of said file and determining content attributes corresponding with the content of the file and storing a copy of said content attributes
    - sending the content attributes to every local computing device containing the file
    - after sending; triggering an action on said local computing device in accordance with said content attributes.

19. A computer program product for executing the method as claimed in claims 18 when executed on a network.

20. A machine readable data storage device storing the computer program product of any of claims 13 or 19.

21. Transmission of the computer program product of any of claims 13 or 19 over a local or wide area telecommunications network.

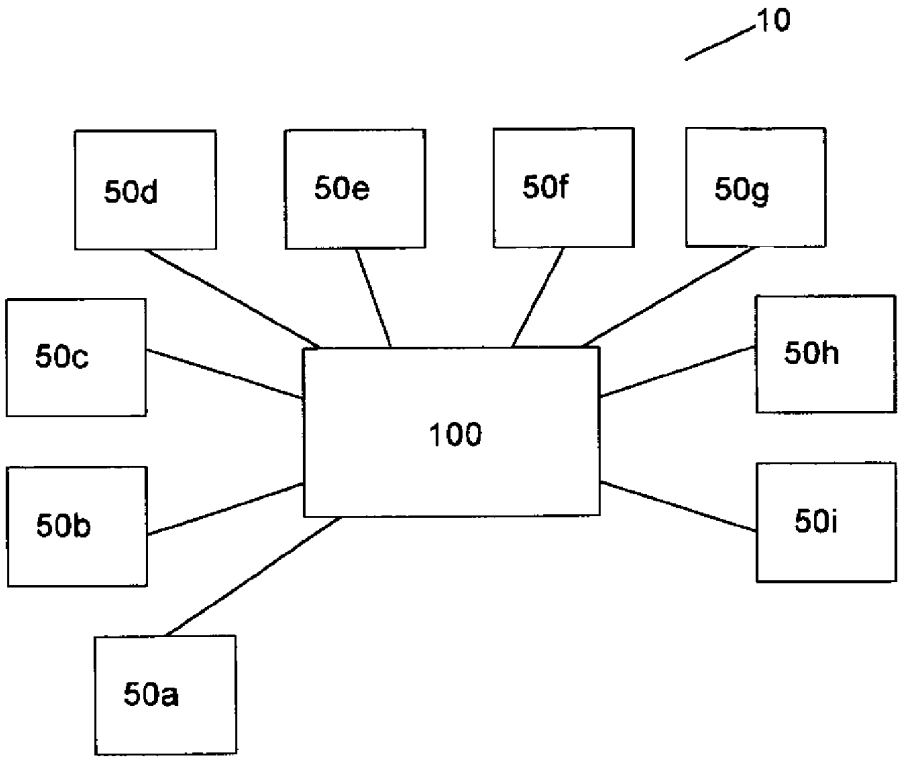

10
50d
50e
50f
50g
50c
50h
100
50b
50i
50a

Fig. 1

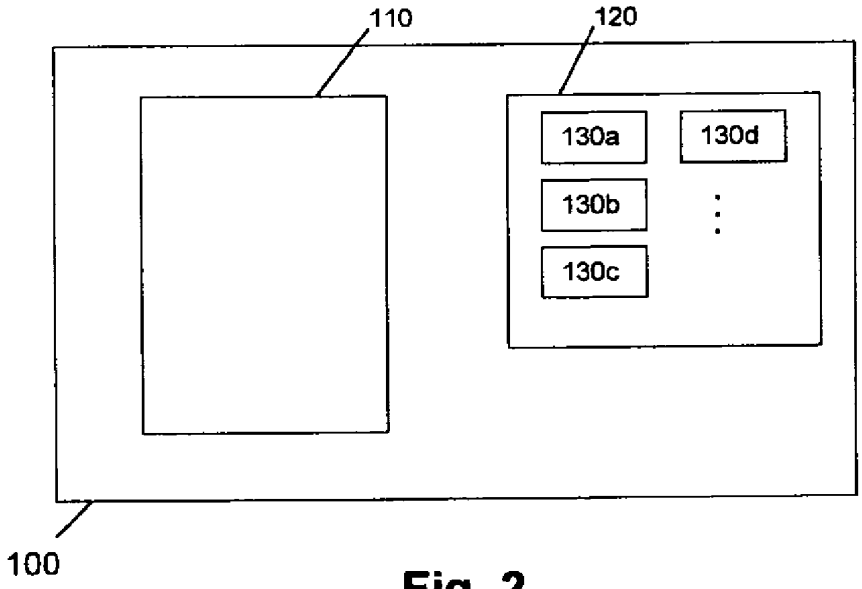

110
120
130a
130d
130b
130c
100

Fig. 2

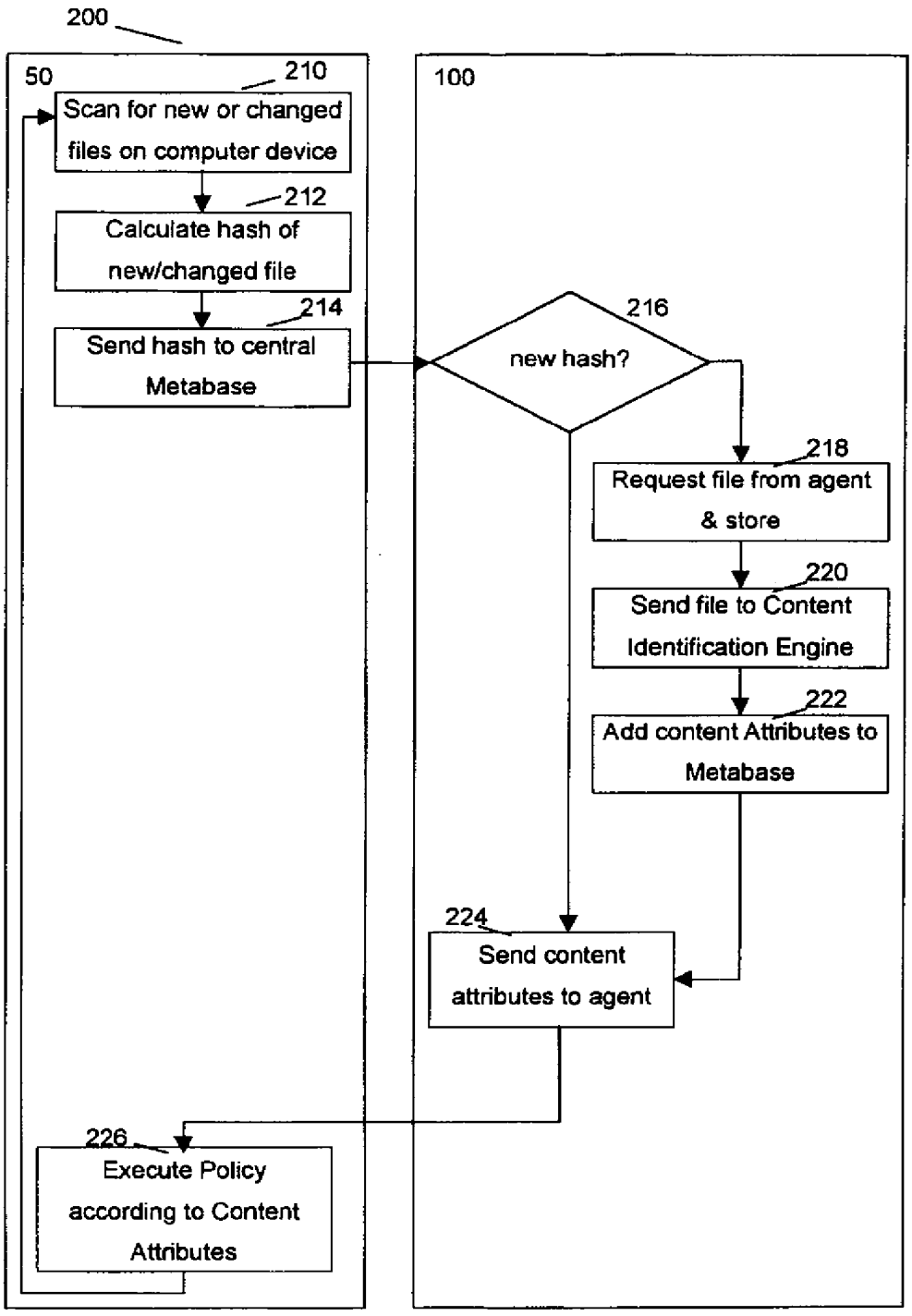

200
50
210
Scan for new or changed files on computer device
212
Calculate hash of new/changed file
214
Send hash to central Metabase
100
216
new hash?
218
Request file from agent & store
220
Send file to Content Identification Engine
222
Add content Attributes to Metabase
224
Send content attributes to agent
226
Execute Policy according to Content Attributes

Fig. 3

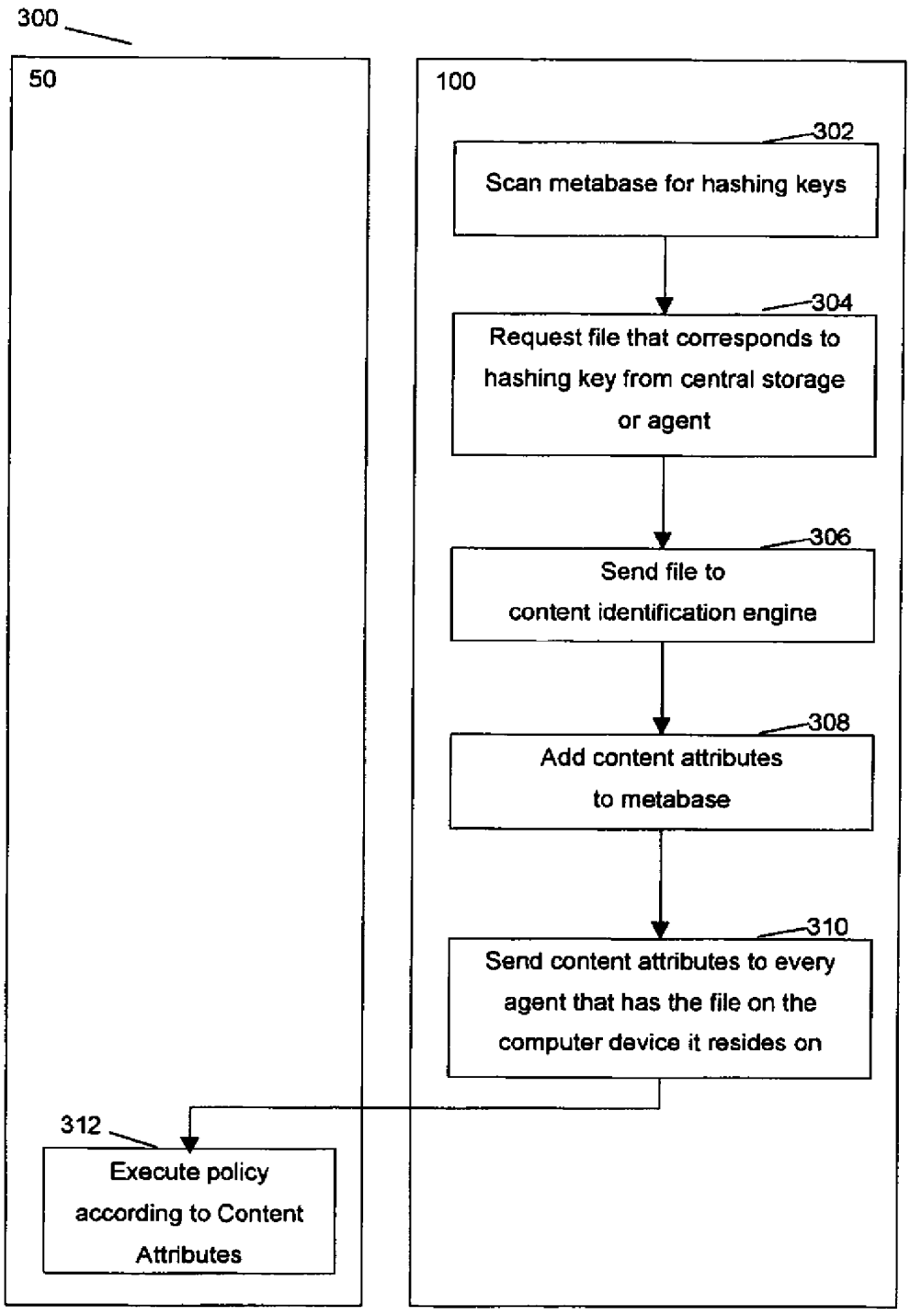

300
50
100
302
Scan metabase for hashing keys
304
Request file that corresponds to hashing key from central storage or agent
306
Send file to content identification engine
308
Add content attributes to metabase
310
Send content attributes to every agent that has the file on the computer device it resides on
312
Execute policy according to Content Attributes

Fig. 4

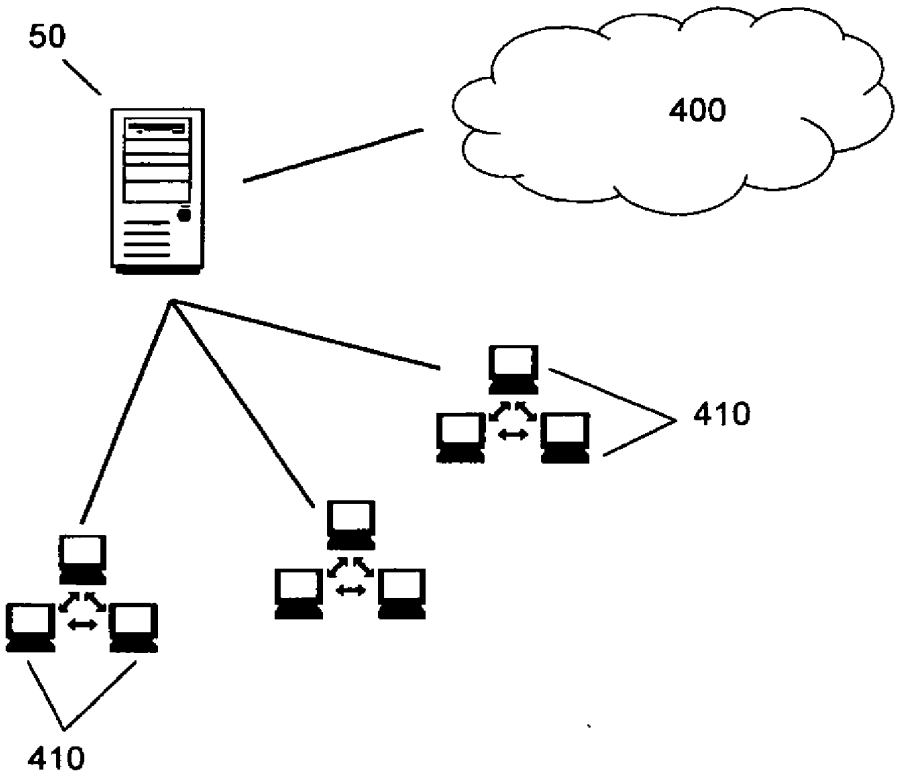


Fig. 5

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 03 44 7310

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	EP 1 280 040 A (NETWORKS ASSOCIATES TECHNOLOGY) 29 January 2003 (2003-01-29) * the whole document *	1-9, 11-17, 20,21	H04L29/06 G06F1/00
A	---	10,18,19	
A	US 6 195 587 B1 (HRUSKA JAN ET AL) 27 February 2001 (2001-02-27) * column 1, line 4-9 * * column 1, line 50 - line 65 * * column 3, line 55 - column 5, line 67 * * column 1, line 50 - column 3, line 3 * * column 3, line 55 - column 5, line 40 *	1-6, 9-11, 13-20	
A	---		
A	US 2003/120953 A1 (RADATTI PETER V) 26 June 2003 (2003-06-26) * paragraphs [0013]-[0015] * * paragraph [0045] * * paragraph [0047] * * paragraph [0051] *	1-4,7,8, 11, 13-15, 17,20	
	---		TECHNICAL FIELDS SEARCHED (Int.Cl.7)
A	EP 1 111 507 A (SECURE OYJ F) 27 June 2001 (2001-06-27) * paragraphs [0002]-[0004] * * paragraphs [0010]-[0012] * * paragraph [0015] * * paragraphs [0025],[0026] * * paragraph [0031] *	1,4-8, 12-16,20	H04L G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	7 June 2004	Ruiz Sanchez, J

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 03 44 7310

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

07-06-2004

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1280040 A	29-01-2003	US 2003023866 A1	30-01-2003
		EP 1280040 A2	29-01-2003
US 6195587 B1	27-02-2001	GB 2283341 A	03-05-1995
		AU 8000394 A	22-05-1995
		DE 69407812 D1	12-02-1998
		DE 69407812 T2	16-07-1998
		EP 0725951 A1	14-08-1996
		WO 9512162 A1	04-05-1995
		JP 9504395 T	28-04-1997
US 2003120953 A1	26-06-2003	NONE	
EP 1111507 A	27-06-2001	GB 2353372 A	21-02-2001
		EP 1111507 A2	27-06-2001
		US 2001005889 A1	28-06-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号
特開2003－196112
（P2003－196112A）
(43)公開日　平成15年７月11日(2003.7.11)

(51)Int.Cl.[7]	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　11/00		Ｇ０６Ｆ　9/06	６６０Ｎ　５Ｂ０７６

審査請求　未請求　請求項の数１　ＯＬ　（全 21 頁）

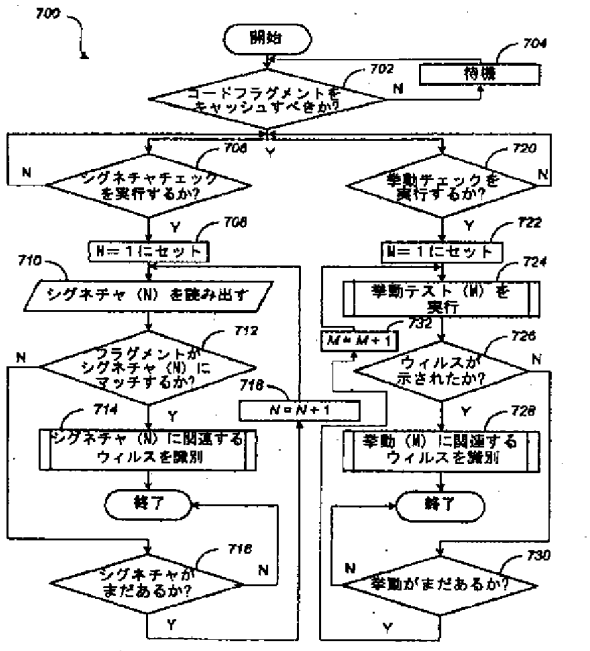
(21)出願番号　特願2002－343392(P2002－343392)
(22)出願日　平成14年11月27日(2002.11.27)
(31)優先権主張番号　０９／９９７，４５４
(32)優先日　平成13年11月29日(2001.11.29)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　398038580
ヒューレット・パッカード・カンパニー
ＨＥＷＬＥＴＴ－ＰＡＣＫＡＲＤ　ＣＯＭＰＡＮＹ
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(72)発明者　ニコライ・マティーブ
アメリカ合衆国01720マサチューセッツ州アクトン、フォックス・ヒル・ロード　３
(74)代理人　100081721
弁理士　岡田　次生　（外２名）

最終頁に続く

(54)【発明の名称】　ウィルスチェックソフトウェアのウィルスチェック方法

(57)【要約】
【課題】感染ソフトウェアコードを検出する改良されたウィルスチェックの方法を提供すること。
【解決手段】本開示は、ウィルスチェックプログラムのシステム（３００）と方法（７００）に関する。一実施形態において、システム（３００）と方法（７００）は、プログラム命令（７０２）をインターセプトし、対応する命令が既知のウィルス（７１２）を識別する１以上の「シグネチャ」バイトを含んでいるかを判断し、インターセプトしたコードにウィルスのシグナチャが全く含まれていないと判断したときだけインターセプトされたコードをコンピュータ・ハードウェア（１０４）に開放する。



1

【特許請求の範囲】
【請求項１】　感染したプログラム命令を識別する方法であって、
動的実行レイヤインタフェース（ＤＥＬＩ）を計算装置ハードウェアと前記プログラム命令の間に挿入するステップと、
前記プログラム命令が前記ＤＥＬＩに入る際に前記プログラム命令を監視して、コードが前記計算装置ハードウェアによって以前に処理されたことがあるかどうかを判断し、前記プログラム命令が以前に処理されたことがない場合には、
前記プログラム命令を解析して、前記プログラム命令が感染しているかどうかを判断するステップとを含む方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本開示は、バイナリプログラムコード実行を動的変換するシステムおよび方法に関する。特に、本開示は、ウィルスチェック実行可能コードのシステムおよび方法に関する。
【０００２】
【従来の技術】コンピュータウィルスは、２つの固有の機能を有するプログラムである。第１に、コンピュータウィルスは、自己をマシンからマシンに拡散させる（自己増殖コード）。これは、他のシステムの実際の感染ならびに可能な限り多くの「キャリア」にコードを隠すことを含む。第２に、コンピュータウィルスは、ウィルスの仕掛け人が計画する「症状」を実施する。これは、モニタ上へのメッセージ表示から特定の日に固定データディスクのコンテンツを消去することまで、かなり多数の事柄でありうる。
【０００３】コンピュータウィルスおよび他の形態の電子「感染」が、コンピュータである限りほとんどに蔓延している。近代コンピュータの父であるJohn Van Neumannは、早くも１９４８年に自己増殖コードの概念を抱いていた。多くの異なる形態の電子感染が認められている。最も一般的なのは、ピギーバックウィルス、電子メールウィルス、ワーム、およびトロイの木馬である。ピギーバックウィルスは、本物のプログラムに「追記する（piggy-back）」小型ソフトウェアである。たとえば、ウィルスは、自己をスプレッドシートプログラムのようなプログラムに添付することができる。スプレッドシートが実行される度に、ウィルスも実行され、自己増殖する（他のプログラムに添付されることにより）かまたは大きな被害を与えるチャンスを持つ。電子メールウィルスは、コンピュータからコンピュータに電子メールメッセージで移動し、通常、自動的に自己を被害者のアドレス帳における多くの人々にメールすることにより、自己複製する。ワームは、コンピュータネットワークおよびセキュリティホールを使用して自己を複製する小型ソフトウェアである。ワームのコピーは、特定のセキュリティホールを有する別のマシンについてネットワークをスキャンする。セキュリティホールを使用して新しいマシンに自己を複製し、次に、そこからも同様に複製を開始する。トロイの木馬は、単に通常のコンピュータプログラムである。プログラムは、１つのことをするものと自称する（たとえば、ゲームであると自称する）が、代わりに、そのプログラムを実行するとダメージを及ぼす（たとえば、ハードディスクを消去する）。トロイの木馬は、自動的に複製する方法を持たない。

2

【０００４】初期のウィルスは、人気のあるゲームまたは人気のあるワードプロセッサのような一般的なプログラムに添付されるコード片であった。掲示板から感染したゲームがダウンロードされると、感染ファイルが実行されうる。このようなウィルスは、より大きな正当なプログラムに埋め込まれる小型のコードである。あらゆるウィルスは、正当なプログラムが実行されるとき最初に実行するように設計される。ウィルスは、自己をメモリにロードし、周りを見てディスク上に他にプログラムを見つけることができるかを調べる。プログラムを見つけることができる場合、ウィルスのコードを疑いのないプログラムに追加するように変更する。次に、ウィルスは、「本物のプログラム」を起動する。ユーザは、実際に、ウィルスが既に実行されたことを知る術がない。不都合なことに、ウィルスは今や自己増殖してしまったため、２つのプログラムが感染している。これらプログラムのいずれかが次に実行されるとき、ウィルスは他のプログラムに感染し、このサイクルが継続する。
【０００５】感染したプログラムの１つがフロッピー（登録商標）ディスクで別の人に与えられる場合、または掲示板にアップロードされる場合、他のプログラムが感染する。これが、ウィルスの拡散の仕方である。
【０００６】拡散する部分は、ウィルスの「感染」段階である。ウィルスは、することが自己の複製だけであったなら、それほどひどく嫌悪されなかったであろう。不都合なことに、ほとんどのウィルスは、あるダメージを引き起こすある種の破壊的な「攻撃」段階も有する。ある種のトリガが、攻撃段階を始動させ、次にウィルスが「何かをする」。何かとは、馬鹿げたメッセージの画面上への表示からデータの消去までである。トリガは、特定の日付、ウィルスが複製された回数、または同様の何かでありうる。
【０００７】ウィルス作成者は、高度になるにつれ、新しいトリックを学んだ。１つの重要なトリックは、コンピュータがオンのままである限り、バックグラウンドで実行し続けることができるように、ウィルスをメモリにロードする能力であった。これは、自己複製するはるかに効果的な方法をウィルスに与えた。別のトリックは、フロッピーディスクおよびハードディスク上のブートセクタに感染する能力であった。ブートセクタは、コンピ

ュータがロードする、オペレーティングシステムの最初の部分である小さなプログラムである。ブートセクタは、オペレーティングシステムの残りをロードする方法をコンピュータに教える非常に小さなプログラムである。コードをブートセクタに配置することにより、ウィルスはその実行を確実にする。即座に自己をメモリにロードすることができ、コンピュータがオンのときはいつでも実行可能である。ブートセクタウィルスは、マシンに挿入されたあらゆるフロッピーディスクのブートセクタに感染可能であり、多くの人がマシンを共有する大学構内では、野火のように拡散する。

【０００８】概して、実行可能ウィルスおよびブートセクタウィルスは双方とも、今はそれほど脅威ではない。衰退の第１の理由は、今日のプログラムのサイズが巨大であることである。プログラムが大きすぎるため、プログラムを持ち運ぶ唯一の容易な方法はコンパクトディスクを購入することである。今日購入される略全てのプログラムは、読み出し専用コンパクトディスクで販売される。これら読み出し専用コンパクトディスクは変更不可能である。データが変更不可能であるため、これらのコンパクトディスクは感染不可能である。プログラムで一杯のフロッピーが野球カードのようにトレーディングされた１９８０年代に行われていたように、フロッピーディスクでアプリケーションを持ち歩くことは間違いなく不可能である。ブートセクタウィルスは、オペレーティングシステムが今ではブートセクタを保護することによっても、衰退した。

【０００９】ブートセクタウィルスおよび実行可能ウィルスは双方とも、依然として可能であるが、今日ではるかに困難であり、以前に可能であったほど高速には拡散しない。フロッピーディスク、小さなプログラム、および脆弱なオペレーティングシステムという環境が１９８０年代においてウィルスを可能にしていたが、その環境上の適所は、巨大な実行可能読み出し専用コンパクトディスク、より徹底的なオペレーティングシステムの安全対策、および巨大なコンピュータネットワークによって大方なくなっている。

【００１０】次に、ウィルス作成者は、便利な送出方法を求めて電子メールおよびネットワークを調べた。Ｍｅｌｉｓｓａウィルスは、１９９９年３月に華々しくデビューした。Ｍｅｌｉｓｓａは、電子メールを介して送信されるＭｉｃｒｏｓｏｆｔＷｏｒｄ文書で拡散した。ウィルスの作成者は、感染コードをＷｏｒｄ文書に配置し、その文書をインターネットニュースグループにアップロードする。その文書をダウンロードし開いたいずれの人も、ウィルスを引き起こした。次に、ウィルスは、その文書（ひいては自己）を電子メールメッセージでユーザのアドレス帳の中の最初の５０人に送信する。電子メールメッセージは、ユーザ氏名を含むフレンドリーなメモを包含したため、受信者は、害のないものと考えて文書を開く。次に、ウィルスは受信者のマシンから新しい５０のメッセージを作成する。その結果、Ｍｅｌｉｓｓａウィルスはそれまでの中で最も高速に拡散した。

【００１１】２０００年５月に出現したＩＬＯＶＥＹＯＵウィルスは、さらに単純であった。このウィルスは、自己複製コードを電子メールの添付ファイルとして包含する。添付ファイルをダブルクリックした（それによって添付ファイルを開いた）人は、コードを実行させた。コードは、自己のコピーを、被害者のアドレス帳の中の全員に送信してから、被害者のマシン上のファイルを破壊し始める。これは、ウィルスがなりうる限り最も単純なものである。実際に、電子メールで分散されるのはウィルスよりもトロイの木馬である。

【００１２】Ｍｅｌｉｓｓａウィルスは、ＶＢＡ(Visual Basic for Applications)と呼ばれるＭｉｃｒｏｓｏｆｔ　Ｗｏｒｄに内蔵されるプログラミング言語を利用した。ＶＢＡは、完全なプログラミング言語であり、ファイル変更および電子メールメッセージ送信のようなことをするようにプログラムすることができる。ＶＢＡは、有用であるが、危険な自動実行機能も有していた。プログラマは、文書が開いているときはいつでも、実行されている文書にすぐにプログラムを挿入することができる。これは、Ｍｅｌｉｓｓａウィルスのプログラムの仕方である。Ｍｅｌｉｓｓａに感染した文書を開いた人は誰でも、即座にウィルスを起動する。Ｍｅｌｉｓｓａは、５０通の電子メールを送信してから、中央ファイルＮＯＲＭＡＬ．ＤＯＴに感染するため、後に保存されたいかなるファイルもウィルスを含むことになった。

【００１３】最も普及しているビジネスアプリケーションであるＭｉｃｒｏｓｏｆｔソフトウェアアプリケーションは、マクロを介しての感染を回避するために、内蔵されたマクロウィルス保護と呼ばれる機能を有する。マクロウィルス保護機能が「オン」である場合、自動実行機能がディスエーブルされる。デフォルトにより、オプションは「オン」である。したがって、文書が感染自動実行コードを試行すると、ユーザに警告するダイアログがポップアップする。不都合なことに、多くの人々は、マクロまたはマクロウィルスが何かわからず、また、ダイアログを見ても無視する。そのため、あまりウィルスに精通していな操作者は、どのみちウィルスを実行してしまうことが多い。他の多くの人々は、保護機構をオフにし、警告ダイアログを受け取ったことがない。その結果、Ｍｅｌｉｓｓａウィルスは、阻止のために安全対策が実施されているにも関わらず拡散する。

【００１４】ＩＬＯＶＥＹＯＵウィルスの場合、感染プロセス全体が人力によるものである。プログラム（電子メールの添付ファイルとして送られる）をダブルクリックすると、プログラムが実行され、したがってウィルスが分散される。実行可能添付ファイルを進んで開こうという人間の意思が、ＩＬＯＶＥＹＯＵウィルスを煽って

いる。

【００１５】他の有名ではないウィルスは、人間の他の弱点を利用する。たとえば、Ａｎｎａ　Ｋｏｕｒｎｉｋｏｖａウィルスは、プロのテニス選手およびモデルにちなんで名付けられた。作成者は、「Ｙｏｕ’ｖｅ　ｇｏｔ　ｔｏ　ｓｅｅ　ｔｈｉｓ．．．」を電子メールメッセージの主題欄に使用した。メッセージの本文では、Annaの写真が約束されている。明らかに、かなりの男性がこの誘惑法に騙されて実行可能添付ファイルをダブルクリックしたであろう。

【００１６】一般に人間のアクティビティに依拠して土台をなすコードを分散するウィルスとは異なり、ワームは、人間の介入なしでマシンからマシンへ自己をコピーする能力を有する。ワームは通常、コンピュータネットワークを通してあちこち移動し、他のマシンに感染する。ネットワークを使用して、ワームは、単一コピーから信じられないほど高速に膨張することができる。たとえば、Ｃｏｄｅ　Ｒｅｄワームは、２００１年７月におおよそ９時間で自己を２５０，０００倍以上に複製した。

【００１７】ワームは、複製されるときにコンピュータ時間およびネットワーク帯域を使い果たし、またある種の悪意を有することが多い。Ｃｏｄｅ　Ｒｅｄワームは、自己複製を開始すると、インターネットトラフィックを低速化させた（予想ほどひどくはなかったが）。ワームの各コピーは、パッチの当てられていない（すなわち、保護されていない）ＷｉｎｄｏｗｓＮＴ（登録商標）またはＷｉｎｄｏｗｓ（登録商標）２０００サーバを求めてインターネットをスキャンする。安全ではないサーバを見つける度に、ワームは自己をそのサーバにコピーする。次に、新しいコピーが、感染する他のサーバを求めてスキャンする。安全ではないサーバの数に応じて、ワームは、破壊段階に入る前に、おそらく、数十万ものコピーを作成することができる。

【００１８】たとえば、Ｃｏｄｅ　Ｒｅｄワームは、（ａ）各月の最初の２０日間自己を複製し、（ｂ）感染したサーバ上のウェブページを、「Ｈａｃｋｅｄ　ｂｙ　Ｃｈｉｎｅｓｅ」と宣言するページで置換し、（ｃ）乗っ取りを試みて１つまたは複数のウェブサーバ上で一斉に攻撃を開始するように設計される。

【００１９】米国の国防総省および大学によって使用されている大規模ネットワークであるＡＲＰＡＮＥＴ(Advanced Research Projects Agency Network)がウィルス感染した１９８７年以来、多くのアンチウィルスプログラムが利用可能になった。これらプログラムは、最もよく知られているタイプのウィルスについてコンピュータシステムを定期的にチェックする。商業的活動が密接に統合され、ローカルエリアネットワークおよびインターネット等広域ネットワークの双方に依拠するようになるにつれ、アンチウィルスプログラムおよび他のコンピュータウィルス保護方法が、データの保護にますます重要になってきている。

【００２０】単に自己を複製する単純なウィルスは、検出が最も容易である。ユーザが感染したプログラムを起動する場合、ウィルスは、コンピュータの制御を得て、自己のコピーを別のプログラムファイルに添付する。拡散後、ウィルスは制御をホストプログラムに戻し、ホストプログラムは正常に機能する。単純なウィルスが、たとえば、新しいファイルまたはフロッピーディスクに何度感染するかに関係なく、感染により常に自己の完全なコピーが作成される。アンチウィルスソフトウェアは、ウィルスを識別するためのシグネチャとして知られる徴候を示すバイトシーケンスについて、探索またはスキャンする必要があるだけである。

【００２１】これに応答して、ウィルス作成者はウィルスを暗号化し始めた。この概念は、ウィルスコードをスクランブルし、ウィルススキャナにとって認識不可能にすることによって「固定シグネチャ」を隠すことであった。暗号化されたウィルスは、ウィルス解読ルーチンおよび暗号化されたウィルス本体からなる。ユーザが感染したプログラムを起動すると、ウィルス解読ルーチンがまずコンピュータの制御を得てから、ウィルス本体を解読する。次に、解読ルーチンがコンピュータの制御を解読されたウィルスに渡す。

【００２２】暗号化されたウィルスは、単純なウィルスがいずれもするように、プログラムおよびファイルに感染する。新しいプログラムに感染する度に、ウィルスは、解読されたウィルス本体および関連する解読ルーチンの双方のコピーを作成し、コピーを暗号化し、双方をターゲットに添付する。ウィルス本体のコピーを暗号化するために、暗号化されたウィルスは、ウィルスが感染毎に変化するようにプログラムされた暗号鍵を使用する。この鍵は変更されるため、ウィルス本体のスクランブリングが変更され、ウィルスが感染毎に異なって見える。これにより、アンチウィルスソフトウェアが、一貫したウィルス本体から抽出されるウィルスシグネチャについて探索することが極めて難しくなる。

【００２３】アンチウィルスプログラマにとって重大なことに、かつ幸運なことに、解読ルーチンは生成毎に一貫したままであり、これは、こういった暗号化されたウィルスを識別するために、アンチウィルスソフトウェアが迅速に利用する弱点である。ただシグナチャについてスキャンする代わりに、ウィルススキャナは、特定の解読ルーチンを識別するバイトシーケンスについても探索するように変更された。

【００２４】これに関連して、ウィルス作成者はポリモーフィック型（polymorphic、多様型）ウィルスを作成した。暗号化されたウィルスのように、ポリモーフィック型ウィルスは、スクランブルされたウィルス本体、およびまずコンピュータの制御を得てからウィルス本体を

解読する解読ルーチンを含む。しかし、ポリモーフィック型は、これら２つの構成要素に、第３の、ウィルスが新しいプログラムに感染する都度変化するランダム化された解読ルーチンを生成する突然変異エンジンを追加する。

【００２５】ポリモーフィック型ウィルスの場合、突然変異エンジンおよびウィルス本体が双方とも暗号化される。ユーザがポリモーフィック型ウィルスに感染したプログラムを実行すると、解読ルーチンがまずコンピュータの制御を獲得してから、ウィルス本体および突然変異エンジンの双方を解読する。次に、解読ルーチンがコンピュータの制御をウィルスに渡し、ウィルスが感染する新しいプログラムを求めて探索する。この時点において、ウィルスは、自己および突然変異エンジン双方のコピーをランダムアクセスメモリ（ＲＡＭ）に作成する。次に、ウィルスは、突然変異エンジンを呼び出し、突然変異エンジンが、ウィルスを解読可能であるが、以前のいかなる解読ルーチンにもほとんどまたはまったく類似しない新しい解読ルーチンをランダムに生成する。次に、ウィルスは、ウィルス本体および突然変異エンジンのこの新しいコピーを暗号化する。最後に、ウィルスは、この新しい解読ルーチンを、新たに暗号化されたウィルスおよび突然変異エンジンと共に、新しいプログラムに追記する。

【００２６】したがって、ウィルス本体が暗号化されるだけではなく、ウィルス解読ルーチンも感染毎に変化する。これは、特定の解読ルーチンを識別するバイトシーケンスを求めて探索するウィルススキャナを混乱させる。スキャンする固定シグネチャがなく、かつ固定の解読ルーチンがない場合、２つの感染は同じに見えない。

【００２７】アンチウィルス研究者はまず、ポリモーフィック型ウィルスそれぞれを１つずつ捕らえるように設計された特別な検出ルーチンを作成することによって反撃した。これら特別なプログラムは、ウィルス本体を解読するために所与の突然変異エンジンが使用するとわかっている様々なコンピュータコードシーケンスを検出する。この手法は先天的に非実用的であり、時間およびコストがかかり、常にウィルス作成者に一歩遅れる。

【００２８】さらに、多くのポリモーフィック型ウィルスは、ポリモーフィック型突然変異エンジンを生み出し、これを共有するウィルス作成者のおかげで、同じ突然変異エンジンを使用する。加えて、他のポリモーフィック型ウィルスが使用する別のエンジンも同様の解読ルーチンを生成することが多く、これにより、解読ルーチンのみに基づくいかなる識別も完全に信頼することが不可能になる。したがって、突然変異エンジンを識別することによってポリモーフィック型ウィルスを識別しようとすると、１つのポリモーフィック型ウィルスを別のものとして誤って識別することにつながりうる。これら欠点により、アンチウィルス研究者は、ポリモーフィック型ウィルスをだまして自己を解読または明らかにさせる汎用解読技術を開発した。

【００２９】汎用解読を使用するウィルススキャナは、いくつかの挙動に基づいてポリモーフィック型ウィルスを検出する。ウィルススキャナは、新しいプログラムファイルをスキャンする都度、ＲＡＭに作成される自己完結型仮想コンピュータにファイルをロードする。仮想コンピュータ内において、プログラムファイルが、実際のコンピュータで実行されるかのように実行される。スキャナは、プログラムファイルが仮想コンピュータ内で実行されるときに、プログラムファイルを監視および制御する。仮想コンピュータ内で実行されるポリモーフィック型ウィルスは、実際のコンピュータからは分離されているため、ダメージを与えることはできない。ウィルススキャナが、ポリモーフィック型ウィルスに感染したファイルを仮想コンピュータにロードする場合、ウィルス解読ルーチンが実行され、暗号化されたウィルス本体を解読する。これにより、ウィルス本体がスキャナに曝され、次に、スキャナが、特定のウィルス株を正確に識別する、ウィルス本体中のシグネチャについて探索することができる。

【００３０】ウィルススキャナが感染していないファイルをロードする場合、曝され監視すべくウィルスはない。非ウィルスの挙動に応答して、スキャナは、仮想コンピュータ内でのファイルの実行を迅速に停止し、ファイルを仮想コンピュータから取り出し、次のファイルのスキャンに進む。汎用解読に伴う主要な問題は、速度である。汎用解読方法は、ポリモーフィック型ウィルスを仮想コンピュータ内で解読するのに時間がかかりすぎる場合、実用的ではない。同様に、汎用解読が単純に短時間で停止されると、ウィルススキャナはウィルスが現れてスキャナがシグネチャを検出する前に、ポリモーフィック型ウィルスを逃しうる。

【００３１】この問題を解決するために、ウィルススキャナはしばしば、非ウィルスの挙動とウィルスの挙動を区別するのを助ける汎用ルールセットである「ヒューリスティクス」を採用する。たとえば、典型的な非ウィルスプログラムは、仮想コンピュータで実行されるときに作成する数学的計算からの結果を使用する。反対に、ポリモーフィック型ウィルスは、同様の計算を実行しうるが、その結果は、ウィルスにとって無関係であるため捨てられる。実際、ポリモーフィック型ウィルスは、ウィルススキャナを避けようとするため、有効なプログラムのように見せるためだけにこのような計算を行うことがある。

【００３２】ヒューリスティックベースのウィルススキャナは、このように一貫しない挙動を捜す。非一貫性は、あるタイプの感染の可能性を増大させ、ヒューリスティックベースのルールに依拠するスキャナに、仮想コンピュータ内での疑いのあるファイルを実行する時間の

長さを延長し、感染した可能性のあるファイルに自己を解読し暗号化されたウィルスを曝すためにより長い時間を与えるように促す。ヒューリスティックベースのウィルススキャナは、ウィルス作成者に遅れないために、継続的なリサーチおよび更新を必要とする。何百もの既知のウィルスを検出するように調整されるヒューリスティックルールは、新しく識別されたウィルスを検出するために更新されるとき、少数のウィルスを見逃しうる。加えて、ウィルス作成者がウィルスを非感染プログラムのように見せかけ続けるため、ヒューリスティックベースのルールは、略全てのプログラムが、ウィルススキャナをトリガしうるいくつかの属性を共有しうるポイントまで容易に膨張して、ファイルの検査にかかる時間量が長くなりうる。

【００３３】さらに、汎用解読技術を利用するウィルススキャナは、何百万もの潜在的なウィルス変形を解析し、シグネチャを抽出し、次に、既知のウィルスの識別に成功したヒューリスティックルールの変更の含意からガードしながら、ヒューリスティックのセットを生成かつ／または変更することが可能なアンチウィルス研究者のチームに依拠する。これには、広範囲かつ網羅的な回帰テストが必要である。

【００３４】これに応答して、Symantec Corporation（Cupertino, California）は、仮想コンピュータにおいて動作するプログラムとは反対に、非ヒューリスティックベースのルールセットを使用するシステムを採用する。Symantecのシステムは、非ウィルスの挙動とウィルスの挙動とを区別する汎用ルールセットに対して、各ウィルスに固有のウィルスプロファイルまたはルールに依拠する。新しいファイルをスキャンする場合、Symantecのシステムはまず、可能な限り多くのウィルスを考慮から外すよう試みる。たとえば、異なるウィルスが異なる実行可能ファイルフォーマットに感染する。ウィルスによっては、．ＣＯＭファイルのみに感染するものもあり、．ＥＸＥファイルにのみ感染するものもある。ウィルスによっては、双方に感染するものもある。非常に少数が．ＳＹＳファイルに感染する。したがって、．ＥＸＥファイルスキャン時に、Symantecのシステムは、．ＣＯＭファイルおよび．ＳＹＳファイルのみに感染するポリモーフィック型ウィルスを無視する。すべてのウィルスが考慮から外される場合、ファイルは感染していないものと見なされ、仮想コンピュータから取り出される。

【００３５】予備スキャンにより観察中のファイル内の感染コードの存在が除外されない場合、Symantecシステムは、そのファイルの挙動が、少なくとも１つの既知のポリモーフィック型ウィルスまたは突然変異エンジンと一致する限り、仮想コンピュータ内でファイルを実行し続ける。たとえば、あるポリモーフィック型ウィルスは、数学的計算を行い、次にその結果を捨てることがわかっている。第２のポリモーフィック型ウィルスは、このような計算は決して行わない。代わりに、特定のランダム命令をその解読ルーチンにおいて使用する。第３のポリモーフィック型ウィルスは、解読時にオペレーティングシステム上で呼び出しを行う。Symantecのシステムは、これらおよび何百もの他の特徴を、ポリモーフィック型ウィルスおよび突然変異エンジンそれぞれに１つずつ、各ウィルスプロファイルにカタログ化する。Symantecのウィルス検出システムは、既知のウィルスプロファイルに一致しない挙動を検出すると、これら既知のウィルスを考慮から外す。このプロセスは、仮想コンピュータで実行中のプログラムの挙動が、任意の既知のポリモーフィック型ウィルスまたは突然変異エンジンの挙動と一貫しなくなるまで、続けられる。

【００３６】Symantecのウィルス検出システムの高度な性質にも関わらず、新しいウィルスを隔離および識別するようにウィルスカタログが更新されるまで、いくつかの「新しい」ウィルスが検出に対して耐性を有することになる。特定のトリガ条件を待つ他のウィルスは、検出されない場合がある。たとえば、ポリモーフィック型ウィルスによっては、プログラムのユーザが特定のキーを押下するのを待つようにプログラムすることができるものがある。このタイプのポリモーフィック型ウィルスは、実際のコンピュータに感染し、ユーザが指定されたキーストロークを入力した後でのみ、ウィルスを実行する。操作者がキーストロークを入力しない場合、ウィルスは起動する機会を得ない。汎用解読方法およびSymantecのウィルス検出システムによって作成される仮想コンピュータ内において、仮想プログラムは必要とされるキーストロークを決して受信しないであろう。

【００３７】

【発明が解決しようとする課題】感染コードの検出に現在利用可能な性能にも関わらず、上記問題の１つまたは複数を回避する、感染ソフトウェアコードを検出する改良されたシステムおよび方法を有することが望ましいことを理解することができる。

【００３８】

【課題を解決するための手段】従来技術のこれらおよび他の欠点に対応して、プログラム命令をインターセプトし、インターセプトしたコードを実際のコンピュータで初めて実行するときに、そのコードを仮想コンピュータにおいて適用するシステムを使用するウィルスチェックソフトウェアのシステムおよび方法が提供される。一構成では、システムおよび方法は、プログラム命令をインターセプトし、コードをコンピュータに送信する前に、感染コードおよび／またはウィルスのような挙動を識別するように設計される１つまたは複数のテストがコードに適用される。コードフラグメントが感染コードを含むと判断される場合、感染コードはキャッシュされず、したがって実際のコンピュータで動作しないようにされる。手短に述べると、アーキテクチャでは、計算装置

と、ランダムアクセスメモリと、少なくとも１つのアプリケーション（すなわち、テスト中のアプリケーション）と計算装置ハードウェアの間に存在する動的実行レイヤインタフェース（ＤＥＬＩ）とを使用して、改良されたウィルスチェックソフトウェアシステムを実現することができる。

【００３９】本開示は、コンピュータ読み取り可能媒体に格納されるプログラム内の感染コードの識別にも関する。一構成では、ウィルス検出マネージャは、プログラム命令をインターセプトするように構成されるロジックと、プログラム命令を実行すべきかどうかを判断するように構成されるロジックと、プログラム命令が感染しているかどうかを識別するように構成されるロジックとを備える。

【００４０】加えて、本開示は、少なくとも１つのアプリケーションと計算システムハードウェアの間に存在する動的実行レイヤインタフェース（ＤＥＬＩ）に関する。一構成では、ＤＥＬＩは、特定のアプリケーションコードフラグメントをキャッシュおよび実行するように構成されるコアと、コアのキャッシュ機能および実行機能にアクセスを提供するように構成されるアプリケーションプログラミングインタフェースと、コアの動作に関するポリシーを提供するように構成されるシステム制御および構成レイヤとを備える。

【００４１】本システムのいくつかの実施形態は、感染したプログラムを検出する方法を提供するものとして見ることができる。この点において、プログラムをウィルスチェックする方法は、以下のステップで要約することができる。すなわち、動的実行レイヤインタフェース（ＤＥＬＩ）を計算装置ハードウェアとプログラムコードの間に挿入するステップと、アプリケーションコードをＤＥＬＩに入る際に監視して、計算装置ハードウェアが以前にそのコードを処理したことがあるかどうかを判断するステップと、アプリケーションコードが以前に処理されていた場合、プログラムコードを解析して、感染したコードが存在するかどうかを判断するステップとである。

【００４２】以下の図面および詳細な説明から、ウィルスチェックソフトウェアに関連する他のシステム、方法、および特徴が当業者には明白になろう。本説明内に含まれるこのようなさらなるシステム、方法、および特徴はすべて、併記の特許請求の範囲によって保護されるウィルスチェックソフトウェアのシステムおよび方法の範囲内にあるものと意図される。

【００４３】ウィルスチェックソフトウェアのシステムおよび方法については、添付図面を参照してよりよく理解することができる。図面中の構成要素は必ずしも一定の縮尺ではなく、代わりにウィルス検出マネージャおよびＤＥＬＩを介しての動作の原理を明確に示すことに重点が置かれている。さらに、図面中、いくつかの図面を通して、同様の参照符号は対応する部品を示す。

【００４４】

【発明の実施の形態】ウィルスチェックおよびプログラムバイナリ実行についてのシステムおよび方法を開示する。好ましい実施形態によれば、ウィルス検出マネージャは、計算装置内に形成される仮想コンピュータと協働して、感染したアプリケーションコードを隔離および識別する。感染したコードが識別されると、ウィルス検出マネージャは、感染したコードが計算装置上で実行されないようにすることで、計算装置を保護する。以下に説明するように、ウィルス検出方法論の性質は、特定のアプリケーションに応じて変化しうる。例として、ウィルス検出は、特に、シグネチャマッチング、実行コード挙動監視を含むことができる。ＤＥＬＩは、各種ウィルス検出方法論を、計算装置ハードウェアによって初めて実行されるコードのみに適用できるようにする。ウィルス検出は、初めて実行しようとするコードに対してのみ連続して適用されるため、改良されたウィルスチェッカは、新しいかまたは変更されたプログラムのコンテンツ全体のチェックに関連するオーバーヘッドを低減しながら、感染したコードの識別および実行の阻止に対してより良好な体制を整えている。

【００４５】本発明のシステムおよび方法の説明を容易にするため、例示的なシステムについて図を参照して考察する。これらシステムについて詳細に説明するが、これは例示のみの目的のために提供されるものであり、本発明の概念から逸脱せずに様々な変更を行いうることが理解されよう。他の例示的なシステムが、２００１年８月８日付けで出願された「Dynamic Execution Layer Interface for Explicitly and Transparently Executing Application or System Binaries」と題する米国特許出願第０９／９２４，２６０号に記載されている。例示的なシステムの説明後、システムの動作例を提供して、プログラムバイナリを実行する前に、ウィルス検出を行うことができる様式を説明する。

【００４６】次に、図１を参照して、動的実行レイヤインタフェース（ＤＥＬＩ）１００の例を示す。概して、ＤＥＬＩ１００は、ハードウェアからアプリケーションバイナリコードをはずすために、オペレーティングシステム（Ｏ／Ｓ）を含むかまたは含まないアプリケーションと、ハードウェアとの間に常駐する高水準または低水準言語で書かれた汎用ソフトウェアレイヤを備える。この構成を通して、ＤＥＬＩ１００は、エミュレーション、動的変換および最適化、トランスペアレントリモートコード実行、仮想化されたハードウェア環境プログラムについてのコンピュータシステム機能性の再マッピング、コード伸張、コード解読等、広範な異なるアプリケーションで使用することのできる動的コンピュータプログラムコード変換、キャッシング、およびリンキングサービスを提供することができる。以下にさらに詳細に考

察するように、ＤＥＬＩ１００は、トランスペアレントモード、非トランスペアレントモード、またはこれら２つの組み合わせで動作している間にサービスを提供することができる。トランスペアレントモードにおいて、ＤＥＬＩ１００は、実行中のプログラムが、コンピュータハードウェア上で直接実行されていないことを知らないように実行中のプログラムの制御を自動的に行う。非トランスペアレントモードでは、ＤＥＬＩ１００は、アプリケーションプログラムインタフェース（ＡＰＩ）を通してサービスをアプリケーション（例えば、ウイルス検出マネージャ）にエクスポートして、どのようにＤＥＬＩ１００が特定のシステムイベントに対して反応するかをアプリケーションが制御できるようにする。

【００４７】図１に示すように、ＤＥＬＩ１００は、少なくとも１つのアプリケーション１０２とコンピュータハードウェア１０４の間に常駐する。特定の構成に応じて、アプリケーション１０２は、ＤＥＬＩの存在を知らない１つまたは複数のユーザアプリケーションおよび／またはＤＥＬＩ１００を知り、ＤＥＬＩのサービスを利用するように構成されるクライアント（たとえば、エミュレータ）を含むことができる。しかし、より一般的に、アプリケーション１０２は、コンピュータプロセッサによって実行される命令を含む任意のタイプのプログラムコードを含む。Ｏ／Ｓが使用される場合、ＤＥＬＩ１００は、提供されるサービスの性質に応じて、Ｏ／Ｓ（図示せず）の上位あるいは下位に存在し得る。たとえば、ＤＥＬＩ１００は、Ｏ／Ｓの上位で動作する場合、アプリケーションの実行を制御することしかできない。しかし、ＤＥＬＩ１００がＯ／Ｓの下位で動作する場合、Ｏ／ＳおよびアプリケーションＯ双方からのシステムおよびユーザコードが混ざったものを含むことができる命令ストリームへのアクセスを有する。ハードウェア１０４は、異なる様々なコンピュータシステムコンポーネントを含むことができるが、通常、少なくともコンピュータプロセッサを含む。

【００４８】ＤＥＬＩ１００は、コア１０６、アプリケーションプログラミングインタフェース（ＡＰＩ）１０８、トランスペアレントモードレイヤ１１０、およびシステム制御および構成レイヤ１１２を含む４つの主要なコンポーネントを含むことができる。概して、コア１０６は、２つの主要なサービスをＡＰＩ１０８およびトランスペアレントモードレイヤ１１０の双方にエクスポートする。これらサービスのうち第１のサービスは、ネイティブコードフラグメントまたはハードウェア１０４の命令セットに対応するコードフラグメントのキャッシングおよびリンキングに関連する。第２のサービスは、先にキャッシュされたコードフラグメントの実行に関する。ＡＰＩ１０８が、非トランスペアレント動作モードにおいて、コア１０６のキャッシングおよびリンキングサービスへのアクセスを提供するアプリケーション１０２から機能をエクスポートする。トランスペアレントモードレイヤ１１０は、コア１０６が、トランスペアレント動作モードにおいてコード実行に対する制御をトランスペアレントに獲得すると共に、キャッシュすべきコードフラグメントをフェッチできるようにする。最後に、システム制御および構成レイヤ１１２は、たとえば、コードのキャッシング、リンキング、および最適化についてのポリシーを含め、コア１０６の動作についてのポリシーを提供することにより、ＤＥＬＩ１００の構成を可能にする。たとえば、これらポリシーは、ＡＰＩ１０８を介してレイヤ１１２からアプリケーション１０２に提供することができる。システム制御および構成レイヤ１１２はまた、ＤＥＬＩ１００のトランスペアレントモードがイネーブルされるかどうかも制御し、したがって、コア１０６がＡＰＩ１０８からか、トランスペアレントモードレイヤ１１０からか、それともこれら双方から入力を受信するかどうかを判断する。

【００４９】図１にさらに示されるように、システム１００は、所望の場合、アプリケーションをハードウェア１０４上で直接実行することができるように、ＤＥＬＩ１００をバイパスするためにアプリケーション１０２が使用することのできるバイパスパス１１４を含むことができる。このような動作は、ＤＥＬＩ１００が、利用可能であるかもしれないし利用不可能であるかもしれないオプションの実行レイヤであるという点において、可能であり得ることに留意されたい。

【００５０】図１に示すように、コア１０６は、コアコントローラ１１６、キャッシュマネージャ１１８、フラグメントマネージャ１２０、および最適化マネージャ１２２を備える。コアコントローラ１１６は、タスクを遂行する責任を負うコア１０６のその他のコンポーネントにタスクを割り当てるディスパッチャとして機能する。キャッシュマネージャ１１８は、フラグメントマネージャ１２０および最適化マネージャ１２２と同様に、システム制御および構成レイヤ１１２によって指定されるポリシーに従って、１つまたは複数のコードキャッシュ１２４（たとえば、キャッシュ１～ｎ）内のコードフラグメントのキャッシングを制御するメカニズム（たとえば、アルゴリズムのセット）を含む。たとえば、コア１０６の１つまたは複数のコードキャッシュ１２４は、ハードウェア１０４の１つまたは複数のプロセッサ上のハードウエアコードキャッシュにあることもでき、また、ハードウェアのメインローカルメモリで作成することもできる。コードキャッシュ１２４がプロセッサに内蔵されたのハードキャッシュにマッピングされる場合、命令キャッシュリフィルオーバーヘッドの低減、メモリ帯域の増大等により、増大したパフォーマンスを得ることができる。フラグメントマネージャ１２０は、コードキャッシュ１２４内のコードフラグメントの構成およびフラグメントに課す変換のタイプを指定する。最後に、最適

化マネージャ１２２は、実行の最適化のためにコードフラグメントに適用することのできる最適化のセットを含む。

【００５１】上述したように、ＡＰＩ１０８は、ＤＥＬＩサービスへのアクセスを提供するアプリケーション１０２から機能をエクスポートする。より具体的には、ＡＰＩ１０８は、ＤＥＬＩの存在を知らないクライアントを含むアプリケーション１０２に、コア１０６のキャッシングおよびリンキングサービスをエクスポートする。ＡＰＩ１０８によってエクスポートされるこれらサービスは、アプリケーション１０２が、（ｉ）キャッシングのためにコードフラグメントをコア１０６に明示的に出すこと、かつ／または（ｉｉ）コードキャッシュ１２４から特定のコードフラグメントを実行するようにＤＥＬＩ１００に命令することにより、非トランスペアレントモードにあるＤＥＬＩ１００の動作を制御できるようにする。加えて、ＡＰＩ１０８はまた、ＤＥＬＩ１００の動作を初期化、停止する機能をエクスポートすることもできる。たとえば、ＡＰＩ１０８は、ＤＥＬＩ１００のトランスペアレント動作を開始し、さらに、ＤＥＬＩ１００がいつその動作を停止すべきかを指示することができる。ＡＰＩ１０８はまた、上述したように、アプリケーション１０２によって指定されるポリシーをコア１０６（たとえば、フラグメントマネージャ１２０および／または最適化マネージャ１２２）に送ることによって、ＤＥＬＩ１００の構成を容易にする。非トランスペアレントモードで操作を容易にするＡＰＩ１０８の使用については、図５に関連して後述する。

【００５２】図１をさらに参照して、トランスペアレントモードレイヤ１１０は、通常、動作中のアプリケーション１０２に対する制御をトランスペアレントに得るために使用されるインジェクタ１２６を含む。ＤＥＬＩ１００が完全なトランスペアレントモードで動作する場合（すなわち、アプリケーションがＤＥＬＩの存在を知らない場合）、インジェクタ１２６を使用して、アプリケーションをＤＥＬＩの制御下で動作することができるように、アプリケーションが実行を開始する前に、ＤＥＬＩをアプリケーション１０２に挿入する。このような状況では、ＤＥＬＩ１００は、例外処理を妨げないために、アプリケーション１０２の実行可能イメージを変更することを避ける。インジェクタ１２６は、バイナリがロードされる仮想アドレスを変更することなく、アプリケーションバイナリをそれぞれロードするいくつかの異なる方法で制御を得ることができる。例として、アプリケーションの実行可能イメージをロードするときに、ＤＥＬＩ１００（たとえば、共有ライブラリとしてコンパイルされる）がカーネルローダによって自動的にロードされるように、Ｏ／Ｓカーネルローダを変更することができる。代替として、ユーザレベルのローダを使用して、中断モードにおいてメモリにアプリケーション１０２をロードし、後にアプリケーションが再開されるときに、ＤＥＬＩ１００の共有ライブラリをロードする命令を、アプリケーション（たとえば、アプリケーションスタック上）に注入するように、カーネルローダを変更することなく利用することができる。

【００５３】別の代替では、ｐｔｒａｃｅを使用して、ＤＥＬＩ１００をアプリケーション１０２に添付できる。当分野では既知のように、ｐｔｒａｃｅは、デバッガが使用することの多い、１つのプロセスが別のプロセスを制御できるようにするメカニズムである。ＤＥＬＩ１００は、ｐｔｒａｃｅを介してアプリケーション１０２に添付され、アプリケーションのバイナリイメージの最上部における実行開始コード（たとえば、ｃｒｔ０）がアプリケーションのエントリポイントをまさに呼び出そうとする時点まで、アプリケーションを実行する別個のプロセスとして構成することができる。次に、アプリケーション１０２の実行を中断し、ＤＥＬＩ１００を使用してアプリケーション命令をフェッチし代わりに実行することができる。

【００５４】さらに別の代替では、アプリケーションのテキストセグメントを実行可能ファイルの別個のコピーで拡張することができる。特に、アプリケーションのバイナリイメージを一時的なロケーションにコピーし、アプリケーションのテキストセグメントを終わりにＤＥＬＩテキストセグメントを追加することによって拡張し、開始シンボル（すなわち、ｃｒｔ０によって呼び出されるエントリポイント）をＤＥＬＩエントリポイントに変更することができる。次に、ｅｘｅｃを使用して、結果得られる実行可能ファイルを実行することができる。元のアプリケーションテキストセグメントはなお、通常ロードされる同じ仮想アドレスにロードされるが、ＤＥＬＩ１００は、実際のアプリケーション１０２が開始する前に制御を得ることになる。

【００５５】別の例では、ＤＥＬＩ１００は、ｃｒｔ０の特別なバージョンを使用してアプリケーション１０２に対する制御を得ることができる。当分野において既知のように、ｃｒｔ０コードは、コマンドライン引数のピックアップ、初期スタックおよびデータセグメントのセットアップ、次に開始シンボル（通常、アプリケーション１０２のｍａｉｎ（　）関数）の値の呼び出し実行の責任を負う。アプリケーション１０２のエントリポイントを呼び出す前に、ｃｒｔ０は、ダイナミックリンクローダｄｌｄをマッピングし、次に、ダイナミックリンクローダｄｌｄが、アプリケーション１０２によって参照されるあらゆるダイナミックリンクライブラリ（ＤＬＬ）をロードする。ｃｒｔ０のカスタムバージョンを使用して、さらにＤＥＬＩコード（それ自体ＤＬＬとしてコンパイルされる）をマッピングし、開始シンボルによって定義されるエントリポイントの代わりにＤＥＬＩのエントリポイントを呼び出すことができる。

【００５６】アプリケーション１０２に対する制御を得る様式に関係なく、次に、命令フェッチコントローラ１２８を使用して、アプリケーションバイナリコードのフラグメント（たとえば、トレース）のコピーを抽出（すなわちフェッチ）し、キャッシングのためにＤＥＬＩコア１０６に渡し、適切なキャッシュされたコピーをコードキャッシュ１２４から実行するようコア１０６に命令することができる。このような動作を容易にするためのトランスペアレントモードレイヤ１１０の使用については、図５に関連して後述する。

【００５７】ＤＥＬＩ１００はトランスペアレントモードレイヤ１１０を含むものとして図示され本明細書において説明されるが、当業者は、この開示全体から、ＤＥＬＩ１００の操作がＡＰＩ１０８を通してアプリケーション１０２（たとえば、クライアント）によって単独に制御される場合、このレイヤを完全に省略し得ることに留意されたい。

【００５８】上述したように、システム制御および構成レイヤ１１２は、コードのキャッシングおよびリンキングについてのポリシーを提供することによってＤＥＬＩ１００の構成を可能にする。ＤＥＬＩ１００は、任意特定タイプのポリシーまたはポリシー内容に限定されないが、ポリシーは通常、ＤＥＬＩがどのように振る舞うかを判断する。たとえば、レイヤ１１２は、コードフラグメントをアプリケーション１０２からどのように抽出するか、フラグメントを元のコードからどのようにして作成するか、複数のコードフラグメントをどのように共にリンクしてより大きなコードフラグメントを形成することができるか等のポリシーを提供し得る。レイヤのポリシーは、静的であってもまた動的であってもよい。前者の場合、ポリシーをＤＥＬＩ１００にハードコードし、構築時に構成を固定することができる。後者の場合、ポリシーはＡＰＩ１０８における関数呼び出しを通してアプリケーション１０２によって動的に提供することができる。ポリシーの実施は、ＤＥＬＩ１００が特定のシステムおよび／またはハードウェアイベント（たとえば、例外および割り込み）に反応する様式を制御する。上述したポリシーに加えて、システム制御および構成レイヤ１１２は、コードキャッシュ１２４のサイズ、ログファイルが作成されるかどうか、コードフラグメントが最適化されるべきかどうか等を指定することができる。

【００５９】システム制御および構成レイヤ１１２は、システムおよびハードウェア機能性において指示されるアプリケーションバイナリコードにおいて命令をインターセプトすることにより、システムおよびハードウェア機能性の抽象化をさらにサポートする。次に、これら命令は、フラグメント形成プロセスの一部として、システム制御および構成レイヤ１１２の指示の下でフラグメントマネージャ１２０によって置換される。システム制御および構成レイヤ１１２は、見つからないかまたは故障したハードウェアにおいて指示される命令を識別し、フラグメントマネージャ１２０に、同様であるが別のハードウェア１０４において指示される対応する命令で、または元のハードウェアのソフトウェアシミュレーションでそれら命令を置換させる。

【００６０】上記コンポーネントに加えて、ＤＥＬＩ１００は、オプションとして、たとえば、感染したアプリケーションコードバイナリをある様式で識別する責任を負う１つまたは複数のアルゴリズムを含むウィルス検出マネージャ１３０をさらに備えることができる。たとえば、ウィルス検出マネージャ１３０は、アプリケーション１０２に由来するコードに適用可能なシグネチャチェッカを提供することができる。ウィルス検出マネージャ１３０については、図４の機能ブロック図を参照してさらに図示および説明する。ウィルス検出マネージャ１３０の使用については提供される場合、図７を参照して後述する。

【００６１】図２は、コア１０６の構成および動作の例を示す。この図に示すように、コア１０６は、２つのタイプの要求をＡＰＩ１０８またはトランスペアレントモードレイヤ１１０から受け入れる。まず、関数インタフェースを通してコードフラグメントをキャッシュしリンクする要求２００を受け入れることができる。このような要求は、たとえば、“ＤＥＬＩ＿ｅｍｉｔ＿ｆｒａｇｍｅｎｔ（ｔａｇ，ｆｒａｇｂｕｆ）”の形態の関数を含むことができる。この関数は、コードフラグメントをパラメータおよび識別子（たとえば、タグ）として受け取り、ＤＥＬＩキャッシュ１２４に格納する。加えて、コア１０６は、実行のためにハードウェア１０４に渡す、キャッシュ１２４に格納されているコードフラグメントを識別する、“ＤＥＬＩ＿ｅｘｅｃｕｔｅ＿ｆｒａｇｍｅｎｔ（ｔａｇ）”等関数インタフェースを通して特定のコードフラグメントタグにおいて実行を開始する要求を受け入れる。

【００６２】コアコントローラ１１６は、これら要求を処理し、適切なコアモジュールにディスパッチする。次に、所与の識別子を有するコードフラグメントを出す要求２０２をフラグメントマネージャ１２０に渡すことができる。フラグメントマネージャ１２０は、フラグメント形成ポリシー２０４に従ってコードフラグメントを変換し、おそらく、インスツルメンテーションポリシー（instrumentation policy）２０６に従ってコードフラグメントをインスツルメント（instrument）し、フラグメントリンキングポリシー２０８に従って先にキャッシュされたフラグメントと共にコードフラグメントをリンクする。たとえば、フラグメントマネージャ１２０は、あるコードフラグメントの実行が終了すると別のコードフラグメントに実行がジャンプし、それによってキャッシュからの実行の長さが増大するように、キャッシュ１２４中の複数のコードフラグメントをリンクすることが

できる。これを達成するために、フラグメントマネージャ１２０は、フラグメント割り当て命令２１０をキャッシュマネージャ１１８に発する。次に、フラグメントマネージャ１２０は、コードキャッシュ１２４中の処理されたコードフラグメントを割り当てる要求をキャッシュマネージャ１１８に送信する。

【００６３】キャッシュマネージャ１１８は、コードフラグメントの割り当てを制御し、通常、キャッシュ空間を管理するために、それ自体のキャッシュポリシー２１２を備える。しかし、フラグメントマネージャ１２０はまた、特定のフラグメント割り当て解除命令２１４もキャッシュマネージャ１１８に発することができる。たとえば、フラグメントマネージャ１２０は、現フラグメントを先に割り当てられたフラグメントと統合するかどうかを判断することができ、この場合、先のフラグメントの割り当てを解除する必要があり得る。構成によっては、キャッシュマネージャ１１８およびフラグメントマネージャ１２０が、２００１年５月２２日付けで出願された「A preemptive Replacement Strategy for a Caching Dynamic Translator Based on Changes in the Translation Rate」と題する米国特許第６，２３７，０６５号に図示し記載される様式で、コードキャッシュ１２４およびコードフラグメントを管理することができる。代替として、コードキャッシュ１２４およびコードフラグメントの管理を、２００１年１月５日付けで出願された「A Partitioned Code Cache Organization to Exploit Program Locality」と題する米国特許出願第０９／７５５，３８９号に図示し記載される様式で実行することも可能である。

【００６４】フラグメントをキャッシュマネージャ１１８に渡す前に、フラグメントマネージャ１２０は、フラグメントを最適化マネージャ１２２に渡して、最適化ポリシー２１８に従ってコードフラグメントの質を高めることができる。構成によっては、最適化マネージャ１２２は、２００１年１月５日付けで出願された「A FastRuntime Scheme for Removing Dead Code Across Linked Fragments」と題する米国特許出願第０９／７５５，３８１号に図示し記載される様式で、コードフラグメントを最適化することができる。代替として、最適化マネージャ１２２は、２００１年１月５日付けで出願された「A Memory Disambiguation Scheme for Partially Redundant Load Removal」と題する米国特許出願第０９／７５５，７７４号に図示し記載される様式で、コードフラグメントを最適化することができる。特に、最適化マネージャ１２２は、冗長計算をなくす、冗長メモリアクセスをなくす、関数をインライン化して手順呼び出し／リターンオーバヘッドをなくすなど、古典的なコンパイラ最適化技術を使用してコードフラグメントを最適化してもよい。

【００６５】上述したように、フラグメントマネージャ１２０は、フラグメント形成ポリシー２０４に従ってコードフラグメントを変換する。フラグメントマネージャ１２０によって行われる変換は、たとえば、相対アドレス、分岐アドレス等を変更することによるメモリアドレス参照の変更によるコードの再配置を含むことができる。コードフラグメントのレイアウトも変更し、機能性（すなわち、意味論（semantic））を変更することなくコードの物理的なレイアウトを変更することができる。これら変換は、フラグメントマネージャ１２０により、ＡＰＩ１０８を通して、命令フェッチコントローラ１２８から受信するフラグメントに対して行われる。

【００６６】コード計測を行うために、フラグメントマネージャ１２０は、コードフラグメントの実行頻度、メモリアドレスのアクセス頻度等についてのデータなど、コードプロファイリングについての計測ポリシー２０６に従ってデータを収集する。フラグメントの形成または割り当て解除を容易にするために、プログラムカウンタを使用して、これら統計を収集することができる。これらポリシーは、システム制御および構成レイヤ１１２によって構成され、ＡＰＩ１０８を通して送信されるか、あるいはＤＥＬＩ構築時に確立されるポリシー命令を受信する。ポリシーは、フラグメントを作成し、インスツルメント（instrument）し、最適化し、リンクする別の方法についてのオプションを包含してもよく、また、ポリシーは単にこれらタスク実行のために、ＤＥＬＩ１００にハードコードされたアルゴリズムであってもよい。

【００６７】ＤＥＬＩコア１０６が受け入れる第２のタイプの要求は、所与の識別子（たとえば、タグ）によって識別されるフラグメントを実行する要求２２０である。このような場合、コアコントローラ１１６は、ルックアップ要求２２２をフラグメントマネージャ１２０に発し、フラグメントマネージャ１２０は、フラグメントがキャッシュ１２４に現在常駐しアクティブである場合、対応するコードキャッシュアドレス２２４を戻す。例として、フラグメントマネージャ１２０は、タグを使用してコードフラグメントのロケーションを識別することができる、常駐しアクティブなコードフラグメントのルックアップテーブルを維持することができる。代替として、フラグメントマネージャ１２０またはキャッシュマネージャ１１８は、コードフラグメントが常駐しアクティブであるかどうかを追跡する他の任意の適した技術を使用することができる。フラグメントがキャッシュ１２４に現在存在しておらずアクティブではない場合、フラグメントマネージャ１２０はエラーコードをコアコントローラ１１６に戻し、コアコントローラ１１６は、フラグメントタグをキャッシュミスアドレスとして最初のリクエスタに戻す。一方、フラグメントが現在存在しアクティブである場合、コアコントローラ１１６は、最初の要求をキャッシュアドレスとともにキャッシュマネージャ１１８にパッチする。次に、キャッシュマネージャ

１１８は、制御をコードキャッシュ１２４中のアドレス指定されたコードフラグメントに渡し、ひいてはアドレス指定されたコードフラグメントを実行する。そして実行は、キャッシュミスが生じるまで、すなわち実行すべき次のアプリケーションアドレスのコピーがキャッシュに現在常駐しなくなるまで、コードキャッシュ１２４において焦点を合わせられたままである。この状況は、たとえば、実行中のコードをキャッシュコード１２４から逃すことによって検出することができる。キャッシュミスがキャッシュマネージャ１１８からコアマネージャ１１６に報告され、次に、最初のリクエスタに報告される。

【００６８】図３は、ＤＥＬＩ１００が実行可能なコンピュータシステム３００のアーキテクチャ例を示す概略図である。概して、コンピュータシステム３００は、デスクトップコンピュータ、ポータブルコンピュータ、専用サーバコンピュータ、マルチプロセッサコンピューティング装置、セルラ電話、個人情報端末（ＰＤＡ）、ハンドヘルドまたはペンベースコンピュータ等、広範な有線および／または無線のコンピューティング装置のいずれか１つを備えることができる。特定の構成に関係なく、コンピュータシステム３００は、たとえば、処理装置３０２、メモリ３０４、１つまたは複数のユーザインタフェース装置３０６、ディスプレイ３０８、１つまたは複数の入出力（Ｉ／Ｏ）装置３１０、および１つまたは複数のネットワーキング装置３１２を備えることができ、これらはそれぞれローカルインタフェース３１４に接続される。

【００６９】処理装置３０２は、コンピュータシステム３００に関連するいくつかのプロセッサから任意のオーダーメードの、または市販のプロセッサ、中央演算処理装置（ＣＰＵ）、または補助プロセッサ、半導体ベースのマイクロプロセッサ（マイクロチップの形態で）、マクロプロセッサ、１つまたは複数の特定用途向け集積回路（ＡＳＩＣ）、複数の適宜構成されたデジタルロジックゲート、および離散要素を個々におよび各種組み合わせで包含し、計算システム３００の全体的な動作を統制する他の周知の電気的構成を具備することができる。

【００７０】メモリ３０４は、揮発性メモリ要素（たとえば、ランダムアクセスメモリ（ＤＲＡＭ、ＳＲＡＭ等のＲＡＭ）および不揮発性メモリ要素（ＲＯＭ、ハードドライブ、テープ、ＣＤ－ＲＯＭ等）の組み合わせのうちのいずれか１つを具備することができる。メモリ３０４は通常、Ｏ／Ｓ３１６、１つまたは複数のアプリケーション１０２（たとえば、ユーザアプリケーションおよび／またはクライアント）、およびすでに詳細に述べたＤＥＬＩ１００を備える。当業者は、メモリ３０４が、簡明にする目的のために省略された他のコンポーネントを備えることができ、通常は備えることを理解しよう。

【００７１】１つまたは複数のユーザインタフェース装置３０６は、ユーザがコンピュータシステム３００と対話できるようにするコンポーネントを備える。たとえば、コンピュータシステム３００がパーソナルコンピュータ（ＰＣ）を備える場合、これらコンポーネントはキーボードおよびマウスを備えることができる。コンピュータシステム３００がハンドヘルド装置（たとえば、ＰＤＡ、携帯電話）を含む場合、これらコンポーネントは、機能キーまたはボタン、タッチセンシティブスクリーン、スタイラス等を備えることができる。ディスプレイ３０８は、ＰＣの場合にはコンピュータモニタもしくはプラズマスクリーンを備え、またはハンドヘルド装置の場合には液晶ディスプレイ（ＬＣＤ）を備えることができる。

【００７２】図３をさらに参照すると、１つまたは複数のＩ／Ｏ装置３１０は、コンピュータシステム３００の別のシステムおよび／または装置への接続を容易にするように適合されるため、１つまたは複数のシリアル、パラレル、ＳＣＳＩ(small computer system interface)、ユニバーサルシリアルバス（ＵＳＢ）、ＩＥＥＥ１３９４（たとえば、Ｆｉｒｅｗｉｒｅ（登録商標））、および／またはパーソナルエリアネットワーク（ＰＡＮ）コンポーネントを具備しうる。ネットワークインタフェース装置３１２は、ネットワークを介してデータを送信かつ／または受信するために使用される各種コンポーネントを備える。例として、ネットワークインタフェース装置３１２は、入力および出力の双方を通信可能にする装置、たとえば、変調器／復調器（たとえば、モデム）、無線（たとえば、無線周波数（ＲＦ））トランシーバ、電話インタフェース、ブリッジ、ルータ、ネットワークカード等を具備する。

【００７３】各種ソフトウェアおよび／またはファームウェアを本明細書において述べた。このソフトウェアおよび／またはファームウェアは、任意のコンピュータ関連システムまたは方法が使用するまたはこれと併用する任意のコンピュータ読み取り可能媒体に格納可能なことを理解されたい。本文書の文脈では、コンピュータ読み取り可能媒体とは、コンピュータ関連システムまたは方法が使用するまたはこれと併用するコンピュータプログラムを含むまたは格納することのできる電子、磁気、光学、または他の物理的な装置または手段を指す。これらプログラムは、コンピュータベースのシステム、プロセッサ包含システム、または命令実行システム、装置、またはデバイスから命令をフェッチし、その命令を実行することのできる他のシステム等、命令実行システム、装置、またはデバイスが使用するまたはこれと併用する任意のコンピュータ読み取り可能媒体で具現することができる。本文書の文脈において、「コンピュータ読み取り可能媒体」は、命令実行システム、装置、またはデバイスが使用するまたはこれと併用するプログラムを格納、通信、伝播、または搬送することのできるあらゆる手段

であることができる。

【００７４】コンピュータ読み取り可能媒体は、たとえば、電子、磁気、光学、電磁、赤外線、または半導体システム、装置、デバイス、または伝播媒体であることができるが、これらに限定されない。コンピュータ読み取り可能媒体のより具体的な例（非網羅的なリスト）としては、１本または複数本のワイヤを有する電気接続、ポータブルコンピュータディスケット、ランダムアクセスメモリ（ＲＡＭ）、読み取り専用メモリ（ＲＯＭ）、消去可能プログラム可能読み取り専用メモリ（ＥＰＲＯＭ）、電気的消去可能プログラム可能読み取り専用メモリ（ＥＥＰＲＯＭ）、またはフラッシュメモリ、光ファイバ、およびポータブルコンパクトディスク読み取り専用メモリ（ＣＤ－ＲＯＭ）が挙げられる。コンピュータ読み出し可能媒体はさらに、たとえば紙または他の媒体を光学走査することを介して、プログラムを電子的に取り込み、次いでコンパイル、解釈、あるいは必要であれば別様に適宜処理してから、コンピュータメモリに格納することができるので、プログラムが印刷される紙または別の適した媒体であってもよいことに留意する。

【００７５】次に、図４の機能ブロック図を参照する。図４に示すようなウィルス検出マネージャ１３０は、コンピュータシステム３００のメモリ３０４において実現することができ、コントローラ４００、コードシグネチャチェッカ４１０、コードシグネチャデータベース４１５、およびコード挙動チェッカ４２０を備えることができる。図４に示すように、コントローラ４００は、新しいフラグメントが出される都度、キャッシュ１２４に出そうとされるコードフラグメントのコンテンツのインジケーションを受信する。このようにして、ウィルス検出マネージャ１３０は、コードの実行される部分のみをチェックするため、感染コードについてソフトウェアをテストする効率が大幅に増大する。

【００７６】その後、コントローラ４００は、コードキャッシュに出すべきこのようなコードフラグメントのコンテンツを、１つまたは複数のウィルス検出エンジン（たとえば、特に、コードシグネチャチェッカ４１０、コード挙動チェッカ４２０（図示せず））に転送することができる。たとえば、ウィルス検出マネージャ１３０は、ウィルスに関連する既知のシグネチャをコードシグネチャデータベース４１５に格納し、キャッシュ１２４のコンテンツを格納されているシグネチャと比較することにより、単純な既知のウィルスについてスキャンすることができる。ウィルス検出マネージャ１３０のコードシグネチャチェッカ４１０が、格納されている既知のウィルスのシグネチャとマッチするシグネチャを識別する場合、コントローラ４００は、１つまたは複数のコマンドをＤＥＬＩ１００に送信して、コードキャッシュへの感染フラグメントの転送を阻止し、したがって、コンピュータハードウェア１０４上で実行されないようにする。

【００７７】上述したウィルスシグネチャ比較の代わりに、またはこれに加えて、ウィルス検出マネージャ１３０は、他のテスト方法も適用することができる。たとえば、コード挙動チェッカ４２０は、コードフラグメントが既知のウィルスと共通する実行可能ステートメントを含むかどうかを判断するために、ヒューリスティックベースのルールをコードフラグメントのコンテンツに適用するように構成されうる。コード挙動チェッカ４２０は、１つまたは複数のチェックを適用して、既知の突然変異エンジンを識別することもできる。さらに、コード挙動チェッカ４２０は、複数の既知のウィルスに固有のルールセットを適用することができる。

【００７８】さらに、コード挙動チェッカ４２０は、一般的なウィルス攻撃を示すプログラムアクティビティを捜すようにプログラムすることもできる。たとえば、コード挙動チェッカ４２０は、スタックオーバーラン、セキュリティおよびオペレーティングシステムの保護をバイパスするように設計される既知のバックドア法を監視し警告するようにプログラムすることができる。重要なことに、ウィルス検出マネージャ４００は、コンピュータシステム３００による実行に向けて設計されるすべてのプログラムをインターセプトおよび処理することができる。このようにして、コンピュータシステム３００のユーザが気付かないことがあるワームおよび他のバックグラウンドアプリケーションを、感染コードの存在についてスクリーニングすることができる。

【００７９】ウィルス検出マネージャ４００はＤＥＬＩ１００と共に、感染コードをインターセプトするよう有利に位置決めされた構成を提示する。コンピュータシステム３００が、暗号化されたウィルスを含むアプリケーション１０２に遭遇すると、ＤＥＬＩ１００は、上述したように、キャッシュ１２４に出そうとしている新しく解読された実行可能コードフラグメントをロードし、ウィルス検出マネージャ４００がこれを解析する。ウィルス検出マネージャ４００が、実行可能コードフラグメントが感染していないことを確認した後でのみ、ＤＥＬＩ１００はコードフラグメントをキャッシュ１２４に出す。

【００８０】ウィルス検出マネージャ４００は、解読された実行可能コードにのみ対処するため、かつ、ウィルス検出マネージャ４００は、コンピュータシステム３００上で特定バージョンのプログラムが最初に実行されるときにのみコードの各実行可能部分をチェック（すなわち、テスト）するため、ウィルスチェックソフトウェアのシステムおよび方法は、感染コードのインターセプトにおいて特に効果的であると共に効率的である。ＤＥＬＩ１００を使用し、本明細書に提示するウィルス検出方法論を実施するコンピュータシステム３００は、トリガイベント（たとえば、１つのキーストロークまたは一連

のキーストローク）が実行されると、または特定の日付または時間を過ぎた後に、解読されるようトリガされる感染コードから特に良好に保護される。

【００８１】ウイルス検出マネジャ４００とＤＥＬＩ１００の大まかな性質について上述したが、次に、ＤＥＬＩ１００の動作例について図５ないし図７を参照して考察する。上述したように、ＤＥＬＩ１００は、２つの一般的な動作モード、すなわちトランスペアレントモードおよび非トランスペアレントモード、ならびにこれらの組み合わせで動作する。これらモードでの動作を説明するに当たり、流れ図を提供する。これら流れ図中の任意のプロセスステップまたはブロックは、プロセスにおける特定の論理機能またはステップを実施するための１つまたは複数の実行可能命令を含むモジュール、セグメント、またはコードの部分を表すことを理解されたい。特定のプロセスステップ例について説明するが、代替の実施も実現可能なことが理解されよう。さらに、ステップは、関わる機能性に応じて、略同時または逆順を含め、図示または説明する順序から外れて実行してもよい。

【００８２】概して、ＤＥＬＩ１００がアプリケーション１０２の実行に対する制御をトランスペアレントに、それとも非トランスペアレントに獲得したかに関係なく、アプリケーションはハードウェア１０４上で直接実行されない。むしろ、アプリケーションコードは、たとえば、コードキャッシュ１２４に維持しうるコードフラグメントの形態で、ＤＥＬＩ１００を通して実行される。

【００８３】図５は、トランスペアレントモードでのＤＥＬＩの単純なＤＥＬＩの動作例を示す。より具体的には、図５は、アプリケーション１０２がＤＥＬＩの存在を知らない完全なトランスペアレントモードでのＤＥＬＩの動作を示す。ブロック５００で開始し、ＤＥＬＩ１００がまず開始される。トランスペアレントモードで動作する場合、この開始は、アプリケーション１０２の開始の結果生じうる。開始されると、ブロック５０２に示すように、ＤＥＬＩがアプリケーションおよびその実行に対する制御を得るように、ＤＥＬＩ１００がトランスペアレントモードレイヤ１１０のインジェクタ１２６を使用してアプリケーション１０２に挿入される。上述したように、この制御を得ることのできる別の様々な方法がある。

【００８４】ＤＥＬＩ１００がアプリケーション１０２に対する制御を持つようになると、ＤＥＬＩを使用して、上述したサービス等いくつかの異なるサービスのいずれか１つを提供することができる。たとえば、ＤＥＬＩ１００は、ハードウェアおよび／またはソフトウェアエミュレーション、動的変換および最適化、トランスペアレントリモートコード実行、仮想化されたハードウェア環境プログラムについてのコンピュータシステム機能性の再マッピング、コード伸張、コード解読、ウィルス検知等を容易にすることができる。これら異なるサービスはそれぞれ、コードキャッシュ１２４内のプログラムコードフラグメントのキャッシングおよびリンキングを含む。アプリケーションバイナリからコピーされた特定のコードフラグメントをキャッシュし、ある様式で変換することにより、後にコードキャッシュ１２４から変換されたコードを実行することで、所望のサービスを提供することができる。

【００８５】コードをキャッシュする前に、ＤＥＬＩ１００は、いずれの特定コードフラグメントをキャッシュするかを判断しなければならない。その際、完全なトランスペアレントモードで動作中の場合、アプリケーション１０２はＤＥＬＩ１００に気づかず、ＤＥＬＩ１００は、どのコードフラグメントをキャッシュするかについてアプリケーション１０２から指導を受けない。コードのキャッシングは、ＤＥＬＩ構築時に作成されるポリシーを通して指示することができるが、より好ましくは、ＤＥＬＩ１００は、少なくとも部分的に、自分でこういった判断を行う能力を有する。ＤＥＬＩ１００は、ブロック５０４に示すように、アプリケーション１０２によるコードの実行を監視することによってこれを行うことができる。そうすることで、ＤＥＬＩ１００は、たとえば、どのフラグメントが最も頻繁に使用されるかを判断することにより、たとえば、どのコードフラグメントがアプリケーション１０２に最も有用であるかについての情報を収集することができる。

【００８６】ＤＥＬＩ１００の制御下でアプリケーション１０２が様々なコードフラグメントを実行するため、ＤＥＬＩ１００は、実行される各コードの断片を「見る」。したがって、監視プロセスを通して、ＤＥＬＩ１００は、どのコードフラグメントが最も頻繁に使用されるかを判断することができる。次に、ＤＥＬＩ１００は、システム制御および構成レイヤ１１２によって提供されるポリシーを参照して、どのコードの部分が「ホット」であるか、すなわちアプリケーションの実行に最も重要であるかを判断することができる。上述したように、この判断は、実行インスタンスを追跡するプログラムカウンタを使用して行うことができる。当業者は、どのコードの部分がホットであるかを判断するために、他の様々な方法を使用しうることを理解しよう。この判断を行うことができる方法の例は、１９９８年１１月５日付けで出願された「Method for Selecting Active Code Tracesfor Translation in a Caching Dynamic Translator」と題する米国特許出願第０９／１８６，９４５号、および１９９９年５月１４日付けで出願された「LowOverhead Speculative Selection of Hot Traces in a Caching Dynamic Translator」と題する米国特許出願第０９／３１２，２９６号に記載されている。

【００８７】図５をさらに参照して、各コードフラグメントが実行されると、判断要素５０６に示すように、D

ＥＬＩ１００は、関連するコードフラグメントが先にキャッシュされたかどうかを判断することができる。先にキャッシュされていた場合、ＤＥＬＩ１００は、キャッシュされた（かつ通常通りに変換された）コードを含むコードキャッシュ１２４にジャンプし、ブロック５０８に示すように、ネイティブアプリケーションコードの代わりにこのコードがハードウェア１０４によって実行される。コードがキャッシュされているかどうかの判断は、上述したように、ネイティブアプリケーションコードと、コードキャッシュ１２４にキャッシュされた類似物との間の関連を識別する識別子（たとえば、タグ）を参照して行うことができる。次に、キャッシュされていないコードへの参照（すなわちキャッシュミス）に遭遇するときまで、コードキャッシュ１２４に常駐するリンクされたコードフラグメントの実行を含む、キャッシュされたコードの実行が続く。判断要素５１０を参照して、キャッシュされていないコードへの参照に遭遇した場合、ＤＥＬＩ１００はアプリケーションコードまでジャンプして戻り、ブロック５１２に示すように、そのコードの実行が再開される。このとき、ＤＥＬＩ１００はこの実行の監視を再開することができる（ブロック５０４）。

【００８８】判断要素５０６に戻り、ＤＥＬＩ１００が、関連するコードフラグメントがコードキャッシュ１２４にないと判断する場合、流れは判断要素５１４に続き、所定のポリシーを参照してコードフラグメントがホットかどうかを判断する。コードがホットではない場合、流れはブロック５０４に戻り、アプリケーションコードの実行の監視が続けられる。一方、コードがホットである場合、ブロック５１６に示すように、トランスペアレントモードレイヤ１１０の命令フェッチコントローラ１２８を使用してフラグメントをフェッチすることにより、コードフラグメントがコピーされる。このような所定のポリシー５１４の極端な例として、全てのフラグメントがホットであることが考えられるが、このような場合、コードフラグメントは、コードキャッシュ１２４へと放出される前に一度だけ監視される（ブロック５１４）。

【００８９】この時点において、ブロック５１８に示すように、コードフラグメントをある様式で変換することができる。加えて、コードリンクに関して確立されているポリシーに従って、キャッシュ１２４内のコードフラグメントをリンクすることができる。コード変換の性質は、ＤＥＬＩ１００が提供すべきサービスのタイプに依存する。たとえば、ＤＥＬＩ１００が単にアプリケーションの実行を最適化するだけの場合、この変換は、より良好なパフォーマンスのためにコードを再配置し、かつ／または再構成することを含むことができる。提供される変換の性質に関係なく、コードの構造は、土台をなす意味論（semantic）を変更することのないように変更される。コードフラグメントが変換されると、ブロック５２０に示すように、変換されたコードをコードキャッシュ１２４にキャッシュすることができ、流れは上述したブロック５０８に続き、ＤＥＬＩ１００内で実行される。

【００９０】上述したように、ＤＥＬＩ１００は、非トランスペアレントモードでは別様に動作する。概して、非トランスペアレントモードで動作する場合、ＤＥＬＩ１００は、たとえば、アプリケーション１０２がアクセス可能なＡＰＩ１０８における機能をエクスポートする、ＤＬＬまたは静的にリンクされたモジュールとして動作しうる。非トランスペアレントモードでは、アプリケーション１０２（すなわち、クライアント）はＤＥＬＩ１００を知り、ＤＥＬＩのサービスを利用するように構成される。最も単純な場合、クライアント（たとえば、エミュレータ）は、ＡＰＩ１０８を通してＤＥＬＩ動作のあらゆる側面を制御する。このような場合、ＤＥＬＩ１００を利用して、ＡＰＩ１０８を介してクライアントによって提供される明示的な命令に従って、コードをキャッシュ、リンク、かつ最適化することができる。たとえば、アプリケーション実行中に最も頻繁に使用される特定のコードフラグメントをキャッシュするようＤＥＬＩ１００に命令することができる。これは、たとえば、タグを識別することでＤＥＬＩ１００にコードのロケーションを提供することによって達成することができる。このような場合、ＤＥＬＩ１００は、ＡＰＩ１０８に命令されるように、コードフラグメントを監視せず代わりに記録する。加えて、クライアントは、ＤＥＬＩ１００に適用可能なタグを識別することにより、キャッシュされたフラグメントのどれをＡＰＩ１０８を介して実行するかについて、ＤＥＬＩ１００に指示することができる。

【００９１】しかし、最も典型的なシナリオでは、クライアントが、ＤＥＬＩ１００にサービスをトランスペアレントで提供するよう求める。このような場合、クライアントはＤＥＬＩ１００の動作を呼び出すとともに、ＤＥＬＩが動作をいつ停止すべきかについての命令を提供する。図６は、この様式での動作の例を提供する。

【００９２】ブロック６００から始まり、ＤＥＬＩ１００が開始される。非トランスペアレントモードでは、この開始は、ＤＥＬＩ１００を呼び出す“ＤＥＬＩ＿ＳＴＡＲＴ”等の開始コマンドがＡＰＩ１０８によって送られるときに発生する。開始されると、ブロック６０２に示すようにトランスペアレントモードレイヤ１１０を起動することができ、図５に関連して上述したトランスペアレントモードと同様に流れが続くことができる。したがって、ＤＥＬＩ１００は、アプリケーション６０４によるコードの実行を監視し、コードフラグメントが以前にキャッシュされたかどうかを判断し（６０６）、該当する場合にはキャッシュされたコードを実行し（６０

８）、「ホット」コードフラグメントをコピーし（６１６）、コピーされたコードフラグメントを変換し（６１８）、変換されたコードフラグメントをキャッシュする（６２０）。ＤＥＬＩ１００の動作は、ＤＥＬＩ１００が、クライアントが要求するように、ＤＥＬＩ１００に動作を停止させる条件に遭遇するまでこのようにして続く。例として、この割り込みは、ＡＰＩ１０８を介してＤＥＬＩ１００に送信される関数呼び出しの終わりに配置される“ＤＥＬＩ＿ＳＴＯＰ”コマンドを使用して行うことができる。

【００９３】上述したように、ＤＥＬＩ１００を使用して、プログラムバイナリのウィルスチェックを促進することができる。たとえば、ＤＥＬＩ１００を使用して、仮想コンピュータにおいて圧縮プログラムコードを伸張および実行することができる。上述したように、たとえば、特に、ウィルスチェッカがアプリケーションプログラムに対して定期的にのみ実行される、ウィルスチェッカがプログラムのように挙動するウィルスを検出するのに十分長く実行することができない（仮想マシンにおいて）、かつウィルスチェッカが各種トリガ条件を適用しないなどを含む、現在のウィルスチェック方法論に伴ういくつかの問題がある。このため、ウィルスチェッカは、プログラムをチェックしない、またはウィルス内の解読エンジンをトリガしないことにより、感染したプログラム全体を見逃しうる。しかし、これら問題は、ＤＥＬＩ１００が上述したウィルス検出マネージャ１３０と組み合わせて使用される場合に回避することができる。ＤＥＬＩ１００は、コードフラグメントさらには個々の命令等のコードの非常に小さな部分を制御するため、ＤＥＬＩ１００は、感染コードを露呈させるのによく適している。

【００９４】アプリケーションバイナリをインターセプトおよび実行する際のＤＥＬＩ１００の動作の一例を図７に示す。本例では、ウィルス検出サービスは、トランスペアレント動作モードで提供される。図７に示すように、感染コード７００を識別する方法は、ステップ７０２のクエリで開始することができ、実行可能コードフラグメントがキャッシュ１２４（図１）に転送するよう指示されているかどうかが判断される。ステップ７０２のクエリから出る否定応答矢印で示すように、キャッシュ１２４が実行可能コードフラグメントを受信しようとしない場合、ウィルス検出マネージャ１３０は、ステップ７０４に示すように待機するようプログラムすることができる。当業者は、ＡＰＩ１０８を介してウィルス検出マネージャ１３０のコントローラ４００と内部で通信して、コードフラグメントをキャッシュ１２４に転送するよう指示されているときをコントローラ４００に通知するようにＤＥＬＩ１００を構成してもよいことを理解しよう。代替として、ステップ７０２のクエリに戻る前に、指示された時間量だけ待つか、またはある他のイベントを待つようにウィルス検出マネージャ１３０をプログラムしてもよい。

【００９５】ステップ７０２のクエリから出る肯定応答矢印で示されるように、実行可能コードフラグメントがキャッシュ１２４に転送するように指示されていることをＤＥＬＩ１００が示した場合、図７の例示的な分岐に示すように、１つまたは複数のウィルス検出方法を実行するように、ウィルス検出マネージャ１３０のコントローラ４００をプログラムすることができる。図７のフローチャートが、感染コードを検出する２つの例示的な方法を示すことに留意することは重要である。コードをコンピュータハードウェア１０４（図１）に転送する前に実行可能アプリケーションコードをスキャンするために、適宜構成されたウィルス検出マネージャ１３０が、上述した他の方法、ならびにＤＥＬＩ１００を利用するよう考案される方法も適用しうる。

【００９６】図７のフローチャートの最も左側の処理分岐に示す第１の方法では、コンピュータシステム３００のウィルス検出マネージャ１３０は、キャッシュすべき（図５のブロック５０４を参照）かまたは既知のウィルスシグネチャについてまだホットではない場合（図５のブロック５１６の前）に実行すべきコードのシグネチャチェックを行うようにプログラムすることができる。所定のポリシー５１４（図５）および６１４（図６）に応じて、シグネチャチェックは、コードが初めて実行される（すべてのフラグメントがホットとみなされる場合）かまたは少数回実行される（フラグメントがホットになる前）ときだけに行われることになる。図７のフローチャートの最も右側の処理分岐に示す第２の方法では、コンピュータシステム３００のウィルス検出マネージャ１３０は、ＤＥＬＩ１００にキャッシュすべきコードの動作を解析して、コードフラグメントが既知のウィルスと一致するように挙動するかどうかを判断するようにプログラムすることができる。最も右側の処理分岐におけるコードフラグメント挙動方法は、ＤＥＬＩ１００からキャッシュ１２４へのコードフラグメントの放出に加えて、特定のイベントの発生と結び付けることができる。たとえば、特に、システム呼び出し、スタックオーバーランをＤＥＬＩ１００により、あるいはコードフラグメントをインスツルメント（instrument）して所望の検出を得る（たとえば、関数呼び出しから戻ったときにスタックポインタをチェックする）ことにより、生成することができる。

【００９７】ステップ７０６のクエリで示すように、ウィルス検出マネージャ１３０は、シグネチャチェックを継続すべきかどうかをチェックするようにプログラムすることができる。ウィルス検出マネージャ１３０は、コードフラグメントがウィルスを含むかどうかの判断を行う際に適用される、１つまたは複数の関連する方法論を定義するように構成しうるフラグまたは他のインジケー

タの外部セットを読み出すように構成してもよいことが理解されよう。ステップ７０６のクエリへの応答が肯定である（すなわち、ウィルス検出マネージャが、既知のウィルスシグネチャセットとのコードフラグメントのシグネチャ比較を実行すべきである）場合、ウィルス検出マネージャ１３０は、カウンタがセットされるステップ７０８に進むようにプログラムすることができる。そうではなく、シグネチャ比較チェックが望ましくない場合、ステップ７０６のクエリから出る否定応答矢印で示すように、ウィルス検出マネージャ１３０は、どのウィルス検出方法が望ましいかを決定するように構成することができる。

【００９８】次に、ステップ７１０に示すように、ウィルス検出マネージャ１３０は、ウィルスに存在する既知のシグネチャを含むウィルスシグネチャデータベース４１５（図４）から次のシグネチャを読み出すようにプログラムすることができる。ステップ７１０においてシグネチャを読み出した後、ウィルス検出マネージャ１３０は、コードフラグメントのコンテンツをシグネチャと比較することができる。ステップ７１２のクエリからの肯定応答で示すように、コードフラグメントが、現在のシグネチャとマッチするバイトを含む場合、ステップ７１４に示すように、ウィルス検出マネージャ１３０はウィルスを識別し、ウィルス検出プロセスを終了することができる。当業者は、ＤＥＬＩ１００に命令するウィルス検出マネージャ１３０から制御信号をＤＥＬＩ１００に供給して、コードフラグメントをクリアし、アプリケーション１０２（図１）を終了することができることを理解しよう。ステップ７１２のクエリから出る否定応答矢印で示すように、コードフラグメントが、現在のシグネチャにマッチするバイトセットを含まない場合、ステップ７１６のクエリに示すように、ウィルス検出マネージャ１３０は、さらなるシグネチャが利用可能かどうかをチェックするようにプログラムすることができる。さらなるシグネチャが利用可能ではない場合、感染コードを識別する方法７００は、図７に示すように終了することができる。その他の場合、ステップ７０８においてセットされたカウンタをステップ７１８において示すように増分することができ、すべての既知のかつ格納されているウィルスシグネチャがコードフラグメントと比較されるまで、ステップ７１０～７１８を繰り返すことができる。感染コードを識別する方法７００のシグネチャ比較分岐が完了すると、フラグをセットしてテストが完了したことを示すことができることが理解されよう。第２のフラグをセットして、コードフラグメントが既知のウィルスシグネチャを持たないことを示すこともできる。

【００９９】その後、かつ／またはシグネチャ比較分岐（すなわち、フローチャートの最も左側の分岐）と略同時に、ウィルス検出マネージャ１３０は、コードフラグメントの動作を監視することにより、１つまたは複数の挙動テストを行うようにプログラムすることができる。ステップ７２０のクエリに示すように、ウィルス検出マネージャ１３０は、コード挙動チェックを継続すべきかどうかをチェックするようにプログラムすることができる。ステップ７２０のクエリに対する応答が肯定である（すなわち、ウィルス検出マネージャがコードフラグメントに対して１つまたは複数のコード挙動テストを行うべきである）場合、ウィルス検出マネージャ１３０は、カウンタがセットされるステップ７２２に進むようにプログラムすることができる。そうではなく、挙動チェックが望ましくない場合、ステップ７２０のクエリから出る否定応答矢印に示すように、ウィルス検出マネージャ１３０は、どのウィルス検出方法が望ましいかを決定するように構成することができる。

【０１００】次に、ステップ７２４に示すように、ウィルス検出マネージャ１３０は、既知のウィルス、ならびに／またはウィルスおよび他の感染コードが一般に使用する既知の攻撃モードを隔離し識別するように構成される、先に格納されたコードの挙動テストを実行するようにプログラムすることができる。上述のように１つまたは複数のイベントの検出器と結び付けることができる、現コードの挙動テストによって示されるように、コードフラグメントに対して各種演算を実行した後、ウィルス検出マネージャ１３０は、ステップ７２６のクエリに示すように、コードフラグメントが１つまたは複数の既知のウィルスのように動作しているかどうか判断を下すことができる。ステップ７２６のクエリからの肯定応答で示すように、コードフラグメントが１つまたは複数の既知のウィルスのように挙動する場合、ウィルス検出マネージャ１３０は、ステップ７２８に示すようにウィルスを識別することができ、ウィルス検出プロセスを終了することができる。

【０１０１】シグネチャ比較ウィルス検出モードを参照して上述したように、ＤＥＬＩ１００に命令するウィルス検出マネージャ１３０から制御信号をＤＥＬＩ１００に供給して、コードフラグメントをクリアし、アプリケーション１０２（図１）を終了することができる。ステップ７２６のクエリから出る否定フロー制御矢印に示すように、コードフラグメントが現在の挙動と一致する挙動を示さない場合、ステップ７３０のクエリを実行する前に、ＤＥＬＩ１００は、コードフラグメントを初期状態（図示せず）に回復するようにプログラムすることができる。さらなる挙動テストが実行されない場合、感染コードを識別する方法７００は、図７に示すように終了することができる。その他の場合、ステップ７２２においてセットされたカウンタをステップ７３２に示すように増分することができ、すべての指示されたコード挙動テストが完了し結果が解析されるまで、ステップ７２４～７３２を繰り返すことができる。感染コードを識別する方法７００のシグネチャ挙動テスト分岐が完了する

と、フラグをセットしてテストが完了したことを示すことができることが理解されよう。第２のフラグをセットして、コードフラグメントが既知のウィルス挙動を持たないことを示すこともできる。

【０１０２】図７の流れ図を参照して図示および説明した感染コードを識別する方法７００は、シグネチャ比較およびコード挙動ウィルス検出方法論の並列処理を反映したものであるが、ウィルスチェックソフトウェアのシステムは、仮想計算資源（すなわち、メモリおよびレジスタ）が限られている場合、これらおよび他のテストを直列に適用するように構成してもよいことを理解されたい。さらに、例示目的のために上記説明および図面において本発明の特定の実施形態を詳細に開示したが、併記の特許請求の範囲に記載される本発明の範囲から逸脱することなく、変形および変更を行いうることが当業者には理解されよう。たとえば、ＤＥＬＩ１００は、ウィルス検出サービスのみを提供するものとして上述したが、ＤＥＬＩ１００が他の様々なサービスを同時に提供可能なことに留意されたい。たとえば、ＤＥＬＩ１００によって提供されるウィルス検出サービスは、たとえば、クライアントによって促進されるハードウェアまたはソフトウェアエミュレーションを含め、他のタスクを実行する場合に利用することができる。本開示は、このような用途を包含するものと意図される。また、この発明は、例として次のような実施形態を含む。

【０１０３】（１）　感染したプログラム命令を識別する方法であって、動的実行レイヤインタフェース（ＤＥＬＩ）（１００）を計算装置ハードウェア（１０４）と前記プログラム命令（１０２）の間に挿入するステップと、前記プログラム命令（１０２）が前記ＤＥＬＩ（１００）に入る際に前記プログラム命令（１０２）を監視して、コードが前記計算装置ハードウェア（１０４）によって以前に処理されたことがあるかどうかを判断し、前記プログラム命令（１０２）が以前に処理されたことがない場合には、前記プログラム命令（１０２）を解析して、前記プログラム命令（１０２）が感染しているかどうかを判断するステップとを含む方法。

【０１０４】（２）　前記プログラム命令を解析する前記ステップは、解読したプログラム命令（１０２）をウィルス検出マネージャに挿入することを含む、（１）記載の方法。

【０１０５】（３）　前記プログラム命令を解析する前記ステップは、仮想計算装置におけるコードフラグメントのコンテンツに対して複数のテスト（７１２、７２４）を適用することを含む、（２）記載の方法。

【０１０６】（４）　リリースされたプログラム命令をコンピュータハードウェア（１０４）において処理するステップをさらに含む、（１）記載の方法。

【０１０７】（５）　アクティブなソフトウェアアプリケーションにおいて感染したプログラム命令を検出するシステムであって、プログラム命令をインターセプトする手段（１１０）と、計算装置が前記インターセプトされたプログラム命令を以前に処理したことがあるかどうかを判断する手段（１２８）と、前記インターセプトされたプログラム命令をコンピュータハードウェア（１０４）に転送する前に、前記計算装置が以前に処理したことのない前記インターセプトされたプログラム命令を解析する手段（１３０）とを備えるシステム。

【０１０８】（６）　前記インターセプトする手段は、動的実行レイヤインタフェース（ＤＥＬＩ）（１００）を備える、（５）記載のシステム。

【０１０９】（７）　前記インターセプトされたプログラム命令を解析する手段は、ウィルス検出マネージャ（１３０）を備える、（５）記載のシステム。

【０１１０】（８）　前記ウィルス検出マネージャ（１３０）は、複数のウィルス検出テスト（４１０、４２０）を前記インターセプトされたプログラム命令（１０２）のコンテンツに対して適用するように構成されるコントローラ（４００）を備える、（７）記載のシステム。

【０１１１】（９）　コンピュータ読み取り可能媒体に格納されるウィルス検出プログラムであって、プログラム命令をインターセプトするように構成されるロジック（１１０）と、計算装置が前記インターセプトされたプログラム命令を以前に処理したことがないかどうかを判断するように構成されるロジック（１２８）と、前記計算装置（１０４）が以前に処理したことのない前記インターセプトされたプログラム命令がウィルスに感染しているかどうかを判断するように構成されるロジック（１３０）とを含むプログラム。

【０１１２】（１０）　インターセプトされた非感染プログラム命令（１０２）を前記計算装置（１０４）に転送するように構成されるロジック（１１６）をさらに含む、（９）記載のプログラム。

【図面の簡単な説明】

【図１】動的変換サービスをアプリケーションおよびオペレーティングシステムに提供するために、コンピュータシステム上で実行される動的実行レイヤインタフェース（ＤＥＬＩ）の例を示すブロック図である。

【図２】図１に示すＤＥＬＩのコアの例示的な構成および動作のブロック図である。

【図３】図１に示すＤＥＬＩを実行可能な、例示的なコンピュータシステムのブロック図である。

【図４】図１に紹介されるウィルス検出マネージャの機能ブロック図である。

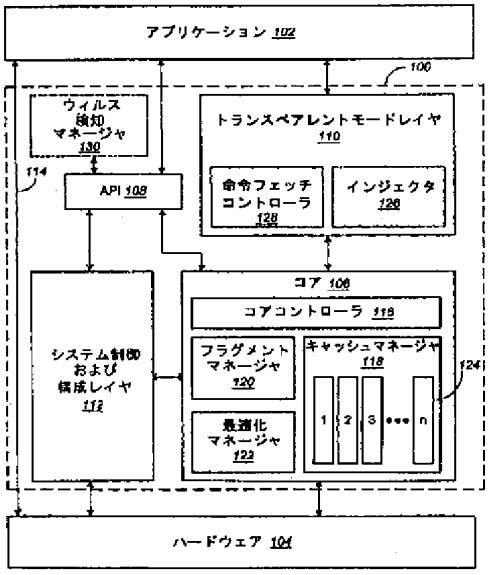
【図５】トランスペアレントモードで動作する図１に示すＤＥＬＩの例を示す流れ図である。

【図６】非トランスペアレントモードで動作する図１に示すＤＥＬＩの例を示す流れ図である。

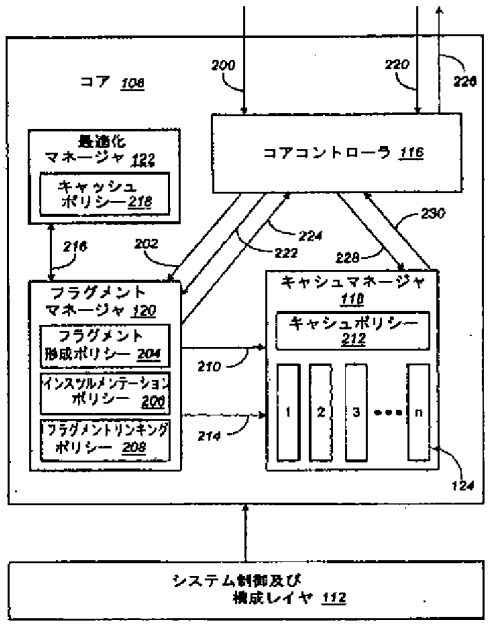
【図７】図４のウィルス検出マネージャの動作を示す流

れ図である。

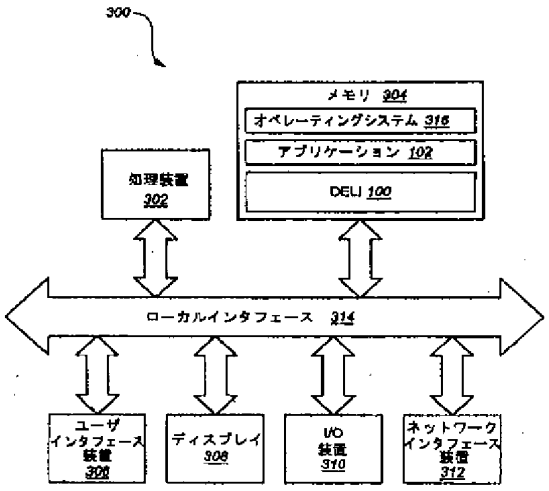
【図１】



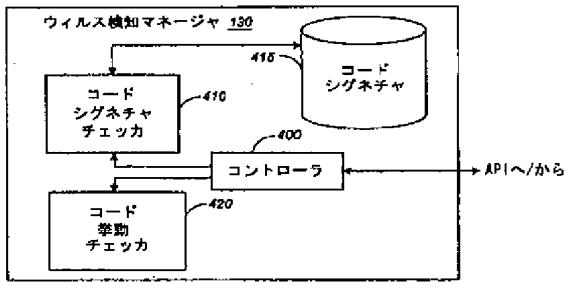
【図２】



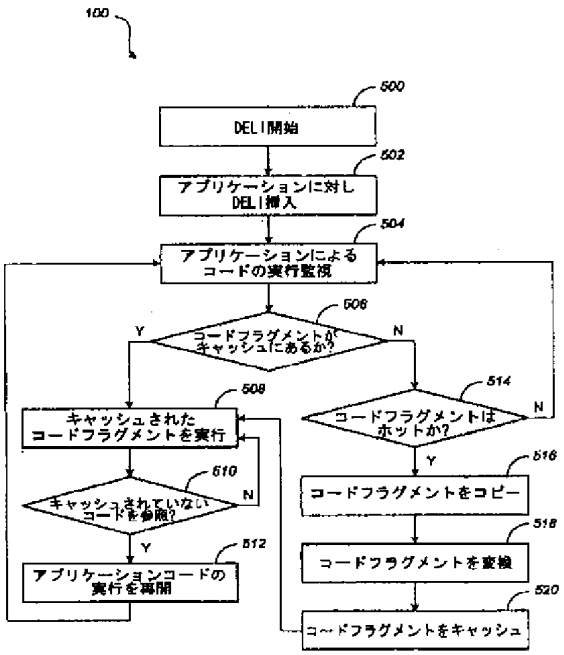
【図３】



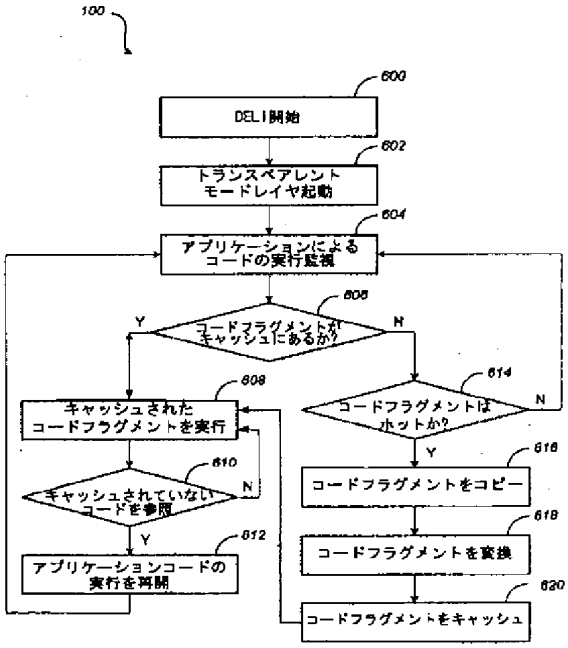
【図４】



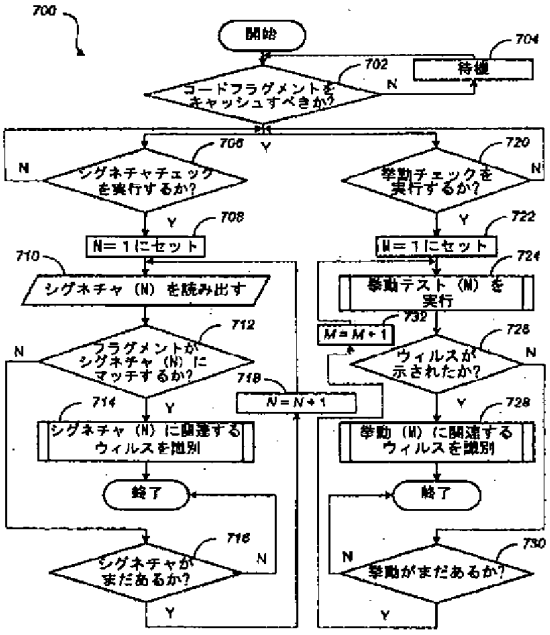
【図５】


100
500
DELI開始
502
アプリケーションに対し DELI挿入
504
アプリケーションによる コードの実行監視
506
コードフラグメントが キャッシュにあるか?
Y
N
508
キャッシュされた コードフラグメントを実行
510
キャッシュされていない コードを参照?
N
Y
512
アプリケーションコードの 実行を再開
514
コードフラグメントは ホットか?
N
Y
516
コードフラグメントをコピー
518
コードフラグメントを変換
520
コードフラグメントをキャッシュ

【図６】


100
600
DELI開始
602
トランスペアレント モードレイヤ起動
604
アプリケーションによる コードの実行監視
606
コードフラグメントが キャッシュにあるか?
Y
N
608
キャッシュされた コードフラグメントを実行
610
キャッシュされていない コードを参照
N
Y
612
アプリケーションコードの 実行を再開
614
コードフラグメントは ホットか?
N
Y
616
コードフラグメントをコピー
618
コードフラグメントを変換
620
コードフラグメントをキャッシュ

【図７】


700
開始
702
コードフラグメントを キャッシュすべきか?
N
Y
704
待機
706
シグネチャチェック を実行するか?
N
Y
708
N＝１にセット
710
シグネチャ (N) を読み出す
712
フラグメントが シグネチャ (N) に マッチするか?
N
Y
714
シグネチャ (N) に関連する ウィルスを識別
716
シグネチャが まだあるか?
N
Y
718
N＝N＋１
720
挙動チェックを 実行するか?
N
Y
722
M＝１にセット
724
挙動テスト (M) を 実行
726
ウィルスが 示されたか?
N
Y
728
挙動 (M) に関連する ウィルスを識別
730
挙動がまだあるか?
N
Y
732
M＝M＋１
終了
終了

フロントページの続き

(72)発明者　ジュゼッペ・デゾリ
アメリカ合衆国02472マサチューセッツ州
ウォータータウン、デズモンド・ストリート　27

Ｆターム(参考)　5B076 FD08

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平3－233629

⑤1Int. Cl.⁵	識別記号	庁内整理番号
G 06 F 9/06	450 Z	7927－5B
11/10	310 B	9072－5B

⑬公開　平成3年(1991)10月17日

審査請求　未請求　請求項の数 2　（全7頁）

⑤4発明の名称　実行形式ファイルの被破壊チェック方式

⑳特　願　平2－29902

㉒出　願　平2(1990)2月9日

㉒発明者　矢津　義之　東京都港区芝5丁目33番1号　日本電気株式会社内

㉑出願人　日本電気株式会社　東京都港区芝5丁目7番1号

㉔代理人　弁理士　境　廣巳

明　細　書

1.発明の名称

実行形式ファイルの被破壊チェック方式

2.特許請求の範囲

（1）各実行形式ファイルのファイル名とファイル長とファイルのバイト列のチェックサムとの対応表から構成されるファイルチェックデータベースと、

前記ファイルチェックデータベースからデータを読み出すファイルチェックデータベース読み出し機構と、

ファイルシステムから全ての実行形式ファイルを逐次取り出す実行形式ファイル取り出し機構と、

取り出した実行形式ファイルが前記ファイルチェックデータベースに存在するか否かを調べ、存在しなければ存在しない旨の警告メッセージを出力するファイル存在チェック機構と、

取り出した実行形式ファイルの実際のファイル長が前記ファイルチェックデータベースに登録された同ファイルのファイル長と一致しているか否かを調べ、一致していなければ一致していない旨の警告メッセージを出力するファイル長チェック機構と、

取り出した実行形式ファイルの実際のバイト列のチェックサムが前記ファイルチェックデータベースに登録された同ファイルのバイト列のチェックサムと一致しているか否かを調べ、一致していなければ一致していない旨の警告メッセージを出力するファイルチェックサムチェック機構と、

前記ファイル存在チェック機構、ファイル長チェック機構およびファイルチェックサムチェック機構の実行を任意に選択可能としたチェック方式選択機構とを備えたことを特徴とする実行形式ファイルの被破壊チェック方式。

（2）各警告メッセージを画面表示装置に出力することを特徴とした請求項1記載の実行形式ファイルの被破壊チェック方式。

3.発明の詳細な説明

〔産業上の利用分野〕

本発明はコンピュータシステムにおける実行形

式ファイルの被破壊チェック方式に関するものであり、より詳しくは、俗にコンピュータウィルスと呼ばれる不正なプログラムのコンピュータシステムへの侵入および実行形式ファイルの破壊を検査する方式に関するものである。

〔従来の技術〕

コンピュータウィルスとは、自己複写機能およびシステム損傷機能を持つプログラムのことである。

ここで、自己複写機能とは、それ自身のプログラム（コンピュータウィルスプログラム）を他の適当な実行形式ファイル（プログラム）に複写または挿入する機能のことである。コンピュータウィルスプログラムを複写または挿入された実行形式ファイルは、コンピュータウィルスに感染したと言い、次にこの実行形式ファイルを実行すると、また別の適当な実行形式ファイルを選び、コンピュータウィルスプログラムを複写または挿入する。そして、これらがコンピュータシステム内の各所で繰り返されて行くことになる。これをコンピュータウィルスの増殖という。また、コンピュータウィルスプログラムがコンピュータシステムへ侵入したことを、コンピュータシステムがコンピュータウィルスに感染したと言う。

一方、システム損傷機能とは、磁気ディスク装置のフォーマットのようなコンピュータシステムに復旧不能な損害を与える処理を行う機能のことであり、特定の日時や、感染してからの時間や、増殖回数等の契機によって生ずる。このようになった状態を、コンピュータシステムが発病したと言う。

従来、上述したコンピュータウィルスについては次のような対策が採られていた。

①ネットワークからの感染を防ぐために、外部のコンピュータシステムからのログインユーザは、なるべく一般ユーザとして扱い、特権ユーザとは区別し、プログラムの書き換えやインストールを行わせない。

②フロッピーディスク等の外部記憶装置からの感染を防ぐために、その発行元を明確にし、怪しいものはインストールを行わないようにする。

③特権ユーザは、正当性が十分信頼できる実行形式ファイルに関してのみ実行するように注意し、コンピュータウィルスの感染する機会を最小限に止め、またシステム破壊等の復元不可能な事態の発生（発病）が起こらないように努める。

④実行時間が普段よりも遅くなったといった兆候によってコンピュータウィルスに感染した疑いのある実行形式ファイルに対しては、既に正しいことが検証されている実行形式ファイルと比較し、書き換えられていないこと、すなわちコンピュータウィルスに感染していないことを確認する。

〔発明が解決しようとする課題〕

上述したように、従来は、コンピュータウィルスの感染そのものを防ぐことが主に考えられており、コンピュータシステムが既に感染しているか否かを判断する方法については、感染の疑いのある実行形式ファイルについて、個々にその正当性をチェックするという方法しかなく、顕著な兆候が現れずに潜伏しているコンピュータウィルスを早期に発見することはできず、よって、コンピュータシステムの発病を十分に防止することができないという欠点があった。

本発明は上記の点に鑑み提案されたものであり、その目的とするところは、ファイルシステムに存在する全ての実行形式ファイルについて容易にコンピュータウィルスの感染をチェックして被疑ファイルを特定することができ、コンピュータシステムの発病を有効に防止することのできる実行形式ファイルの被破壊チェック方式を提供することにある。

〔課題を解決するための手段〕

本発明の実行形式ファイルの被破壊チェック方式では、コンピュータシステムへのコンピュータウィルスの感染のタイプと、実行形式ファイルへのコンピュータウィルスの感染のタイプとをそれぞれ分析した結果に基づき、コンピュータウィル

スが感染した実行形式ファイル（被疑ファイル）を統一的に検出することを可能にしている。

すなわち、コンピュータシステムへのコンピュータウィルスの感染には、以下の2種類のタイプがある。

1つ目は、コンピュータシステムのファイルシステムに元来存在する実行形式ファイルの感染である。

2つ目は、既にコンピュータウィルスに感染した実行形式ファイルの新規インストールである。これは、かかる実行形式ファイルをファイルシステム上の適当な場所にインストールしておき、特権モードの利用者がこれを実行するのを待つというものである。これを実行したユーザがたまたま特権モードであったときに、コンピュータウィルスの増殖または発病を引き起こす。

一方、実行形式ファイルへのコンピュータウィルスの感染には、以下の2種類のタイプがある。

1つ目は、コンピュータウィルスプログラムが実行形式ファイルに上書きされてしまったものである。このように侵されたものは、本来その実行形式ファイルの持っている機能は失われ、これを実行しても期待した動作はせず、その代わりに発病を引き起こす。

2つ目は、本来の機能を損傷しない状態で、実行形式ファイルにコンピュータウィルスプログラムが付け加えられたものである。この実行形式ファイルを実行すると、コンピュータウィルスの増殖を引き起こすが、本来の機能も正しく動作するため、実行速度が普段より遅くなるといったコンピュータウィルスプログラムの実行に伴った副次的な兆候によってのみ判断されるため、ユーザは感染していることに気付きにくい。そして、増殖回数がある一定の数値に達するなどの契機により発病を引き起こす。

しかして、本発明の実行形式ファイルの被破壊チェック方式では、コンピュータシステムへのコンピュータウィルスの感染の2タイプのうち、2つ目の、既にコンピュータウィルスに感染した実行形式ファイルの新規インストールについては、ファイルシステムに存在する全ての実行形式ファイルについて、ファイルシステムに存在することが確認されている実行形式ファイルであるか否かをファイル名から判断することにより、被疑ファイルを検出するようにしている。すなわち、今までになかった実行形式ファイルがインストールされていることにより、その感染を知ることができる。なお、1つ目の、コンピュータシステムのファイルシステムに元来存在する実行形式ファイルの感染については、ファイル名としては正当であるため、この方式では検出できず、その判断は次に委ねる。

次に、実行形式ファイルへのコンピュータウィルスの感染の2タイプのうち、2つ目の、実行形式ファイルにコンピュータウィルスプログラムが付け加えられたものについては、ファイル長を以前の値と比較することにより被疑ファイルを検出するようにしている。すなわち、このタイプのものは、ファイルのサイズが元のものより大きくなることから、その感染を知ることができる。

また、1つ目の、コンピュータウィルスプログラムが実行形式ファイルに上書きされてしまったものについては、ファイル長の変化では感染を検出することはできないが、ファイル内容の相違を反映するファイルチェックサムを以前の値と比較することにより被疑ファイルを検出するようにしている。すなわち、このタイプのものは、ファイルの内容をチェックすることによってのみ、その感染を知ることができる。

これらの手法により、あらゆるタイプの感染について検出が可能である。

本発明では、上記の手法を実現するため、各実行形式ファイルのファイル名とファイル長とファイルのバイト列のチェックサムとの対応表から構成されるファイルチェックデータベースと、

前記ファイルチェックデータベースからデータを読み出すファイルチェックデータベース読み出し機構と、

ファイルシステムから全ての実行形式ファイルを逐次取り出す実行形式ファイル取り出し機構と、

取り出した実行形式ファイルが前記ファイルチェックデータベースに存在するか否かを調べ、存在しなければ存在しない旨の警告メッセージを出力するファイル存在チェック機構と、

取り出した実行形式ファイルの実際のファイル長が前記ファイルチェックデータベースに登録された同ファイルのファイル長と一致しているか否かを調べ、一致していなければ一致していない旨の警告メッセージを出力するファイル長チェック機構と、

取り出した実行形式ファイルの実際のバイト列のチェックサムが前記ファイルチェックデータベースに登録された同ファイルのバイト列のチェックサムと一致しているか否かを調べ、一致していなければ一致していない旨の警告メッセージを出力するファイルチェックサムチェック機構と、

前記ファイル存在チェック機構、ファイル長チェック機構およびファイルチェックサムチェック機構の実行を任意に選択可能としたチェック方式選択機構とを備えるようにしている。

〔作用〕

本発明の実行形式ファイルの被破壊チェック方式にあっては、各実行形式ファイルのファイル名とファイル長とファイルのバイト列のチェックサムとの対応表から構成されるファイルチェックデータベースを予め構築しておくことにより、チェック実行の指示に応じて、ファイルチェックデータベース読み出し機構が前記ファイルチェックデータベースからデータを読み出し、実行形式ファイル取り出し機構がファイルシステムから全ての実行形式ファイルを逐次取り出し、取り出した実行形式ファイルが前記ファイルチェックデータベースに存在するか否かをファイル存在チェック機構が調べて存在しなければ存在しない旨の警告メッセージを出力し、取り出した実行形式ファイルの実際のファイル長が前記ファイルチェックデータベースに登録された同ファイルのファイル長と一致しているか否かをファイル長チェック機構が調べて一致していなければ一致していない旨の警告メッセージを出力し、取り出した実行形式ファイルの実際のバイト列のチェックサムが前記ファイルチェックデータベースに登録された同ファイルのバイト列のチェックサムと一致しているか否かをファイルチェックサムチェック機構が調べて一致していなければ一致していない旨の警告メッセージを出力する。また、チェック方式選択機構が前記ファイル存在チェック機構、ファイル長チェック機構およびファイルチェックサムチェック機構の実行を任意に選択可能とする。

〔実施例〕

以下、本発明の実施例につき図面を参照して説明する。

第1図は本発明の実行形式ファイルの被破壊チェック方式の一実施例を示す構成図である。第1図において、本実施例は、ファイルシステム1とファイルチェックデータベース2とチェック方式選択機構3とファイルチェックデータベース読み出し機構4と実行形式ファイル取り出し機構5とファイル存在チェック機構6とファイル長チェック機構7とファイルチェックサムチェック機構8と画面表示装置9とから構成されている。なお、矢印を付した太線は制御の流れを示し、細線はデータの流れを示し、破線は選択が行われることを示している。

各部の機能は次の通りである。

ファイルシステム1；コンピュータシステムのファイルシステムであり、実行形式ファイルが格納されている。

ファイルチェックデータベース2；ファイルシステム1に正当に格納されている実行形式ファイルのファイル名と、そのファイル長と、そのファイルのバイト列のチェックサムとを一つのエントリとするデータベースである。これは、コンピュータシステムを最初にインストールしたときに作成しておく。そのコンピュータシステムの管理者がエディタで登録してもよいし、別途ツールを作成して登録してもよい。

チェック方式選択機構3；チェック方式を選択し、実行する機構を確定するものである。チェック方式としては、ファイル存在チェックとファイ

ル長チェックとファイルチェックサムチェックとがある。なお、どのチェックを行うかは、起動時に引数として与えられる。

ファイルチェックデータベース読み出し機構4：ファイルチェックデータベース2からデータを読み出すものである。

実行形式ファイル取り出し機構5：ファイルシステム1から全ての実行形式ファイルを逐次取り出すものである。

ファイル存在チェック機構6：実行形式ファイル取り出し機構5がファイルシステム1から取り出した実行形式ファイルのファイル名がファイルチェックデータベース2から読み出したデータ中に存在するか否かを調べ、存在しなければ存在しない旨の警告メッセージを画面表示装置9に出力するものである。なお、チェック方式選択機構3により当該機構が選択された場合にだけ動作する。

ファイル長チェック機構7：実行形式ファイル取り出し機構5がファイルシステム1から取り出した実行形式ファイルの実際のファイル長がファイルチェックデータベース2に登録された同ファイルのファイル長と一致しているか否かを調べ、一致していなければ一致していない旨の警告メッセージを画面表示装置9に出力するものである。なお、チェック方式選択機構3により当該機構が選択された場合にだけ動作する。

ファイルチェックサムチェック機構8：実行形式ファイル取り出し機構5がファイルシステム1から取り出した実行形式ファイルの実際のバイト列のチェックサムがファイルチェックデータベース2に登録された同ファイルのバイト列のチェックサムと一致しているか否かを調べ、一致していなければ一致していない旨の警告メッセージを画面表示装置9に出力するものである。なお、チェック方式選択機構3により当該機構が選択された場合にだけ動作する。

画面表示装置9：警告メッセージをコンピュータシステムの管理者等に対して表示するものである。

第2図は、上記の実施例の処理を示すフローチャートであり、以下、第1図および第2図を参照して実施例の動作を説明する。

先ず、動作が開始されると（ステップＳ１）、オペレータ等の指示に基づき、チェック方式選択機構3によりチェック方式の特定が行われる（ステップＳ２）。具体的には、起動時に与えられるプログラムの引数に応じて、３つのチェック方式（ファイル存在チェック，ファイル長チェック，ファイルチェックサムチェック）のそれぞれに対応するビットを持ったファイルチェックフラグのビットを設定する。ビットの値がその方式を採用するか採用しないかを示す。

次いで、ファイルチェックデータベース読み出し機構4により、ファイルチェックデータベース2からデータがメモリ（図示せず）上に読み出される（ステップＳ３）。

次いで、実行形式ファイル取り出し機構5により、ファイルシステム1から順次に実行形式ファイルが取り出される（ステップＳ４）。そして、実行形式ファイルが存在しないか否かが判断され（ステップＳ５）、存在しない場合（ステップＳ５のＹＥＳ）は全ての実行形式ファイルについて処理が終了したものとして処理を終了する（ステップＳ６）。存在する場合は次の処理に移行する。

次いで、ファイル存在チェック機構6では、先にチェック方式選択機構3により設定されたファイルチェックフラグのうちファイル存在チェックに対応する値が調べられ（ステップＳ７）、「ＯＮ」の場合（ステップＳ７のＯＮ）は、メモリに読み出したファイルチェックデータベース2のデータ中に、取り出した実行形式ファイルと同じファイル名が存在するか否かが調べられ（ステップＳ８）、存在しない場合（ステップＳ８のＮＯ）には、存在しない旨の警告メッセージが画面表示装置9に出力される（ステップＳ９）。その後、ステップＳ４に戻る。

ファイルチェックフラグのうちファイル存在チェックに対応する値が「ＯＦＦ」の場合（ステップＳ７のＯＦＦ）およびファイルチェックデータベース2に同じファイル名が存在する場合（ステ

ップS8のYES）には、次の処理に移行する。

次いで、ファイル長チェック機構7では、ファイルチェックフラグのうちファイル長チェックに対応する値が調べられ（ステップS10）、「ON」の場合（ステップS10のON）は、ファイルチェックデータベース2に登録された同ファイルのファイル長が、取り出した実行形式ファイルの実際のファイル長と一致しているか否かが調べられ（ステップS11）、一致していない場合（ステップS11のNO）は、一致していない旨の警告メッセージが画面表示装置9に出力される（ステップS12）。その後、ステップS4に戻る。

ファイルチェックフラグのうちファイル長チェックに対応する値が「OFF」の場合（ステップS10のOFF）およびファイルチェックデータベース2の同ファイルとファイル長が一致する場合（ステップS11のYES）には、次の処理に移行する。

次いで、ファイルチェックサムチェック機構8では、ファイルチェックフラグのうちファイルチェックサムチェックに対応する値が調べられ（ステップS13）、「ON」の場合（ステップS13のON）は、ファイルチェックデータベース2に登録された同ファイルのファイルチェックサムが、取り出した実行形式ファイルの実際のバイト列のチェックサムと一致しているか否かが調べられ（ステップS14）、一致していない場合（ステップS14のNO）は、一致していない旨の警告メッセージを画面表示装置9に出力する（ステップS15）。その後、ステップS4に戻る。

また、ファイルチェックフラグのうちファイルチェックサムチェックに対応する値が「OFF」の場合（ステップS13のOFF）およびファイルチェックデータベース2の同ファイルとファイルチェックサムが一致する場合（ステップS14のYES）にも同様にステップS4に戻る。

これらの処理が全ての実行形式ファイルについて行われる。

警告メッセージで特定された実行形式ファイルについては、コンピュータウィルスの感染の可能性が高いので、個別にチェックした上で適当な処置がとられる。

〔発明の効果〕

以上説明したように、本発明の実行形式ファイルの被破壊チェック方式にあっては、コンピュータウィルスの感染の性質に着目して統一的かつ網羅的にチェックを行うため、コンピュータウィルスに感染しているか否かの判定および被疑ファイルの特定が遅滞なく行え、コンピュータウィルスによるコンピュータシステムの発病を有効に防止できる効果がある。

4.図面の簡単な説明

第1図は本発明の実行形式ファイルの被破壊チェック方式の一実施例を示す構成図および、

第2図は第1図の実施例の処理のフローチャートである。

図において、

1……ファイルシステム

2……ファイルチェックデータベース

3……チェック方式選択機構

4……ファイルチェックデータベース読み出し機構

5……実行形式ファイル取り出し機構

6……ファイル存在チェック機構

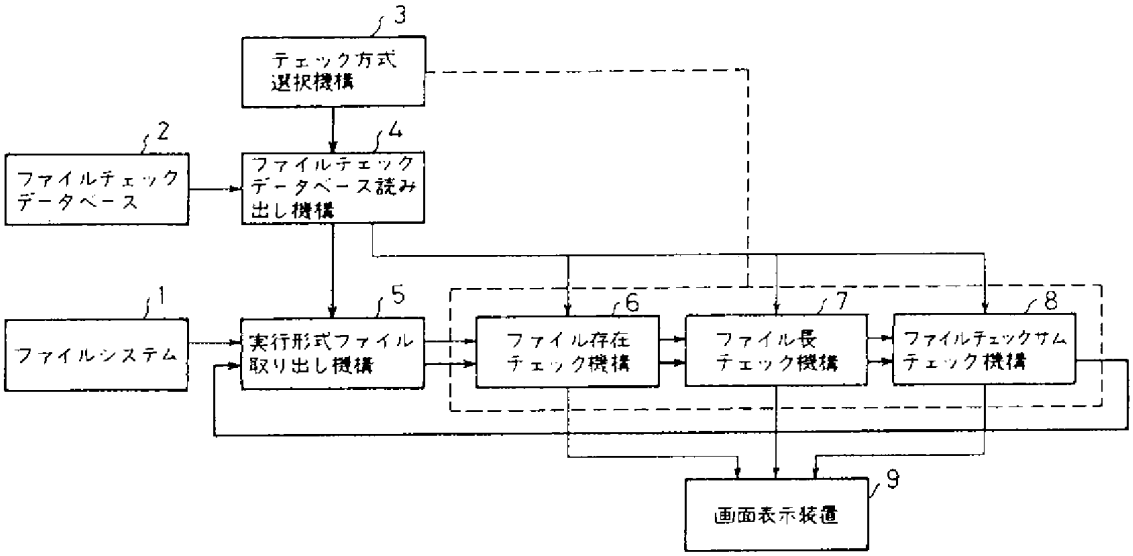
7……ファイル長チェック機構

8……ファイルチェックサムチェック機構

9……画面表示装置

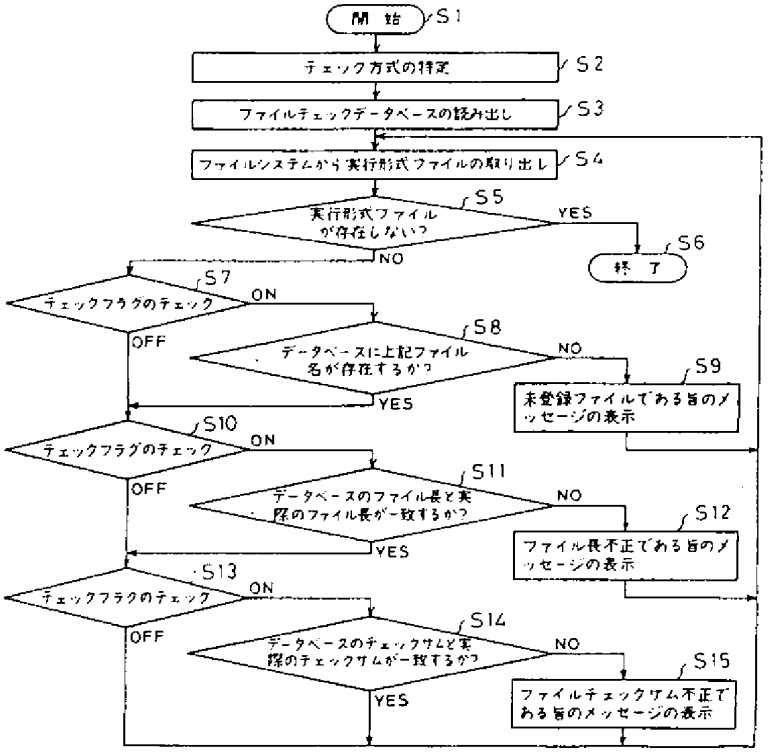
特許出願人　日本電気株式会社

代理人　弁理士　境　廣巳



実施例の構成図

第 1 図



処理のフローチャート

第 2 図

(19)日本国特許庁（ＪＰ）　(12) **公表特許公報**（Ａ）　(11)特許出願公表番号

特表平9－504395

(43)公表日　平成９年(1997)４月28日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		
Ｇ０６Ｆ	9/06	５５０	9367－5B	Ｇ０６Ｆ	9/06	５５０Ｚ
	12/00	５３７	7623－5B		12/00	５３７Ｚ
	12/14	３１０	7623－5B		12/14	３１０Ｚ

審査請求　未請求　　予備審査請求　有　　（全 20 頁）

(21)出願番号　特願平7－512495
(86)(22)出願日　平成６年(1994)10月28日
(85)翻訳文提出日　平成８年(1996)４月30日
(86)国際出願番号　ＰＣＴ／ＧＢ９４／０２３７８
(87)国際公開番号　ＷＯ９５／１２１６２
(87)国際公開日　平成７年(1995)５月４日
(31)優先権主張番号　９３２２２９２．５
(32)優先日　1993年10月29日
(33)優先権主張国　イギリス（ＧＢ）

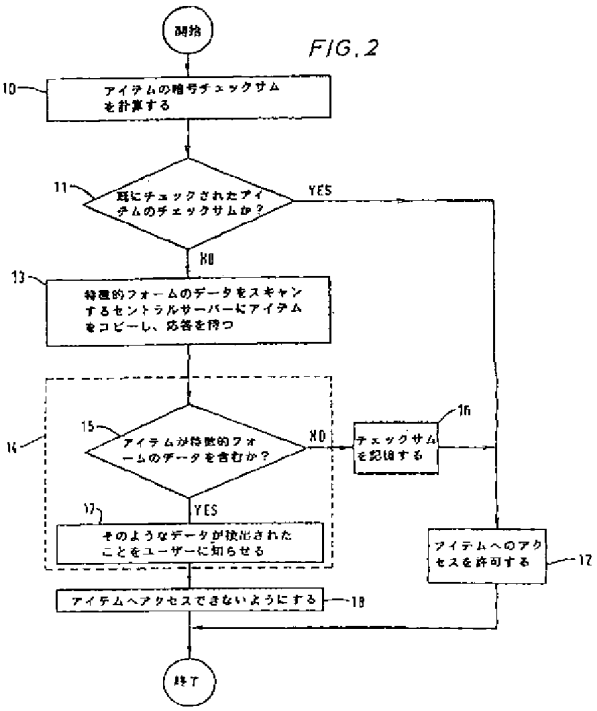
(71)出願人　ソフォス　ピーエルシー
英国　ＯＸ14　３ＹＳ　オックスフォードシャー　アビンドン　アビンドンサイエンスパーク　ザ　クアドラント　21番
(72)発明者　ルスカ　ジャン
英国　オックスフォードシャー　タブニー　サッチファームコテージス　１番
(72)発明者　ラマー　ピーター
英国　オックスフォードシャー　アビンドン　ドライサンドフォード　マナーコテージ
(74)代理人　弁理士　井上　一　（外２名）

最終頁に続く

(54)【発明の名称】　妥当性のチェック

(57)【要約】
本発明は、データ処理システムにおけるデータの妥当性のチェックに関するものである。

FIG.2



【特許請求の範囲】

（１）少なくともふたつの相互接続されたデータプロセッサを含むデータ処理ネットワークの中の第１のデータプロセッサによるアクセスのために記憶されるデータ・アイテムの妥当性をチェックする方法であって、

第１のデータプロセッサにより第２のデータプロセッサに対してデータ・アイテムをコピーするステップと、

第２のデータプロセッサが、アイテムの妥当性又は非妥当性を示す特徴的フォームのデータがそのアイテムに存在するかどうかを判断し、アイテムの妥当性について第１のデータプロセッサに報告するステップとを含むことを特徴とする。

（２）請求項１に記載の方法であって、

妥当であることがわかったデータ・アイテムの特徴を各レコードが示すレコード組が、第１のデータプロセッサによるアクセスのために記憶され、

妥当性をチェックしようとするデータ・アイテムの特徴を示すレコードを作成するステップと、

該レコードをレコード組から探し出すステップと、

レコード組中に該レコードが見つからない場合にのみ第２のデータプロセッサに対して前記データ・アイテムをコピーするステップと、

を含むことを特徴とする方法。

（３）請求項２に記載の方法であって、

データ・アイテムが妥当であるとする第２のデータプロセッサの報告に応じて、該データ・アイテムの特徴を示すレコードを、第１のデータプロセッサによるアクセスのために記憶されるレコード組に加えることを特徴とする方法。

（４）請求項１に記載の方法であって、、

第１のデータプロセッサが、データ・アイテムにアクセスするコマンドをインターセプトし、検出されたそのようなコマンドに応じて、データアイテムに特徴的フォームのデータがあるかどうかをチェックさせることを特徴とする方法。

（５）請求項１に記載の方法であって、

データ・アイテムが妥当であることがわからない場合又は妥当であることがわ

かるまで、第１のデータプロセッサが、データ・アイテムへのアクセスを妨げることを特徴とする方法。

（６）請求項１に記載の方法であって、

データ・アイテムが、実行可能な命令シークエンスを含むことを特徴とする方法。

（７）請求項１に記載の方法であって、

第１のデータプロセッサがワークステーションであることを特徴とする方法。

（８）請求項１に記載の方法であって、

第２のデータプロセッサがファイルサーバーであることを特徴とする方法。

（９）ネットワークとして相互接続された複数のデータプロセッサを含むデータ処理システムであって、

ネットワークの第１のデータプロセッサ中に設けられた手段であって、データ・アイテムをネットワークの第２のデータプロセッサにコピーさせる手段と、

第２のデータプロセッサに設けられた手段であって、アイテムの妥当性又は非妥当性を示す特徴的フォームのデータがそのアイテム中にあるかどうかをテストし、そのテストに基づいてそのようなデータがアイテム中に検出されたかどうかを示す妥当性信号を生成する手段と、

該妥当性信号を第１のデータプロセッサに伝送する手段と、

を含むことを特徴とするデータ処理システム。

（１０）請求項９に記載のデータ処理システムであって、

第２のデータプロセッサが、非妥当性を示す複数の特徴的フォームを規定する情報組を記憶する記憶手段を含むことを特徴とするデータ処理システム。

（１１）請求項９に記載のデータ処理システムであって、

第２のデータプロセッサが、妥当性を示す複数の特徴的フォームを規定する情報組を記憶する記憶手段を含むことを特徴とするデータ処理システム。

（１２）請求項９に記載のデータ処理システムであって、

第１のデータプロセッサが、特徴的フォームのデータを規定する情報組へのアクセスを持たないことを特徴とするデータ処理システム。

（１３）請求項９に記載のデータ処理システムであって、

第１のデータプロセッサが、データ・アイテム中に１又は複数の特徴的フォームのデータがあるかどうかをチェックするコマンドを受けて、そのようなコマンドに応じてアイテム中にそのようなデータがあるかどうかをチェックする手段を含むことを特徴とするデータ処理システム。

（１４）請求項９に記載のデータ処理システムであって、

ネットワークとして接続された少なくとも３つのデータプロセッサと、

ネットワークの他のデータプロセッサから第２のデータプロセッサへデータ・アイテムをコピーさせて、アイテム中に１又は複数の特徴的フォームのデータがあるかどうかをテストする手段とを含むことを特徴とするデータ処理システム。

（１５）請求項９に記載のデータ処理システムであって、

第１のデータプロセッサがワークステーションであることを特徴とするデータ処理システム。

（１６）請求項９に記載のデータ処理システムであって、

第２のデータプロセッサがファイルサーバーであることを特徴とするデータ処理システム。

（１７）請求項９に記載のデータ処理システムであって、請求項１乃至６のいずれかの方法を実行するためのデータ処理システム。

【発明の詳細な説明】

妥当性のチェック

［技術分野］

本発明は、データ処理ネットワークにおけるデータの妥当性をチェックする方法及び装置に関し、例えばそのデータが、ウィルスや他の好まざるデータを含むかどうか、あるいはそのデータが、ネットワークまたはその一部において使用を許可されているか禁止されているかをチェックするための方法及び装置に関する。

［背景技術］

一般的に、コンピューター・ファイル又はディスク・セクターのデータ（コンピューター・プログラム等のデータ）に対しては、望ましくないデータのチェック或いはファイルが使用を許可されているか禁止されているかを示す情報のチェックを行うことができ、これは所定フォームのデータをファイル中でサーチすることにより行われる。この所定フォームは、（順序は様々かもしれないが）ファイル中のどこかの場所に特定の情報が存在するといった所定の特徴、或いは（他のそのようなデータとの組み合わせとしてかもしれないが）ファイル中の特定の場所に特定の情報が存在するといった所定の特徴を含むことがある。例えばコンピューター・ウィルスは、ウィルスの作動命令となることができる一組のウィルス・データとして、コンピューター・ファイルのデータ中に記憶される。周知のウィルスに対するファイルのチェックは、各ウィルスの存在を示唆するものとして知られている特性をファイル中でサーチするウィルス検出手順によって行うことができる。チェックすべき周知のウィルスの数は増加しているため（現在約３０００個が知られている）、周知の全てのウィルスの特性を規定する情報を記憶するためには、かなりの記憶容量が必要となる。

ワークステーションとファイルサーバから成るのコンピューター・ネットワークでは、従来から、各ワークステーション自身が、自ら保持するデータの妥当性をチェックしていた。しかしそれは、全てのワークステーションが、その記憶容量の一部を使用して、サーチされる全ての特徴的フォームの規定情報を記憶しな

ければならないということを意味する。このことによって全体では膨大な記憶容量が必要となる。また新規のウィルスが確認されるなどして、より多くの特徴的フォームをサーチすることになれば、オペレーティング・システムの限界のため、ワークステーションは自主検索を実行することが不可能となる。また、新たな特徴フォームを取り入れるためには各ワークステーションを個々に更新しなければならなく、ワークステーションが数多くある場合には、これは不便なものとなる。

ＷＯ９３／０１５５０は、ライセンスプロダクトの使用を調整するシステムについて開示しており、ライセンスを受けた者のデータプロセッサによるアクセスのために記憶されたライセンスプロダクトに対してアクセスが可能かどうかを決定するために、ライセンス・データグラムが、ライセンス許諾者のデータプロセッサにコピーされ、データプロセッサが応答メッセージを返送する。このライセンス・データグラムは、ライセンスプロダクトからの様々なデータを含む。

［発明の開示］

本発明の第１の態様が提示するのは、少なくとも２つの相互接続されたデータプロセッサを含むデータ処理ネットワークの中の第１のデータプロセッサによるアクセスのために記憶されるデータ・アイテムの妥当性をチェックする方法であって、第１のデータプロセッサにより第２のデータプロセッサに対してデータ・アイテムをコピーするステップと、第２のデータプロセッサが、アイテムの妥当性又は非妥当性を示す特徴的フォームのデータがそのアイテムに存在するかどうかを判断し、アイテムの妥当性について第１のデータプロセッサに報告するステップとを含むことを特徴とする。第１のデータプロセッサはネットワークのワークステーションであることが望ましく、第２のデータプロセッサはネットワークのファイルサーバーであることが望ましい。特徴的フォームのデータは、データ・アイテムの非妥当性を適切に示すものであり、例えば、ウィルス又は他の好ましくないデータであることを示したり、あるいはデータ・アイテムの使用が禁止されていることを示すものである。

このタイプのシステムでは、テストされる１又は複数の特徴的フォームを規定する情報は、第２のデータプロセッサのみによるアクセスのために記憶される必

要がある。新たな特性が加わった場合には、（第２のデータプロセッサがアクセスする）単一の記憶手段のみが更新される必要がある。ネットワークが、第１のデータプロセッサと同等のデータプロセッサを他に有する場合、それによるアクセスのために記憶された、妥当性をチェックすべきいずれのデータ・アイテムも、これらのデータプロセッサにより第２のデータプロセッサにコピーされることが好ましい。第１のデータプロセッサおよびそれと同等の如何なるデータプロセッサも、テストされるデータの如何なる特徴的フォームを規定する情報のリストを記憶せず、通常はアクセスしないことが好ましい。

第２のデータプロセッサは、データ・アイテムの妥当性（または非妥当性）について、第１のデータプロセッサにリポートすることが好ましい。

テストされるデータの複数の特徴的フォームを規定する情報は、第２のデータプロセッサによって記憶され或いはアクセスされることが好ましい。そして第２のデータプロセッサは、そのような特徴的フォームがデータ・アイテム中にあるかどうかを、アイテム中に特徴的フォームの何れかでもあるかをテストすることによってテストすることが好ましい。

データ・アイテムは、例えば、第１のデータプロセッサによるローディングまたは実行によってアクセスされるファイルまたはプログラムであることが適切である。データ・アイテムは、実行命令のシークエンスを含むことが好ましい。

第１のデータプロセッサは、データ・アイテムにアクセスするコマンドをインターセプトし、検出されたそのようなコマンドに応じて、データ・アイテムの妥当性をチェックすることが好ましい。アイテムが妥当であるとわかるまで（即ち望ましくない特徴的フォームのデータが無い或いはアイテムが使用を禁じられているものであることを示すデータが無いとわかるまで）、第１のデータプロセッサは、例えば第１のデータプロセッサのユーザーによって、そのデータ・アイテムへのアクセスを妨げることが好ましい。これを達成するため、第１のデータプロセッサに、データ・アイテムへのアクセスコマンドを検出する手段を持たせ、そのコマンドをインターセプトさせ、アクセスされる前にアイテムが妥当なものとなっていることが保証されるようにすることが適切である。また第１のデータプロセッサによるアクセスのために記憶されたアイテム・データの何れか或いは

全ての妥当性についてシステムがチェックすることを、第１のデータプロセッサのユーザが指示できるようにすることが好ましい。また第１のデータプロセッサは次のように構成されていることが望ましい。即ち１又は複数の特徴的フォームのデータがアイテム中に存在するかについてのリポートを第２のデータプロセッサから受け、特徴的フォームのデータを含むとされたデータ・アイテムへのアクセスを妨げる或いは拒否し、そして／あるいは、そうしたデータを含まないとされたデータ・アイテムへのアクセスを許すように、構成されていることが好ましい。このようにすることで、ウィルスなどの望ましくないデータ或いは使用を禁じられたデータを含むアイテムへのアクセスを防止することができる。

好ましくは、データの特徴的フォームは、データ・アイテムの妥当性、例えばそのアイテムが使用を許可されているかどうかを示すデータのフォームを含むことができる。この時、第１のデータプロセッサは、そのようなデータを含まないアイテムへのアクセスを妨げる或いは拒否し、そして／あるいは、そのようなデータを含むアイテムのみへのアクセスを許可するようにしてもよい。

好ましくは本発明の方法は、妥当であることがわかったデータ・アイテムの特徴を各レコードが示すレコード組を第１のデータプロセッサが記憶し或いは第１のデータプロセッサが該レコード組のアクセスを持ち、妥当性をチェックしようとするデータ・アイテムの特徴を示すレコードを作成するステップと、該レコードをレコード組から探し出すステップと、レコード組中に該レコードが見つからない場合にのみ第２のデータプロセッサに対して前記データ・アイテムをコピーさせるステップとを含むことを特徴とする。第１のデータプロセッサは、レコード組を記憶する手段、および／またはレコードを作成しそれらをレコード組の内容と比較する処理手段とを含むことが好ましい。またデータ・アイテムが妥当であるとする第２のデータプロセッサの報告に応じて、該データ・アイテムの特徴を示すレコードがレコード組に加えられることが好ましい。各レコードは、対応するデータ・アイテムに対して計算されたチェックサム（検査合計）であることが好ましい。

本発明の第２の態様により提示されるのは、ネットワークとして相互接続された複数のデータプロセッサを含むデータ処理システムであり、このデータ処理シ

ステムは、ネットワークの第１のデータプロセッサ（好ましくはワークステーション）中に設けられた手段であって、データ・アイテムをネットワークの第２のデータプロセッサ（好ましくはファイルサーバー）にコピーさせる手段と、第２のデータプロセッサに設けられた手段であって、アイテムの妥当性又は非妥当性を示すデータがそのアイテム中にあるかどうかをテストし、そのテストに基づいてアイテムの妥当性を示す妥当性信号を生成する手段と、該妥当性信号を第１のデータプロセッサに伝送する手段とを含むことを特徴とする。

第２のデータプロセッサは、妥当性信号を第１のデータプロセッサへ、リポート・メッセージ、ファイル、またはパケットという形でリポートあるいは伝送することが好ましい。第２のデータプロセッサは定期的に適切にスキャンし、テストすべきデータ・アイテムを受け取ったかどうかを判断できる。あるいは第１のデータプロセッサが、例えばパケットとして信号を第２のデータプロセッサに伝送し、テストすべきデータ・アイテムを送ったことを知らせることもできる。

［図面の簡単な説明］

第１図は、本発明が使用される代表的なデータ処理システムの概略図である。

第２図は、本発明の一実施例を示すフローチャートである。

第３図は、本発明の他の実施例を示すフローチャートである。

［発明を実施するための最良の形態］

次に、添付図面を用い、例をあげて本発明を説明する。

図１に、ファイルサーバー１及びワークステーション２ａ、２ｂ、２ｃを構成するデータプロセッサ群を含むネットネットワーク形式のデータ処理システムを示す。ネットワークの汎用アーキテクチャは従来からあるもので、たとえばＩＢＭ社のＰＣタイプ、あるいはアップル社のマッキントッシュ・ワークステーション、およびＶＡＸ／ＶＭＳ、Ｎｏｖｅｌ またはＯＳ２ファイルサーバーがある。各ワークステーションは、データファイルと実行プログラムを記憶できる。ファイルサーバー１は、データを記憶するための記憶手段３、データ処理手段４、ワークステーションとの通信のための通信手段５を含む。ワークステーション２ａは、記憶手段６、データ処理手段７、ファイルサーバーとの通信のための通信手

段８を含む。他の各ワークステーションも同等の構成要素を有する。各ワークステーションの記憶手段は、他のワークステーションから離れた位置、例えばファイルサーバーの位置に配置してもよい。

ファイル（例えばプログラムのような実行可能な命令シークエンス、或いはより一般的にはアイテムデータ、即ちワークステーションによるアクセスのため記憶されるもの又はワークステーションにより実行されようとされているもの）の妥当性をチェックする必要がある場合には、そのファイルはファイルサーバーへとコピーされる。そしてファイルサーバーは、ファイル内に特徴的なフォームのデータがあるか否かをテストし、リポートメッセージ、一般的には、ファイルがそのようなデータを含むか否か或いはファイルが妥当か否かを示す信号を戻す。

更に詳しく述べれば、各ワークステーションは、例えばローディングや実行によってワークステーション上のいずれかのファイルへのアクセスコマンドをインターセプトし、そのファイルの妥当性をチェックして妥当であると判明するまでそのファイルへのアクセスを妨げることによって、ファイルの妥当性をチェックする必要がある場合を検出するように構成されている。ワークステーションは、プログラムやブートセクターのような特定の「保護」アイテムへのアクセスコマンドだけをインターセプトするように構成してもよい。

図２に示す手順を実行することで、ファイルの妥当性がチェックされる。まずワークステーションは、ファイルが、これまでにチェックされ妥当であると判断されたものであるかどうかを調べる予備手続きを行う。これにより各ファイルに対して一回を超える妥当性チェックが行われることを回避できる。ワークステーションは、チェックされるファイルの特徴を示すものである暗号チェックサム(cryptographic checksum)の計算を行う（図２のボックス１０）。これは、標準的なＡＮＳＩ　Ｘ９．９或いはＩＳＯ８７３１のパート２手順を用いて、３２又は６４ビットチェックサムを計算することにより適切に行うことができる。このチェックサムは、既にチェックされ妥当と判断されたファイルのチェックサムへのアクセスをワークステーションが持っているリスト内においてサーチされる（ボックス１１）。このリストは、ワークステーションの記憶手段に記憶しても

よいし、ネットワークサービスの１つとしてファイルサーバーにより記憶してもよい。テストにおいてファイルのチェックサムがリスト内に見つかりファイルが妥当であるとされた場合には、ファイルへのアクセスが許可される（ボックス１２）。ファイルのチェックサムがリスト内に見つからなければ、そのファイルはファイルサーバーにコピーされ（ボックス１３）、妥当性が直接テストされることになる。

図２のボックス１４には、ファイルサーバーが行うステップが概略的に示されている。ファイルの妥当性または非妥当性を示すデータの特徴的フォームを規定する情報は、ファイルサーバーに記憶されている。ファイルがウイルスなどの望ましくないデータを含んでいるかどうか或いは使用を許可するか禁ずるかを、こうした特徴的フォームが示す場合がある。例えばウイルスの場合、ファイルの最初にある命令（たいていはジャンプ命令）、ファイルのいずこかにある命令、あるウイルスがあるシークエンスで現れるなど、そのウィルスに特徴的なデータのフォームを特性が示すことがある。ファイルサーバーが、特徴的フォームのデータの存在をテストすべきファイルのコピーを受け取ると、ファイルサーバーはファイルをスキャンし（ボックス１５）、ファイル内に特徴的フォームのデータがないかどうかを調べ、チェックするファイルを送り出したワークステーションに対してそのようなデータが見つかったかどうかを示すリポートメッセージを戻す。もし、その時に既に使用が許可されていることを示すファイルのデータのテストをファイルサーバーがするのであれば（図２には示さず）、ワークステーションへ送られるリポートメッセージは、そのデータが妥当か非妥当かを定義するものでなければならない。そうでなければリポートメッセージは直接そのファイルが妥当か妥当でないかを報告する。

リポートメッセージが、ファイルが望ましくないデータを含まないということ、または禁ずるべきデータを持たないということ、および／またはファイルの含むデータは使用が許可されるものであるということ（すなわちメッセージがファイルは妥当であると報告する）を示した場合には、ワークステーションは、ファイルのチェックサムを妥当ファイルのチェックサムリストに加え（ボックス１６）、ファイルへのアクセスを許可する。リポートメッセージがファイルが妥当で

ない

ことを示せば、ワークステーションはユーザーに報告し（ボックス１７）、例えばメッセージを表示するなどして、ファイルへのアクセスを防ぐ（ボックス１８）。

あるいは、ワークステーションによるアクセスのために記憶されたファイルの何れか又は全ての妥当性をチェックし、記憶されたファイルの確認（authenticate）を行うことを、ワークステーションのオペレータが指示することもできる。この確認（authentication）は、記憶されたリストに対するファイルのチェックサムの検査ステップ（ボックス１１）を省略し、各ファイルが確実に、ファイルサーバーによって特徴的フォームのデータの有無を直接テストされるようにすることで行うことができる。

ファイルの妥当性・非妥当性を示すデータのチェックに加え、システムに導入されたいずれのファイルも、アクセス前にネットワーク・スーパーバイザーによって許可されたものであることをシステムが知っておかなければならないように、システムを構成することもできる。これを達成するための一つの方法が、図３に示した手順である。これはファイルがテストのためファイルサーバーにコピーされた後に行うことができる。ファイルサーバーはファイルに対するチェックサムを計算する（ボックス２０）。そしてファイルサーバーによって記憶されたリストであり、スーパーバイザーによって既に使用を許可されたファイルのチェックサムのリスト内で、そのチェックサムを探す（ボックス２１）。チェックサムがリスト内に見つかれば、ファイルを使用できるというリポートメッセージが、関連ワークステーションへ戻される（ボックス２２）。別のワークステーションがそれまでにそのファイルをテストのためファイルサーバーに渡していても、これは起こる場合がある。チェックサムが見つからなければ、ファイルは、上述の方法（ボックス２３～２５）により特徴的フォームのデータの有無がテストされ、ネットワークスーパーバイザーへ、ファイルの妥当性に関するメッセージと共に送られる（ボックス２６）。その際ファイルがスーパーバイザーによって許可されれば、ファイルサーバーによって計算されたチェックサムは、ファイルサー

バーの許可済みファイルのチェックサムリストに加えられ（ボックス２７）、リポートメッセージが関連ワークステーションに戻され、ファイルへのアクセスが許可されたことが示される（ボックス２２）。ファイルがスーパーバイザーによっ

て許可されなければ、そのチェックサムはリストに加えられず、リポートメッセージが関連ワークステーションに戻されて、ファイルがアクセスされるべきものではないことが示される（ボックス２８）。この手順は、ファイルが使用を許可されたか禁じられたかを示すデータがファイルに含まれるかどうかということに加えて用いられる（図３には示さず）。

以上のシステムによれば、ファイルサーバーによって記憶される必要があるのは、テストすべき特徴を規定する情報のリストひとつだけであるため、システムが更新されるときに変更する必要があるのはリストの１コピーだけとなる。この点は、全てのワークステーションの持つコピーを変更しなければならない従来のシステムと比べ非常に便利である。より多くの特徴的フォームをサーチしなければならないようになった場合（例えば新しいウィルスが明らかになった場合）でも、ファイルサーバー（通例ワークステーションよりも強力なオペレーティングシステムを持っている）は、特徴的フォームのテストを続けることができる。また、妥当なファイルのチェックサムのリストひとつが、全ワークステーションによるアクセスのために記憶されていれば、その後、特徴的データのためのファイルチェックが必要になるのは、いずれかのワークステーションによってファイルが最初にアクセスされた時のみとなり、各ワークステーションの最初のアクセス毎にチェックをする必要はなくなる。

ファイルをファイルサーバーへ転送する二つの方法と、ワークステーションに戻されるリポートメッセージについて説明する。第１の方法によれば、ワークステーションは、ワークステーションを識別するデータと共に、テストするファイルを、所定のフォーマット（たとえば所定のファイルエクステンション）を持つランダム選択された名前のファイルとして、ファイルサーバーへ（暗号化されたフォームで）コピーする。ファイルサーバーは定期的にそういったファイルをス

キャンするよう構成され、ひとつ見つかるとファイルサーバーがそれを解読し、特徴的フォームのデータの有無をテストするように構成されている。ファイルサーバーは、ワークステーションへの伝送のための、応答メッセージを含む応答ファイルを作成することによって、受け取ったファイル内で識別されたワークステーションへ応答メッセージを戻す。応答ファイルの名前は、ワークステーションによって伝送される対応ファイルの名前の関数として作成される。従って、複数のワークステーションがチェックのためにファイルを送った場合でも、それぞれがチェック要請に対する応答を含むファイルを識別できることになる。

第２の方法によれば、ファイル伝送はネットワークパケットに依存する。テストするファイルは上述のようにファイルサーバーへコピーされるが、ファイルサーバーが、テストするファイルを定期的にスキャンする代わりに、伝送を行うワークステーションが、ファイルサーバーへパケットメッセージを送り、テストするファイルが送られたことをファイルサーバーに知らせる。これを受けると、ファイルサーバーはファイルをテストする。リポートメッセージは、パケットとして、伝送を行ったワークステーションに戻される。

このシステムは、互いに通信するワークステーションまたは中間ネットワークを介したファイルサーバーによって作動するようにしてもよい。

ファイルサーバーが特徴的フォームのデータの有無をテストする代わりに、この機能を、ネットワークの中の選択されたワークステーションによって行ってもよい。

好適な実施例を用いて本発明を詳細に示し説明したが、発明の範囲と精神から離れることなく、細部においてさまざまな変形が可能であることは、当業者には理解されるはずである。

【図１】

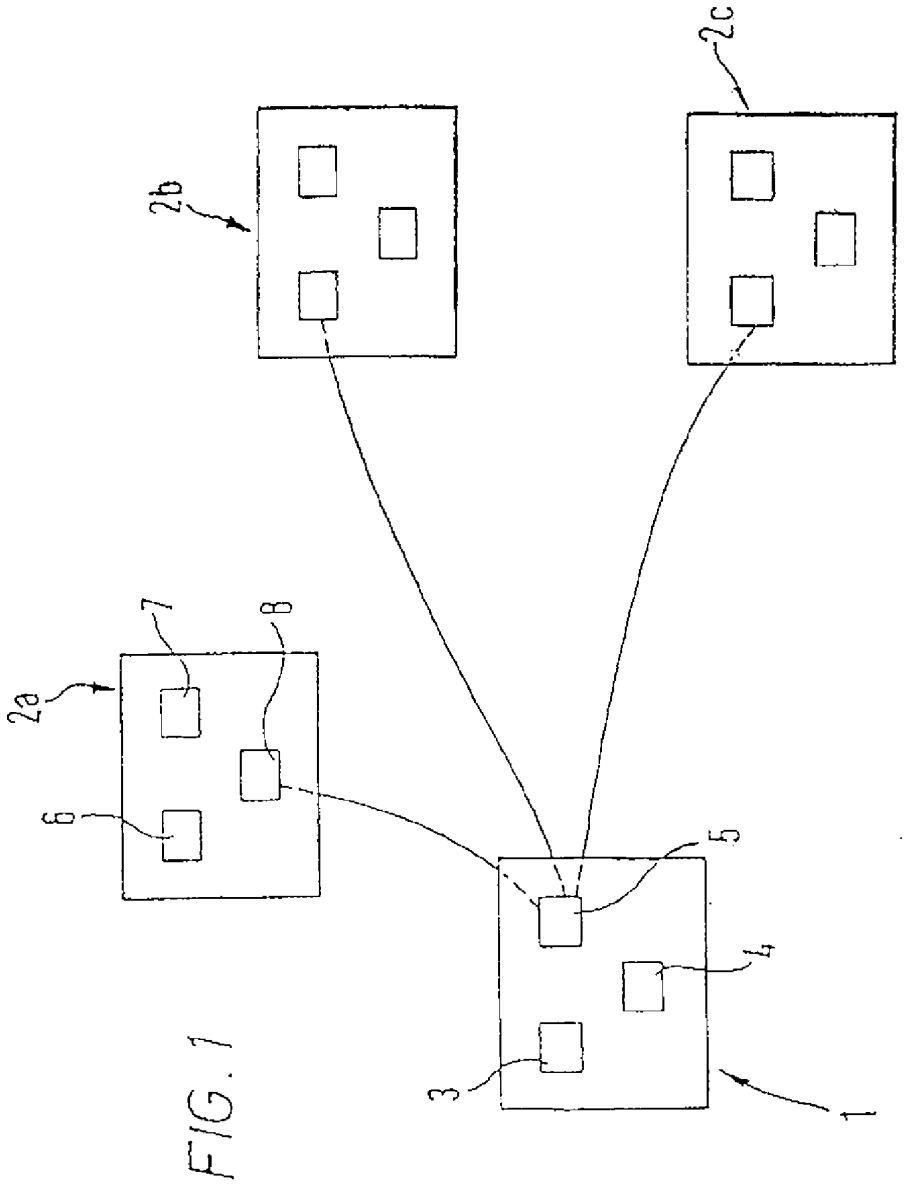


FIG. 1

【図２】

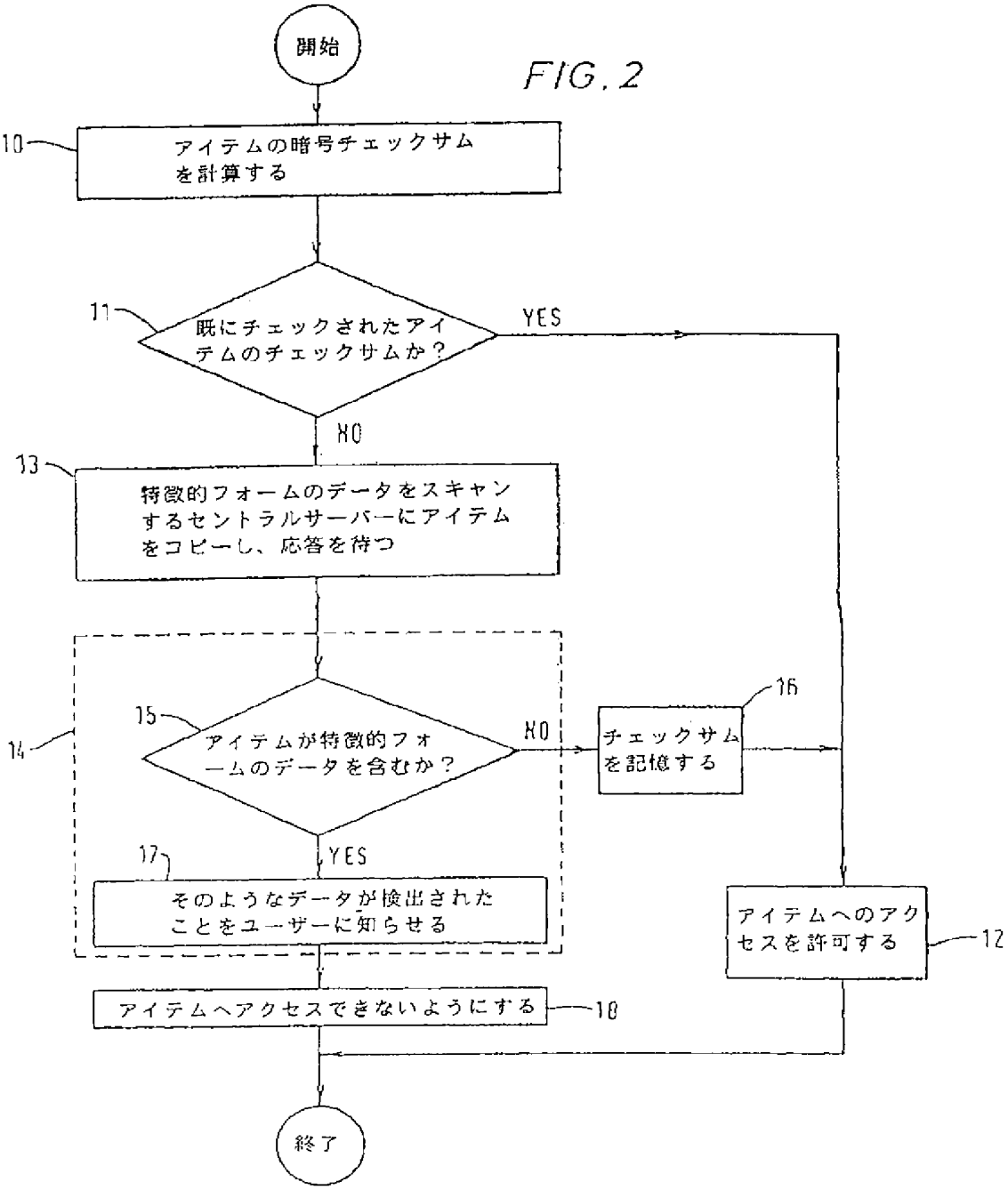
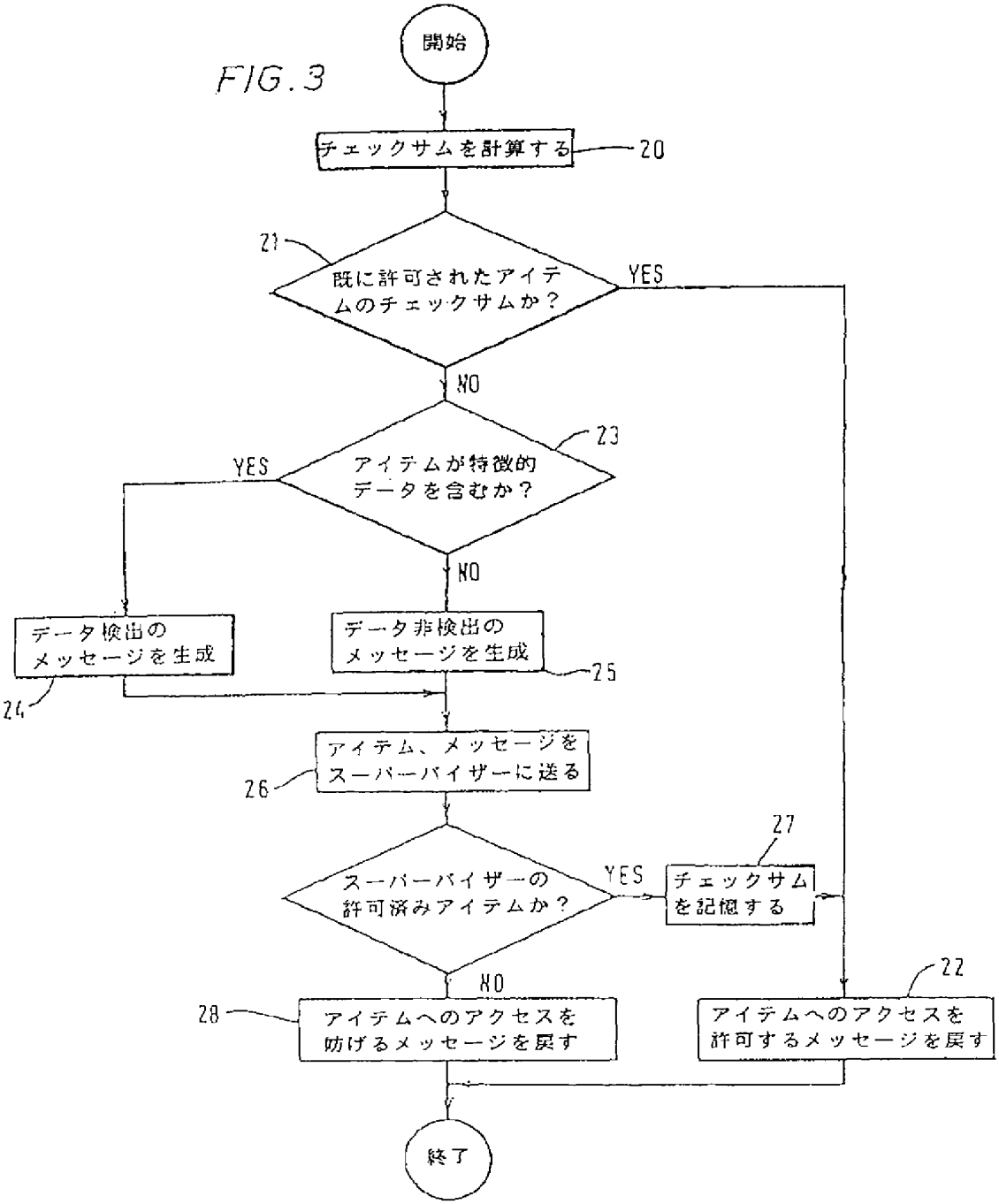

開始
10
アイテムの暗号チェックサムを計算する
11
既にチェックされたアイテムのチェックサムか？
YES
NO
13
特徴的フォームのデータをスキャンするセントラルサーバーにアイテムをコピーし、応答を待つ
14
15
アイテムが特徴的フォームのデータを含むか？
NO
16
チェックサムを記憶する
YES
17
そのようなデータが検出されたことをユーザーに知らせる
アイテムへのアクセスを許可する
12
アイテムへアクセスできないようにする
18
終了

FIG. 2

【図３】

FIG. 3


開始
チェックサムを計算する
20
21
既に許可されたアイテムのチェックサムか？
YES
NO
23
アイテムが特徴的データを含むか？
YES
NO
データ検出のメッセージを生成
データ非検出のメッセージを生成
25
24
アイテム、メッセージをスーパーバイザーに送る
26
27
スーパーバイザーの許可済みアイテムか？
YES
チェックサムを記憶する
NO
28
アイテムへのアクセスを妨げるメッセージを戻す
22
アイテムへのアクセスを許可するメッセージを戻す
終了

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Intern. al Application No
PCT/GB 94/02378

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F11/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	EP,A,0 449 242 (NATIONAL SEMICONDUCTOR CORPORATION) 2 October 1991 see claims 1,11 ---	1-17
Y	EP,A,0 400 233 (MICHAEL H. FRANCISCO) 5 December 1990 see claim 1 ------	1-17

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 January 1995	31.01.95

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Corremans, G

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Intern al Application No
PCT/GB 94/02378

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0449242	02-10-91	NONE		
EP-A-0400233	05-12-90	US-A-	4845715	04-07-89
		GB-A-	2232279	05-12-90

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(81)指定国 EP(AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OA(BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), AP(KE, MW, SD, SZ), AM, AT, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, CZ, DE, DE, DK, DK, EE, ES, FI, FI, GB, GE, HU, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SI, SK, SK, TJ, TT, UA, US, UZ, VN

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平6－110718

(43)公開日　平成６年(1994)４月22日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/00	３１０　Ｎ	7313－5B		
9/06	４１０　Ｐ	9367－5B		
12/00	５３１　Ｒ	8526－5B		

審査請求　未請求　請求項の数３(全 15 頁)

(21)出願番号　特願平4－261336

(22)出願日　平成４年(1992)９月30日

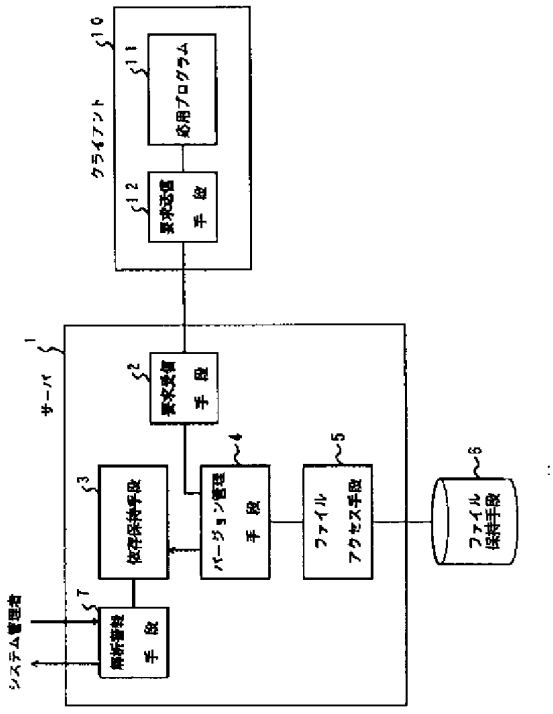
(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　関戸　一紀
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝総合研究所内

(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】　ウィルス防御方式

(57)【要約】
【目的】ウィルス感染の範囲を推定すると共に、感染したプログラムを元に戻すウィルス防御方式を実現する。
【構成】バージョン管理手段４によるバージョン管理によって、既存のファイルを損なうことなくファイル変更が行なわれる。よって、いつでもウィルス感染等によって変更が加えられたバージョンを削除し、前の状態に戻すことができる。また、依存保持手段３が管理しているクライアント間のファイルの依存関係およびファイルアクセスの履歴を用いることにより、ウィルス感染の可能性のあるファイルを必ずシステム管理者に警報できるとともに、その検査結果に基づいてバージョンをもとに戻しているので、ウィルスが感染したバージョンは必ず削除される。よって、ウィルス感染を完全に防御できる。



1

【特許請求の範囲】

【請求項１】　ファイルアクセスを要求する複数のクライアントと、これら各クライアントからのファイルアクセスに応じてファイルアクセスを行うサーバとから構成されるクライアント／サーバシステムにおいて、

前記サーバは、前記各クライアントからのアクセス要求に従って、前記複数のクライアントのファイルアクセスの履歴およびクライアント間におけるファイルアクセスの依存関係を管理する第１の管理手段と、

前記各クライアントからのファイル変更要求に応じて変更対象ファイルのバージョンを追加作成し、各ファイル毎にバージョンを管理する第２の管理手段と、

前記第１の管理手段で管理されているファイルアクセスの履歴を解析してウィルス検査の必要性の有無を検出すると共に、前記クライアント間におけるファイルアクセスの依存関係に従って検査対象のファイルおよびクライアントを追跡する解析手段と、

所定のウィルス検査の結果にしたがって前記各ファイル毎に不要バージョンを削除する手段とを具備することを特徴とするクライアント／サーバシステムのウィルス防御方式。

【請求項２】　ファイルをアクセスを要求する応用プログラムと、そのアクセス要求をネットワーク経由でサーバに伝える要求送信手段から構成される複数のクライアントと、

クライアントからのアクセス要求を受け取る要求受信手段と、この要求に従ってファイルアクセスを行うファイルアクセス手段と、ファイルを記憶しているファイル保持手段とから構成されるサーバとをネットワークで接続したクライアント／サーバシステムにおいて、

前記サーバに、ファイル変更（挿入、削除、更新）が必要な場合にそのファイルの新しいバージョンを作成管理し、ファイル保持手段内のすでに存在するファイルを損なうこと無くファイル変更を実現するとともに、指定された不必要なバージョンを削除するバージョン管理手段と、

バージョン管理手段が行ったファイルアクセスの履歴から、ファイルを挿入・更新したクライアントとそのファイルを参照したクライアントとの間の依存関係と、各クライアント内でのファイルアクセスの順序関係によるファイル間の依存関係を作成保持する依存保持手段と、

依存保持手段の情報を常に解析し、ウィルス検査の必要がないか調べ、必要な場合にはシステム管理者にすべて検査すべきクライアントとファイルを指定して警報するとともに、システム管理者がウィルス検査した結果を元に解析し、不必要となったバージョンをバージョン管理手段に伝え、さらに検査の必要なクライアントやファイルをシステム管理者に警報する解析警報手段を有し、

システム管理者が解析警報手段の指示に従いクライアントやファイルのウィルス検査を行い、その結果を解析警報手段に教えることを特徴とするウィルス防御方式。

2

【請求項３】　前記サーバは、さらにウィルス検査手段を有することを特徴とする請求項２記載のウィルス防御方式。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明はクライアント／サーバシステムにおけるサーバのウィルス防御方式に関する。

【０００２】

【従来の技術】近年、低価格で高性能なパソコン（ＰＣ）・ワークステーション（ＷＳ）の出現とネットワーク技術の進歩により、ファイルをクライアント／サーバ方式で共有するシステムが増えている。このシステムは、図１２に示すように、独立して動作するのに必要なファイルしか持たない多数の小型計算機（クライアント）１６－１～Ｎと、大量のファイルを保持する少数の（１つだけでもよい）大型計算機（サーバ）１５をネットワーク１７で接続した構成である。この方式により、これまで個々の計算機で重複して保持していたファイルを、１カ所に集中管理でき、クライアントが変わってもファイルをコピーして移動させる必要がない。さらに、クライアントには、大型のディスクを登載する必要がなく、安価でコンパクトな計算機を使える。

【０００３】また、最近、ＰＣを中心にウィルスと呼ばれる計算機に障害を起こすプログラムによる被害が色々と報告されている。このプログラムは、予め設定された時期になると計算機内のファイルを消去したりＯＳの基本的なプログラムを破壊したりするもので、プログラムやＯＳ等を経由して他のプログラムに伝染する機能を持っている。ウィルスに感染したプログラムを実行すると、まず、他のプログラムにもウィルスを感染させるため、ウィルスプログラムはその計算機からアクセス可能な全ファイル／ディレクトリを調べ、ウィルスの感染可能なプログラムやウィルス感染したプログラムを格納可能なディレクトリを見つける。また、感染可能なプログラムを順次読み出してウィルスプログラムを組み込んでから書き戻したり、ウィルス感染したプログラムを格納可能なディレクトリに書き込む。さらに、その計算機のプログラム実行を制御するＯＳにもウィルスプログラムを組み込み、ＯＳが起動される毎に感染を試みる。なお、ウィルスチェッカと呼ばれるウィルスの存在を確認するプログラムも開発されている。

【０００４】これまでは、フロッピー・ディスク等の媒体を介してウィルス感染する場合が主な経路で、各計算機の管理者が身元のはっきりしないプログラムを実行しないように注意すれば良く、その被害もウィルスプログラムを実行した計算機に限られた。しかし、クライアント／サーバ方式のシステムでは、ユーザのファイルはすべてサーバに格納されており、各クライアントからそれらにアクセスできる必要がある。よって、クライアント

からサーバ内の全ファイルが原則としてアクセス可能であり、１人の不注意なユーザによるウィルス感染したプログラムの実行が、サーバ内の全ファイルに感染し、さらに全クライアントに障害をもたらす危険性がある。よって、ウィルス防御手段がファイルサーバには必須である。しかし、従来のファイルサーバにはアクセス権による方法しかなく不十分であった。以下、従来のファイルサーバで採られているアクセス権方式とその問題点について簡単に説明する。

【０００５】クライアント／サーバシステムにおける、ファイルサーバ２０とクライアント２５の構成を図１３に示す。ファイルサーバ２０は、クライアントからのファイル要求を受け取る要求受信手段２１と、ファイルへのアクセス権を検査するアクセス権検査手段２２と、ファイルやそのアクセス属性の読み出し書き込みを行うファイルアクセス手段２３と、実際に多数のファイルを保持するファイル保持手段２４から構成される。一方、クライアント２５は、サーバ内にあるファイルを必要とする応用プログラム２６と、現在クライアント２５を使っているユーザの識別子を保持するユーザ識別子保持手段２７と、応用プログラム２６の指示に従ってサーバへ要求を送り出す要求送信手段２８から構成される。なお、ユーザ識別子保持手段２７内の識別子は、クライアント２５がサーバ２０へ論理的接続（ログインやマウント）を行った時に設定される。論理的接続ではパスワード検査等を行い、正規のユーザであることを確認している。

【０００６】このシステムの動作について簡単に説明する。まず、クライアントの応用プログラム２６が、ファイル名前とアクセス種別（読み出し／書き込み）を指定してアクセス要求を要求送信手段２８に伝える。この要求を受け取った要求送信手段２８は、ユーザ識別子保持手段２７からユーザ識別子を読み出して、ファイル名、アクセス種別、ユーザ識別子からなるファイル要求を作成してサーバ２０に送る。

【０００７】サーバ２０側では、この要求を要求受信手段２１が受け取り、この要求内容に従って、ユーザ識別子のユーザが、ファイル名のファイルへ、アクセス種別のアクセスを行うものとして、ファイルアクセス手段２３へアクセス要求を伝える。この要求を受け取ったファイルアクセス手段２３は、まず指定されたファイルのアクセス属性をファイル保持手段２４から取り出し、アクセス種別、ユーザ識別子とともにアクセス権検査手段２２に送る。アクセス権検査手段２２では、格納されていたアクセス属性を解析し、このファイルはユーザ識別子のユーザに対してアクセス種別のアクセスが許されているか検査する。この結果がファイルアクセス手段に伝えられ、許可される場合にのみアクセス種別のアクセスがファイル保持手段２４内のファイル名のファイルに対して行われる。

【０００８】ここで、応用プログラム２６がウィルスに

感染している場合を考える。応用プログラムを実行するとウィルスプログラムが動き出す。ウィルスプログラムはサーバ内のファイルへウィルスを感染させようと試み、ファイル名とアクセス種別を指定してアクセスを要求送信手段２８に伝える。要求送信手段２８は、ユーザ識別子保持手段２７から現在クライアントを使っているユーザのユーザ識別子を読み出して、アクセス要求を作成してサーバ２０に送る。

【０００９】サーバ２０側では、この要求を要求受信手段２１が受け取り、ファイルアクセス手段２３へ伝える。ファイルアクセス手段２３は、まず指定されたファイルのアクセス属性をファイル保持手段２４から取り出して、アクセス種別やユーザ識別子とともアクセス検査手段２２に送る。アクセス検査手段２２では、格納されていたアクセス属性を解析し、このファイルがユーザ識別子のユーザにアクセス種別のアクセスが許可されているかを検査する。許可されている場合にのみ、ファイルアクセス手段２３はファイル保持手段２４内のファイルに対してアクセスを行う。よって、各ユーザがアクセス（特に変更）できるファイル／ディレクトリの属性を非常に限られたものに設定することにより、サーバ内の大部分のファイルにはアクセス検査手段２２からアクセス許可が出ず、ウィルスプログラムが感染できるプログラムは非常に制限される。

【００１０】以上のように、サーバ側ではユーザ識別子とアクセス属性を用いてファイルへのアクセスを制限しており、結果としてウィルスプログラムを実行したユーザが変更を許されているプログラムにしか影響が及ばない。よって、ファイルへのアクセス権方式を用いて、ウィルスプログラムを実行したユーザの範囲内にウィルス汚染の影響を抑えられる。しかし、本方式には次のような問題点がある。

【００１１】第１に、ウィルスプログラムを実行したユーザが管理している他のプログラムやディレクトリにはアクセス許可が常に出ているので、それにウィルスが感染するのを防げない。

【００１２】第２に、一般にはユーザ毎に別々にプログラムやディレクトリが存在するのではなく、共通のプログラムやディレクトリ、即ちウィルス感染の可能なプログラムやウィルスプログラムを格納可能なディレクトリが存在する。よって、あるユーザがウィルスプログラムを実行すると、共通のプログラムに感染していたり、ディレクトリにウィルス感染プログラムを格納したりする。よって、それ以降にそのプログラムを実行したユーザにもウィルス汚染が及んでしまう。

【００１３】第３に、ウィルスプログラムを実行してクライアントのＯＳに感染した場合、そのクライアントをそのまま他のユーザが使うと、ＯＳ内のウィルスプログラムが作動してそのユーザにアクセスの許されているプログラム／ディレクトリにまで感染してしまう。

5

【００１４】第４に、このようなサーバには先のアクセス権によってアクセスが禁止されない特権ユーザが存在しており、クライアント／サーバシステムを管理・運用している。よって、特権ユーザとしてウィルスプログラムをクライアントかサーバで実行すると、サーバ内の全プログラムに感染させることが可能となる。さらに、これらの感染原因は複合して作用し、一層感染の範囲を広げてしまう。

【００１５】また、サーバ内の全ファイルと全クライアントに感染する可能性があるので、ウィルスチェッカがウィルスが存在しないことや感染が及んでいる範囲を確認するには、全ファイルと全クライアントを検査する必要があり、ファイル数やクライアント数に比例した膨大な作業を必要とする。

【００１６】

【発明が解決しようとする課題】以上のように、従来のアクセス権方式ではサーバ内にウィルスが感染するのを完全に防げないばかりでなく、サーバ内にウィルスが存在しないことやウィルス感染の範囲を確認するのに膨大な作業が必要であった。本発明は、ウィルス感染の範囲を推定するとともに感染したプログラムを元にもどすウィルス防御方式を提供することを目的とする。

【００１７】

【課題を解決するための手段および作用】本発明のウィルス防御方式は、ファイルアクセスを要求する複数のクライアントと、これら各クライアントからのファイルアクセスに応じてファイルアクセスを行うサーバとから構成されるクライアント／サーバシステムにおいて、前記サーバに、前記各クライアントからのアクセス要求に従って、前記複数のクライアントのファイルアクセスの履歴およびクライアント間におけるファイルアクセスの依存関係を管理する第１の管理手段と、前記各クライアントからのファイル変更要求に応じて変更対象ファイルのバージョンを追加作成し、各ファイル毎にバージョンを管理する第２の管理手段と、前記第１の管理手段で管理されているファイルアクセスの履歴を解析してウィルス検査の必要性の有無を検出すると共に、前記クライアント間におけるファイルアクセスの依存関係に従って検査対象のファイルおよびクライアントを追跡する解析手段と、所定のウィルス検査の結果にしたがって前記各ファイル毎に不要バージョンを削除する手段とを具備することを特徴とする。

【００１８】このシステムにおいては、第２の管理手段によるバージョン管理によって、既存のファイルを損なうことなくファイル変更が行なわれる。よって、いつでもウィルス感染等によって変更が加えられたバージョンを削除し、前の状態に戻すことができる。また、第１の管理手段が管理しているクライアント間のファイルの依存関係およびファイルアクセスの履歴を用いることにより、ウィルス感染の可能性のあるファイルを必ずシステム管理者に警報できるとともに、その検査結果に基づいてバージョンをもとに戻しているので、ウィルスが感染したバージョンは必ず削除される。よって、ウィルス感染を完全に防御できる。

【００１９】さらに、解析手段が常に情報を解析してウィルス存在の可能性をチェックしているので、ウィルス感染しても広範囲に広がる前に発見できる。また、クライアント間の依存関係の情報を使ってウィルス感染の可能性があるクライアントやファイルを限定できる。よって、サーバ内におけるウィルスの存在や存在するときの感染範囲を確認するのに少数のファイルやクライアントを検査するだけでよい。

【００２０】

【実施例】以下に本発明の実施例を説明する。図１は本発明のウィルス防御方式を適用して構成した、サーバ１とクライアント１０からなるクライアント／サーバシステムの構成図である。

【００２１】サーバ１は、クライアントからのアクセス要求を受け取る要求受信手段２と、クライアントからのアクセス要求に従ってクライアント間やファイル間の依存関係を作成保持する依存保持手段３と、ファイルへの変更（挿入、削除、更新）をファイルに直接反映せずにバージョンを作って管理するバージョン管理手段４と、ファイルの読み書きを行うファイルアクセス手段５と、実際に多数のファイルを保持するファイル保持手段６と、依存保持手段３が保持しているクライアント間やファイル間の依存関係を解析しウィルス検査の必要性をシステム管理者に警報する解析警報手段７から構成される。一方、クライアント１０は、サーバ内にあるファイルを必要とする応用プログラム１１と、応用プログラム１１の指示に従いサーバ２へアクセス要求を送り出す要求送信手段１２から構成される。このシステムの動作について以下で簡単に示す。

（１）クライアントの応用プログラム１１がファイル名とアクセス種別（読み出し、書き込み）からなるアクセス要求を要求送信手段１２に伝える。

（２）この要求を受け取った要求送信手段１２は、クライアント名を含むファイル要求を作成してサーバ１に送る。

（３）サーバ１では、この要求を要求受信手段２が受け取り、ファイル名、アクセス種別およびクライアント名をバージョン管理手段４に伝える。

【００２２】（４）バージョン管理手段４は、指定されたファイル名のファイルがすでに管理されているか調べる。もし、管理されていなければ、ファイル保持手段６に存在するかをファイルアクセス手段５に問い合わせた後、管理対象ファイルとして登録する。

【００２３】（５）次に、バージョン管理手段４は、登録された情報に従って、指定されたファイルにアクセス種別のアクセスが可能か調べ、そのアクセス形態（読み

出し、挿入、更新、削除）を決定する。なお、アクセス可能な場合にのみアクセス処理が継続する。

【００２４】（６）さらに、バージョン管理手段４は、確定したアクセス形態、ファイル名、クライアント名を依存保持手段３に伝えるとともに、既存ファイルの内容を損なわないよう、バージョンを作るファイル要求に変えてファイルアクセス手段５に伝える。

【００２５】（７）ファイルアクセス手段５は、要求されたファイルアクセスをファイル保持手段６内のファイルに対して行い、その結果がバージョン管理手段４、要求受信手段２、要求送信手段１２を経由して応用プログラム１１に伝えられる。

【００２６】（８）一方、確定したアクセス形態、ファイル名、クライアント名を受け取った依存保持手段３は、その情報を元にクライアント間やファイル間の依存関係を作成保持し、その情報を解析警報手段７に提供する。

【００２７】（９）解析警報手段７は依存保持手段３の情報を常に解析し、ウィルス検査の必要がある場合にはシステム管理者に検査すべきクライアントやプログラムとともに警報する。また、システム管理者からのウィルス検査の結果に基づき、ウィルス感染の危険性が無くもはや必要なくなったバージョンやウィルスに感染していたファイルのバージョンをバージョン管理手段４に伝える。

【００２８】（１０）バージョン管理手段４は、もはや必要が無くなったバージョンの削減と、ウィルスに感染したファイルの古いバージョンへの復元を行う。また、登録されているそのファイルに関する情報を更新する。

【００２９】ここで、応用プログラム１１がウィルスに感染しており、ファイルＡ，Ｂ，Ｃへこのウィルスが伝染する場合を考える。応用プログラムを実行するとウィルスプログラムが動き出す。ウィルスプログラムはサーバ内のファイルＡ，Ｂ，Ｃへ感染を試み、図２に示す順序でファイル名とアクセス種別を指定してアクセス要求を要求送信手段１２に伝える。要求送信手段はこれらアクセス要求からサーバ１へのファイル要求を作成して送る。

【００３０】サーバ１側では、これらの要求を要求受信手段２が受け取り、そのファイル名、アクセス種別、クライアント名をバージョン管理手段４に伝える。バージョン管理手段４は、これらのファイルをバージョン管理されるものとして登録し、そのアクセス形態（参照、挿入、更新、削除）を決定する（図３参照）。さらに、アクセス形態が更新の場合には、その更新要求を、既存のファイル内容を失わないように新しいバージョンファイルを作る、削除の場合には何もしない、といったアクセス要求に変えてファイルアクセス手段５に伝える。ファイルアクセス手段５は、これらのアクセスをファイル保持手段６内のファイルに対して行い、その結果をバージョン管理手段４、要求受信手段２、要求送信手段１２を経由して応用プログラム１１に伝える。よって、クライアントからのアクセス要求の処理後には、図４に示すように、ファイルＡ，Ｂ，Ｃはそれぞれバージョンa_1，b_1，c_1からウィルスに感染したバージョンa_2，b_2，c_2に変わる。しかし、バージョンa_0，b_1，c_1のファイルも変更が加えられずに残っているので、あとでバージョンa_1，b_1，c_1に戻すことで復元が可能である。

【００３１】また、バージョン管理手段４は、決定したアクセス形態、ファイル名、クライアント名を依存保持手段３に伝える。依存保持手段３は、これまでに保持していたクライアント間とファイル間の依存関係にこれらの情報を追加する。よって、クライアント１０におけるファイル間の依存関係の情報として図３が追加され、クライアント１０についてのファイル依存関係の情報は図５のようになる。

【００３２】解析警報手段７はこの依存関係の情報を常に解析し、ウィルス存在の可能性を示すアクセスパターンがないか調べる。もし、該当するアクセスパターンが発見されウィルス検査の必要がある場合には、システム管理者に検査すべきクライアントやファイルを指定して警報する。よって、この例の場合、図５に示すように３つものファイル更新が短時間に連続して行われていることが発見され、ウィルス検査すべきクライアントとしてクライアント１０、検査すべきファイルとしてＰ（Ｐは応用プログラム１１が格納されているファイル），Ａ，Ｂ，Ｃがシステム管理者へ警報される。システム管理者は、ウィルスチェッカなどの手段で、ウィルスの存在を調べ、ファイルＰ，Ａ，Ｂ，Ｃ内のウィルスを発見する。この結果をシステム管理者は解析警報手段７に入力する。解析警報手段７は、ウィルスに感染していた場合にはそのファイルの新しいバージョンを、感染していなかった場合には古いバージョンを不要なバージョンとしてバージョン管理手段４に伝え、不必要なバージョンを削除する。よって、この例ではファイルＰ，Ａ，Ｂ，Ｃの新しいバージョン即ち、ファイルＰのp_1、ファイルＡ，Ｂ，Ｃのa_2，b_2，c_2がそれぞれ削除され、ウィルスが感染する前のファイルバージョンに復元される。

【００３３】なお、これらのウィルスに感染したバージョンp_1，a_2，b_2，c_2のどれかを他のクライアントが参照していた場合、依存保持手段３には、図６のようなクライアント間の依存関係が保持されている。よって、解析警報手段７は、ウィルス感染したバージョンp_1を参照しているクライアントＢとその参照の後で更新されたファイルＤを検査するようにシステム管理者にさらに警報する。このシステム管理者とのウィルス検査のやりとりはウィルス感染の可能性があるバージョンがなくなるまで続けられる。

【００３４】次に、従来のアクセス権方式で問題となった場合を考える。本方式では、バージョン管理手段４により既存のファイルを古いバージョンとして残し、システム管理者によりウィルスに感染していないことが確認された場合だけそれを削除しているので、ウィルスに感染したことが判った時はいつでもウィルス感染の影響を取り除ける。よって、第１の場合でもウィルス感染を防げる。また、依存保持手段３ではファイル間の依存関係とクライアント間依存関係を保持しており、解析警報手段７は第２の場合でも第３の場合でもその感染の可能性のあるファイルを追跡でき、システム管理者の検査によりウィルス感染の影響を取り除ける。さらに、本方式はアクセス権を全く使っていないので、第４の場合でも防御できる。最後に、本方式では、依存保持手段３の依存情報を元に感染の可能性のあるクライアントやファイルを求めるとともに、解析警報手段７が常に依存情報を解析してウィルスの感染が大きく広がる前にシステム管理者に警報するので、ウィルスチェッカなどでウィルスが存在しないことや感染が及んでいる範囲を確認するにも少数のファイルやクライアントを検査するだけでよい。最後に、本実施例を構成する各要素の具体例について説明する。

【００３５】要求送信手段１２と要求受信手段２は、クライアント／サーバシステムに必須の構成要素である。ＵＮＩＸのクライアント／サーバ型ファイルシステムＮＦＳを例にとると、要求送信手段１２はＮＦＳクライアントとその通信プログラム（ＸＤＲ，ＲＰＣ，ＵＤＰ，ＩＰ）に相当し、要求受信手段２はＮＦＳサーバとその通信プログラムに相当し、その仕様はＴＣＰ／ＩＰの標準化組織（ＩＡＢ：Ｉｎｔｅｒｎｅｔ　Ａｃｔｉｖｉｔｅｓ　Ｂｏａｒｄ）からＲＦＣとして公開されている。よって、当業者にとっては自明であるのでこれ以上の説明はここでは省略する。

【００３６】ファイル保持手段６は磁気ディスク、又は光ディスク、又は半導体メモリ等からなる不揮発性の記憶媒体で、多数のファイルを保持し、その読み出し／書き込みが可能なものである。一般には、磁気ディスク装置が用いられる。

【００３７】ファイルアクセス手段５は、ファイル名（またはその識別子）からファイル保持手段内におけるそのファイルの格納位置を求め、アクセス要求に従ってファイルの読み出し／書き込みを行うものである。ＵＮＩＸを例にとると、ファイルシステムがこれに相当し、その詳細は文献「ＵＮＩＸカーネルの設計：共立出版、Ｍａｕｒｉｃｅ　Ｊ　Ｂａｃｈ著）に述べられている。

【００３８】バージョン管理手段４は、ファイルへの変更（挿入、削除、更新）をファイル保持手段６内のファイルを損なうことなく実現するため、バージョン（版）を作成してその情報を管理する。バージョン情報を管理するバージョン情報テーブルは、図７に示すように、ファイル名フィールドと、複数のバージョンフィールドとから構成されており、ユーザから見えるファイルのファイル名と、その各バージョンの実際のファイル名の対応関係を保持している。バージョンフィールドの一番左のファイル名がもっとも古いバージョンを意味し、もっとも右が最新のバージョンを意味する。また、（ｄｅｌｅｔｅｄ）は削除されたことを意味する。よって、図７においてＤＥＦファイルは削除されていることを表わす。このテーブルを使ったバージョン管理手段４の概略フローを図８に示す。

【００３９】すなわち、バージョン管理手段４は、まず、指定されたファイル名のファイルがすでに管理されているか調べる（ステップＳ１）。もし、管理されていなければ、ファイル保持手段６に存在するかをファイルアクセス手段５に問い合わせた後、管理対象ファイルとして登録する（ステップＳ２）。次に、バージョン管理手段４は、登録された情報に従って、指定されたファイルにアクセス種別のアクセスが可能か調べ、そのアクセス形態（読み出し、挿入、更新、削除）を決定する（ステップＳ３，Ｓ４）。なお、アクセス不可能な場合には、エラーとなり処理が中断される。

【００４０】さらに、バージョン管理手段４は、確定したアクセス形態、ファイル名、クライアント名を依存保持手段３に伝えるとともに、既存ファイルの内容を損なわないよう、バージョンを作るファイル要求に変えてファイルアクセス手段５に伝える（ステップＳ５）。例えば、アクセス形態が参照の場合には、図７のテーブルのフィールドがサーチされ、最も右側の最新のバージョーンが見つけられ、それがファイルアクセス手段５によって読み出される（ステップＳ６）。アクセス形態が挿入の場合には、新しいバーション名のファイルが作成されてファイル保持手段６に登録されると共に、そのバーション名がバージョンフィールドの先頭に書き込まれる（ステップＳ７）。アクセス形態が削除の場合には、フィールドの最右端に（ｄｅｌｅｔｅｄ）が書き込まれる（ステップＳ８）。アクセス形態が更新の場合には、新しいバージョン名のファイルが作成され、それがファイル保持手段６に登録されると共に、新しいバージョン名がフィールドの最右端に書き込まれる（ステップＳ９）。なお、ウィルス検査の結果、もはや不要になったバージョンはテーブルからその名前が削除されるときに、ファイル保持手段６からも実際に削除される。

【００４１】依存保持手段３は、バージョン管理手段５からのアクセス形態、ファイル名、クライアント名に基づいて、クライアント間とファイル間の依存関係を作成保持する。依存関係を保持する依存関係テーブルは、図９に示すように、各クライアント毎に存在し、ファイル名、アクセス形態、ポインタフィールドから構成されている。各テーブルはそのクライアントで実行されたファイルアクセスの内、ウィルス感染の範囲を限定するのに

必要なものが格納されている。図１０に依存保持手段３の依存関係テーブルを作成するフローを示す。なお、ウィルス検査の結果、もはや不要になった情報は削除される。

【００４２】すなわち、まず、アクセス形態が参照か否かが判断され（ステップＳ１１）、参照以外の場合には、クライアント名で指定されたテーブルにファイル名と対応するアクセス形態が追加される（ステップＳ１２）。一方、アクセス形態が参照の場合には、指定されたファイルの更新，挿入がテーブルにすでにあるか否かが判断され（ステップＳ１３）、存在する場合には、クライアント名で指定されたテーブルに、ファイル名とアクセス形態が追加されると共に、ステップＳ１３で見つけた更新、挿入のエントリ間がポインタで結合される（ステップＳ１４）。解析警報手段７は次の４つを処理する。

（１）依存保持手段３の情報を常に解析し、次に示すようなウィルス検査が必要なアクセスパターンがあるか調べる。

パターンＡ；　ウィルス動作の可能性を示すアクセスパターン

ａ．サーバ全体で予め定められた頻度以上のファイル変更が行われる。

ｂ．１つのクライアントで予め定められた頻度以上のファイル変更が行われる。

ｃ．予め定められた名前のファイルが変更される。

パターンＢ；　ウィルスの存在が判明したとき、その影響が広範囲に及ぶ可能性があるアクセスパターン

ａ．１つのクライアントで累積のファイル変更が予め定めた回数を越える。

ｂ．１つの変更されたファイルで累積の参照が予め定めた回数を越える。

（２）ウィルス検査が必要な場合、ウィルス検査するクライアントとファイルを次のようにして求めて、システム管理者に警報する。

【００４３】パターンＡの場合は、アクセスパターン内で変更されているファイル、アクセスパターンの前で参照されたファイル、アクセスパターンが発見されたクライアントを検査対象にする。パターンＢの場合は、アクセスパターンが発見されたクライアント、又はアクセスパターンに含まれるファイルを検査対象にする。

（３）システム管理者が行ったウィルス検査の結果に基づいて、次のようにしてもはや不要になったバージョンを求めてバージョン管理手段に伝える。

・ウィルスに感染したいないことがわかった場合、そのバージョンより前のものを不要なバージョンとする。

・ウィルスに感染していることがわかった場合、そのバージョンより後のものを不要なバージョンとする。

【００４４】（４）システム管理者が行ったウィルス検査の結果に基づいて、次のようにしてさらにウィルス検査が必要なファイルやクライアントを求めシステム管理者に警報する。

・ウィルスが発見されたファイルを参照したクライアント。

・ウィルスが発見されたクライアントで変更されたファイル。

【００４５】なお、この実施例では、ウィルスに感染した可能性を検出する機能と感染していた場合に元に戻す機能が提供されているだけである。よって、感染したか否かの判断は、システム管理者等がウィルスチェッカ等の別の手段を使うことを想定していた。これではシステム管理者の仕事が大変になってしまうので、次に、ウィルスに感染したか否かを判断する感染判断機能について述べる。

【００４６】ウィルス感染をサーバで検出するには、まず、クライアント又はサーバにウィルスが何時でも感染させられるような被感染ファイルを用意し、クライアント側にある被感染ファイルの内容をウィルスが感染する前にサーバ側にコピーを取って置く。そして、サーバ側にある被感染ファイルに対してはそのファイルが更新アクセスが起きないか、クライアント側にある被感染ファイルに対しては定期的にサーバ内にあるコピーと比較して変更された形跡がないかを調査する。もし、更新アクセスや更新された形跡があれば、その更新アクセスに出したクライアントやその更新された形跡のあるクライアントはウィルスに感染している事を意味し、ウィルス感染を検出できる。以下、図１１を参照して具体的に説明する。図１１はウィルス検出のためのシステム構成である。

【００４７】クライアントには、誰でもアクセスできる被感染ファイルＢが存在する。サーバには、誰でもアクセス出来る被感染ファイルＡと、その被感染ファイルＡに対する更新を監視するファイル更新監視手段と、クライアントにある被感染ファイルＢのコピーファイルと、定期的にクライアント内の被感染ファイルＢの内容とサーバ内の被感染ファイルＢのコピーファイルの内容が一致するか比較監視するファイル比較監視手段が存在する。ここで、クライアントでウィルスプログラムが起動された場合を考える。

【００４８】（１）まず、ウィルスプログラムは、自分自身を感染させられるファイルを探す。ここで、ウィルスプログラムは、サーバ内の誰でもアクセス出来る被感染ファイルＡやクライアント内の誰でもアクセス出来る被感染ファイルＢを発見する。

（２）次に、ウィルスプログラムは、発見したファイルに対してウィルス感染を試み、ファイルの更新を行なう。よって、被感染ファイルＡ，Ｂは更新され、ウィルスが感染してしまう。ここで、サーバ内にある被感染ファイルＡに対する更新は、サーバ内のファイル更新監視手段により検出され、ファイル更新を要求したクライア

ントにウィルスプログラムが存在する事が判る。

【００４９】（３）サーバ内のファイル比較監視手段は、定期的にクライアント内の被感染ファイルとサーバ内のそのコピーを比較し、一致しているか検査する。ここで、クライアント内の被感染ファイルＢは、ウィルスプログラムによりステップ２で更新されているので、一致しない。よって、被感染ファイルＢがあるクライアントにウィルスプログラムが存在することが判る。

【００５０】以上のように、クライアントにウィルスプログラムが存在すれば、サーバ内の被感染ファイルＡかクライアント内の被感染ファイルＢのどちらかが変更され、ファイル更新監視手段かファイル比較監視手段によりウィルスの存在が検出される。

【００５１】なお、被感染ファイルＡ，Ｂは、テキストファイルに限定するものではなく、コマンドファイル、ライブラリファイル、実行ファイル、ＯＳ自身を格納するファイルなど計算機に格納されるあらゆる種類のファイルを用いることができる。以上のように、この実施例においては、バージョン管理手段４によるバージョン管理によって、既存のファイルを損なうことなくファイル変更が行なわれる。よって、いつでもウィルス感染等によって変更が加えられたバージョンを削除し、前の状態に戻すことができる。また、依存保持手段３が管理しているクライアント間のファイルの依存関係およびファイルアクセスの履歴を用いることにより、ウィルス感染の可能性のあるファイルを必ずシステム管理者に警報できるとともに、その検査結果に基づいてバージョンをもとに戻しているので、ウィルスが感染したバージョンは必ず削除される。よって、ウィルス感染を完全に防御できる。

【００５２】さらに、解析警報手段７が常に情報を解析してウィルス存在の可能性をチェックしているので、ウィルス感染しても広範囲に広がる前に発見できる。また、クライアント間の依存関係の情報を使ってウィルス感染の可能性があるクライアントやファイルを限定できる。よって、サーバ内におけるウィルスの存在や存在するときの感染範囲を確認するのに少数のファイルやクライアントを検査するだけでよい。

【００５３】

【発明の効果】以上のように、本ウィルス防御機能ではサーバ内にウィルスが感染するのを完全に防げるばかりでなく、サーバ内にウィルスが存在しないことやウィルス感染の範囲を確認するのも少数のファイルを検査するだけでよい。よって非常に効率の良いウィルス対策を実現できる。

【図面の簡単な説明】

【図１】この発明の一実施例に係わるウィルス防御方式を実現するためのクライアント／サーバシステムの構成を示すブロック図。

【図２】同実施例のシステムにおけるクライアントからのアクセス要求の一例を示す図。

【図３】同実施例のシステムにおいてサーバで確認されるアクセス形態の一例を示す図。

【図４】同実施例のシステムにおいてサーバで管理されるバージョン関係の一例を示す図。

【図５】同実施例のシステムにおいてサーバで管理されるクライアントのファイル依存関係を示す図。

【図６】同実施例のシステムにおいてサーバで管理されるクライアント間の依存関係を示す図。

【図７】同実施例のシステムにおいてサーバで管理されるバージョン管理テーブルの一例を示す図。

【図８】同実施例のシステムにおいてサーバ内で実行されるバージョン管理動作の一例を示すフローチャート。

【図９】同実施例のシステムにおいてサーバで管理される依存関係テーブルの一例を示す図。

【図１０】同実施例のシステムにおいてサーバ内で実行される依存関係テーブル作成動作の一例を示すフローチャート。

【図１１】同実施例のシステムで採用されるウイルス検査方式の一例を説明するための図。

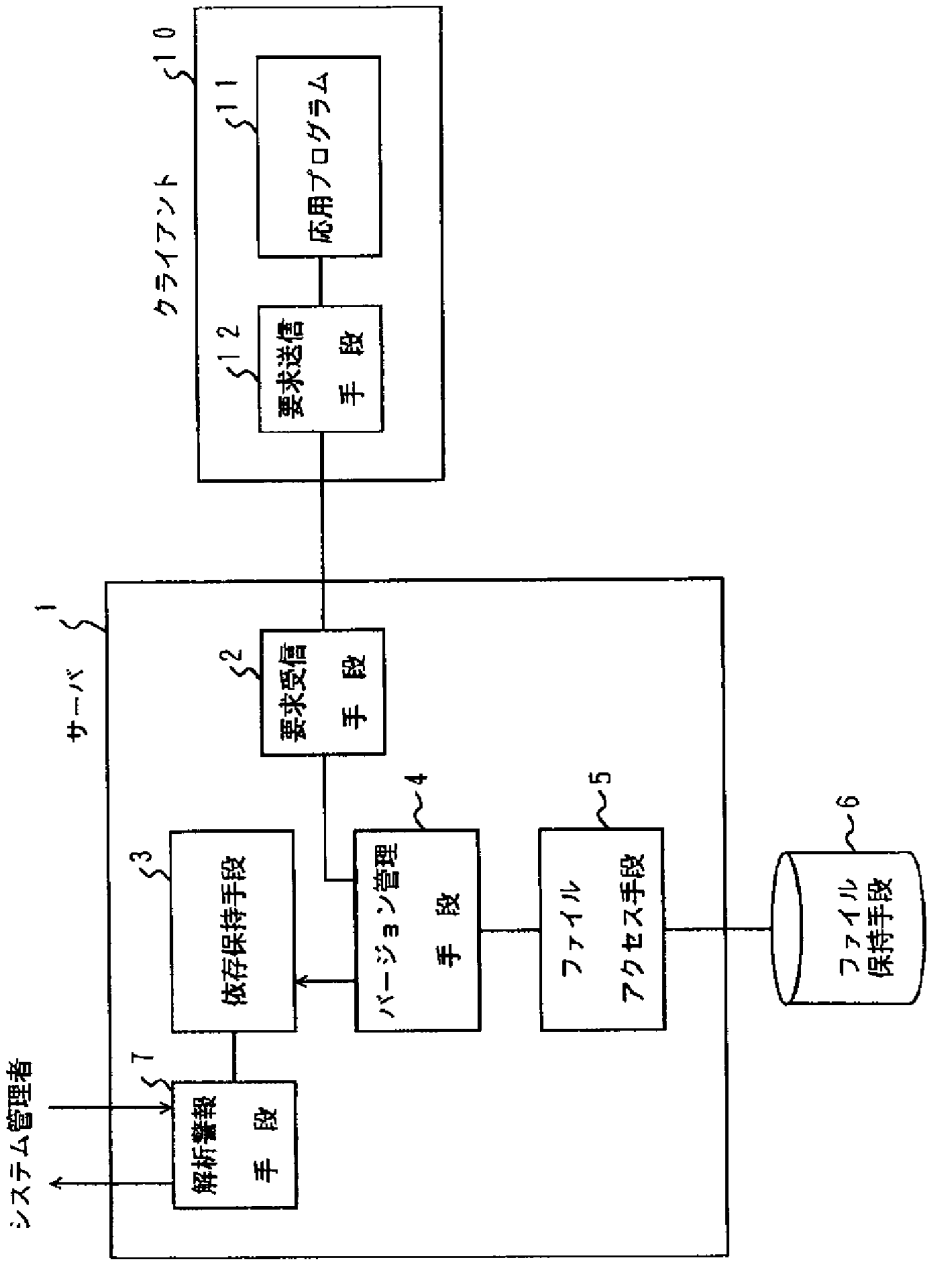
【図１２】従来のクライアント／サーバシステムを概念的に示すブロック図。

【図１３】従来のクライアント／サーバシステムにおけるファイルアクセス動作を説明するための図。

【符号の説明】

１…サーバ、２…要求受信手段、３…依存関係保持手段、４…バージョン管理手段、５…ファイルアクセス手段、６…ファイル保持手段、７…解析警報手段、１０…クライアント。

【図１】


クライアント
10
11
応用プログラム
12
要求送信
手段
サーバ
1
2
要求受信
手段
3
依存保持手段
4
バージョン管理
手段
5
ファイル
アクセス手段
6
ファイル
保持手段
システム管理者
7
解析警報
手段

【図２】

ファイル名	アクセス種別
ファイルA	読み出し
ファイルA	書き込み
ファイルB	読み出し
ファイルB	書き込み
ファイルC	読み出し
ファイルC	書き込み

クライアントのアクセス要求

【図３】

ファイル名	アクセス形態
ファイルA	参照
ファイルA	更新
ファイルB	参照
ファイルB	更新
ファイルC	参照
ファイルC	更新

アクセス形態

【図４】

ファイル名	バージョン
P	p_1
A	$a_1 \rightarrow a_2$
B	$b_1 \rightarrow b_2$
C	$c_1 \rightarrow c_2$

バージョン関係

【図５】

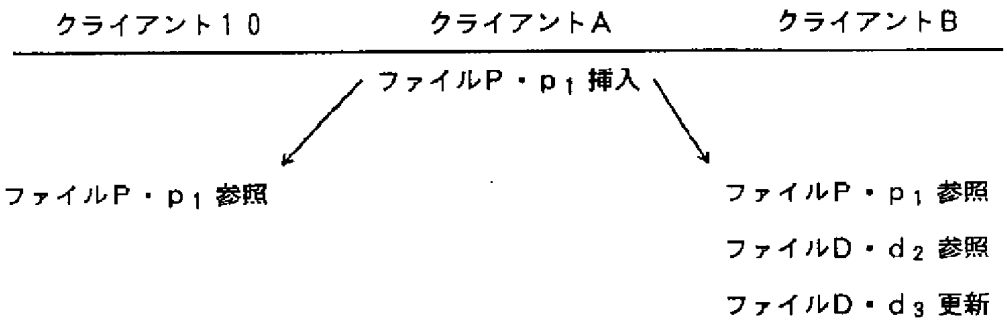
ファイル名	バージョン	アクセス形態
P	p_1	参照
A	a_1	参照
A	a_2	更新
B	b_1	参照
B	b_2	更新
C	c_1	参照
C	c_2	更新

クライアント１０のファイル依存関係

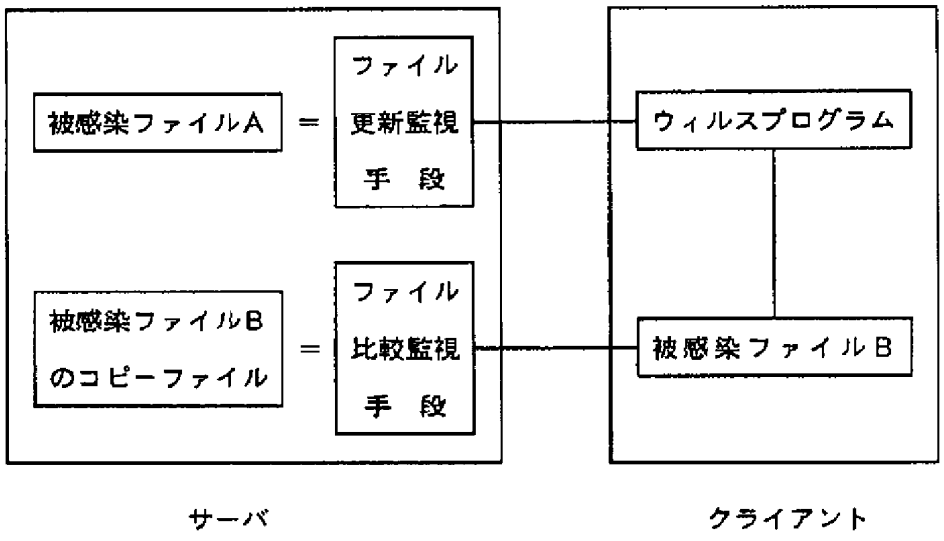
【図７】

ファイル名	バージョン1	バージョン2	バージョン3	バージョン4	バージョン5 ……
ABC	ABC	ABC. 1	ABC. 2	ABC. 3	
DEF	DEF	DEF. 1	(deleted)		
GHI	GHI. 3	GHI. 4			
KLM	(deleted)				

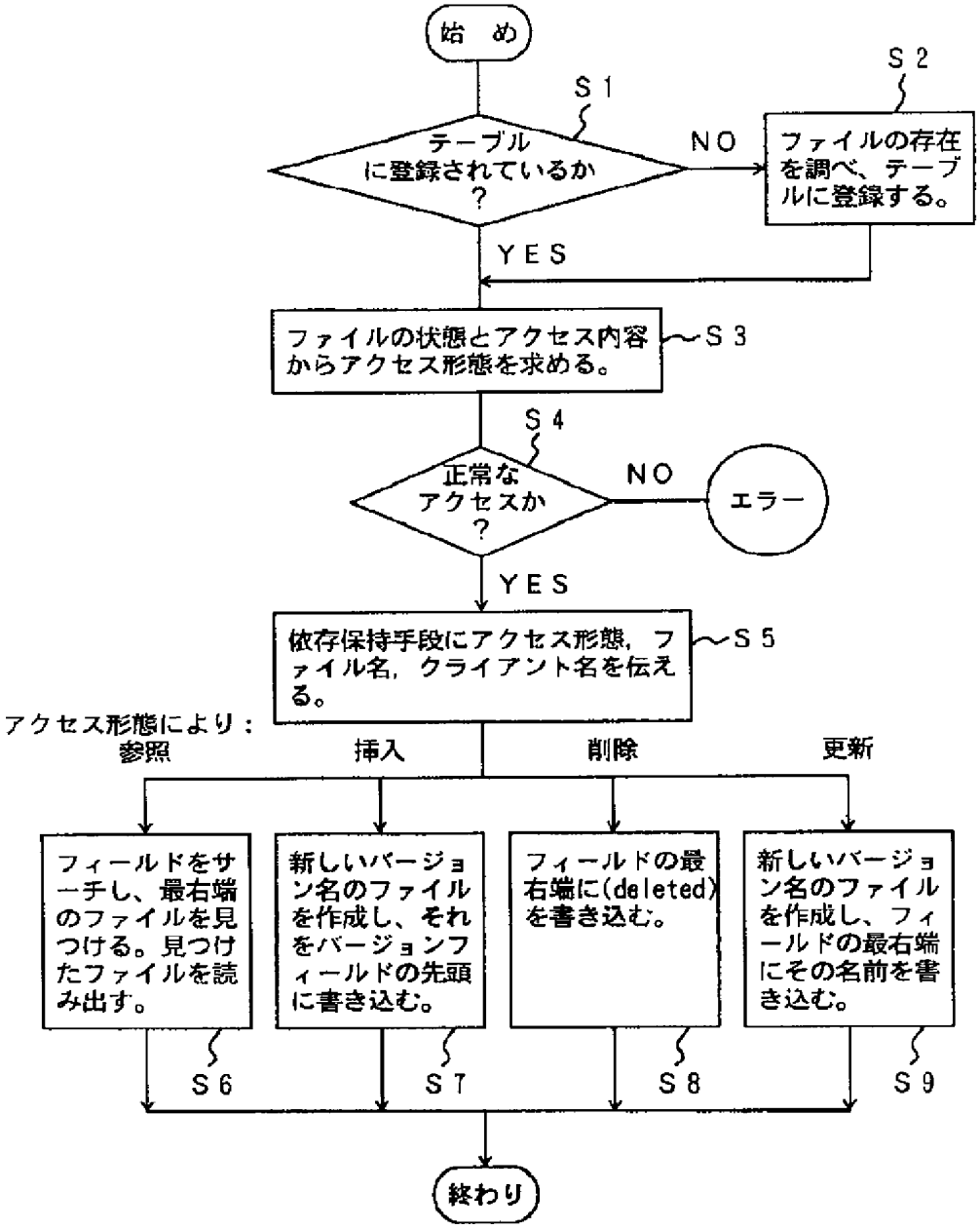
【図６】


クライアント１０
クライアントA
クライアントB
ファイルP・p1 挿入
ファイルP・p1 参照
ファイルP・p1 参照
ファイルD・d2 参照
ファイルD・d3 更新

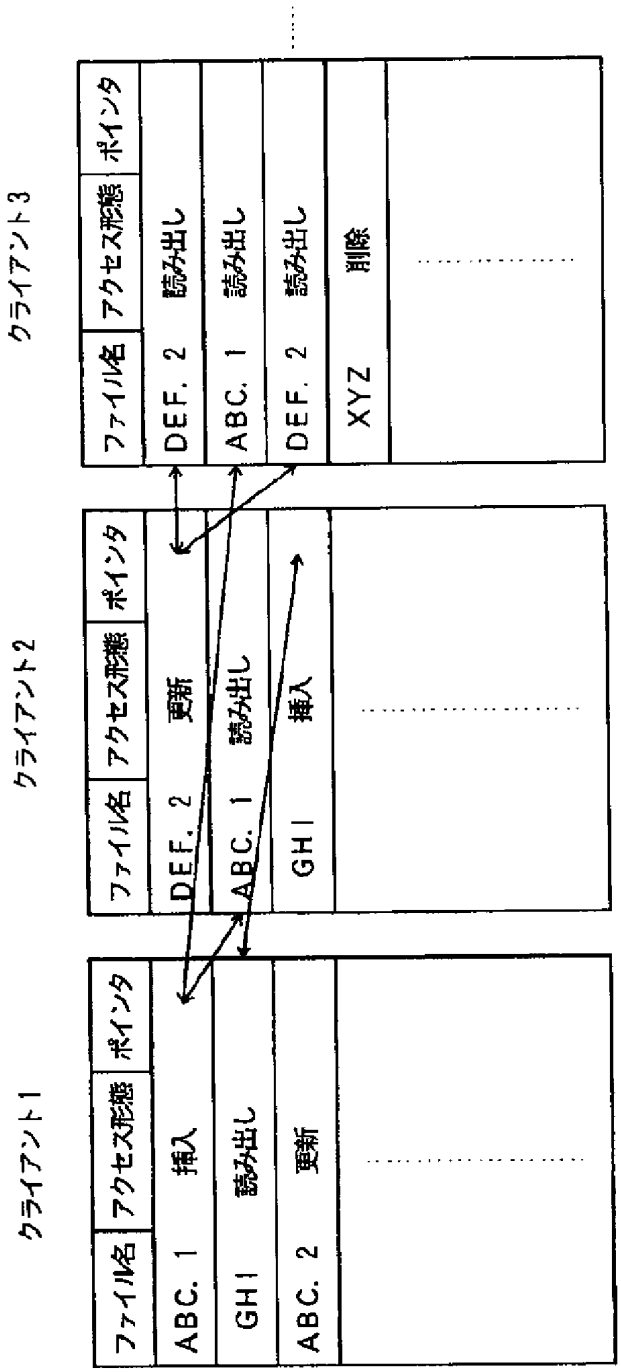
【図１１】


被感染ファイルA
＝
ファイル
更新監視
手　段
被感染ファイルB
のコピーファイル
＝
ファイル
比較監視
手　段
ウィルスプログラム
被感染ファイルB
サーバ
クライアント

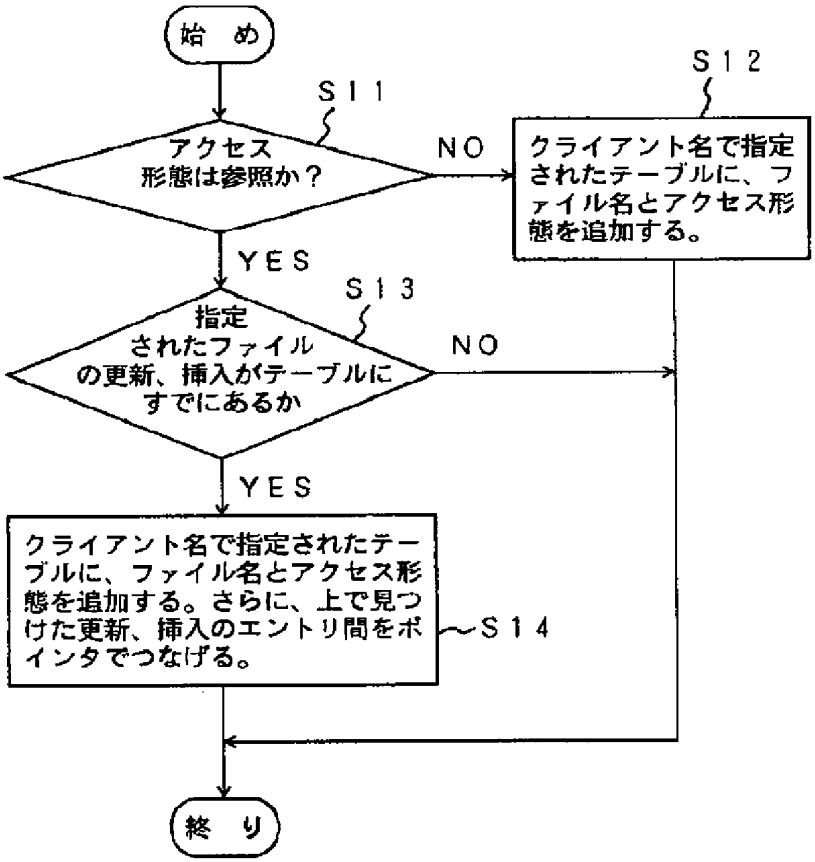
【図８】


始　め
S１
テーブルに登録されているか？
NO
S２
ファイルの存在を調べ、テーブルに登録する。
YES
ファイルの状態とアクセス内容からアクセス形態を求める。
S３
S４
正常なアクセスか？
NO
エラー
YES
依存保持手段にアクセス形態，ファイル名，クライアント名を伝える。
S５
アクセス形態により：
参照
挿入
削除
更新
フィールドをサーチし、最右端のファイルを見つける。見つけたファイルを読み出す。
新しいバージョン名のファイルを作成し、それをバージョンフィールドの先頭に書き込む。
フィールドの最右端に(deleted)を書き込む。
新しいバージョン名のファイルを作成し、フィールドの最右端にその名前を書き込む。
S６
S７
S８
S９
終わり

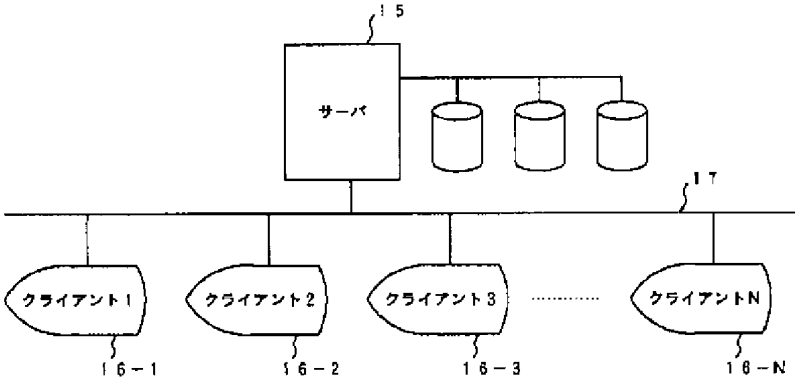
【図9】


クライアント1
ファイル名
アクセス形態
ポインタ
ABC. 1　挿入
GHI　読み出し
ABC. 2　更新
クライアント2
ファイル名
アクセス形態
ポインタ
DEF. 2　更新
ABC. 1　読み出し
GHI　挿入
クライアント3
ファイル名
アクセス形態
ポインタ
DEF. 2　読み出し
ABC. 1　読み出し
DEF. 2　読み出し
XYZ　削除

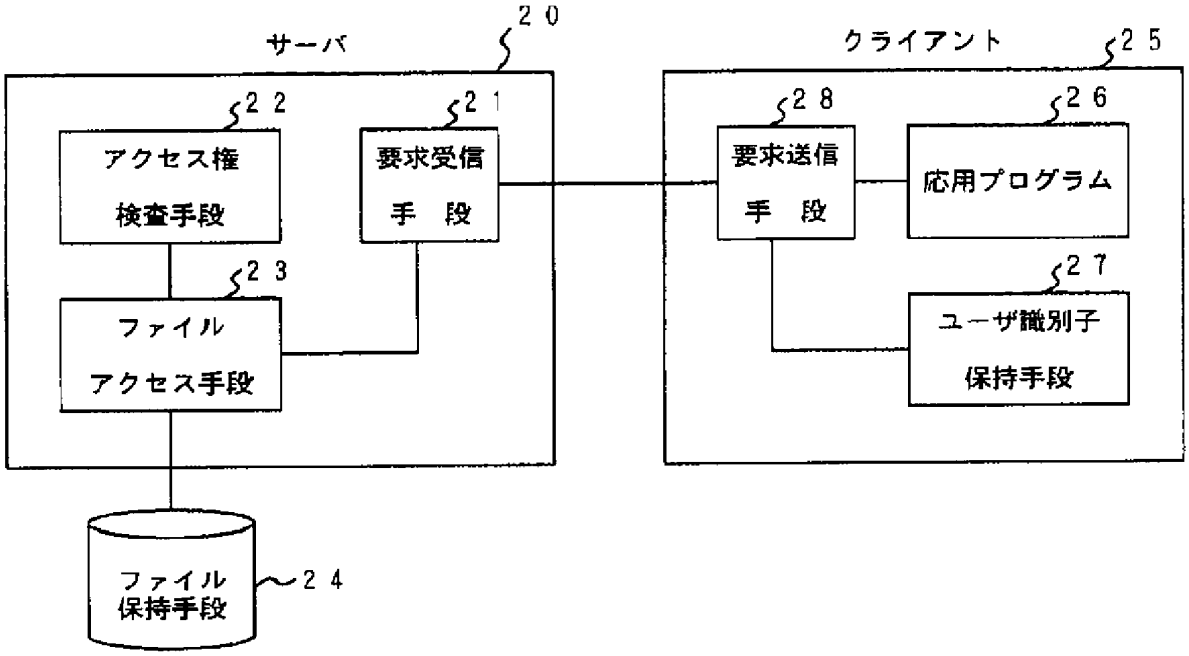
【図１０】


始　め
S１１
アクセス
形態は参照か？
NO
S１２
クライアント名で指定されたテーブルに、ファイル名とアクセス形態を追加する。
YES
S１３
指定
されたファイル
の更新、挿入がテーブルに
すでにあるか
NO
YES
クライアント名で指定されたテーブルに、ファイル名とアクセス形態を追加する。さらに、上で見つけた更新、挿入のエントリ間をポインタでつなげる。
S１４
終　り

【図１２】


１５
サーバ
１７
クライアント１
クライアント２
クライアント３
クライアントN
１６－１
１６－２
１６－３
１６－N

【図１３】


サーバ
２０
２２
アクセス権
検査手段
２１
要求受信
手　段
２３
ファイル
アクセス手段
ファイル
保持手段
２４
クライアント
２５
２８
要求送信
手　段
２６
応用プログラム
２７
ユーザ識別子
保持手段

拒絶理由通知書

特許出願の番号	特願２００８－５１８９７５
起案日	平成２３年　９月１３日
特許庁審査官	後藤　彰　　４２２６　５Ｓ００
特許出願人代理人	江上　達夫（外　１名）　様
適用条文	第２９条第１項、第２９条第２項、第３７条

この出願は、次の理由によって拒絶をすべきものです。これについて意見がありましたら、この通知書の発送の日から３か月以内に意見書を提出してください。

理　由

Ａ．この出願は、下記の点で特許法第３７条に規定する要件を満たしていない。

記

請求項１に係る発明は、下記理由Ｂに記載したように、引用文献１に記載された発明であるから、請求項１に係る発明は特別な技術的特徴を有しない。

よって、請求項１に係る発明と、請求項２－４７に係る発明との間に、同一の又は対応する特別な技術的特徴を見いだすことができない。

ただし、請求項２－１２、１３－２４、２５－３０係る発明については、発明の単一性の要件を問わないこととする（「特許・実用新案　審査基準」第Ｉ部第２章４．２を参照）。

以上のように、請求項１に係る発明と、請求項３１－３３、３４－３６、３７－３９、４０、４１、４２－４７に係る発明とは、発明の単一性の要件を満たす一群の発明に該当しないから、この出願は特許法第３７条に規定する要件を満たさない。

この出願は特許法第３７条の規定に違反しているので、請求項１－３０以外の請求項に係る発明については特許法第３７条以外の要件についての審査を行っていない。

Ｂ．この出願の下記の請求項に係る発明は、その出願前に日本国内又は外国において、頒布された下記の刊行物に記載された発明又は電気通信回線を通じて公衆に利用可能となった発明であるから、特許法第２９条第１項第３号に該当し、特許を受けることができない。

記　　（引用文献等については引用文献等一覧参照）

・請求項１，１３
・引用文献等１
・備考
引用文献１には、
ファイルサーバ（１）において、複数のファイルを記憶する複数のリモートコンピュータ（２ａ，２ｂ，２ｃ）の夫々からのファイルに関するデータを受信する工程（図２の１３）と、
前記複数のコンピュータから受信した前記ファイルに関する前記データを、前記ファイルサーバにおいて比較する工程（図２の１５）と、
前記比較に基づいて前記ファイルをマルウェアとして分類する工程（図２の１７）と
を備える分類方法が記載されていると認められる。
したがって、請求項１に係る発明は、引用文献１に記載された発明である。
請求項１３に係る発明も、同様の理由によって、引用文献１に記載された発明である。

・請求項２，１４
・引用文献等１
・備考
引用文献１記載の発明は、ファイルを受信しており、ファイルには、複数の実行可能命令が含まれている。

Ｃ．この出願の下記の請求項に係る発明は、その出願前に日本国内又は外国において、頒布された下記の刊行物に記載された発明又は電気通信回線を通じて公衆に利用可能となった発明に基いて、その出願前にその発明の属する技術の分野における通常の知識を有する者が容易に発明をすることができたものであるから、特許法第２９条第２項の規定により特許を受けることができない。

記　　（引用文献等については引用文献等一覧参照）

・請求項１，１３
・引用文献等１
・備考
引用文献１には、上記理由Ｂにおいて指摘したとおりの発明が記載されており、請求項１，１３に係る発明は、引用文献１に記載された発明に基いて、当業者が容易に発明できたものである。

・請求項２，１４
・引用文献等１，２
・備考
引用文献２には、特に［００４２］、［００４３］の記載に注意すると、中央インフラに送るファイルに関するデータとして、ファイル名、パス、変更日、タイプを採用することが記載されていると認められる。
引用文献２記載の発明における「中央インフラ」、「ファイル」は、それぞれ、請求項２に係る発明の「ベースコンピュータ」、「コンピュータオブジェクト」に相当するから、請求項２，１４に係る発明は、引用文献１及び２に記載された発明に基いて、当業者が容易に発明できたものである。

・請求項３－６，１５－１８
・引用文献等１，２，３
・備考
引用文献３には、特に【００３１】、【００３２】の記載に注意すると、複数のファイルの依存関係を保持しておき、ウイルスすなわちマルウェアが発見されると、依存関係にあるファイルは全て感染の疑いがあるとする技術が記載されていると認められる。
コンピュータは、通常、複数のオブジェクトを記憶しているから、必要に応じて、引用文献３記載の技術を採用することは、当業者が適宜に為し得る事項である。
その際、判定基準は適宜に決めることができるので、マルウェアの疑いがあると判定するか、マルウェアであると判定するかは、当業者が適宜に設計できる設計変更である。

・請求項７，１９，２５，３０
・引用文献等１，２，３
・備考
引用文献２には、特に［００５７］、［００５８］、図３のステップ２１２及び２１４の記載に注意すると、ハッシュ値すなわちハッシングプロセスによって取得される鍵を送信する技術が記載されていると認められる。
引用文献１記載の発明においても、ハッシングの一形態であるチェックサムを生成しているので、当業者ならば、引用文献２記載の技術を採用することは容易に想起し得た事項である。

・請求項８－１０，２０－２２，２６－２８
・引用文献等１，２，３
・備考
引用文献２記載の発明のファイルには、複数の実行可能命令が含まれており、

また、引用文献２の［００４２］、［００４３］には、ファイル名、パス、変更日、タイプなどを送信することが示されているので、請求項８－１０，２０－２２，２６－２８に係る発明は、引用文献１及び２に記載された発明に基いて、当業者が容易に発明できたものである。

・請求項１１，２３，２９
・引用文献等１，２，３，４
・備考
引用文献４には、ウイルス（マルウェア）検出のためにファイル（オブジェクト）の物理的サイズを比較する（第２図のＳ１１）技術が記載されていると認められる。
引用文献１又は２に記載の発明では、マルウェア検出に必要なデータをベースコンピュータに送っているので、当業者ならば、引用文献４記載の技術を採用し、オブジェクトの物理的サイズを送信することは、容易に想起し得た事項である。

・請求項１２，２４
・引用文献等１，２，３，４，５
・備考
引用文献５には、特に【００９９】、【０１００】、図７のステップ７２０－７３０の記載に注意すると、オブジェクトの振る舞いを監視することによって、マルウェアを分類する技術が記載されていると認められる。
引用文献１記載の発明も、マルウェアを分類する技術であるから、当業者ならば、引用文献５記載の技術を付加することは容易に想起し得た事項である。

引　用　文　献　等　一　覧

1.特表平９－５０４３９５号公報
2.欧州特許出願公開第１５４９０１２号明細書
3.特開平６－１１０７１８号公報
4.特開平３－２３３６２９号公報
5.特開２００３－１９６１１２号公報

先行技術文献調査結果の記録

・調査した分野　　ＩＰＣ　Ｇ０６Ｆ　２１／２２
ＤＢ名

・先行技術文献　　特表２００７－５２０７９６号公報
（引用文献２のファミリー）

この先行技術文献調査結果の記録は、拒絶理由を構成するものではありません。

－－

この拒絶理由通知の内容に関するお問い合わせ、または面接のご希望がございましたら下記までご連絡下さい。

特許審査第四部　情報処理（情報セキュリティ）　後藤　彰
ＴＥＬ．０３（３５８１）１１０１　内線３５４６
ＦＡＸ．０３（３５８０）６９０６



Japanese Patent Application No. 2008-518975

Your Ref:AF/JG/P10717JP-W, Our Ref:071842JP

Our Brief Comment

[Reason A]

The Examiner stated that this application does not comply with the requirements for Unity of Invention. Please let us have your comment and/or instructions.

For your reference, Article 37 of the Japanese Patent Law and its Examination guidelines can be referred at the following English website (Part I, Chapter 2) of Japanese Patent Office.

.....

http://www.jpo.go.jp/cgi/linke.cgi?url=/tetuzuki_e/t_tokkyo_e/1312-02_e.htm

.....

[Reasons B and C]

The Examiner stated that the inventions disclosed in claims 1, 2, 13 and 14 are not new on the basis of D1.

The Examiner stated that the inventions disclosed in claims 1 to30 do not involve an inventive step on the basis of D1 to D5. Incidentally, the Examiner did NOT examine the novelty and inventiveness of the remaining claims 31 to 47.

Please let us have your comment and/or instructions.

Reference No: , Mailing No: 633227, Mailing date: September 20, 2011

Notification of Reason(s) for Rejection

(English Translation)

Patent application number	2008-518975
Drafting date	September 13, 2011
Examiner of the Patent Office	Akira GOTO 4226 5S00
Agent	Tatsuo EGAMI et al.
Applicable articles	Art 29(1), Art 29(2), Art 37

This application shall be rejected for the following reasons. If the applicant has any arguments against the reason, please file an argument within 3 months from the mailing date of this notification.

Reasons

[A] This application does not comply with the requirement prescribed in Article 37 of the Japanese Patent Law, in the following point.

[B] The invention according to the following claims of this application cannot be patentable under section 29, paragraph 1, Item 3 of Japanese Patent Law, since the invention is an invention that was described in a distributed publication, or that were made publicly available through an electric telecommunication line in Japan or a foreign country, prior to the filing of the patent application.

[C] The invention according to the following claims of this application cannot be patentable under section 29, paragraph 2 of

Japanese Patent Law, since the invention could be easily have been made, prior to the filing of the patent application, by a person with an ordinary skill in the art to which the invention pertains, on the basis of the inventions which were described in the following distributed publications or made available to the public through electric telecommunication lines in Japan or elsewhere prior to the filing of the patent application.

Note

(See the List of Reference about the cited references)

<Reason A>

The invention disclosed in claim 1 does not have a STF (Special Technical Feature), because it is an invention that was described in D1 as described in the below Reason B.

Therefore, there is no same or corresponding Special Technical Feature between the invention disclosed in claim 1 and the inventions disclosed in claims 2 to 47.

However, the Examiner does not require the Unity of Invention, with respect to claims 2 to 12, 13 to 24 and 25 to 30 (see Part I, Chapter 2, item 4.2 in the examination guidelines related to Article 37 of the Japanese Patent Law).

As described above, the present patent application does not meet the requirement of Article 37 of the Japanese Patent Law, because the invention disclosed in claim 1 and the inventions disclosed in claims 31 to 33, 34 to 36, 37 to 38, 39, 40 to 41, 42 to 46 and 47 do not belong to one invention group which meets the requirement of the Unity of Invention.

Since the present patent application does not meet the

requirement of Article 37 of the Japanese Patent Law, the Examiner did not examine whether the remaining claims 31 to 47 other than claims 1 to 30 met the requirement other than the requirement of Article 37 of the Japanese Patent Law.

<Reason B>

・Claims 1 and 13

・D1

D1 discloses a method comprising:

at a file server (1), receiving data about a file from each of a plurality of computers (2a, 2b, 2c) on which a plurality of files are stored (see step 13 in Fig. 2);

comparing in the file server the data about the file received from the plurality of computers (see step S15 in Fig. 2); and

classifying the file as malware on the basis of the comparison.

Therefore, the invention disclosed in claim 1 is an invention described in D1.

The invention disclosed in claim 13 is also an invention described in D1, for the same reason.

・Claims 2 and 14

・D1

The invention described in D1 receives the file. The file includes a plurality of executable instructions.

<Reason C>

・Claims 1 and 13

・D1

D1 discloses the invention that is indicated in the above <Reason B>. The inventions disclosed in claims 1 and 13 are easily

made by a person with an ordinary skill in the art to which the invention pertains on the basis of the invention described in D1.

・Claims 2 and 14

・D1 and D2

D2 discloses that the filename, the path, the modification date and the type are used as the data that is sent to the central infrastructure (paragraphs [0042] to [0043]).

The "central infrastructure" and the "file" of D2 correspond to the "base computer" and the "computer object" of the invention disclosed in claim 2, respectively. Thus, the inventions disclosed in claims 2 and 14 are easily made by a person with an ordinary skill in the art to which the invention pertains on the basis of the inventions described in D1 and D2.

・Claims 3 to 6 and 15 to 18

・D1 to D3

D3 discloses that (i) the dependency-relationship between the plurality of files is stored and that (ii) if the virus (malware) is detected in a certain file, all of the files that have the dependency-relationship with the certain file.

Because the computer usually stores the plurality of objects, a person with an ordinary skill in the art to which the invention pertains could easily use the technique described in D3 as occasion demand.

At this time, because the judgment standard could be freely determined, a person with an ordinary skill in the art to which the invention pertains could easily design whether it is judged that the file may be the malware or it is judged that the file is the malware as occasion demand.

・Claims 7, 19, 25 and 30

・D1 to D3

D2 discloses that the hash value (i.e. the key obtained by the hashing process) is transferred (paragraphs [0057] to [0058] and steps 212 and 214 in Fig. 3).

Because the invention described in D1 generates the check-sum which is one aspect of the hashing, a person with an ordinary skill in the art to which the invention pertains could easily use the technique described in D2.

・Claims 8 to 10, 20 to 22 and 26 to 28

・D1 to D3

The file described in D2 includes a plurality of executable instructions. D2 discloses that the filename, the path, the modification date and the type are transferred (paragraphs [0042] to [0043]). Thus, the inventions disclosed in claims 8 to 10, 20 to 22 and 26 to 28 are easily made by a person with an ordinary skill in the art to which the invention pertains on the basis of the inventions described in D1 and D2.

・Claims 11, 23 and 29

・D1 to D4

D4 discloses that the physical sizes of the files (object) are compared in order to detect the virus /malware (step S11 in Fig. 2). Because the invention described in D1 or D2 transfers data, that is necessary to detect the malware, to the base computer, a person with an ordinary skill in the art to which the invention pertains could easily combine the transferring of the physical size of the object by using the technique described in D4.

・Claims 12 and 24

・D1 to D5

D5 discloses that the malware is classified by monitoring the behavior of the object (paragraphs [0099] and [0100] and steps 720 to 730 in Fig. 7). Because the invention described in D1 is a technique for classifying the malware, a person with an ordinary skill in the art to which the invention pertains could easily add the technique described in D5.

List of Reference

D1: JP H09-504395 A (JP 1997-504395 A)
D2: EP 1 549 012 A
D3: JP H06-110718 A (JP 1994-110718 A)
D4: JP H03-233629 A (JP 1991-233629 A)
D5: JP2003-196112 A

Prior Art Search Report

・Field searched IPC G06F 21/22
DB name
・Prior art JP 2007-520796 A (family application of D2)

This prior art search report does not constitute the reason(s) for rejection.

If you have any inquiries about the content of the Notification of Reason(s) for Rejection or the requirement for the interview, please contact the following address.

4th department of the patent examination
Information Processing (Information Security)
Examiner Akira GOTO
TEL. 03(3581)1101 ext.3546 FAX. 03(3501)6906

(End)

PATENT ABSTRACTS OF JAPAN

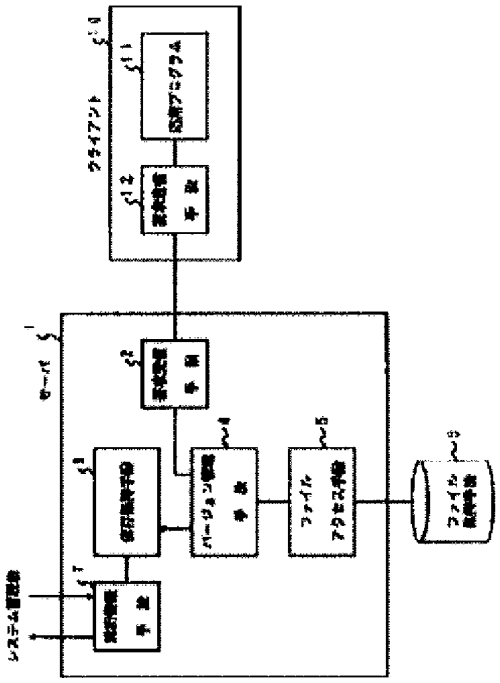
(11)Publication number : **06-110718**
(43)Date of publication of application : **22.04.1994**

(51)Int.Cl. G06F 11/00
G06F 9/06
G06F 12/00

(21)Application number : **04-261336**
(22)Date of filing : **30.09.1992**

(71)Applicant : **TOSHIBA CORP**
(72)Inventor : **SEKIDO KAZUNORI**

(54) VIRUS PROTECTION SYSTEM



(57)Abstract:
PURPOSE: To realize a virus protection system estimating the range of virus infection and restoring an infected program.
CONSTITUTION: A file change is performed without damaging an existing file by the version control with a version control means 4. Thus, the version for which a change is added due to a virus infection, etc., is deleted any time and the previous state can be restored. By using the dependence of files between clients controlled by a dependence holding means 3 and the history of the file access, a file having the possibility of the virus infection can be alarmed to a system controller without fail. The version infected with virus is deleted without fail since the version is restored based on the inspection result. Thus, virus infection can be completely protected.

* NOTICES *

JPO and INPIT are not responsible for any
damages caused by the use of this translation.

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Industrial Application]This invention relates to the virus defense system of the server in a client/server system.
[0002]
[Description of the Prior Art]In recent years, the systems which share a file between a client/server system are increasing in number with the low price by the highly efficient appearance of a personal computer (PC) workstation (WS) and progress of network technology. This system is the composition of having connected the mainframe (at least one is good) (server) 15 of a small number holding a lot of file with many minicomputers (client) 16-1 only with a file required to operate independently - N in the network 17, as shown in drawing 12. Central control can be carried out to one place, and even if a client changes, it is not necessary to copy a file and to move the file currently overlapped and held by the computer of former each with this system. It is not necessary to record a large-sized disk in a client, and an inexpensive and compact computer can be used for it.
[0003]The damage caused by the program which causes an obstacle to the computer called a virus centering on PC is reported [that it is various and] these days. If this program becomes the stage set up beforehand, it eliminates the file in a computer, or destroys the fundamental program of OS, and it has a function transmitted to other programs via a program, OS, etc. If the program infected with the virus is executed, in order to infect a virus with other programs first, A virus program investigates accessible all the files / directory from the computer, and finds the directory which can store the program which can be infected for a virus, and the program which carried out the viral infection. After beginning to read the program which can be infected one by one and incorporating a virus program, it returns, or it writes in the directory which can store the program which carried out the viral infection. A virus program is included also in OS which controls the program execution of the computer, and infection is tried whenever OS is started. The program which checks existence of the virus called a virus checker is also developed.
[0004]Until now, the cases where a viral infection was carried out via media, such as a floppy disk, were the main courses, what is necessary is just to have taken care that the administrator of each computer did not execute the program in which an identity does not clarify, and the damage was also restricted to the computer which executed the virus program. However, in the system of a client/server system, all the user's files are stored in the server and access them from each client. Therefore, from a client, all the files in a server are accessible as a principle, execution of the program by one careless user which carried out the viral infection is infected with all the files in a

server, and there is a danger of bringing an obstacle further to all the clients. Therefore, a virus defending means is indispensable to a file server. However, there was only a method by the right to access in the conventional file server, and it was insufficient for it. [just] Hereafter, the right-to-access system taken by the conventional file server and its problem are explained briefly.

[0005]The composition of the file server 20 and the client 25 in a client/server system is shown in drawing 13. The request reception means 21 from which the file server 20 receives the file request from a client, It comprises the right-to-access inspection means 22 which inspects the right to access to a file, the file access means 23 which performs a file and the read-out writing of the access attribute, and the file holding mechanism 24 which actually holds many files. The application program 26 which, on the other hand, needs the file which the client 25 has in a server, It comprises the user-identification child holding mechanism 27 holding the identifier of the user who is using the present client 25, and the demand transmitting means 28 which sends out a demand to a server according to directions of the application program 26. The identifier within the user-identification child holding mechanism 27 is set up when the client 25 performs logical connection (login and mount) to the server 20. In logical connection, password verification etc. are conducted and it is checking that he is a regular user.

[0006]Operation of this system is explained briefly. First, the application program 26 of a client specifies access classification (read-out/writing) a file name front, and tells an access request to the demand transmitting means 28. The demand transmitting means 28 which received this demand reads a user-identification child from the user-identification child holding mechanism 27, creates the file request which consists of a file name, access classification, and a user-identification child, and sends it to the server 20.

[0007]In the server 20 side, the request reception means 21 receives this demand, and a user-identification child's user tells an access request to the file access means 23 as what accesses access classification to the file of a file name according to this request content. The file access means 23 which received this demand takes out the access attribute of the file specified first from the file holding mechanism 24, and sends it to the right-to-access inspection means 22 with access classification and a user-identification child. In the right-to-access inspection means 22, the stored access attribute is analyzed and this file inspects whether access of access classification is allowed to a user-identification child's user. Only when this result is told and permitted to a file access means, access of access classification is performed to the file of the file name within the file holding mechanism 24.

[0008]Here, the case where the application program 26 is infected with the virus is considered. If an application program is executed, a virus program will begin to move. A virus program infects a virus to the file in a server, tries as like, specifies a file name and access classification, and tells access to the demand transmitting means 28. The demand transmitting means 28 reads the user-identification child of the user who is using the present client from the user-identification child holding mechanism 27, creates an access request, and sends it to the server 20.

[0009]In the server 20 side, the request reception means 21 receives this demand, and it tells to the file access means 23. the file access means 23 takes out the access attribute of the file specified first from the file holding mechanism 24 -- access classification and a user-identification child -- it sends to the access inspection means 22. In the access inspection means 22, the stored access attribute is analyzed and this file inspects whether access of access classification is permitted to the user-

identification child's user. Only when the permission is granted, the file access means 23 is accessed to the file within the file holding mechanism 24. Therefore, by setting the attribute of the file/directory which each user can access (it changes especially) as the dramatically limited thing, an access permit does not appear from the access inspection means 22 in the file of most in a server, but the program with which a virus program can be infected is restricted dramatically.
[0010]As mentioned above, in the server side, access to a file is restricted using an access attribute with the user-identification child, and influence attains to only the program by which the user who executed the virus program as a result is allowed change. Therefore, the influence of virus contamination is suppressed within the limits of the user who executed the virus program using the right-to-access system to a file. However, there are the following problems in this system.
[0011]Since the access permit has always come out to other programs and directories which the user who executed the virus program to the 1st has managed, it cannot prevent infecting a virus with it.
[0012]Generally neither a program nor a directory exists in the 2nd independently for every user, but a common program and directory, i.e., the directory which can store the possible program and virus program of a viral infection, exist. Therefore, if a certain user executes a virus program, it is infected with the common program or a viral infection program is stored in a directory. Therefore, virus contamination will also attain to the user who executed the program after it.
[0013]It will be infected even with the program/directory to which the virus program within OS operates and the user is allowed access, if other users use the client as it is when a virus program is executed to the 3rd and it is infected with OS of a client.
[0014]The privileged user to which access is not forbidden by the previous right to access exists in the 4th at such a server, and the client/server system is managed and employed. Therefore, if a virus program is executed by the client or a server as a privileged user, it will become possible for you to make it infected with all the programs in a server. It will compound, and these causes of infection will act, and will extend the range of infection further.
[0015]Since it may be infected with all the files and all the clients in a server, in order for a virus checker to check the range which a virus's not existing and infection have attained to, It is necessary to inspect all the files and all the clients, and the huge work proportional to the number of files or the number of clients is needed.
[0016]
[Problem to be solved by the invention]As mentioned above, huge work was required in the conventional right-to-access system to check the range of infect(ing), that it not only cannot prevent infecting a virus in a server thoroughly, but a virus does not exist in a server, or a viral infection. An object of this invention is to provide the virus defense system which returns the program infected while presuming the range of a viral infection.
[0017]
[Means for Solving the Problem and its Function]Two or more clients as which a virus defense system of this invention requires file access, In a client/server system which comprises a server which performs file access according to file access from these each client, The 1st management tool that follows said server at an access request from said each client, and manages a dependency of a history of file access of two or more of said clients, and file access between clients, The 2nd management tool that carries out additional creation of the version of a change target file according to a file change demand from said each client, and manages a version for every file,

Analyze a history of file access managed by said 1st management tool, and detect existence of the necessity for a virus inspection, and. An analysis means to pursue a file to be examined and a client according to a dependency of file access between said clients, and a means to delete an unnecessary version for said every file according to a result of a predetermined virus inspection are provided.

[0018]In this system, a file change is performed by version management by the 2nd management tool, without spoiling the existing file. Therefore, by a viral infection etc., a version to which change was added can be deleted and it can return to a front state at any time. While certainly being able to carry out the alarm of the file which has the possibility of a viral infection by using a dependency of a file between clients which the 1st management tool has managed, and a history of file access to a system administrator, Since a version is returned based on the inspection result, a version with which a virus was infected is certainly deleted. Therefore, a viral infection can be defended thoroughly.

[0019]Since an analysis means always analyzes information and is checking the possibility of virus existence, it can discover, before spreading broadly, even if it carries out a viral infection. Possible client and file of a viral infection can be limited using information on a dependency between clients. Therefore, what is necessary is just to inspect a small number of file and client, although existence of a virus in a server and an infection range when it exists are checked.

[0020]

[Working example]An working example of this invention is described below. Drawing 1 is the block diagram of a client/server system which consists of the server 1 and the client 10 constituted with the application of a virus defense system of this invention.

[0021]The request reception means 2 from which the server 1 receives the access request from a client, The dependence holding mechanism 3 which carries out creation maintenance of the dependency between clients and during a file according to the access request from a client, A version management means 4 to make and manage a version without reflecting the change (insertion, deletion, updating) to a file in a file directly, The file access means 5 which write a file, and the file holding mechanism 6 which actually holds many files, It comprises the analysis alarming means 7 which analyzes the dependency between the clients which the dependence holding mechanism 3 holds, and during a file, and carries out the alarm of the necessity for a virus inspection to a system administrator. On the other hand, the client 10 comprises the demand transmitting means 12 which sends out an access request to the server 2 according to directions of the application program 11 which needs the file in a server, and the application program 11. Operation of this system is shown briefly below.

(1) The application program 11 of a client tells the access request which consists of a file name and access classification (reading writing) to the demand transmitting means 12.

(2) The demand transmitting means 12 which received this demand creates a file request including a client name, and sends it to the server 1.

(3) In the server 1, the request reception means 2 receives this demand, and tell a file name, access classification, and a client name to the version management means 4.

[0022](4) The version management means 4 investigates whether the file of the specified file name is already managed. If not managed, after asking the file access means 5 whether to exist in the file holding mechanism 6, it registers as an administration object file.

[0023](5) Next, according to the registered information, the version management

means 4 investigates whether access of access classification is possible to the specified file, and determines the access configuration (it reads and deletes [insert, update and]) as it. Only when accessible, access processing continues.
[0024](6) Further, the version management means 4 is changed into the file request which makes a version, and tells the file access means 5 so that the contents of the existing file may not be spoiled, while it tells the settled access configuration, file name, and client name to the dependence holding mechanism 3.
[0025](7) The file access means 5 performs demanded file access to the file within the file holding mechanism 6, and the result is told to the application program 11 via the version management means 4, the request reception means 2, and the demand transmitting means 12.
[0026](8) The dependence holding mechanism 3 which, on the other hand, received the settled access configuration, file name, and client name carries out creation maintenance of the dependency between clients and during a file based on the information, and provides the analysis alarming means 7 with the information.
[0027](9) The analysis alarming means 7 always analyzes the information on the dependence holding mechanism 3, and when there is the necessity for a virus inspection, it carries out an alarm with the client and program which should be inspected to a system administrator. Based on the result of the virus inspection from a system administrator, there is no danger of a viral infection and the version of the file infected with the version which became unnecessary any longer, or the virus is told to the version management means 4.
[0028](10) The version management means 4 performs reduction of the versions whose necessity has already been lost, and restoration to the old version of the file infected with the virus. The information about the file registered is updated.
[0029]Here, the application program 11 is infected with the virus and the case where this virus is infectious to the file A, B, and C is considered. If an application program is executed, a virus program will begin to move. A virus program tries infection to the file A, B, and C in a server, specifies a file name and access classification in an order shown in drawing 2, and tells an access request to the demand transmitting means 12. A demand transmitting means creates and sends the file request from these access requests to the server 1.
[0030]In the server 1 side, the request reception means 2 receives these demands, and the file name, access classification, and a client name are told to the version management means 4. The version management means 4 registers these files as that by which version management is carried out, and determines the access configuration (reference, insertion, updating, deletion) (refer to drawing 3). When an access configuration is updating, it changes into the access request which makes a new version file so that the existing file content may not be lost for the update request of doing nothing in deletion, and tells the file access means 5. The file access means 5 performs these accesses to the file within the file holding mechanism 6, and tells the result to the application program 11 via the version management means 4, the request reception means 2, and the demand transmitting means 12. Therefore, as shown in drawing 4 after processing of the access request from a client, the file A, B, and C changes to version a_1, b_1, version a_2 infected with the virus from c_1, b_2, and c_2, respectively. However, since the file of version a_1, b_1, and c_1 also remains without adding change, it can restore by returning to version a_1, b_1, and c_1 later.
[0031]The version management means 4 tells the access configuration, file name, and client name which were determined to the dependence holding mechanism 3. The dependence holding mechanism 3 adds these information to the dependency between

the clients currently held so far and during a file. Therefore, drawing 3 is added as information on the dependency during the file in the client 10, and the information on the file dependency about the client 10 becomes like drawing 5.
[0032]It is investigated whether the analysis alarming means 7 has an access pattern which always analyzes the information on this dependency and in which the possibility of virus existence is shown. When an applicable access pattern is discovered and there is the necessity for a virus inspection, the alarm of the client and file which should be inspected to a system administrator is specified and carried out. Therefore, it is discovered that in the case of this example three file updates also continue for a short time, and are performed as shown in drawing 5, P (file in which, as for P, the application program 11 is stored), A, B, and C are carried out to the client 10 as a client which should carry out a virus inspection, and an alarm is carried out to a system administrator as a file which should be inspected. System administrators are means, such as a virus checker, and discover the virus in ** BE, the file P, A, and B, and C for existence of a virus. A system administrator inputs this result into the analysis alarming means 7. The analysis alarming means 7 is conveyed to the version management means 4 by making an old version into an unnecessary version, when the version in which the file is high when infected with the virus is not infected, and it deletes an unnecessary version. Therefore, in this example, the file P, A, and B, the high version of C, i.e., p_1 of the file P, a_2 of the file A, B, and C, b_2, and c_2 are deleted, respectively, and it is restored to the file version before a virus is infected.
[0033]When other clients are referring to version p_1 infected with these viruses, a_2, b_2, or c_2, the dependency between clients like drawing 6 is held at the dependence holding mechanism 3. Therefore, the alarm of the analysis alarming means 7 is further carried out to a system administrator so that the client B which is referring to version p_1 which carried out the viral infection, and the file D updated after the reference may be inspected. The exchange of a virus inspection with this system administrator is continued until a version with the possibility of a viral infection is lost.
[0034]Next, the case where it becomes a problem by the conventional right-to-access system is considered. In this system, it leaves the existing file as an old version by the version management means 4, and since it is deleted only when not being infected with a virus by a system administrator is checked, when it turns out that it was infected with the virus, the influence of a viral infection is removed always. Therefore, a viral infection can be prevented also by the 1st case. In the dependence holding mechanism 3, the dependency during a file and the dependency between clients are held, and the analysis alarming means 7 can pursue the file which has the possibility of the infection also by the 2nd case or the 3rd case, and removes the influence of a viral infection by a system administrator's inspection. Since this system is not using the right to access at all, it can be defended also by the 4th case. While asking for possible client and file of the infection by the dependency information of the dependence holding mechanism 3 in origin by this system finally, What is necessary is just to inspect a small number of file and client to also check the range which a virus's not existing by a virus checker etc. and infection have attained to, since an alarm is carried out to a system administrator before the analysis alarming means 7 always analyzes dependency information and infection of a virus spreads greatly. Finally, the example of each element which constitutes this example is explained.
[0035]The demand transmitting means 12 and the request reception means 2 are components indispensable to a client/server system. If UNIXclient/server type file system NFS is taken for an example, The demand transmitting means 12 is equivalent

to an NFS client and its communications program (XDR, RPC, UDP, IP), The request reception means 2 is equivalent to an NFS server and its communications program, and the specification is exhibited as RFC from the standardization organization (IAB:Internet ActivitesBoard) of TCP/IP. Therefore, since it is obvious for a person skilled in the art, the explanation beyond this is omitted here.

[0036]It is a nonvolatile storage which consists of a magnetic disk, an optical disc, or semiconductor memory, and the file holding mechanism 6 holds many files, and its read-out/writing is possible for it. Generally, a magnetic disk drive is used.

[0037]The file access means 5 searches for the storing position of the file within file holding mechanism from a file name (or the identifier), and performs read-out/writing of a file according to an access request. If UNIX is taken for an example, a file system is equivalent to this and the details are given to design:KYORITSU SHUPPAN of a document "UNIX kernel, and Maurice J Bach work.

[0038]In order that the version management means 4 may realize change (insertion, deletion, updating) to a file, without spoiling the file within the file holding mechanism 6, it creates a version (version) and manages the information. The version information table which manages version information comprises the file name field and two or more version fields, as shown in drawing 7.

The correspondence relation of the file name and the actual file name of each version of the file which is visible from a user is held.

The file name of the leftmost of the version field means the oldest version, and the rightmost means the newest version. It means that (deleted) was deleted. Therefore, it means that the DEF file is deleted in drawing 7. The outline flow of the version management means 4 using this table is shown in drawing 8.

[0039]That is, the version management means 4 investigates whether the file of the specified file name is already managed first (Step S1). If not managed, after asking the file access means 5 whether to exist in the file holding mechanism 6, it registers as an administration object file (Step S2). Next, according to the registered information, the version management means 4 investigates whether access of access classification is possible to the specified file, and determines the access configuration (it reads and deletes [insert, update and]) as it (Step S3, S4). When it cannot access, it becomes an error and processing is interrupted.

[0040]The version management means 4 is changed into the file request which makes a version, and tells the file access means 5 so that the contents of the existing file may not be spoiled, while it tells the settled access configuration, file name, and client name to the dependence holding mechanism 3 (Step S5). For example, when an access configuration is reference, the field of the table of drawing 7 is searched, the newest Bar Joan of most right-hand side is found, and it is read by the file access means 5 (Step S6). When an access configuration is insertion, the file of a new bar SHON name is created and it registers with the file holding mechanism 6, and the bar SHON name is written in the head of the version field (Step S7). When an access configuration is deletion, (deleted) is written in the right end of the field (Step S8). When an access configuration is updating, the file of new version names is created, it is registered into the file holding mechanism 6, and new version names are written in the right end of the field (step S9). As a result of a virus inspection, the version which has already become unnecessary is actually deleted also from the file holding mechanism 6, when the name is deleted from a table.

[0041]The dependence holding mechanism 3 carries out creation maintenance of the dependency between clients and during a file based on the access configuration from the version management means 5, a file name, and a client name. As shown in

drawing 9, the dependency table holding a dependency exists for every client, and comprises a file name, an access configuration, and the pointer field. The thing required for each table to limit the range of a viral infection among the file accesses performed by the client is stored. The flow which creates the dependency table of the dependence holding mechanism 3 is shown in drawing 10. The information which has already become unnecessary is deleted as a result of a virus inspection.
[0042]That is, first, it is judged whether an access configuration is reference (Step S11), and when it is except reference, the access configuration corresponding to the table specified by the client name with a file name is added (Step S12). In judging whether renewal of the specified file and insertion are already shown in a table on the other hand when an access configuration is reference (Step S13) and existing, A file name and an access configuration are added to the table specified by the client name, and between the entries of updating found at Step S13 and insertion is combined with a pointer (Step S14). The analysis alarming means 7 processes the following four.
(1) Always analyze the information on the dependence holding mechanism 3, and investigate whether there is any access pattern which needs a virus inspection as shown below.
Pattern A; The file change more than the frequency beforehand defined by the whole access pattern a. server which shows the possibility of virus operation is performed.
b. The file change more than the frequency beforehand defined by one client is performed.
c. The file of the name defined beforehand is changed.
Pattern B; When existence of a virus becomes clear, the file change of accumulation exceeds the number of times defined beforehand by the client of a.1 access pattern in which the influence may reach far and wide.
b. Reference of accumulation exceeds the number of times defined beforehand by one changed file.
(2) When a virus inspection is required, ask for the client and file which conduct a virus inspection as follows, and carry out an alarm to a system administrator.
[0043]In the case of the pattern A, the file changed within the access pattern, the file referred to in front of the access pattern, and the client by which the access pattern was discovered are made into a subject of examination. In the case of the pattern B, the client by which the access pattern was discovered, or the file included in an access pattern is made into a subject of examination.
(3) Even if it performs it as follows based on the result of the virus inspection which the system administrator conducted, tell a version management means in quest of the version which already became unnecessary.
- When you understand that there are not to be infected with a virus, let the thing before the version be an unnecessary version.
- When it turns out that it is infected with the virus, let the thing after the version be an unnecessary version.
[0044](4) Carry out an alarm to a system administrator in quest of the file and client which still need a virus inspection as follows based on the result of the virus inspection which the system administrator conducted.
- The client which referred to the file by which the virus was discovered.
- The file changed by the client by which the virus was discovered.
[0045]In this working example, the function returned when infected with the function to detect a possibility of having been infected with the virus is only provided. Therefore, judgment whether it was infected or not assumed that a system administrator etc. used another means, such as a virus checker. Since a system

administrator's work becomes serious now next, the infection judgment function which judges whether it was infected with the virus is described.
[0046]In order to detect a viral infection by a server, an infected file which is infected with a virus by a client or the server always is prepared first, and before a virus is infected in the contents of the infected file in a client side, a copy is taken and put on the server side. And it is investigated whether to the infected file in the server side, there is any trace changed as compared with the copy which has the file in a server periodically to the infected file in whether updating access breaks out and a client side. If there is trace which updating-accessed or was updated, the client taken out to the updating access and the client with the updated trace mean being infected with the virus, and can detect a viral infection. Hereafter, with reference to drawing 11, it explains concretely. Drawing 11 is a system configuration for virus detection.
[0047]Infected file B which anyone can access exists in a client. The file update monitor means which supervises in a server updating to infected file A which anyone can access, and its infected file A, The file comparison monitor means which carries out the comparison surveillance of whether the contents of infected file B in a client and the contents of the copy file of infected file B in a server are periodically in agreement with the copy file of infected file B in a client exists. Here, the case where a virus program is started by a client is considered.
[0048](1) A virus program looks for the file infected in itself first. Here, a virus program discovers infected file B which anyone in infected file A which can be accessed [whom in a server or], or a client can access.
(2) Next, a virus program tries a viral infection to the discovered file, and updates a file. Therefore, infected file A and B will be updated and a virus will be infected. Here, updating to infected file A in a server is detected by the file update monitor means in a server, and it turns out that a virus program exists in the client which required the file update.
[0049](3) The file comparison monitor means in a server compares the infected file in a client with its copy in a server periodically, and inspects whether it is in agreement. Here, since infected file B in a client is updated by the virus program at Step 2, it is not in agreement. Therefore, it turns out that a virus program exists in a client with infected file B.
[0050]As mentioned above, if a virus program exists in a client, either infected file A in a server or infected file B in a client will be changed, and existence of a virus will be detected by the file update monitor means or a file comparison monitor means.
[0051]Infected file A and B cannot be limited to a text file, and all the kinds stored in computers, such as a command file, a library file, an executable file, and a file that stores the OS itself, of file can be used for them. As mentioned above, in this working example, a file change is performed by the version management by the version management means 4, without spoiling the existing file. Therefore, by a viral infection etc., the version to which change was added can be deleted and it can return to a front state at any time. While certainly being able to carry out the alarm of the file which has the possibility of a viral infection by using the dependency of the file between the clients which the dependence holding mechanism 3 has managed, and the history of file access to a system administrator, Since the version is returned based on the inspection result, the version with which the virus was infected is certainly deleted. Therefore, a viral infection can be defended thoroughly.
[0052]Since the analysis alarming means 7 always analyzes information and is checking the possibility of virus existence, it can discover, before spreading broadly, even if it carries out a viral infection. Possible client and file of a viral infection can

be limited using the information on the dependency between clients. Therefore, what is necessary is just to inspect a small number of file and client, although existence of the virus in a server and the infection range when it exists are checked.
[0053]
[Effect of the Invention]As mentioned above, in this virus defense function, checking the range of infect(ing), that it not only can prevent infecting a virus in a server thoroughly, but a virus does not exist in a server, or a viral infection should just also inspect a small number of file. Therefore, the dramatically efficient measure against a virus is realizable.

CLAIMS

[Claim(s)]
[Claim 1]A client/server system which comprises two or more clients characterized by comprising the following which require file access, and a server which performs file access according to file access from these each client.
The 1st management tool that said server follows an access request from said each client, and manages a dependency of a history of file access of two or more of said clients, and file access between clients.
The 2nd management tool that carries out additional creation of the version of a change target file according to a file change demand from said each client, and manages a version for every file.
An analysis means to analyze a history of file access managed by said 1st management tool, and to detect existence of the necessity for a virus inspection, and to pursue a file to be examined and a client according to a dependency of file access between said clients.
A means to delete an unnecessary version for said every file according to a result of a predetermined virus inspection.

[Claim 2]An application program which requires access for a file, and two or more clients which comprise a demand transmitting means which tells the access request to a server via a network, A request reception means which receives an access request from a client, and a file access means which performs file access according to this demand, In a client/server system which connected in a network a server which comprises file holding mechanism which has memorized a file, While realizing a file change, without carrying out creation management of the high version of the file, and spoiling a file within file holding mechanism which already exists to said server, when a file change (insertion, deletion, updating) is required, From a history of file access which a version management means to delete a specified unnecessary version, and a version management means performed. A dependency between a client which inserted and updated a file, and a client which referred to the file, Always analyze information on dependence holding mechanism which carries out creation maintenance of the dependency during a file by order relation of file access within each client, and dependence holding mechanism, and it is investigated whether there is any necessity for a virus inspection, In being required, while specifying and carrying out the alarm of a client and a file which should be inspected altogether to a system administrator, Analyze based on a result in which a system administrator conducted the virus inspection, and a version which became unnecessary is told to a version management means, A virus defense system having an analysis alarming

means which carries out the alarm of a client and a file furthermore to be inspected to a system administrator, and a system administrator's conducting a virus inspection of a client or a file according to directions of an analysis alarming means, and teaching the result to an analysis alarming means.
[Claim 3]The virus defense system according to claim 2, wherein said server has a virus inspection means further.

[Translation done.]

PATENT ABSTRACTS OF JAPAN

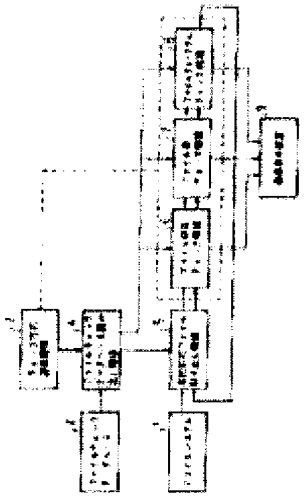
(11)Publication number : **03-233629**
(43)Date of publication of application : **17.10.1991**

(51)Int.Cl. G06F 9/06
G06F 11/10

(21)Application number : **02-029902**
(22)Date of filing : **09.02.1990**

(71)Applicant : **NEC CORP**
(72)Inventor : **YATSU YOSHIYUKI**

(54) SYSTEM FOR CHECKING DESTRUCTION OF EXECUTION FORMAT FILE



(57)Abstract:
PURPOSE: To easily specify a suspected file and to prevent a computer system from being infected by computer virus by totally comprehensively executing a check while paying attention onto the character of infection caused by the computer virus.
CONSTITUTION: A file existence checking mechanism 6 investigates whether the file name of an execution format file taken out from a file system 1 by an execution format file take-out mechanism 5 exists in data read out from a file checking data base 2 or not. A file length checking mechanism 7 investigates whether the real file length of the execution format file is coincident with the file length of the same file registered to the file checking data base 2 or not, and a file checksome checking mechanism 8 investigates whether the checksome of a byte string is coincident or not. Then, alarming messages are respectively outputted to a picture display device 9. Thus, the suspected file is specified by easily checking the infection of the computer virus and the computer system is prevented form being infected.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-196112
(43)Date of publication of application : 11.07.2003

(51)Int.Cl. G06F 11/00

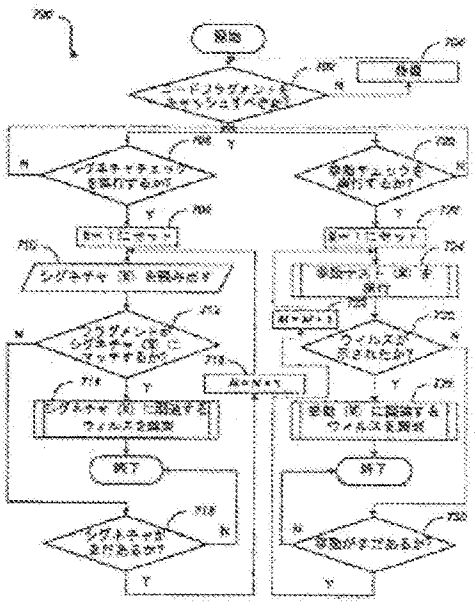
(21)Application number : 2002-343392
(22)Date of filing : 27.11.2002

(71)Applicant : HEWLETT PACKARD CO <HP>
(72)Inventor : MATEEV NIKOLAY
DESOLI GIUSEPPE

(30)Priority
Priority number : 2001 997454 Priority date : 29.11.2001 Priority country : US

(54) VIRUS CHECK METHOD FOR VIRUS CHECK SOFTWARE

(57)Abstract:
PROBLEM TO BE SOLVED: To provide an improved virus check method for detecting an infected software code.
SOLUTION: In one embodiment of the system 300 and method 700 for a virus check program, a program command 702 is intercepted to determine whether the corresponding command contains at least one 'signature' bite for identifying a known virus 712 or not, and the intercepted code is opened to computer hardware 104 only when it is determined that no signature of virus is contained in the intercepted code.



PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
11/477,807	06/30/2006	

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY ☐ RATE ($)	SMALL ENTITY ☐ FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
	12/23/2011									
	Total (37 CFR 1.16(i))	* 47	Minus	** 49	= 0	X $ =		OR	X $60=	0
	Independent (37 CFR 1.16(h))	* 8	Minus	***8	= 0	X $ =		OR	X $250=	0
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	0

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/SHERRY DAVIS/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2004/039921	2/26/2004	Chuang, Shyne-Song	
		US -2005/086500	4/21/2005	Albornoz, Jordi	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document: Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		EP 1 315 066	5/28/2003			

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		International Search Report issued in PCT/GB2006/002439 issued January 15, 2007	
		ZENKIN, D., "Fighting Against the Invisible Enemy - Methods for detecting an unknown virus" Computers & Security, Elsevier Science Publishers; Amsterdam, NL; vol. 20; no. 4; 31 July 2001; pages 316-321, XP004254268	

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 315 066 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
28.05.2003 Bulletin 2003/22

(51) Int Cl.[7]: **G06F 1/00**, H04L 29/06

(21) Application number: **01309795.1**

(22) Date of filing: **21.11.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(71) Applicant: **BRITISH TELECOMMUNICATIONS public limited company**
London EC1A 7AJ (GB)

(72) Inventor: **The designation of the inventor has not yet been filed**

(74) Representative: **Nash, Roger William et al**
BT Group Legal Services,
Intellectual Property Department,
Holborn Centre, 8th Floor,
120 Holborn
London EC1N 2TE (GB)

(54) Computer security system

(57) A computer security system for use in a network environment comprising at least a plurality of user computers arranged to communicate over a network, the system comprising a warning message exchange system operable to allow the communication from the user computers of warning messages relating to suspect data identified as a possible security threat; a message counting system operable to maintain a count for every particular piece or set of suspect data based on the number of warning messages communicated relating thereto; and network security means operable to act against any particular piece or set of suspect data for which the count maintained therefor exceeds at least one threshold value

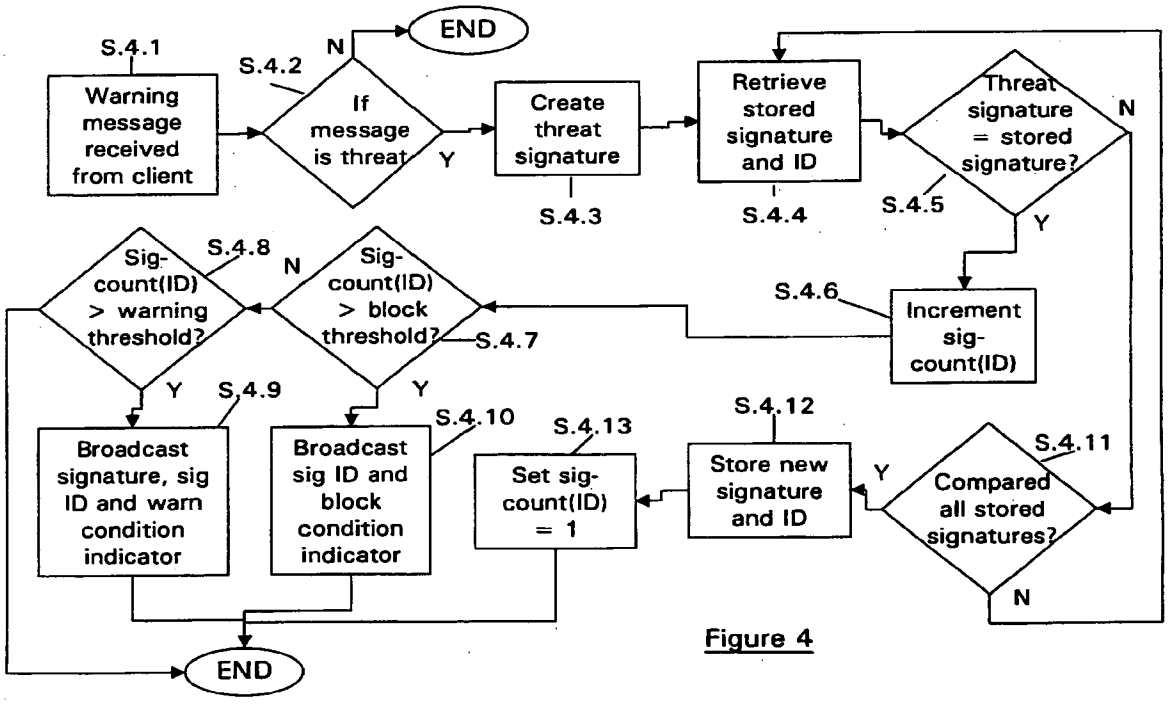


Figure 4

EP 1 315 066 A1

Printed by Jouve, 75001 PARIS (FR)

Description

Technical field

[0001] The present invention relates to a computer security system for use in a networked environment, the system providing a measure of security against suspect data such as computer viruses and the like.

Background to the Invention

[0002] Malicious attacks on computer systems have plagued individuals and organisations for many years. Initially these attacks mostly took the form of viruses in which a rogue program infected a machine via an infected floppy disk or network location for example. More recently so called worms and Trojan horses have caused much disruption in which the user is deceived into running a program via an attachment to an e-mail or a rogue program masquerading as something more innocuous. A large industry has grown up around protection from such attacks. These companies provide anti-virus software that typically resides on individual machines and monitors the system to check for the presence of known viruses. These viruses can take many forms - some more advanced forms are fully polymorphic in that the byte code "signature" is entirely different from one instance of the virus to the next. This is achieved through the use of encryption technology and/or the addition of spurious and random code to the virus. However, the majority of viruses that are in the "wild" and the cause of costly disruption to computer systems are relatively simple and can be detected by simple byte-code signature matching. Many of the current anti-virus programs use just such techniques and are successful if the signature of a discovered virus can be delivered to machines before the virus strikes.

[0003] The operation of such anti-virus programs and systems is well known in the art, and is usually as follows. A computer user will have installed on their computer an anti-virus program which is provided with a database of known computer virus byte-code signatures. The program will tend to run in the background continuously monitoring operations performed on the computer, and data received at and transmitted from the computer. If the byte-code signature of a known virus stored in the anti-virus program's database is detected by the program, the anti-virus program informs the user and takes appropriate action against the data, such as deleting it or storing it in a protected drive.

[0004] Such prior approaches tend to suffer from a big problem, however - delay. Even the simplest of viruses and worms may still cause great disruption at enormous cost to organisations. This is because the process from discovery of a new virus to delivering its signature to all protected machines takes too long, and requires an administrative authority such as the anti-virus program manufacturer (or in some cases an organisation's IT department) to recognise the problem, take action to identify the virus's signature, update the anti-virus database, and distribute the updated database to each user. By the time such a sequence of actions is complete the damage has already been done in many cases. What is required is a much more rapid approach - one that can operate on the same time scales as the spread of the virus and thus provide much more rapid and cost-saving protection.

Prior Art

[0005] There is much work exploring the use of agent technology to provide rapid reaction to malicious attack. In this work computer systems are inhabited by a number of agents whose job it is to detect intrusions. One interesting approach aims to draw metaphor from immune systems, which allow an organism to rapidly respond to previously unseen foreign objects through a "real-time" evolutionary system, as disclosed in Artificial Life IV, Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems, Rodney A. Brooks and Pattie Maes, eds., MIT Press, Cambridge, Massachusetts, 1994, pp. 130-139 These kinds of agent systems may prove useful in future security systems, but rely on the successful development and deployment of software agents which can detect intrusions. Many other potential future anti-virus systems are also discussed at http://www.research.ibm.com/antivirus/SciPapers.htm. In the meantime until the potential of these ideas is exploited, there is still a need for a system which allows for more rapid prevention of the spread of computer viruses.

[0006] One such prior art system which claims to increase response times is that of the Symantec® Digital Immune System (DIS) , which is described at http://securityresponse.symantec.com/avcenter/reference/dis.tech.brief.pdf. The DIS operates by providing an heuristic search engine which automatically identifies possible viruses at either the user's desktop, the server, or the network gateway. A secure channel is then provided direct to Symantec's security response centre over which the suspect data can be transmitted. At the security response centre the received data is subject to a number of automatic filters to detect, for example, known clean files, files which are known to return false positives from the heuristic search engine, and known virus files. If the suspect data is not filtered out by any of these filters then it is directed to an analysis program for further analysis and action.

[0007] The DIS system therefore allows for the automatic identification, secure submission, automatic prioritisation, and subsequent analysis of suspect data, but ultimately still involves a central authority in the loop to analyse suspect files which make it through the filters. Thus, whilst DIS may improve response times to virus infection through it's automatic filtering processes, it still relies on a central authority to analyse the suspect data

and decide on appropriate action, which must then be communicated outwards to each user. There is therefore still a need for a system which removes this centralised analysis step to speed the response.

Summary of the Invention

[0008] In order to address the above problems, the present invention provides a collaborative computer security system wherein the responsibility for detection of malicious data such as a computer virus is removed from that of any central authority, and is instead placed in the hands of each and every user of the network. More particularly, the present invention provides a distributed virus identification system which allows individual users or software agents running on a user's computer to identify malicious data when they receive it, and to pass a warning message concerning the received data to either their peers, or to a central server. A record is kept either at the server or at each peer computer as to the number of warning messages communicated concerning any particular piece or set of suspect data, and then appropriate security actions such as issuing warnings to users or blocking the transmission of the suspect data can be taken once the number of warning messages communicated from users has passed a certain threshold level. The advantages of the invention are that an organisation's response to a computer virus intrusion can be much more rapid than was previously the case where a central authority had to identify the problem and take action.

[0009] In view of the above, the present invention provides a computer security system for use in a network environment comprising at least a first group of user computers arranged to communicate over a network, the system comprising a warning message exchange system operable to allow the communication from the user computers of warning messages relating to suspect data identified as a possible security threat; a message counting system operable to maintain a count for every particular piece or set of suspect data based on the number of warning messages communicated relating thereto; and network security means operable to act against any particular piece or set of suspect data for which the count maintained therefor exceeds at least one threshold value

[0010] The primary advantage of the invention is that it allows the security system to work on the same time scales as the spread of the attack. The more the computer virus or worm spreads the more likely it is that enough positive identifications are made to allow its signature to be ascertained and protection to be installed. This overcomes the problem of traditional security systems where an authority has to learn of the virus, take action, and then distribute associated protective measures to the users.

[0011] A further advantage of the system is that it allows a lot of information to be rapidly collected about a computer virus. For example, current "worms" that are "in the wild" are mostly identical at the byte code level, however it is likely that in the future this will no longer be the case, and that there will be variations in different instances of the same worm. In such a case, the more instances of the worm that can be collected the more likely that it will be that suitable protection can be devised. By allowing for individual identification of a worm at each computer by the user, the present invention allows for multiple instances of the same virus to be identified and collected.

[0012] The one or more "threshold values" against which the respective counts for pieces or sets of suspect data are compared may be any value, and may be fixed or dynamically adaptive. Furthermore the actual values may be generated by any appropriate function, such as for example by a random number generator, or as a function of some metric of accuracy of users in identifying viruses. Furthermore, in other embodiments it is foreseen that the threshold values may not simply be actual numbers which the respective counts must be greater than, but could instead be probabilistic in nature. Furthermore, various logical functions or rules can be used for their creation, including for example, fuzzy logic.

[0013] Preferably, the judgement as to whether any piece or set of data present at any one or more of the user computers is a possible threat is performed by the respective human user or users of the or each user computer. Such an arrangement exploits the massive untapped resources and expertise of many computer users, and again allows for a rapid response to the spread of a computer virus.

[0014] Preferably, the network security means is further operable to compare the maintained count for a particular piece or set of suspect data against a plurality of different thresholds, and to take different action against the data dependent upon which threshold or thresholds are exceeded.

[0015] Such an approach allows the computer security systems to have a graded response to a computer virus intrusion, and to take different actions depending on the number of warnings received from users. Furthermore, it is also possible to weight warnings from different users, such that a warning from a user who is particularly expert in the field is weighted more heavily than a casual user of the system.

[0016] Preferably, the action taken by the network security means comprises warning each of the users as to the suspect nature of a particular piece or set of data. Furthermore, the action may also comprise preventing the transmission of the particular piece or set of suspect data across the network. Usually, the warning action will be taken when the number of messages communicated exceeds a lower threshold than a second threshold which must be crossed in order for the transmission blocking action to take place.

[0017] Preferably, the or each warning message com-

prises at least an identifier of the piece or set of suspect data to which it relates. The identifier may be the actual piece or set of suspect data itself, or a repeatably generatable signature substantially unique to the piece or set of data. By allowing for the collection of such identifiers, the security system also provides a means of collecting information on multiple computer viruses, and also on adaptive computer viruses whose byte code signature may change in each instantiation on a users machine.

[0018] In a preferred embodiment, the computer security system further comprises a network server arranged to receive the or each warning message communicated from the user computers, and further arranged to host the message counting system and at least part of the network security system, wherein that part of the network security system hosted by the server is operable to determine the action which should be taken against a particular piece or set of suspect data, and to communicate an action condition indicator indicative of the determined action to each of the plurality of users.

[0019] By using a central server to maintain the warning message counts and to receive the warning messages themselves, the network traffic generated by the computer security system is minimised, as users need only send a warning message direct to the server, which can then broadcast warnings about suspect data as appropriate to all the users.

[0020] In another embodiment, the warning messages generated by a user computer are broadcast to every other user computer, and each user computer is further arranged to host its own message counting system and network security system operable in accordance with any of the above. Such an approach represents a "peer-to-peer" system, and provides advantages that it is not dependent upon a central server for its operation, thereby presenting a more robust distributed system. Such a peer-to-peer system has a potential disadvantage, however, in that depending on the exact implementation it may be that the network traffic required to broadcast the warning messages from each user to every other user is unacceptably high.

[0021] To at least partially overcome this problem, in another embodiment there is provided an inter-group communications system operable to allow the communication of warning messages relating to suspect data identified as a possible security threat from the first group of user computers to one or more other groups of other user computers. By splitting the user computers intro groups, the number of warning messages which are required to be transmit is substantially reduced, thereby conserving network signalling bandwidth.

[0022] Preferably, the inter sub-set warning messages are transmitted by the inter-sub-set communications system if the number of warning messages communicated by the user computers in the sub-set concerning a particular piece or set of data exceeds at least one threshold value. Preferably the threshold value or values is/are the same as those at which the network security system takes action.

[0023] In further embodiments, the message counting system is further arranged to store one or more weighting coefficients relating to one or more particular user computers, and to increment the count maintained for a particular piece or set of suspect data by an amount based upon the weighting coefficient when a warning is received from the one or more particular user computers relating to the particular piece or set of suspect data

[0024] From a second aspect, the present invention also provides a computer readable storage medium storing one or more computer programs which when run on a computer cause the computer to operate in the manner of the computer security system as described above. Such a computer security system may provide all the additional features and advantages mentioned previously.

Brief Description of the Drawings

[0025] An overview of the operation of the invention, together with a description of a number of embodiments thereof, presented by way of example only, will now be made with reference to the accompanying drawings, wherein like reference numerals refer to like parts, and wherein:-

Figure 1 illustrates a typical network environment in which the computer security system of the present invention is intended to operate;

Figure 2 illustrates a computer storage medium located within each users computer, and depicts the programs and data stored thereon in a first embodiment of the invention;

Figure 3 depicts a computer storage medium located at a server used within the first embodiment of the inventions, and further depicts the programs and data stored thereon;

Figure 4 is a flow diagram showing the operation steps performed by the warning creation program stored at the server in the first embodiment of the present invention;

Figure 5 is a flow diagram showing the method steps performed by the warning receipt program stored at a users computer in the first embodiment;

Figure 6 is a flow diagram showing the steps performed by a filter program in either of the embodiments of the invention;

Figure 7 depicts a computer readable storage medium located at a users computer in a second embodiment of the present invention, and further depicts the programs and data stored thereon;

Figure 8 is a flow diagram of the steps performed by a warning creation program in the second embodiment of the present invention;

Figure 9 is a flow diagram of the steps performed by a warning receipt program in the second embod-

iment of the present invention;
Figure 10 is a flow diagram illustrating the steps performed by one of the functions in both Figures 8 and 9; and
Figure 11 illustrates an organisational architecture of user computers which may be used in another embodiment of the invention.

Overview of the operation of the invention

[0026] The present invention recognises and makes use of the fact that a computer system is already inhabited by a number of intelligent agents - the users themselves. In order to summarise the operation of the invention consider an attack by an e-mail "worm" (a type of computer virus such as the "Melissa" virus which was encountered in 2001). A worm is sent to an employee of a given organisation. With some probability the employee opens the e-mail attachment and unwittingly unleashes the worm that causes some damage to the individuals machine and propagates itself to a number of other machines through use of the individuals e-mail system and address book. In each of these cases there is some probability that the user will open the attachment and further propagate the virus. Thus, the statistics are in the worm's favour - not everyone will open the attachment but it only takes a small fraction to do so for the worm to propagate throughout an organisation's computer system. However, for every user who opens the attachment there are likely to be many more who correctly recognise the e-mail as a potential threat and discard it. Vital information is thus gained in this process - an intelligent agent has successfully detected an intrusion - and it is this information that the present invention exploits to its full effect. Doing so opens up the potential for the statistics to be turned against the worm and in favour of the protection system.

[0027] Within the invention the user informs the system of the recognition by forwarding the mail or passing the mail or other program to an anti-virus program that is running on that machine, which then passes it to the server (or in an alternative embodiment direct to the user's peers). The signature of the worm is then quickly ascertained and a count kept of the number of warnings received from users about the worm. Once the count reaches a certain threshold the signature of the worm is distributed to all the other individual machines, who then use the signature to filter any received data or mail. As an alternative the server itself could simply filter out all e-mails with such a signature. In either case the technique of allowing users (or user machines provided with appropriate software agents) to detect and provide warning of the worm aligns well with the trends in the industry towards peer to peer systems where the vast untapped resources at the edges of the network are beginning to be used to greatly enhance storage and computational power for example. The invention is therefore essentially a peer to peer security system that may tap the vast unused resource of the user's knowledge and experience, or the user computer's processing power.

Description of the embodiments

[0028] Two specific embodiments of the invention will now be described in detail, followed by description of various alterations that may be made to each to provide additional embodiments. Both main embodiments of the invention implement a computer security system for use in a network environment. A typical network environment in which the invention may be used is depicted in Figure 1.

[0029] Figure 1 illustrates a typical local area network (LAN) configuration. Here, a plurality of user computers 15 are connected to a network 10 which is another control of a network server 12. The network 10 provides typical network functionality, such as the ability to allow users 15 to communicate with each other and with the server over the network. The network 10 in the invention may be any particular network technology, may be wired or wireless, and may be scaled to any network size or architecture known in the art. Therefore, whilst the present invention is particularly useful for use within individual organisations provided with a LAN, it may also be used on a wider scale in a metropolitan area network (MAN) or a wide area network (WAN). Within the embodiments, each of the user computers 15, the network itself 10, and the server 12 are provided with appropriate equipment and software to permit the usual network communication functions as are known in the art.

[0030] Although the above makes reference to the user computers 15 being part of the same network, it should be noted that it is not essential that each user-computer be located in the same actual network (although this may be preferable). This is because all that the invention requires is the ability for any particular user computer to be able to send messages either to its peers or to a server, and hence all that is required is some message transport mechanism between such parties. Such a message transport mechanism may be a single network to which all the user computers and servers (where required) are connected, or may be a sequence of interconnected networks such as the Internet, with different user-computers actually being part of different networks.

[0031] Having described the context in which the invention is intended to operate, a first, preferred, embodiment which represents the best mode of the invention will now be described with reference to Figures 2 to 6.

[0032] The first embodiment of the invention provides a computer security system wherein identification of suspect data is performed by the users or suitable software agents installed and running on the user's computers at the user's computers themselves, and upon identification of a suspect piece or set of data a warning message is transmitted from the user's computer to the network server 12. At the server the number of warning

messages received about a particular piece or set of data is counted, and once the count passes a first warning threshold, a warning message is broadcast to all users, the message containing a signature of the suspect data, such that an anti-virus program located at each user's computer can filter incoming data for the suspect data. If further warnings are received from users by the server, and the count of warning messages exceeds a second threshold, then the server broadcasts a second message to all of the users on the network instructing the anti-virus programs located at each user computer to block the suspect data to prevent any onward transmission or infection.

[0033] Given the above overview, within the first embodiment each user computer 15 contains a computer readable storage medium 22, as shown in Figure 2. The computer readable storage medium will typically be a magnetic hard drive, or may equally be a DVD RAM, DVD-RW, DVD+RW, solid state memory, optical disc, magneto optical disc, or the like. The computer readable storage medium 22 at each user computer 15 is arranged to store a data filter program 24, and a warning receipt program 26. In addition, storage space is also provided for suspect data signatures and ID's, stored in the form of a data base 28.

[0034] The server 12 within the first embodiment also contains a computer readable storage medium 32 which will typically be a magnetic hard disc, but may also be any of the storage devices mentioned previously in respect of the or each user computer 15. Within the first embodiment the computer readable storage medium 32 stores a warning creation program 34, and a suspect data signature and ID database 36. Optionally, the computer readable storage medium 32 may also store a data filter program 38, to provide an alternative method of operation as will be described later.

[0035] Having described the main system hardware elements required by the first embodiment, the operation thereof will now be discussed with reference to the flow diagrams of Figures 4, 5, and 6.

[0036] Assume that a user computer 15 connected to the network 10 receives a message over the network which contains a computer virus or the like (referred to hereafter as suspect data). Within the first embodiment of the invention a user of the user computer 15 upon reading the message recognises that the message contains such suspect data, and hence within the first embodiment then forwards the suspect data as the payload in a warning message of predetermined format to the server 12. By forwarding the data as the payload encapsulated in a warning message of known format, the server knows that the payload is potentially suspect data, and hence will not become victim to the computer virus itself. At the server, upon receipt of such a warning message from one of the user computers 15, the warning creation program 34 is run, and the steps performed by this program are shown in Figure 4.

[0037] Firstly, at step 4.1 the server receives a warning message from one of its clients, being one of the user computers 15 (as described above). Then, at step 4.2 the server parses the warning message to retrieve the suspect data, and checks the message to see if it could be a computer virus. Such checking preferably takes the form of determining the data type to see if it could feasibly be a virus. Viruses are frequently executable files, such as .exe or .bat on a PC, or macros such as are found in Microsoft® Word® or Excel®. Such a check can conveniently be performed by looking at any file extension on the data. If such checking does not indicate that the data could be a virus then processing ends, as it is assumed that the user has simply made a mistake. Instead, if the warning message is in fact a true warning and the suspect data could possibly be a virus, then processing proceeds to step 4.3.

[0038] Note here that the checking step of step 4.2 is performed in the preferred embodiment, but in other embodiments may not be performed, and as such is optional. Where step 4.2 is not performed processing proceeds directly from step 4.1 to 4.3.

[0039] At step 4.3 the server reads the suspect data from the payload, and creates a "signature" for the suspect data, to permit the particular piece or set of suspect data to be identified in the future. Such a signature may be conveniently created by running the suspect data through a hash function. Hash functions are well known in the art, and commonly provide a 16 or 20 bit output word unique to any particular piece or set of data input thereto. The key feature of hash functions which may be of use in the present invention, however, is that the same output word is always output from the hash function for the same piece or set of input data. Thus the hash value for a certain piece or set of suspect data will always be the same, no matter where the suspect data has been received from.

[0040] It should be noted that other forms of signature creation other than the use of hash functions may also be of use, the only requirement being that an identifiable unique signature is generated for any particular piece or set of suspect data input into the signature creation function at any time.

[0041] Following the creation of the data signature, at step 4.4 the signature and ID database 36 is accessed to retrieve the first previously stored signature and accompanying signature ID therefrom. The signature and ID database 36 in the server 12 is used to store the generated signature ID's from pieces of suspect data received in previous warning messages. At step 4.4 the first signature and ID in the data base is retrieved the first time the controlling function for step 4.4 is called, and thereafter at succeeding calls the next signature and ID in the database list are successfully retrieved.

[0042] At step 4.5, the threat signature of the received suspect data generated at step 4.3 is compared with the stored signature retrieved from the database at step 4.4. If the threat signature is not the same as the retrieved stored signature then processing proceeds to step 4.11,

wherein a check of the signature and ID database 36 is made to see if all stored signatures have been compared with the new signature. If not then processing proceeds to step 4.4, which retrieves the next signature and ID from the database 36 for evaluation at step 4.5 with the generated data signature for the presently received.

[0043] If at step 4.11 it is determined that all of the stored signatures have been compared with the newly created threat signature then it must be the case that the warning message received from the user computer at step 4.1 is the first warning message from any of the user computers to bring to the server's attention the particular piece or set of suspect data contained in the warning message payload. In this case it is necessary for the server to store information on the suspect data, and this is achieved by storing the suspect data signature created at step 4.3 together with a new unique signature ID within the signature and ID database 76. This storing function is accomplished at step 4.12, and thereafter processing proceeds to step 4.13, wherein a count of warning messages received for the particular piece of suspect data is established, and initiated to a value of one. It will be appreciated that a separate count is maintained for every different piece or set of suspect data received in a warning message, and hence at any one time the server will be maintained in several different counts. Therefore, conveniently the various counts may be maintained in a one dimensional array structure, indexed by the particular suspect data's signature ID. Such storage of the count is shown in step 4.13.

[0044] Following the storage of the data signature, signature ID, and establishment of the message count for the suspect data, then in the case that the suspect data had never been received before processing then ends.

[0045] Returning to a consideration of step 4.5, if it turns out that the particular piece or set of suspect data included in the warning message has been received before in a previous warning message then at some point in the loop formed by steps 4.4, 4.5, and 4.11, the evaluation performed at step 4.5 will return a positive value, in that the threat signature generated at step 4.3 will match a previously stored signature. In this case processing then proceeds to step 4.6, wherein the ID of the stored signature is used at an index into the signature count array, and the particular count for the stored signature is incremented by one to reflect the receipt of an additional warning message relating to the particular piece or set of suspect data to which the signature relates. Following the incrementing of the signature count, processing proceeds to step 4.7.

[0046] Step 4.7 represents the first thresholding evaluation performed to determine whether or not a sufficient number of warnings have been received from the user computers concerning a particular piece or set of data such that action should then be taken against that data. Therefore, at step 4.7 an evaluation is performed to determine whether the signature count for the particular piece or set of suspect data is greater than a "block" threshold value. If so processing proceeds to step 4.10, wherein the server then broadcasts a message to all of the user computers 15 on the network, the message containing the particular data signature ID, and a message condition indicator field which in this case contains data instructing the user computers to block the suspect data if it is received thereat. Blocking of the suspect data is performed by the data filter program 24 provided at each user computer 15, the operation of which is described later.

[0047] It should be noted that at step 4.10 the message from the server to the clients contained only the signature ID and the condition indicator and does not contain the actual signature itself. The reason for this is that it is assumed that prior to any particular piece or set of suspect data becoming blocked a warning message will have been transmitted from the server to the user computers containing the suspect data signature, and hence there is no need to retransmit the signature at step 4.10.

[0048] If the evaluation at step 4.7 returns a negative, processing proceeds to the evaluation at step 4.8 wherein the signature count for the particular piece or set of suspect data is compared against a warning threshold level. It should be noted here that the warning threshold level is set less than the block threshold level such that the evaluation of step 4.8 is always found positive before the evaluation of step 4.7. If the evaluation of step 4.8 returns a negative such that the count is less than the warning threshold then this means that not enough warning messages have been received from user computers concerning the particular piece or set of suspect data such that no action should be taken as yet. In this case processing of the warning creation program 34 then ends.

[0049] In contrast, if the evaluation at step 4.8 is positive, the processing proceeds to step 4.9 wherein the server broadcasts a message over the network to all the user computers 15, the message containing the suspect data's signature as generated at step 4.3, a new data signature ID, being simply an identifier of the signature which can be used as a short hand means of identifying the particular piece or set of suspect data, as well as a "warn" flag in a data condition field of the message. The presence of the warn flag in the data condition field of the message causes the data filter program 24 at the user computer to warn the user whenever the suspect piece of data is received, as will be described later. Following the broadcast of the warning message at step 4.9, processing by the warning creation program 34 ends.

[0050] The output of the warning creation program 34 is therefore a warning message broadcast to all the user computers 15 instructing the user computers to upgrade their threat levels for a particular piece or set of suspect data provided that enough warning messages have been received at the server concerning the particular

suspect data. At each user computer upon receipt of a warning message from the server the warning receipt program 26 is run, and the steps performed thereby are shown in Figure 5.

[0051] With reference to Figure 5, at step 5.1 a warning message containing the information described earlier is received from the server. Following this, at step 5.2 the user computer 15 passes the warning message to determine whether or not the message contains a data signature.

[0052] If the message does not contain a data signature, then this means that a warning message must have been received at some point in the past from the server containing the data signature, and therefore the data signature will be already stored in the ID database 28 at the user computer 15. In this case processing proceeds to step 5.4, wherein the signature ID contained within the received warning message is used to index and retrieve the appropriate signature from the database 28, whereupon processing then proceeds to step 5.6.

[0053] At step 5.6 the data condition indicator which is contained in the warning message received from the receiver is read, and a condition field in the database 28 related to the particular suspect data signature is updated to match the condition indicated in the warning message from the server. In this case, as a warning message will have already been received concerning the particular suspect data and hence a "warn" condition already set, it is likely that the condition is being updated to a "block" condition. Following step 5.6 the processing by the warning receipt program ends.

[0054] If at step 5.2 it is determined that the warning message from the server does contain a data signature, then this will be the first warning message received from the server concerning the particular suspect data. In this case, processing proceeds to step 5.3 wherein the received signature is stored in the database 28 indexed by the ID which is also contained in the received message. Furthermore, a condition field in the database related to the signature is also set, at step 5.4. In this case, as this will be the first warning message received from the server about the particular suspect data the condition will be set to a "warn" condition. Following step 5.4 the processing by the warning receipt program 26 ends.

[0055] It has thus been described how users can generate warnings about suspect data which are then transmitted to the server, which then broadcasts warnings to all users if enough user's specific warnings have been received. Users can then record the particular security advice broadcast from the server concerning a piece or set of suspect data, and we now describe how user computers 15 may use the broadcast warnings within the data filter program 24.

[0056] More particular, as mentioned previously each user computer 15 is provided with a data filter program 24 which is arranged to run in the background of the computer all the time the computer is operating in a similar manner to anti-virus and other security programs of the prior art. The data filter program 24 acts to monitor all data received at the user computer 15 upon which it is stored to determine if any of the received data matches the signatures of any suspect data stored in the signature and ID database 28. The steps performed by the data filter program 24 in performing this function are shown in Figure 6.

[0057] At step 6.1 assume that the user computer 15 has received some data in the form of an email message or the like over the network 10. At step 6.2 the data filter program 24 acts to create a signature of the received data using the same signature generation function as is used in the server 12. That is, the signature of the received data may be created using a hash function or indeed any other signature generating function which provides a repeatably generatable unique signature for a given set of input data.

[0058] Following the creation of the data signature, at step 6.3 the signature and ID database 28 at the user computer 15 is accessed and the first signature and signature ID within the database is retrieved therefrom. Upon subsequent calls to the controlling function of step 6.3 the next signature and signature ID in the database list is retrieved successively.

[0059] Following step 6.3 processing proceeds to the evaluation of step 6.4, wherein the data signature created at step 6.2 is compared with the retrieved signature from the database 28. If these signatures do not match then processing proceeds to the evaluation of step 6.10, wherein the database is again accessed to see if all signatures have been accessed and compared. If at step 6.10 it turns out that all of the signatures stored in the database have been compared against the signature of the received data and no match has been found, then it can be concluded that the received data is not suspect data, and hence the data filter program ends. In such a case the user is then free to use the received data in whatever manner she chooses.

[0060] In contrast, if at step 6.10 it is determined that not all of the stored signatures have been compared against the received data signature, processing returns to step 6.3 wherein the next stored signature and ID in the database are retrieved, and the evaluation of 6.4 is performed once again.

[0061] If at step 6.4 it is determined that the received data signature is the equivalent of a previously stored signature then processing proceeds to step 6.5. Here, the database is again accessed to retrieve the signature condition associated with the stored signature to determine what security condition was attached to the signature by the server in the last warning message received from the server concerning the particular signature. Following step 6.5 the retrieved condition is checked at step 6.6 to see if it is a "warn" condition, and if so processing proceeds to step 6.7, wherein a warning is displayed on the user computer's screen to warn the user about the received data. Following the display of the warning the particular process performed by the data filter program

24 ends, although it should be noted that the data filter program will continue to run continuously in the background to check whenever data is received.

[0062] If the retrieved signatory condition is not a "warn" condition then processing proceeds to the evaluation of step 6.8 wherein the condition is checked to see if it is a "block" condition. If so then at step 6.9 the data filtering program 24 acts to block the receipt of the suspect data either by preventing the computer from downloading it from the server or from merely deleting it from the user computer's local storage, and again the user is preferably informed of this action. Following this action the processing performed by the data filter program 24 ends, although again the program will continue to run in the background to detect the receipt of new data.

[0063] If the evaluation of step 6.8 does not indicate that the retrieved signature is a "block condition", then the data filter program 24 processing ends, and the program returns to running in the background, as described previously.

[0064] The first embodiment of the invention therefore presents a computer security system whereby computer viruses and the like can be detected by individual users, who transmit warnings to a server which then broadcasts warnings as appropriate to all users if the number of individual warnings received from individual users exceeds certain thresholds. The use of thresholding in the server instils in the invention a degree of order, in that it is only once a particular threshold level of warnings have been received that action is taken by the server and user computers against the suspect data. This prevents, for example, inexperienced users who may mistake acceptable data for suspect data from initiating a security alert on the acceptable data which may prevent other users from receiving that data, for example.

[0065] Furthermore, the threshold levels for both the "warn" and "block" conditions may of course be set at any level required, and may even be simply one. The assumption is made, however, that in order for the embodiment to operate in a meaningful manner the warn threshold is less than or equal to the block threshold.

[0066] Within the preferred embodiment we have described the data filter program 24 as being located at each of the user computers 15 and running thereat. However, in a slight variation of the first embodiment it would of course be possible for the data filter program 24 to also run at the server to screen messages and data routed therethrough. In this case the operation of the data filter program would be exactly the same as previously described with respect to Figure 6, however it would mean that the user computers 15 would neither need the data filter program 24, the warning receipt program 26, or the signature and ID database 28, as all blocking operations could be performed centrally. However, in the case that it is desirable for users to be warned about data to be received, then it will still be necessary to have the data filter program 24, warning receipt program 26, and signature and ID database 28 at the user computer 15 as described previously in order to allow such warnings.

[0067] In another variation of the first embodiment, in another embodiment a step equivalent to step 4.2 can be performed at the user computers before a warning message is generated. Here, after a user has identified a piece of data as suspect and instructed her computer to send a warning message to the server, prior to compiling and sending the message the warning creation program 34 controls the computer to check if the data identified as suspect by the user could possibly be a virus. Such checking is substantially identical to the check performed at step 4.2, i.e. takes the form of determining the data type to see if it could feasibly be a virus. Viruses are frequently executable files, such as .exe or .bat on a PC, or macros such as are found in Microsoft® Word® or Excel®. Such a check can conveniently be performed by looking at any file extension on the data. If such checking does not indicate that the data could be a virus then no warning message is sent, as it is assumed that the user has simply made a mistake. Instead, if the data is of a data type which could be a virus (e.g. an executable or the like) then the warning message is transmitted.

[0068] The primary advantage of the first embodiment which makes use of the server to keep track of the number of warning messages received about a particular piece of suspect data is that the network traffic overhead introduced by the security system is minimised, as users need only send warning messages to the server, which can then broadcast warning condition messages as appropriate.

[0069] A second embodiment of the invention will now be described with reference to Figures 7 to 10.

[0070] The second embodiment of the invention presents a pure "peer to peer" system which does without the server 12 of the first embodiment. Instead, within the second embodiment each user computer 15 itself individually keeps count of the number of warning messages received from its peers concerning suspect data, and in turn when a user computer detects suspect data it generates and broadcasts a warning message to all of its peers. This arrangement has the advantage that it is not dependent upon a central server for operation, and is therefore much more robust and immune to the loss of one or more of its elements than the client-server architecture of the first embodiment.

[0071] Figure 7 illustrates the software elements present at a user computer 15 in the second embodiment of the invention. More particularly, as in the first embodiment the user computer 15 is provided with a computer readable storage medium 22 on which is stored a data filter program 24, a warning creation program 72, a warning receipt program 74, and a signature and ID database 76.

[0072] Within the second embodiment the data filter program 24 and the warning creation program 72 both

continuously run in the background on the users computer at all times. The data filter program 24 operates in exactly the same manner as previously described in the first embodiment with respect to Figure 6, that is it acts to check the signature of received data against signatures of suspect data stored in the signature database 76, and takes action against any received suspect data as necessary. In this respect, the operation of the second embodiment is exactly the same as that of the first embodiment. The difference between the second embodiment and the first embodiment lies, however, in the fact that each user computer 15 directly broadcasts warnings to each of its peer computers (i.e. the other user computers 15), and in addition each user computer 15 also keeps its own track of the count of warning messages received against each piece of suspect data. These functions are performed by the warning creation program 72 and the warning receipt program 74, as described next.

[0073] The operation of the warning creation program 72 is shown in Figure 8. Here, at step 8.1 data is received at the user computer 15 over the network 10. At step 8.2 an evaluation is performed to determine whether the received data is a security threat or not. As with the first embodiment, this evaluation is preferably performed by the actual human user of the computer 15 who employs her expertise to determine whether or not the received data is a computer virus or the like. In alternative embodiments, however, it may be that this evaluation could be performed by a software agent installed on the users computer, the software agent acting to monitor the functions of the computer to determine if it is under attack from a computer virus. Within the first and second embodiments, however, we prefer the use of a human operator to perform the evaluation.

[0074] If the evaluation at step 8.2 concludes that the received data is not a threat then the processing performed by the warning creation program on the received data ends and the program returns to running in the background until further data is received in the future.

[0075] If it is determined at step 8.2 that the received data is a threat, then at step 8.3 a data signature for the received data is created. As in the first embodiment, the signature generation function may be a hash function or the like or any other suitable signature generation function which is capable of repeatably generating the same signature for a given set of input data. Furthermore, it should be noted here that each user computer 15 must use the same signature generation function, such that when the same set or piece of suspect data arrives at any of the user computers, each of them creates the same signature therefor.

[0076] At step 8.4 a check is made to see whether the generated data signature has already been stored in the signature and ID database 76, and if not then at step 8.8 the signature is stored in the database, and is allocated a globally unique ID. The ID for the signature is generated by the user computer, preferably from a list of ID's which have been specifically allocated to that user computer for allocation to signatures thereby.

[0077] Following step 8.8, at step 8.9 a message count is instantiated for the particular signature generated by the received data, and is set to the value one. As in the first embodiment, the various count values for each piece of suspect data may be stored in a one dimensional array indexed by signature ID.

[0078] Following step 8.9 at step 8.10 the warning creation program causes the user computer to broadcast the data signature and the signature ID to all of its peers as a warning message. As in the first embodiment, the warning message preferably takes a known format, with the signature and signature ID as particular fields within the warning message. Following the broadcast of the warning message, at step 8.10 processing of the warning creation program ends, and it returns to run in the background.

[0079] Returning to the evaluation at step 8.4, if it is determined that the data signature for the received data has already been stored (i.e. the data has already been identified as suspect by either the present user, or one of the other users who have then broadcast a warning message which has been received at each user computer) then processing proceeds to step 8.5, wherein the message count maintained for the particular signature is incremented to reflect the fact that the data has in fact been identified once again as a threat.

[0080] Next, at step 8.6 the signature ID is broadcast to every other user in the system as part of a warning message, and then following that at step 8.7 the appropriate warning or block conditions are set for the signature based on the warning and block threshold values. The specific operations required by step 8.7 are shown in more detail in Figure 10.

[0081] With reference to Figure 10, the steps required for setting the warning or block conditions for a particular data signature are as follows. At step 10.1 the message count for the particular data signature is retrieved, by indexing into the counter array using the signature ID. Then, at step 10.2 the retrieved message count is compared against the block threshold, and if larger than the threshold at step 10.3 the security condition for the data signature is set to "block". As in the first embodiment, this security condition is stored in the signature and ID database 76, and is used by the data filter program 24 to filter received data.

[0082] If the warning message count for the particular data signature is not greater than the block threshold then the message count is compared against the warning threshold. If it is not greater than the warning threshold then no warning or block condition is set for the particular data signature. In contrast, if it is greater than the warning threshold then the security condition for the particular data signature is set to "warn", which is again used by the filter data program 24 as described before in respect of the first embodiment.

[0083] Returning to Figure 8, once the warning or

block conditions have been set for the data signature, the specific processing performed by the warning creation program 72 ends, and it returns to running in the background until further data is received.

[0084] Turning now to the operation of the warning receipt program 74, this program operates whenever a warning message generated by a warning creation program stored on another user's computer is received. The operation of the warning receipt program 74 is shown in Figure 9.

[0085] At step 9.1, a warning message is received from one of the other user computers. The warning message is of course in a predetermined format which identifies it as such. Then, at step 9.2 the warning receipt program 74 acts to parse the message to see if it contains a data signature relating to suspect data. Recall here that the data will have been determined as suspect by the user of the user computer from which the message was received. If it is determined that the message does not contain a signature, then this means that the data will have already been determined as suspect previously, and hence should already be stored in the user computer's signature and ID database 76. Therefore, at step 9.3 the warning receipt program acts to retrieve the message count for the particular suspect data to which the warning message relates, using the signature ID which will have been included in the warning message (recall here that at step 8.6 in the warning creation program where the suspect data has already been identified as such a warning message is still sent out including the suspect data signature ID). Then, at step 9.4 the retrieved message count for the suspect data is incremented to reflect the additional warning, and at step 9.5 the warning or block conditions for the data signature are also set, following the procedure of Figure 10, as described previously.

[0086] If it is determined at step 9.2 that the message does contain a signature, then processing proceeds to the evaluation of step 9.6 wherein it is determined whether or not the received signature has already been stored. If not processing proceeds to step 9.7 wherein the signature is stored in the signature and ID database 76 indexed by the signature ID included in the received warning message. At step 9.8 a message count is instantiated for the newly stored data signature, and is set to one. As described previously, the count values for each particular signature can be stored in a one dimensional array, indexed by each signature ID.

[0087] Following step 9.8 processing proceeds to step 9.5, wherein the warning or block conditions are set for the particular signature, using the procedure of Figure 10.

[0088] Returning to step 9.6, if it is determined that the signature received in the warning message has already been stored then processing proceeds to step 9.9, wherein the received signature ID is stored for use as another key to the signature. In such a case a count value should be instantiated related to the received signature ID, with its value set to the same value as the count indexed by the ID of the already stored signature which matches the received signature. Then, at step 9.10 both of the count values stored for each ID (the received ID, and the ID of the previously stored ID which matches the received signature) are incremented to reflect the receipt of the additional warning. Finally, following the increment of the message counts for the signature the warning and block conditions are set for the signature once more, using the procedure of Figure 10.

[0089] Therefore, as described the second embodiment provides a pure peer to peer system which incorporates the distributed warning capabilities of the security system of the present invention with the ability to threshold the warnings so as to be able to provide a graded reaction locally at each user computer. The provision of such a peer-to-peer operation provides for a more robust system, but as mentioned earlier might have detrimental network overheads due to the need to broadcast every warning message to every user.

[0090] As a modification to the operation of the second embodiment, an additional checking step analogous to step 4.2 of the first embodiment may be performed between steps 8.2 and 8.3 of Figure 8 depicting the operation of the warning creation program 72. Here, after the human user has identified data as suspect at step 8.2, an additional check is made to see if the data could possibly be a computer virus. Such checking preferably takes the form of determining the data type to see if it could feasibly be a virus. Viruses are frequently executable files, such as .exe or .bat on a PC, or macros such as are found in Microsoft® Word® or Excel®. Such a check can conveniently be performed by looking at any file extension on the data. If such checking does not indicate that the data could be a virus then processing ends, as it is assumed that the user has simply made a mistake. Instead, if the warning message is in fact a true warning and the suspect data could possibly be a virus, then processing proceeds to step 8.3, and onwards as previously described. The operation of the other programs in the second embodiment remains unchanged.

[0091] It should be noted here that as the second embodiment does not require a server to act centrally, but rather is a true peer-to-peer system, it is particularly suitable for implementation on networks such as ETHERNET and the like.

[0092] Various further modifications may be made to either of the first or second embodiments to produce further embodiments, as described next.

[0093] In further embodiments it is possible to add more sophistication to increase the power of the system. For example, users could be given the ability to respond to a warning to inform the server of whether they agree with the assessment or not - this would give the potential of reducing the count number if appropriate. The decision as to whether a particular program was a threat or not would then be a collective decision from the whole community.

[0094] Furthermore it is also possible to build in trust models for each user such that some users have greater influence than others. Such trust models could be predefined according to the users position within the community/organisation, such that a warning received from a user with a high trust rating increments the signature count for a particular piece or set of suspect data more than another user with a lower trust rating. Each trust model may then also be adaptively varied by judging the accuracy of a prediction that a user has made against a final collective decision, and any further warnings registered by that user could be weighted according to the trust in that users demonstrated decision-making ability.

[0095] In other embodiments a community of organisations could be formed that would extend the collaboration beyond the limits of a single organisation through sharing of knowledge. Any discoveries in one organisation could be passed on to all other organisations in the community and vice versa. This would greatly extend the power of the system and in many cases allow rapid response to intrusions experienced in other organisations without any internal hits being experienced to the remaining organisations using the system.

[0096] Moreover, in another embodiment the actual number of warnings received concerning a particular piece or set of suspect data could be displayed to the user if the suspect data is received at the user's computer, and the user could then decide what action to take himself. Such a scheme is akin to the "warn" action of the primary embodiments, but provides the user with the additional information of how many other users have determined the data to be threat, thus providing the user with an assessment analogous to a community "confidence-level" in the data, such a level being of greater resolution than the simple "warn" or "block" conditions. Furthermore, such a scheme would not require the system to actually perform automatic blocking of suspect data, as this could be left to the user to do himself. This mode of operation provides the additional potential advantage that a highly skilled or confident user may not wish to act on the community warning if he is sure that it may be wrong, thus increasing the flexibility of the system.

[0097] Furthermore, in another embodiment based on the peer-to-peer architecture of the second embodiment, rather than broadcast to all peers e.g. all users in BT - sub-communities of peers are formed so that the initial broadcasting of warning messages concerning a particular is limited to a smaller number of peers. An example of such an architecture is shown in Figure 11, whence it will be seen that a plurality of sub-communities 111, 112, 113, and 114 may be provided, each connected to each other. Only when the count passes the warning threshold in one of the sub-communities is the suspect data signature distributed further. This limits the overheads incurred by false positives in that any particular sub-community would need to agree that a particular piece or set of data was a threat before it was broadcast to the wider community. "Crossover points" are provided between the sub-communities so that inter-group (or alternatively "inter-community") warnings may be passed on to other sub-communities by the provision of one of the peers 115 in a particular sub-community acting as the bridge via a communications link to another peer in a neighbouring sub-community, and only communicating with that other peer when the warning (or block) threshold has been passed in its own parent sub-community i.e. the sub-community has decided that the particular piece or set of suspect data really could be a threat.

[0098] The processing steps involved in such an embodiment are almost identical to those required in the second embodiment, with the addition that the processing steps depicted in Figure 10 are altered to include an additional step after step 10.3 or 10.5 wherein once the filter condition has changed (as a consequence of either step 10.3 or 10.5) if the computer is a bridging peer to another sub-community then an inter-group warning message is sent to the connected sub-community informing the other peer that a change in the filter condition has occurred for a particular piece or set of suspect data. Such an inter-group warning message should preferably include at least the data signature, as well as a warning condition indicator to communicate the threat level (e.g. warn or block) decided by the sending sub-community. By sending such a warning message to other sub-communities, the bridging peer is effectively declaring to the other sub-communities to which it is connected that its peers in its own sub-community have found the suspect data identified in the warning message to be a security threat.

[0099] Conversely, when an inter-group warning message is received at a bridging peer from another sub-community, the bridging peer should broadcast the inter-group warning message onwards to all its own peers as an intra-group message, to inform them of the threat assessment made by the neighbouring community. Each peer can then store the suspect data signature in its signature and ID database 76, for future reference by the data filter program 24 as needed (and in the same manner as already described in relation to the second embodiment).

[0100] Whether or not a user computer in a sub-community uses a forwarded inter-group warning message directly to initiate action (such as "warn" or "block" for the indicated suspect data) is optional, as it could be that it may wish to wait until it's own community has also agreed on action in respect of the same suspect data before actually initiating a "warn" or "block" condition for the data. Conversely, where a particular user-computer's own sub-community has already come to agreement on a particular piece or set of suspect data (i.e. either the "warn" or "block" threshold has been passed), then it may be that the user-computer may wish to wait until one or more connected sub-communities have also made the same finding before setting the appropriate

condition in its database for the data. It will be apparent that there are a multitude of possibilities as to when any particular user computer may wish to actually set an action condition for a particular piece or set of suspect data. For example, an action condition may be set for a particular piece or set of suspect data only if a user computer's own sub-community has agreed on the action in addition to a threshold number of other sub-communities, or alternatively if merely a threshold number out of the total number of sub-communities has agreed on the action regardless of the particular user computer's own sub-community's finding. Other such alternatives representing possible combinations of the logical conditions required in order to set an action condition will be apparent to the intended reader.

[0101] It is clear that such an architecture of sub-communities is analogous to a plurality of inter-connected local area networks, although it is not essential that each peer in a particular sub-community need actually be located in the same actual LAN (although this may be preferable). This is because all that defines any particular sub-community is the common cause that upon detecting suspect data a user broadcasts a message to all of its peers in the same sub-community, and hence all that is required is some message transport mechanism between peers. Such a message transport mechanism may be a single LAN to which all the peers are connected, or may be a sequence of interconnected networks such as the Internet, with different peers in a sub-community actually being part of different networks.

[0102] The modification to the second embodiment to split the user computers into sub-communities provides the additional advantage over the basic second embodiment that it, at least partially, overcomes the primary disadvantage of the high signalling bandwidth which would be required for the transmission of the warning messages to every other peer, and hence allows for scalability to add more users. As an example, consider the case where a community consists of n peers, such that when a warning message is broadcast from one of them to each of the others $(n-1)$ individual intra-group warning messages are sent. Now, assume that in order to agree on action about a particular piece or set of data a proportion of $1/a$ peers have to agree i.e. the action threshold is n/a. In such a case a total of $n(n-1)/a$ intra-group warning messages have to be transmitted in total for any one particular piece or set of data. Thus, where there are 40 peers, and half of them have to agree before action can be taken, a total of 780 warning messages would have to be transmitted (without error) before the action is agreed upon.

[0103] Now consider that the community of n peers is split equally into x sub-communities, as shown in Figure 11. In this case, within a sub-community for any single warning from one peer a total of $(n/x - 1)$ intra-group warning messages are broadcast. Assuming that in order to agree on action about a particular piece or set of data a proportion of 1/a peers have to agree i.e. the action threshold is n/ax, then a total of $n(n-x)/ax^2$ intra-group warning messages have to be broadcast within a single sub-community for action to be agreed on a single piece of suspect data. Assuming that each sub-community has to reach agreement within itself and communicate to each other group that it has so agreed before action can be taken by any peer in any sub-community, there will still only be a total of $n(n-x)/ax$ intra-group warning messages broadcast in total (the sum of the number of warning messages in total broadcast in each sub-community), plus $x(x-1)$ inter-group messages between sub-communities, plus $(n/x - 1)$ further intra-group messages from the bridging peer in each sub-community to each other peer in its respective community informing them of the findings of other sub-communities (i.e. forwarding received inter-group messages onwards). Note that with respect to this latter number of intra-group messages, $(n/x - 1)$ messages are sent from a bridging peer in a sub-community for each inter-group warning message received from another sub-community, such that where full knowledge of every other sub-community's findings about a particular piece or set of suspect data is required to be disseminated throughout a particular sub-community, the total number of intra-group messages transmitted by a single bridging node forwarding on the findings of other sub-communities would be $(x-1)(n/x - 1)$. This leads to the total number of intra-group messages forwarding inter-group messages across the entire community of users of $x(x-1)(n/x-1)$.

[0104] For example, if there are 40 peers split into 2 sub-communities of 20, then in each sub-community 190 messages are sent before action is agreed (i.e. the action threshold is reached) in each sub-community, 2 messages are exchanged between the respective bridging peers of each sub-community, one in each direction, and then the respective bridging peers each send 19 messages to inform their own peers of the finding of the other sub-community, to give a total of 420 messages. This is substantially less than the 780 messages which were required when the peers were not split into sub-communities.

[0105] It should be apparent that the above numeric examples are dependent solely on the logical conditions which were ascribed to the group operation, and in particular in the example above the need for every sub-community to agree on action about a particular piece or set of data and each issue its own inter-group warning message before any action is taken. Where slightly lower thresholds are set for action (e.g. if only half the number of sub-communities are required to agree), then the number of both inter- and intra-group warning messages which would be required to be broadcast before action is agreed would be further reduced.

[0106] In any of the embodiments of the invention described above and in other embodiments not described it should be noted that the one or more "threshold values" against which the respective counts for pieces or sets of suspect data are compared may be any value,

and may be fixed or dynamically adaptive. Furthermore the actual values may be generated by any appropriate function, such as for example by a random number generator, or as a function of some metric of accuracy of users in identifying viruses. Furthermore, it is foreseen that the threshold values may not simply be actual numbers which the respective counts must be greater than, but could instead be probabilistic in nature. Furthermore, various logical functions or rules can be used for their creation, including for example, fuzzy logic. The threshold values may therefore be any values, derived by any function or method, and are not limited to the particular examples of threshold values given in the specific embodiments described herein.

[0107] As described, the present invention therefore provides a collaborative computer security system wherein a distributed decision is taken involving all or substantially all of the users of the system, to effectively allow the users to "vote" on whether or not a particular piece or set of suspect data is a threat. By providing such a distributed system the response time to a computer virus attack can be improved over the prior art case where a central authority is required to perform analysis and take action, thereby reducing the potential damage which may be caused by such an attack.

Claims

1. A computer security system for use in a network environment comprising at least a first group of user computers arranged to communicate over a network, the system comprising a warning message exchange system operable to allow the communication from the user computers of warning messages relating to suspect data identified as a possible security threat; a message counting system operable to maintain a count for every particular piece or set of suspect data based on the number of warning messages communicated relating thereto; and network security means operable to act in respect of any particular piece or set of suspect data for which the count maintained therefor is substantially equal to or greater than at least one threshold value.

2. A computer security system according to claim 1, wherein the judgement as to whether any piece or set of data present at any one or more of the user computers is a possible threat is performed by the respective human user or users of the or each user computer.

3. A computer security system according to any of the preceding claims, wherein the network security means is further operable to compare the maintained count for a particular piece or set of suspect data against a plurality of different thresholds, and to take different action in respect of the data dependent upon which threshold or thresholds are met.

4. A computer security system according to any of the preceding claims, wherein the action taken by the network security means comprises warning each of the users as to the suspect nature of a particular piece or set of data.

5. A computer security system according to any of the preceding claims, wherein the action taken by the network security means comprises preventing the transmission of the particular piece or set of suspect data across the network.

6. A computer security system according to any of the preceding claims, wherein the or each warning message comprises at least an identifier of the piece of or set of suspect data to which it relates.

7. A computer security system according to claim 6, wherein the identifier is the piece or set of suspect data itself.

8. A computer security system according to claim 6, wherein the identifier is a repeatably generatable signature substantially unique to the piece or set of data.

9. A computer security system according to any of the preceding claims, and further comprising a network server arranged to receive the or each warning message communicated from the user computers, and further arranged to host the message counting system and at least part of the network security system, wherein that part of the network security system hosted by the server is operable to determine the action which should be taken against a particular piece or set of suspect data, and to communicate an action condition indicator indicative of the determined action to each of the plurality of users.

10. A computer security system according to any of the preceding claims, wherein a warning message generated by a user computer is broadcast to every other user computer, and each user computer is further arranged to host its own message counting system and network security system operable in accordance with any of the preceding claims.

11. A computer security system according to any of the preceding claims, and further comprising an inter-group communications system operable to allow the communication of warning messages relating to suspect data identified as a possible security threat from the first group of user computers to one or more other groups of other user computers.

12. A computer security system according to claim 11, wherein the inter group warning messages are transmitted by the inter-group communications system if the number of warning messages communicated by the user computers in the first group concerning a particular piece or set of data is substantially equal to or greater than at least one threshold value.

13. A computer security system according to claim 12, wherein the threshold value or values is/are the same as those at which the network security system takes action.

14. A computer security system according to any one of claims 11, 12, or 13, wherein the inter-group communications system is further arranged to receive inter-group warning messages from the one or more other groups of user computers.

15. A computer security system according to claim 14, and further comprising a second message counting system operable to maintain a count for every particular piece or set of suspect data based on the number of inter-group warning messages communicated relating thereto; and wherein the network security means is further operable to act against any particular piece or set of suspect data for which the count maintained therefor by the second message counting system is substantially equal to or greater than at least one inter-group threshold value.

16. A computer security system according to claim 15, wherein the network security means is further arranged to act against a particular piece or set of suspect data only if both the respective counts maintained by the message counting system and the second message counting system are substantially equal to or greater than at least the threshold value and the inter-group threshold value respectively.

17. A computer security system according to any of the preceding claims, wherein the message counting system is further arranged to store one or more weighting coefficients relating to one or more particular user computers, and to increment the count maintained for a particular piece or set of suspect data by an amount based upon the weighting coefficient when a warning is received from the one or more particular user computers relating to the particular piece or set of suspect data.

18. A computer readable storage medium storing one or more computer programs which when run on a computer causes the computer to operate in the manner of a system according to any of the preceding claims.

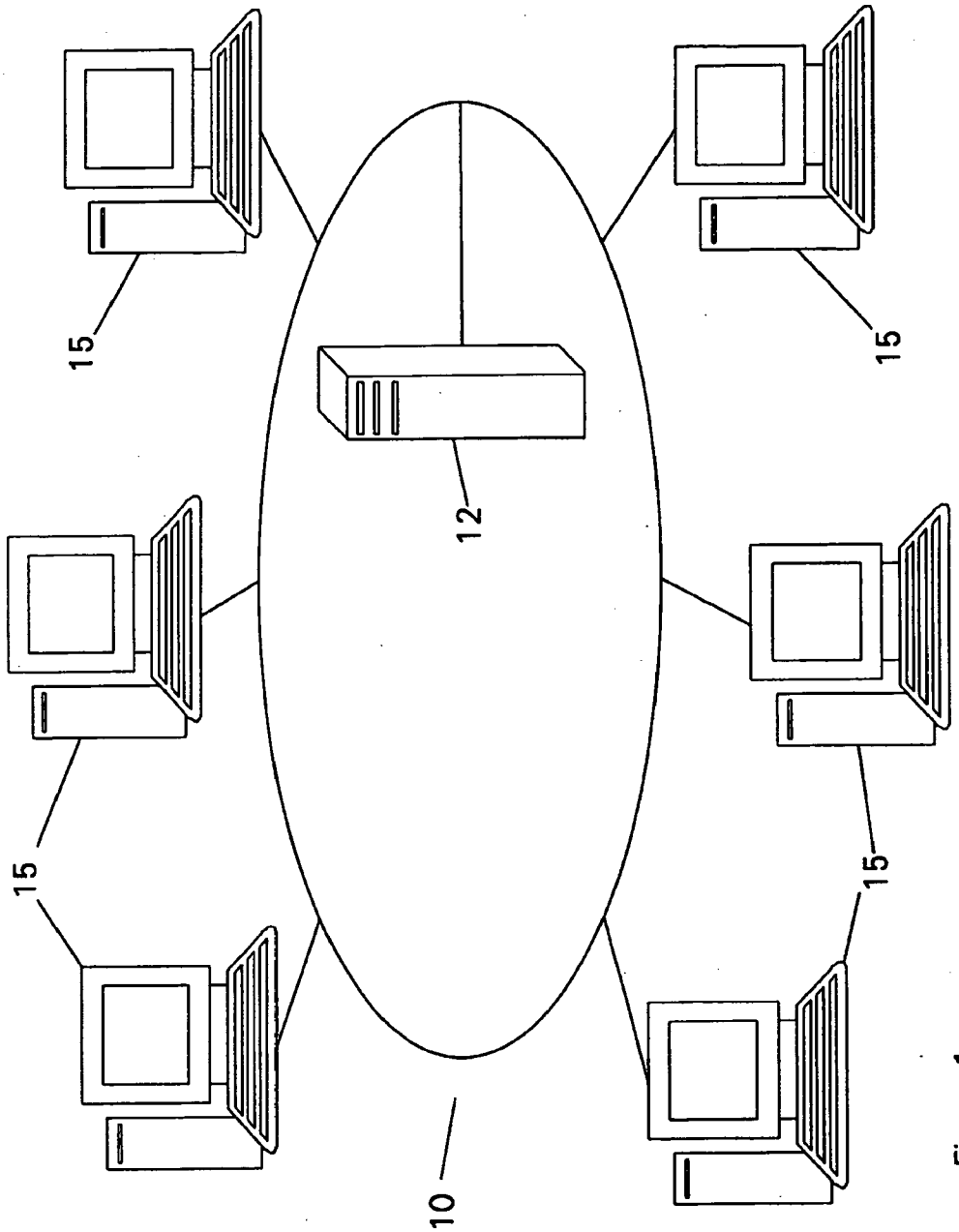


Figure 1

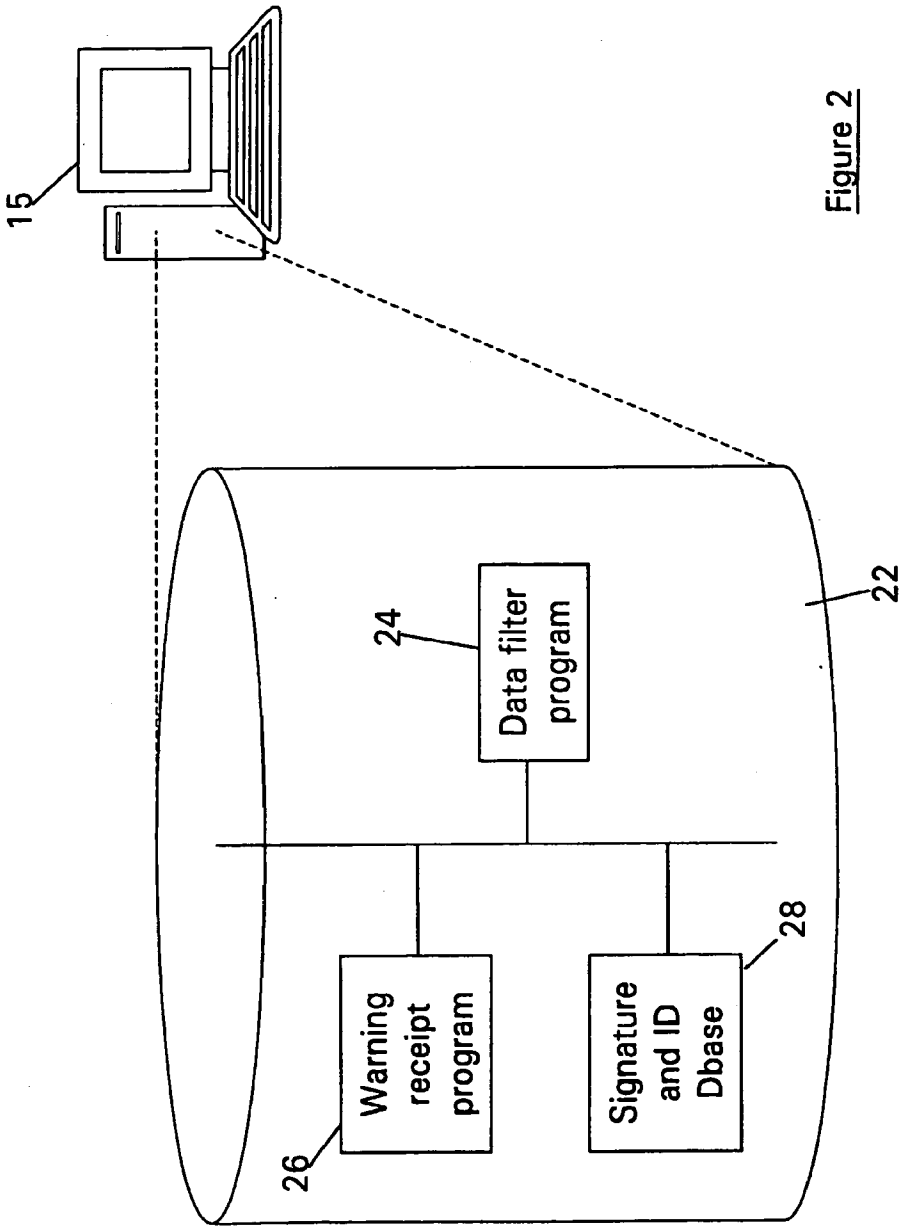

15
22
24
Data filter program
26
Warning receipt program
28
Signature and ID Dbase

Figure 2

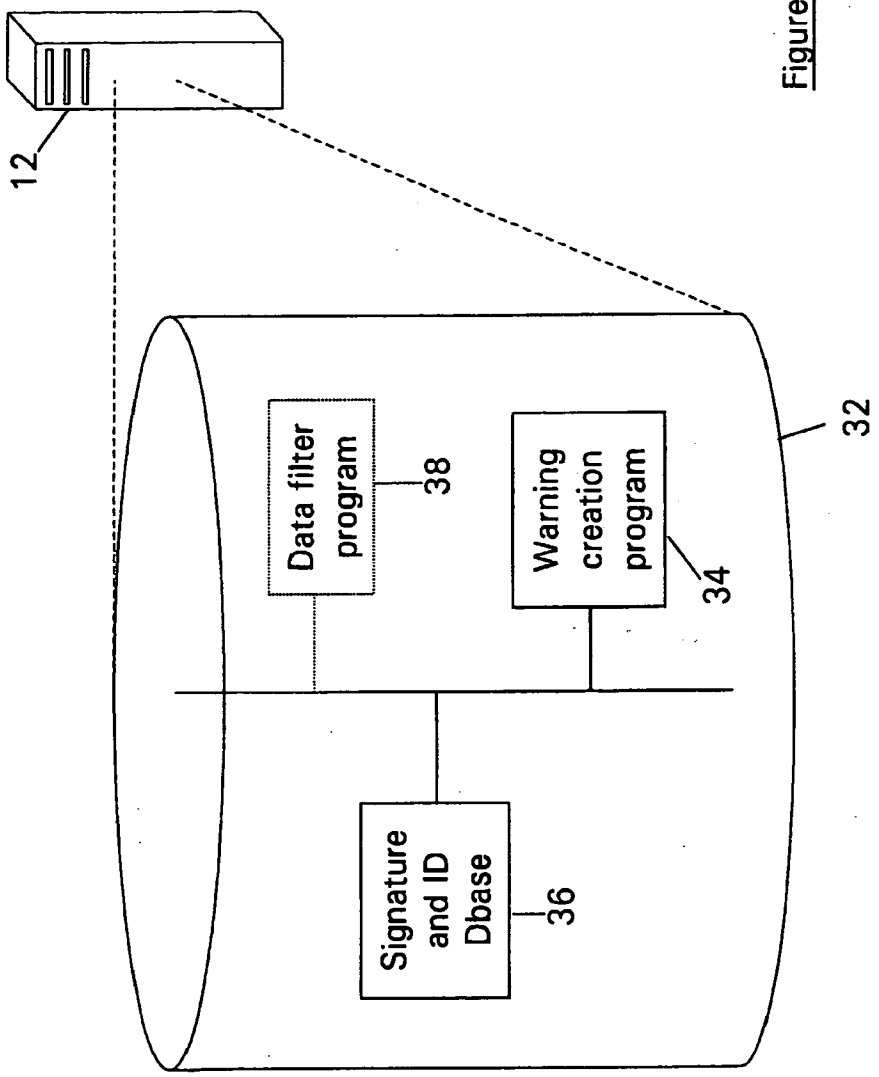

12
32
Data filter program
38
Warning creation program
34
Signature and ID Dbase
36

Figure 3

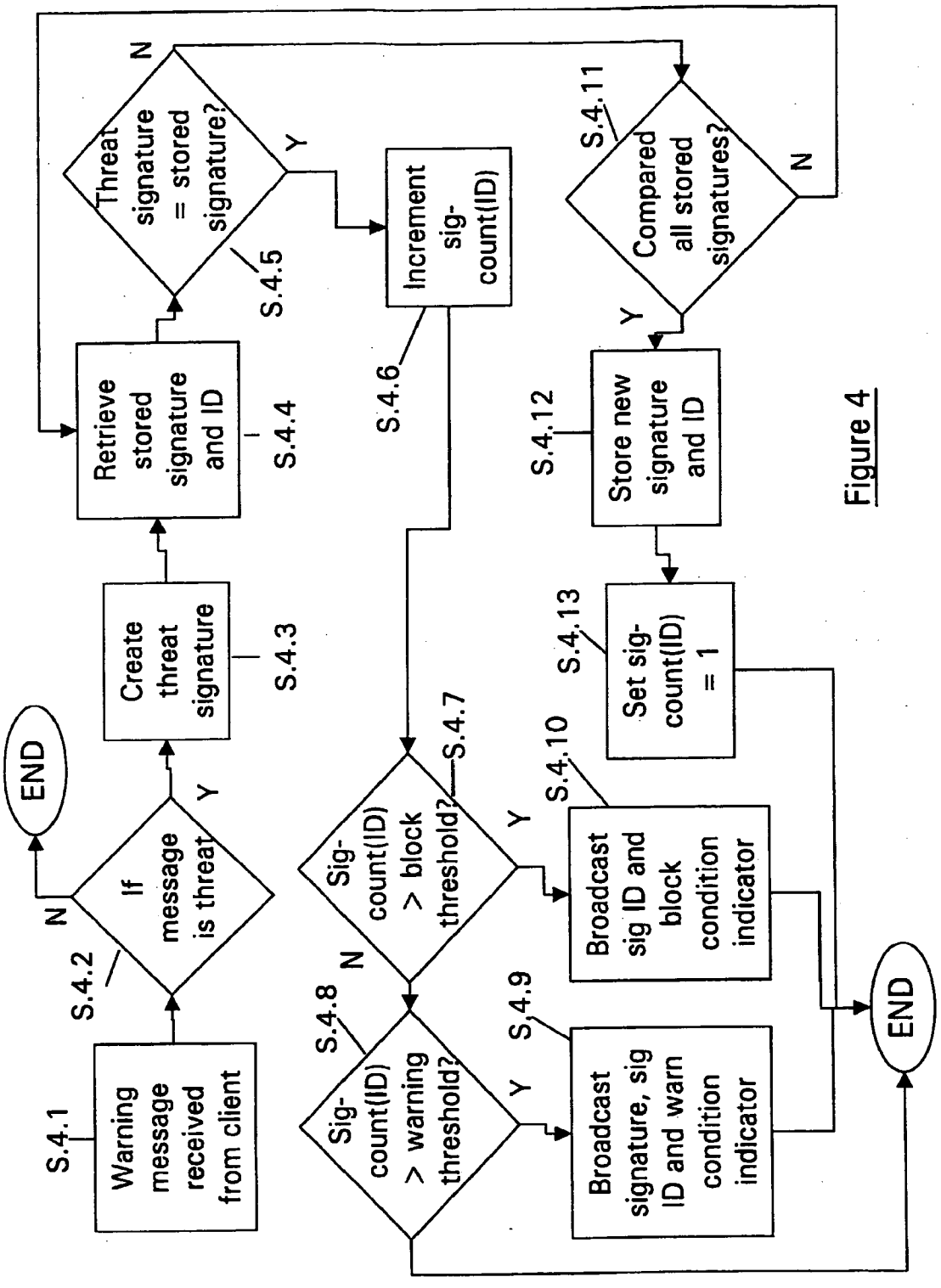

Warning message received from client
S.4.1
If message is threat
S.4.2
N
Y
END
Create threat signature
S.4.3
Retrieve stored signature and ID
S.4.4
Threat signature = stored signature?
S.4.5
N
Y
Increment sig-count(ID)
S.4.6
Sig-count(ID) > warning threshold?
S.4.8
N
Y
Broadcast signature, sig ID and warn condition indicator
S.4.9
Sig-count(ID) > block threshold?
S.4.7
Y
Broadcast sig ID and block condition indicator
S.4.10
Set sig-count(ID) = 1
S.4.13
Compared all stored signatures?
S.4.11
N
Y
Store new signature and ID
S.4.12
END

Figure 4

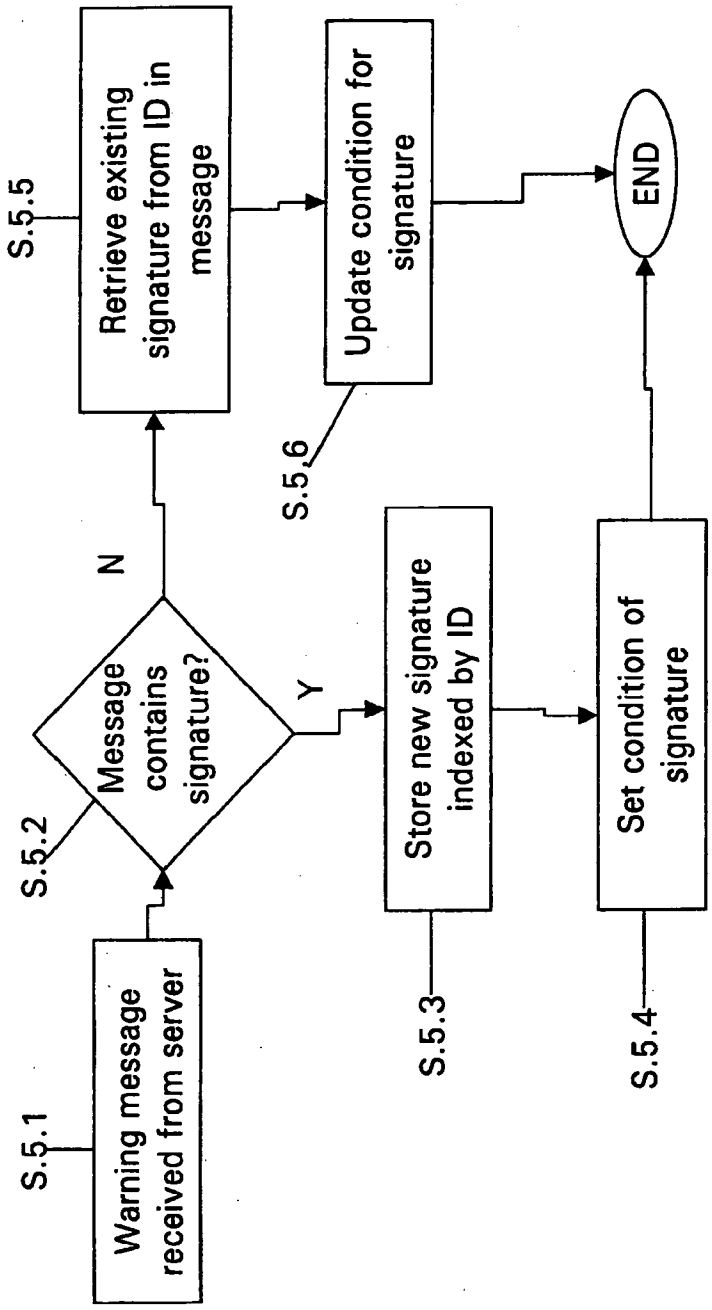

Warning message received from server
S.5.1
Message contains signature?
S.5.2
N
Y
Retrieve existing signature from ID in message
S.5.5
Update condition for signature
S.5.6
Store new signature indexed by ID
S.5.3
Set condition of signature
S.5.4
END

Figure 5

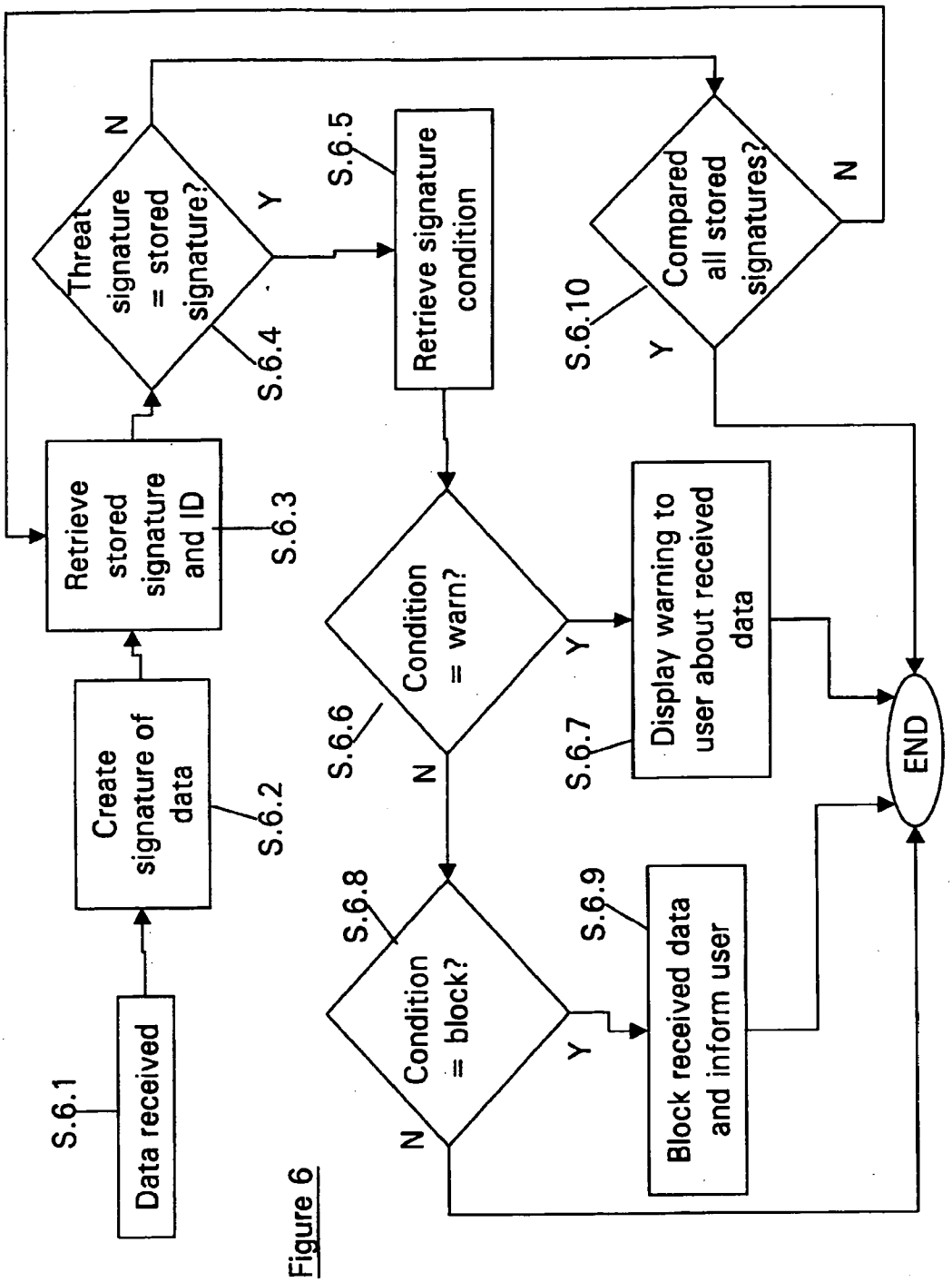

Data received
S.6.1
Create signature of data
S.6.2
Retrieve stored signature and ID
S.6.3
Threat signature = stored signature?
S.6.4
N
Y
Retrieve signature condition
S.6.5
Condition = warn?
S.6.6
Y
N
Display warning to user about received data
S.6.7
Condition = block?
S.6.8
Y
N
Block received data and inform user
S.6.9
Compared all stored signatures?
S.6.10
N
Y
END

Figure 6

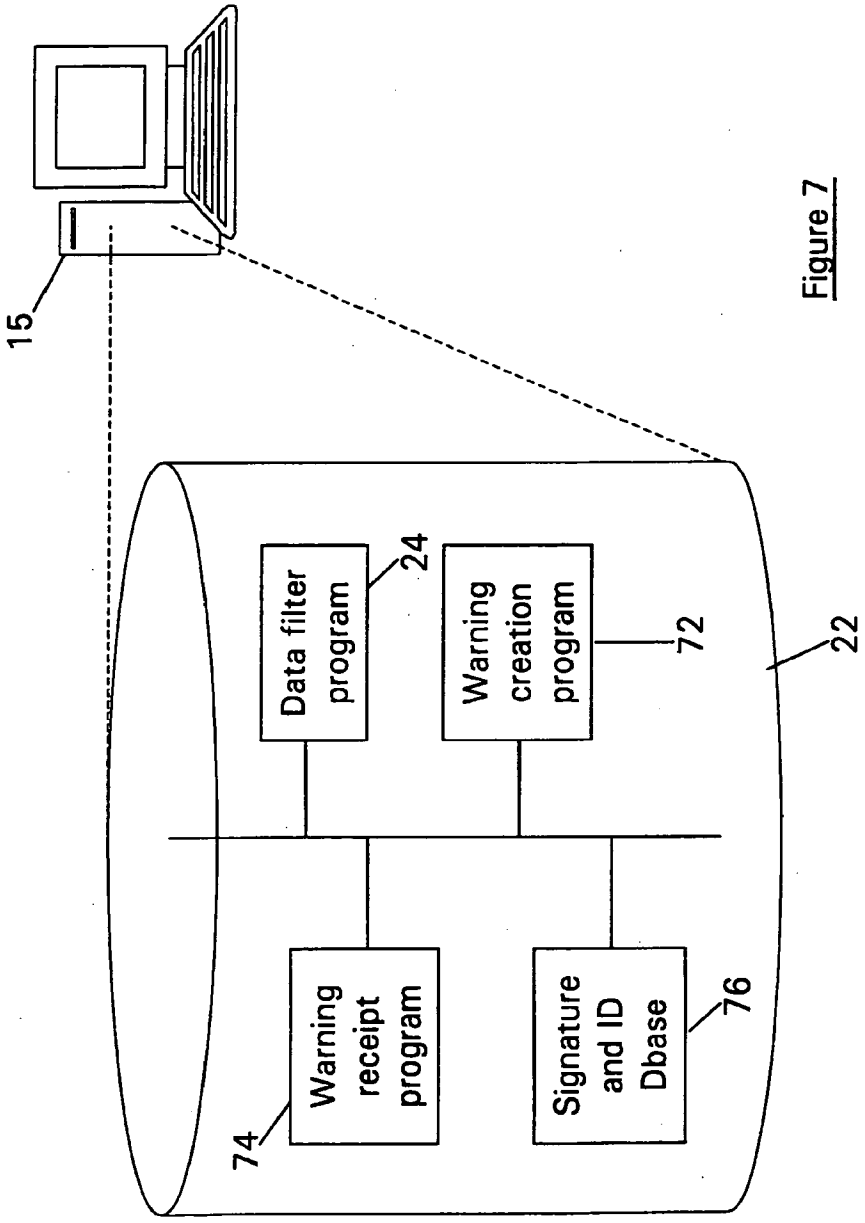

15
Data filter program
24
Warning creation program
72
22
Warning receipt program
74
Signature and ID Dbase
76

Figure 7

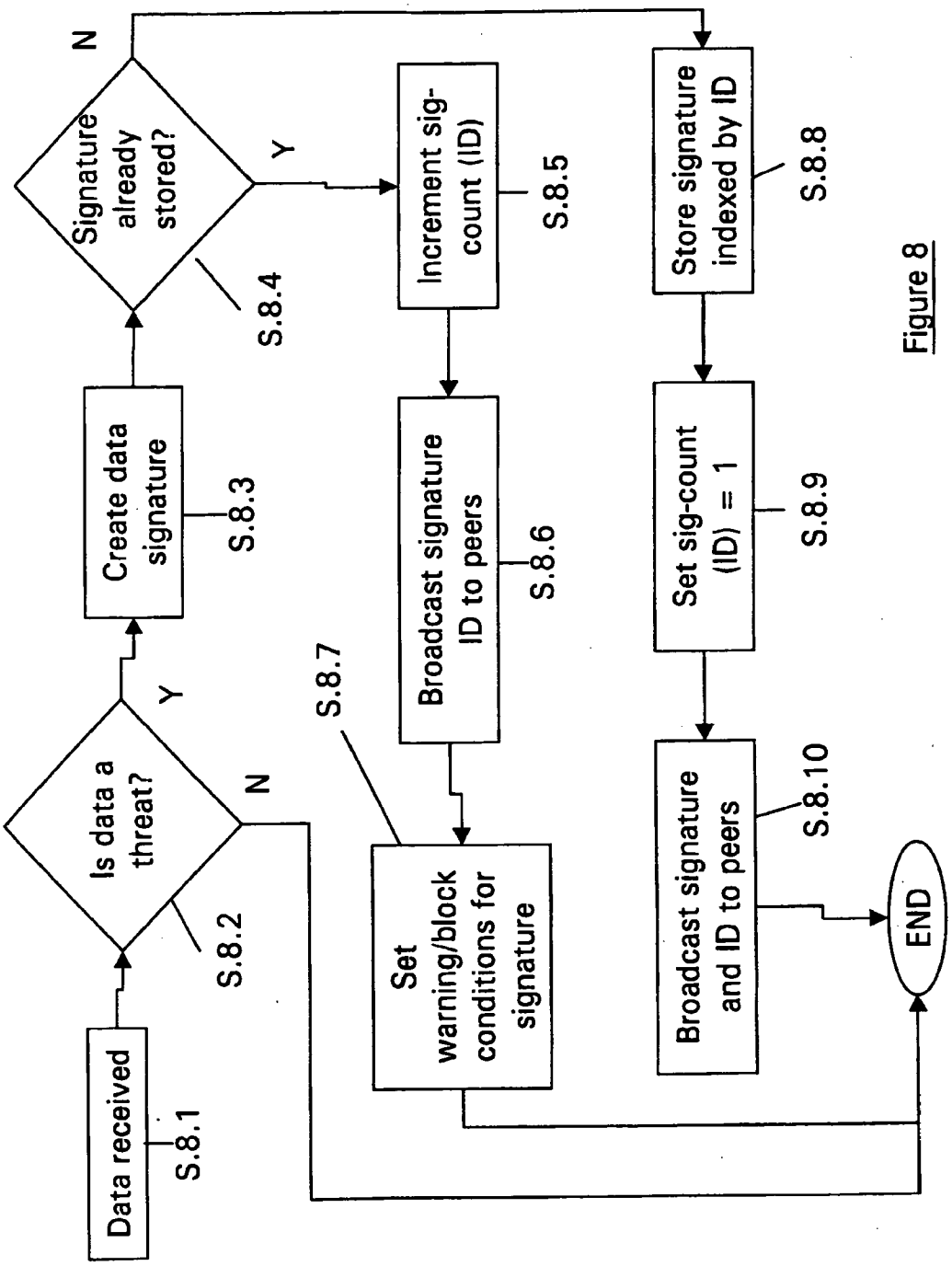

Data received
S.8.1
Is data a threat?
S.8.2
N
Y
Create data signature
S.8.3
Signature already stored?
S.8.4
N
Y
Increment sig-count (ID)
S.8.5
Store signature indexed by ID
S.8.8
Broadcast signature ID to peers
S.8.6
Set sig-count (ID) = 1
S.8.9
S.8.7
Set warning/block conditions for signature
Broadcast signature and ID to peers
S.8.10
END

Figure 8

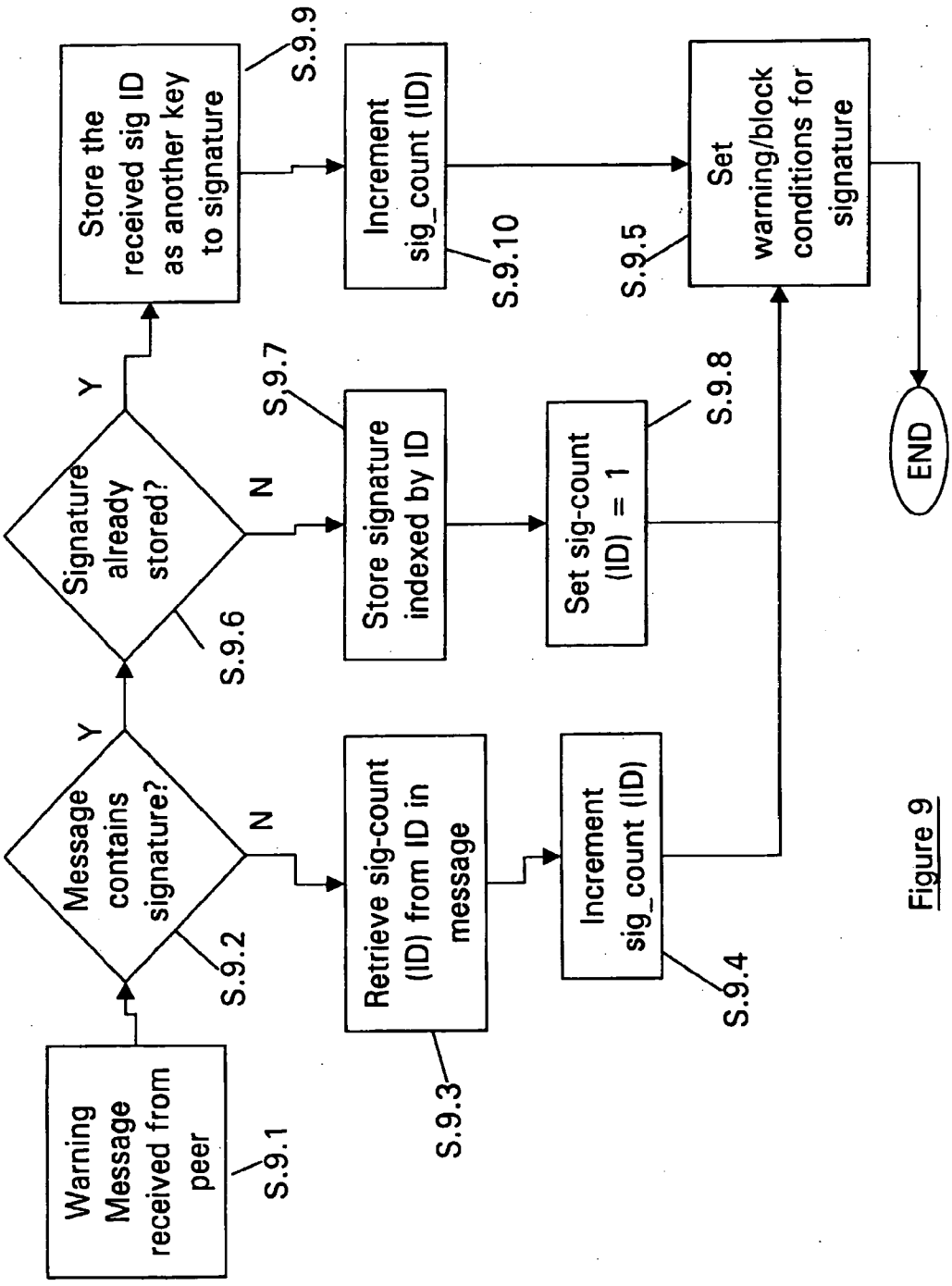

Warning Message received from peer
S.9.1
Message contains signature?
S.9.2
Y
N
Retrieve sig-count (ID) from ID in message
S.9.3
Increment sig_count (ID)
S.9.4
Signature already stored?
S.9.6
Y
N
Store signature indexed by ID
S.9.7
Set sig-count (ID) = 1
S.9.8
Store the received sig ID as another key to signature
S.9.9
Increment sig_count (ID)
S.9.10
Set warning/block conditions for signature
S.9.5
END

Figure 9

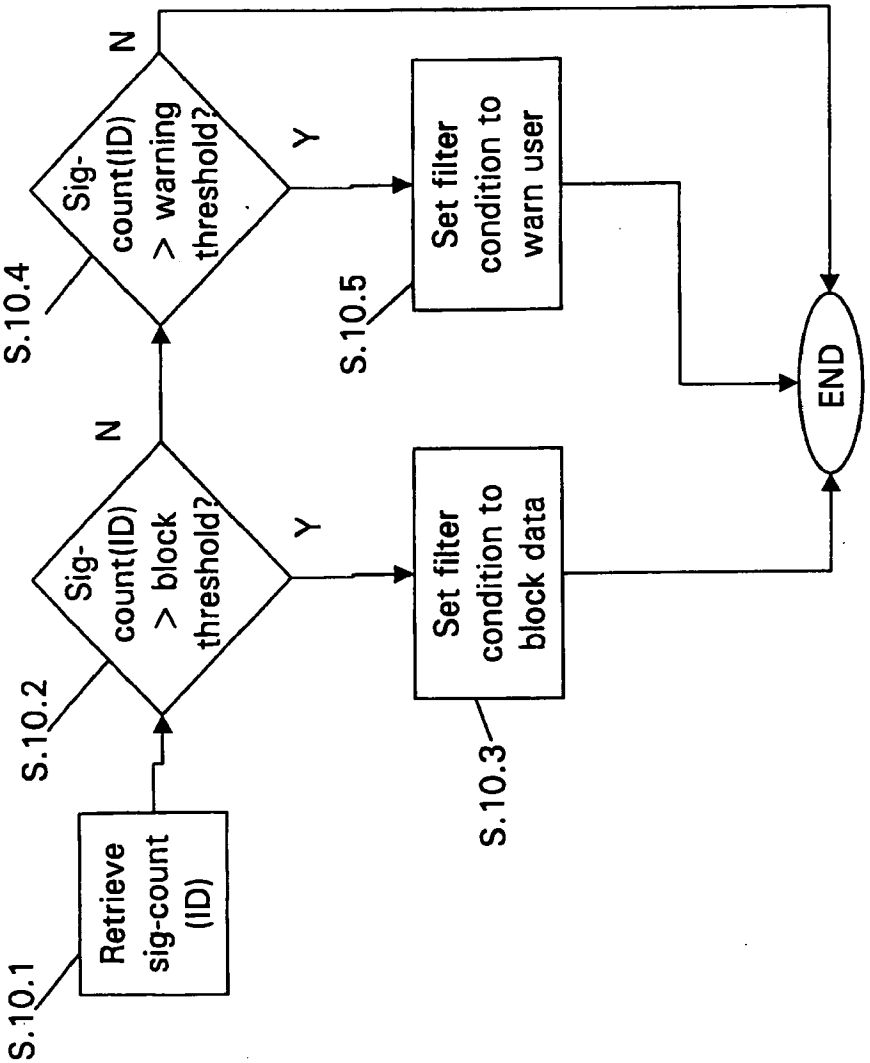

Retrieve sig-count (ID)
S.10.1
Sig-count(ID) > block threshold?
S.10.2
Y
N
Set filter condition to block data
S.10.3
Sig-count(ID) > warning threshold?
S.10.4
Y
N
Set filter condition to warn user
S.10.5
END

Figure 10

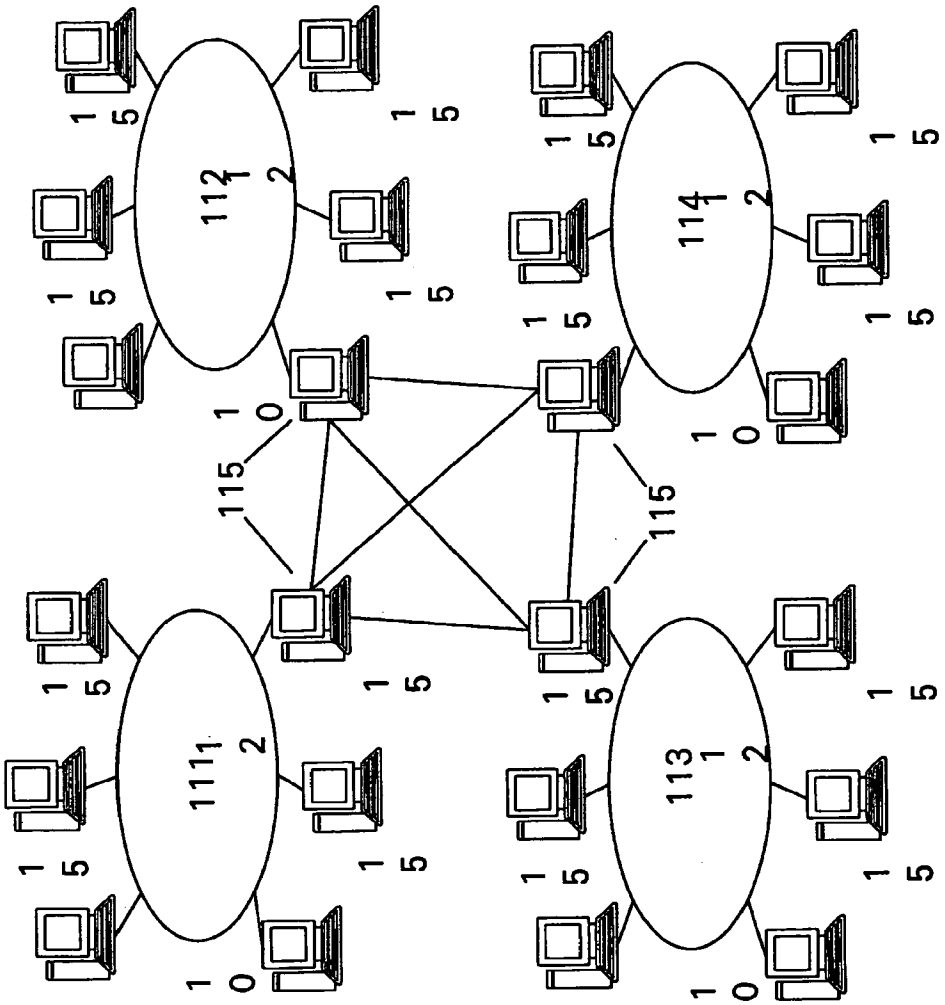


Figure 11

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 30 9795

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	US 5 440 723 A (KEPHART JEFFREY O ET AL) 8 August 1995 (1995-08-08) * abstract; figures 2,3,7,8 * * column 4, line 29 - column 5, last line * * column 19, line 45 - column 20, last line *	1	G06F1/00 H04L29/06
A		2,3,5-8, 10-14	
A	US 5 621 889 A (EMERY THIERRY ET AL) 15 April 1997 (1997-04-15) * abstract; figure 2 * * column 2, line 9 - line 55 * * column 3, line 30 - column 4, line 43 *	1-3,17	
A	WO 01 71499 A (COUNTERPANE INTERNET SECURITY) 27 September 2001 (2001-09-27)		
A	ZENKIN D: "Fighting Against the Invisible Enemy - Methods for detecting an unknown virus" COMPUTERS & SECURITY. INTERNATIONAL JOURNAL DEVOTED TO THE STUDY OF TECHNICAL AND FINANCIAL ASPECTS OF COMPUTER SECURITY, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, vol. 20, no. 4, 31 July 2001 (2001-07-31), pages 316-321, XP004254268 ISSN: 0167-4048		TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	31 July 2002	Powell, D

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 30 9795

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

31-07-2002

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5440723 A	08-08-1995	NONE	
US 5621889 A	15-04-1997	FR 2706652 A1	23-12-1994
		CA 2125445 A1	10-12-1994
		DE 69428631 D1	22-11-2001
		EP 0629940 A1	21-12-1994
		ES 2164091 T3	16-02-2002
		NO 942108 A	12-12-1994
WO 0171499 A	27-09-2001	US 2002087882 A1	04-07-2002
		AU 4735101 A	03-10-2001
		WO 0171499 A1	27-09-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

Annex to Form PCT/ISA/206
COMMUNICATION RELATING TO THE RESULTS OF THE PARTIAL INTERNATIONAL SEARCH

International Application No
PCT/GB2006/002439

1. The present communication is an Annex to the invitation to pay additional fees (Form PCT/ISA/206). It shows the results of the international search established on the parts of the international application which relate to the invention first mentioned in claims Nos.:
see 'Invitation to pay additional fees'
2. This communication is not the international search report which will be established according to Article 18 and Rule 43.

3. If the applicant does not pay any additional search fees, the information appearing in this communication will be considered as the result of the international search and will be included as such in the international search report.

4. If the applicant pays additional fees, the international search report will contain both the information appearing in this communication and the results of the international search on other parts of the international application for which such fees will have been paid.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 549 012 A (DATACENTERTECHNOLOGIES N.V) 29 June 2005 (2005-06-29) cited in the application page 2, paragraphs 6,7	1-11, 13-23, 25-30, 37,39,40
Y	page 3, paragraphs 17,18 page 6, paragraph 40 - paragraph 43 page 8, paragraph 50 - paragraph 51 page 10, paragraph 68 - paragraph 72	12,24, 38,41
X	US 2004/153644 A1 (MCCORKENDALE BRUCE ET AL) 5 August 2004 (2004-08-05) cited in the application page 2, paragraphs 21,25 page 4, paragraph 43 - page 5, paragraph 53	1-11, 13-23, 25-30, 37,39,40
X	EP 1 315 066 A (BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY) 28 May 2003 (2003-05-28) page 3, paragraph 8 page 6, paragraph 36 - page 7, paragraph 46 abstract	1,13,25, 30,37, 39,40

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

3

(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/039921 A1 (CHUANG SHYNE-SONG) 26 February 2004 (2004-02-26) cited in the application page 2, paragraph 33 - page 4, paragraph 51	1,13,25, 30,37, 39,40
Y	US 2005/086500 A1 (ALBORNOZ JORDI A) 21 April 2005 (2005-04-21) page 1, paragraph 8	12,24, 38,41
A	ZENKIN D: "Fighting Against the Invisible Enemy - Methods for detecting an unknown virus" COMPUTERS & SECURITY, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, vol. 20, no. 4, 31 July 2001 (2001-07-31), pages 316-321, XP004254268 ISSN: 0167-4048	

3

Patent Family Annex

Information on patent family members

International Application No
PCT/GB2006/002439

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1549012 A	29-06-2005	NONE	
US 2004153644 A1	05-08-2004	NONE	
EP 1315066 A	28-05-2003	CA 2467186 A1 WO 03046699 A1 US 2005022014 A1	05-06-2003 05-06-2003 27-01-2005
US 2004039921 A1	26-02-2004	AU 9620501 A WO 0233525 A2	29-04-2002 25-04-2002
US 2005086500 A1	21-04-2005	NONE	

Whole issue.qxd 6/22/01 1:34 PM Page 316

Computers & Security, 20 (2001) 316-321

Fighting Against the Invisible Enemy

Methods for detecting an unknown virus

Dr Denis Zenkin

Kaspersky Lab, 10, Geroyev Panfilovtcev Str., Moscow, 123363, Russia. Denis@kaspersky.com

Protection from unknown viruses is, indisputably, the issue of the day in computer virology. As is well known, the main deficiency of commonly used anti-virus scanners is that these scanners are able to detect and delete only those malicious programs described (by the so-called virus signatures) in their anti-virus databases. And for the new virus signature to be included in the anti-virus database, first, the scanner manufacturer must receive this virus from one of its customers, then the company experts must analyse it and develop a corresponding update. Then they must test this update and deliver it to the end users. Sometimes this procedure takes several days, and for the most complicated viruses, it can take up to a week. To our regret, during this time, end users of these scanners remain unprotected from this new virus threatening their computer security.

To shorten the period between when a new virus appears and when the corresponding update is delivered to end-users – and to speed-up the development of those updates - anti-virus manufacturers have created and implemented certain technology allowing for automation of the analysis and development and testing of updates for anti-virus databases. E-mail and advantages of other contemporary data-transmission technologies (for example, the so-called "Push" technology) allow delivery of ready-made anti-virus database updates to end-users within seconds. But all these efforts still haven't solved the key problem: protection from unknown (new) viruses before they actually appear.

So, let us discuss the so-called unknown virus in terms of conventional anti-virus scanners. The approach that is used in these programs allows them to check files on a computer when searching for virus-code fragments within their contents. The scanner acts just like a police officer checking every suspicious person (in this case, it is a suspicious program code) against the police database of criminals. The main disadvantage to this approach is that the scanner cannot detect a virus if the database doesn't contain its description (its code fragment). At the same time, to detect a malicious program (as well as a criminal) one does not need its detailed picture; it is enough to have just its rough description and to understand the features that are shared by all computer viruses. This advanced approach underlies the most up-to-date software tools allowing search for unknown viruses: a heuristic analyzer of the program code, redundant scanning, and behavior blocker and integrity checker.

Heuristic Analyzer

The main feature of computer viruses is their ability to embed in other programs while partially or completely retaining those programs' operability. During

0167-4048/01$20.00 © 2001 Elsevier Science Ltd

BNSDOCID: <XP_____4254268A__I_>

Whole issue.qxd 6/22/01 1:34 PM Page 317

the process of infection (when a virus embeds itself into a program), viruses modify the parent program so that when the system calls this program, the malicious code is executed. In addition, while trying to infect files, to mask their presence etc., all viruses use the same algorithms.

This feature was detected by anti-virus experts, and at the beginning of the 1990s, they developed and implemented in anti-virus scanners a tool allowing for the detection of unknown viruses – a heuristic analyzer. This tool checked objects by analyzing the instructions sequence in the objects' contents, accumulating statistics and making a decision about whether the object was infected with an unknown computer virus or not.

This generation of heuristic tools used a fairly primitive approach when detecting unknown viruses – if the tool detected a certain instruction sequence that is appropriate for computer viruses, the program, where this sequence had been detected, was automatically identified as infected. As a result, this technology produced a large number of false alarms and was ineffective. Very soon, virus-writers responded to this challenge by implementing various techniques, like encryption or polymorphism, allowing viruses to deceive heuristic analyzers. Then anti-virus experts developed the second-generation heuristic tool. Unlike its predecessor, this analyzer reconstructed the behavior of the program it was checking, and used it as a basis for conclusions about the potential danger from this program. This approach proved to be much more effective than that of the first generation and is still used by most heuristic tools today.

However, the heuristic technology is far from being a cure-all for a computer infection. This approach still has too many weak points to be the absolute defense: false alarms, deficient reliability, slowdown in system operation and the inability to delete the viruses detected are problematic of heuristic analyzers. Furthermore, to be able to detect different virus types, you must use different approaches and, as a result, different heuristic algorithms.

Redundant Scanning

Unlike the heuristic analyzer that is currently used in every competent anti-virus program, the redundant scanning technique is not so common. At the same time, while having a lot of disadvantages (the main one is that it tremendously slows down computer operation – up to several times), this technique allows for the detection of viruses that use extraordinary methods when infecting files. One of these methods is used by viruses "not having an entry point" (Entry Point Obscuring viruses (EPO-viruses)). The EPO family includes viruses that don't write the transfer-of-control instructions in the header of COM files (JMP) and do not change the starting point address in the header of EXE files. These viruses write the transfer-to-virus instruction somewhere in the file contents and obtain control when the procedure containing a code transferring control to the virus body is executed rather than directly after the infected file has been started. And this procedure might well be one of those that are rarely executed (for example, the procedure displaying a message about some program error). As a result, this virus may well be a "sleeper" inside of a file for years, breaking loose under certain circumstances. Before writing the transfer-to-virus instruction into a file, the virus must choose a "correct" address in this file; otherwise this file might well be corrupted. There are several methods that viruses use when defining these addresses:

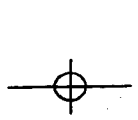
The first method is to search for a sequence of standard C/Pascal code in the file ("Lucretia" and "Zhengxi" viruses). These viruses search for standard C/Pascal header procedures and overwrite these with their code.

The second method is to trace and disassemble the file code ("CNTV," "MidInfector" and "NexivDer" viruses). These viruses load the file into the computer memory, then trace or disassemble it, and depending on various conditions, choose an instruction (or instructions) to be overwritten with the transfer-to-virus code.

The third method is used by memory-resident viruses only. These viruses control various system

BNSDOCID: <XP_____4254268A__1_>

Whole issue.qxd 6/22/01 1:34 PM Page 318



interrupts (INT 21h most often). When the file to be infected summons this interrupt, the virus overwrites this call-interruption instruction with its code ("Avatar.Positron" and "Markiz" viruses).

Redundant scanning - thorough scanning - allows for the scanning of not only the system processing entry points, but the entire contents of any given object. In fact, this method is not absolutely an unknown-virus search technology. This function allows an anti-virus program to look far inside a file in order to see its entire contents are inaccessible for others.

Integrity Checker

An integrity checker is yet another anti-virus technology that allows for unknown-virus detection. It is based on the fact that viruses can be considered average programs that have the ability for the unauthorized creation of new and modifying or self-planting into existing objects (such as files, boot sectors or system registry). In other words, they make changes in the file system that can be detected, and the presence of the malicious code is revealed.

An integrity checker's operating mode is based on a collection of original "prints" (CRC-values) of files, boot sectors and a system registry. These "prints" are stored in the database. When started up anew, the integrity checker compares information from its database with current "prints" and informs a user of changes that have occurred, marking virus-like changes and other unsuspicious changes. Some integrity checkers offer improved file analysis for the fastest and the most effective checking of a computer requiring minimum system resources. Thanks to the support of the most popular executable-file formats—such as PE EXE (Win32), NE EXE (Win16), MZ EXE (DOS), SYS, COM, OLE2 (Word, Excel and Access), and ELF (Linux)—an integrity checker can create Fast-CRC sums for the file system. This increases productivity up to five times while keeping the highest reliability of data control.

In 1990, the first stealth viruses, "Frodo" and "Whale," caused discredit to this anti-virus technology, because the operating mode of a stealth virus is based on hiding its activity within an infected system by substituting the original (not infected) files and boot sectors while the integrity checker tries to scan them. To do so, these viruses intercept disk-access interrupts and, in the case that an attempt to read, write or execute an infected object is detected, the virus passes off the clean copy of the object. Fortunately, this difficulty with detecting stealth-viruses was solved soon, and integrity checkers began accessing disks directly through an IOS disk driver (Input/Output Supervisor), leaving the system interrupts. This allowed integrity checkers to effectively protect computers against even stealth viruses.

Considered amongst the advantages of the most improved integrity checkers can be: the extremely high operating speed; low hardware and software requirements; and the high rate of file- and boot-sector restoration that previously have been infected with computer viruses, including unknown ones. An integrity checker's approach to infected-object restoration is based on the knowledge of what should be inside a clean file or boot sector rather than on the knowledge of a virus' appearance: everything that violates this rule is considered as a change and is subject to being reported to the user and having the original content restored. This is the reason why integrity checkers don't require a multi-megabyte, anti-virus database; instead being content with the virus-infecting method descriptions that usually need no more than 500 Kbytes. Knowing these methods, an integrity checker can quickly and effectively delete a virus and restore a file or a boot sector regardless in which part of an infected object the virus code is found - whether it is in the beginning, middle or end of the file, or the virus code is scattered all around it. It doesn't matter whether a virus is known and presented in the anti-virus database of the traditional virus scanner or not.

Integrity checkers are used not only in basic standalone home computers, but also on the server level of computing systems. For example, they are very useful for control over a Web server's content in order to prevent any unauthorized alternations. In real-time mode, an integrity checker performs the continuous checking of the Web server's integrity,



BNSDOCID: <XP_____4254268A__1_>

Whole issue.qxd 6/22/01 1:34 PM Page 319

tracking all changes. In case an unauthorized alternation is detected, the program can immediately send an SMS or pager message or an e-mail notification to the system administrator reporting the incident, and allows for full recovery of the original Web server's contents from back up. Therefore, integrity checkers can be considered powerful tools not only against known and unknown computer viruses, but also as strong alternative intrusion detection and anti-hacker utilities.

However, this type of anti-virus program has some disadvantages. Firstly, integrity checkers cannot detect a virus upon entry into the system and can only do so after a period of time. By then, the virus has already spread into the computer. Secondly, they cannot detect a virus in new files (e-mail, files on disks, files restored from backup, and files extracted from archive), since there is not any information in their databases about these objects. Some viruses even use this feature and infect only newly developed files, thereby staying "invisible" to the integrity checkers. Finally, in order to provide anti-virus protection, these anti-virus programs must be run on a regular basis. The more often they are run, the more reliable the virus-activity control will be.

Behaviour Blocker

The common disadvantage of all of the aforementioned techniques for the detection of unknown viruses is lack of guaranteed protection. None of them can provide 100% defense. It seems that the absolute solution for this problem could be found only after the discovery of an artificial intellect that would be able to analyze computer data as today's best anti-virus experts do. But nowadays, we can predict the most likely trends for anti-virus software development. In our opinion, these trends will include the so-called behaviour blocker (sandbox) anti-virus. Such programs can guarantee 100% protection against attacks from new viruses.

What is a behaviour blocker? It is a memory-resident program intercepting various computer operations. If a virus-suspected action (actions that can be performed by a virus or some other harmful program) is detected, a behaviour blocker thwarts it or asks the user for instructions. In other words, the behavior blocker does not search for virus signatures (i.e., a virus code as do traditional anti-virus scanners), and does not compare files with their original content, but rather monitors any program's activity and prevents harmful actions that could be done by malicious code.

The blocker idea is not new. These anti-virus programs were developed long ago, but did not achieve widespread use in the majority of desktop applications, because they were very intrusive, difficult to customize and required an in-depth knowledge of computer technologies on the part of the user. Nevertheless, the behavior blocker has found its niche in other areas of information technology. For instance, the widely used Java standard, developed by the Sun company, provides each single Java application with a strictly limited virtual space (a set of allowed activities) that prevents any attempts by any Java-based program to execute functions (such as deleting files) that a user may be suspicious of and could pose a threat to his or her computer security.

Let us look at the pros and cons of the behaviour blocker in detail. In theory, the behaviour blocker may prevent the distribution of any known and/or unknown (written after the blocker was developed) virus, warning the user about the virus before it is able to infect other files or damage his or her computer. However, the operating system itself or some useful utilities can also perform virus-like actions (such as creating files or modifying the system registry). Here we need to define two types of blockers: behaviour blockers for file systems and behaviour blockers for certain applications.

The behaviour blocker for file systems (a standard blocker used to combat file viruses) cannot independently determine the source of a virus-suspected action (virus, operating system or utility), and asks the user for instructions. This means that ultimately, it is the user who has to make the decision. And in order to make the correct decision, the user must possess sufficient knowledge and experience, otherwise the operating system or utility will be disabled

BNSDOCID: <XP_____4254268A__1_>

Whole issue.qxd 6/22/01 1:34 PM Page 320

from performing the action required, or a virus will penetrate the system.

This is the main reason why blockers have yet to become popular. Their strength is their weakness - they annoy users with their intrusive questions and have been uninstalled. This situation can only be improved with the invention of an artificial intelligence able to independently define the reasons for any virus-suspected action.

The behaviour blockers for applications have much more of a chance of becoming as popular as the traditional anti-virus scanners today, because their area of operating is strictly limited by the structure of a given application. Therefore, the blocker should only keep its eye on the defined number of activities that can be performed by programs developed especially for this application.

The most illustrative example of how efficient and easy-to-operate a behaviour blocker could be is Microsoft Office and the problem of fighting against unknown macro-viruses. For macro-programs written in the most widespread programming language VBA (Visual Basic for Application), good actions can be automatically differentiated from harmful with a fairly high success rate.

The behavioural analysis of macro-viruses reveals the most frequent sequences of operations. It allows embedding in the blocker a highly intelligent system of macro-instructions filtering that is able to detect the macro-instructions and, with a high probability rate, to define those that can be considered threatening. Due to this, behaviour blockers for applications are not so "intrusive" as its file-brothers. Although asking fewer questions, this blocker still provides reliable anti-virus protection and is able to protect a user from virtually all known and unknown macro-viruses.

Furthermore, a behaviour blocker intercepts and blocks the execution of even multi-platform macro-virus instructions; i.e., viruses that are able to operate in various Office applications. The program is equally reliable in Microsoft Word, Excel, Access and PowerPoint and even applications that are not included in the Microsoft Office package, but still use VBA (such as Visio and AutoCAD)

The blocker also monitors macros working with external applications, including e-mail programs. In so doing, it eliminates the possibility of macro-virus distribution through e-mail, as was the case in May 2000 when the ILOVEYOU virus infected hundreds of thousands of computers worldwide. Unlike traditional anti-virus technologies, a behaviour blocker solves this problem by blocking macro access to e-mail. In case the blocker detects an attempt by a macro-program to access an e-mail program, it can block the function or even terminate the whole macro (according to the pre-defined user preferences).

While employing a behaviour blocker for Microsoft Office, a user avoids the boring need to regularly download and install updates for the anti-virus database in order to protect his or her computer from new macro-viruses, since the program intercepts any new macro-virus by default. This solves the problem of the most dangerous period between the appearance of a new virus and the release of a corresponding update. Once installed, a behaviour blocker reliably protects a computer from macro-viruses until the next version of VBA (with expanded capabilities that can be used by macro-viruses) is released.

The behaviour blocker solves the problems of macro-virus detection and distribution prevention, but is not designed to delete macro-viruses and restore the infected files. This is why it should be used in conjunction with an anti-virus scanner that is able to delete viruses. The blocker allows a user to run Microsoft Office applications during the period between the appearance of a new macro-virus and the release of a corresponding scanner-database update without fear of losing vital data or serious damage to the hardware.

In our opinion, with the development of computer technologies - especially in the field of artificial intelligence - the magnitude, efficiency and simplicity-of-use of behaviour blockers (even those designed for a file system) will rapidly increase. In the near future, this type of anti-virus program preventing new and,

BNSDOCID: <XP_____4254268A_1_>

Whole issue.qxd 6/22/01 1:34 PM Page 321

thus unknown viruses from penetrating and distributing will become the major tool for anti-virus protection.

Conclusion

This overview of the modern technologies for protection against unknown viruses would not be complete if we did not mention the best way of using them. Which method is the best and most effective for use with home computers and enterprise-wide networks? We recommend something pretty simple: the best choice for protection against unknown viruses is a well-organized combination of all of the aforementioned techniques. Following the well-known proverb - don't put all your eggs in one basket - we also recommend not relying on integrity checkers or heuristic analyzers only. All of them have advantages and disadvantages. When used together, they ideally compensate for each other's deficiencies, and many years of their successful combined usage has proved this to be the best way to protect computing systems of any scale and complexity, whether it is a stand-alone home computer or huge corporate network.

BNSDOCID: <XP_____4254268A_|_>

Electronic Acknowledgement Receipt

EFS ID:	11803413
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Dale S. Lazar/Margaret Filipowicz
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000003
Receipt Date:	10-JAN-2012
Filing Date:	30-JUN-2006
Time Stamp:	16:43:02
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	339000-000003IDS.pdf	1324709 d228b770b9aea6785216925ad829ff9b1ccca40c	no	3

Warnings:	
Information:	

2	Information Disclosure Statement (IDS) Form (SB08)	339000-000003PTO1449.pdf	487696 f4430a1bdd561afb2c10c06c2a74079153169554	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Foreign Reference	EP1315066.pdf	1583075 d8c3f6a46c6600f19fa1b6b9fd563d14c7bdf653	no	28
Warnings:					
Information:					
4	Non Patent Literature	PCTGB2006002439ISR.pdf	107345 4c0b2751253afbe487d4e23726e5b49a6b5c6620	no	3
Warnings:					
Information:					
5	Non Patent Literature	Zenkin.pdf	414382 c7a3c005153cafde9fc53de87dd29aa0cc304ff1	no	6
Warnings:					
Information:					
		Total Files Size (in bytes):	3917207		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Docket No.: 339000-000003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Melvyn Morris	Conf. No.	4785
SERIAL NUMBER:	11/477,807	Group:	2432
FILED:	June 30, 2006	Examiner:	C. Zecher
FOR:	METHODS AND APPARATUS FOR DEALING WITH MALWARE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

January 10, 2012

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☒ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☐ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/maf
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2004/0083408	4/29/2004	Spiegel	
		US -2005/0022014	6/27/2005	Shipman	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Electronic Acknowledgement Receipt

EFS ID:	11813333
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	Methods and apparatus for dealing with malware
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Dale S. Lazar/Margaret Filipowicz
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000003
Receipt Date:	11-JAN-2012
Filing Date:	30-JUN-2006
Time Stamp:	16:21:23
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		339000-000003IDS.pdf	1279369 88f475a717db1f83d49cf02c4309192bfce43b2b	yes	4

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Transmittal Letter	1	3
	Information Disclosure Statement (IDS) Form (SB08)	4	4
Warnings:			
Information:			
	Total Files Size (in bytes):	1279369	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Docket No.: 339000-000003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Melvyn Morris	Conf. No.	4785
SERIAL NUMBER:	11/477,807	Group:	2432
FILED:	June 30, 2006	Examiner:	C. Zecher
FOR:	METHODS AND APPARATUS FOR DEALING WITH MALWARE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

January 11, 2012

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☒ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☐ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/maf
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

47604 7590 03/21/2012

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

EXAMINER
ZECHER, CORDELIA P K

ART UNIT	PAPER NUMBER
2432	

DATE MAILED: 03/21/2012

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	339000-000003	4785

TITLE OF INVENTION: METHODS AND APPARATUS FOR DEALING WITH MALWARE

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1740	$300	$0	$2040	06/21/2012

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

47604 7590 03/21/2012

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	339000-000003	4785

TITLE OF INVENTION: METHODS AND APPARATUS FOR DEALING WITH MALWARE

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1740	$300	$0	$2040	06/21/2012

EXAMINER	ART UNIT	CLASS-SUBCLASS
ZECHER, CORDELIA P K	2432	726-024000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____

2 _____

3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:

❑ Issue Fee

❑ Publication Fee (No small entity discount permitted)

❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

❑ A check is enclosed.

❑ Payment by credit card. Form PTO-2038 is attached.

❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

❑ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

❑ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	339000-000003	4785

47604 7590 03/21/2012

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

EXAMINER	
ZECHER, CORDELIA P K	

ART UNIT	PAPER NUMBER
2432	

DATE MAILED: 03/21/2012

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 712 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 712 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Notice of Allowability	Application No.	Applicant(s)	
	11/477,807	MORRIS ET AL.	
	Examiner	Art Unit	
	CORDELIA ZECHER	2432	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *Interview on 3/12/12*.

2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

3. ☒ The allowed claim(s) is/are *1-11,13-23,25-30,49 and 51*.

4. ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   a) ☒ All b) ☐ Some* c) ☐ None of the:

   1. ☒ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

6. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.

   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached

   1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.

   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

7. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☒ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☒ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

/CORDELIA ZECHER/
Primary Examiner, Art Unit 2432

Examiner-Initiated Interview Summary	Application No.	Applicant(s)	
	11/477,807	MORRIS ET AL.	
	Examiner	Art Unit	
	CORDELIA ZECHER	2432	

All participants (applicant, applicant's representative, PTO personnel):

(1) *Cordelia Zecher*. (3)_____.

(2) *Dale Lazar*. (4)_____.

Date of Interview: *12 March 2012*.

Type: ☒ Telephonic ☐ Video Conference
☐ Personal [copy given to: ☐ applicant ☐ applicant's representative]

Exhibit shown or demonstration conducted: ☐ Yes ☒ No.
If Yes, brief description: _____.

Issues Discussed ☒101 ☐112 ☐102 ☐103 ☐Others
(For each of the checked box(es) above, please describe below the issue and detailed description of the discussion)

Claim(s) discussed: *30-47*.

Identification of prior art discussed: *None*.

Substance of Interview
(For each issue discussed, provide a detailed description and indicate if agreement was reached. Some topics may include: identification or clarification of a reference or a portion thereof, claim interpretation, proposed amendments, arguments of any applied references etc...)

Applicant's representative was contacted for an examiner's amendment to place the application in condition for allowance. The examiner proposed adding "non-transitory" to claim 30 to make the claim statutory under 101. In addition, the examiner requested the cancelling of withdrawn claims 31-47. All the the suggested amendments were accepted by applicant's representative.

Applicant recordation instructions: It is not necessary for applicant to provide a separate record of the substance of interview.

Examiner recordation instructions: Examiners must summarize the substance of any interview of record. A complete and proper recordation of the substance of an interview should include the items listed in MPEP 713.04 for complete and proper recordation including the identification of the general thrust of each argument or issue discussed, a general indication of any other pertinent matters discussed regarding patentability and the general results or outcome of the interview, to include an indication as to whether or not agreement was reached on the issues raised.

☐ Attachment

/CORDELIA ZECHER/ Primary Examiner, Art Unit 2432	

DETAILED ACTION

Information Disclosure Statement

The information disclosure statements (IDS) submitted on December 23, 2011, January 10, 2012, and January 11, 2012 were filed after the mailing date of the Final Rejection on July 25, 2011. The submissions are in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statements are being considered by the examiner.

EXAMINER'S AMENDMENT

An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Dale S. Lazar on March 12, 2012.

The application has been amended as follows:

Claim 30 (currently amended): A non-transitory storage medium storing a computer program comprising program instructions for causing a computer to perform the method of claim 25.

Claims 31 – 47 (Cancelled)

EXAMINER'S COMMENTS

This communication warrants No Examiner's Reason for Allowance, applicant's reply make evident the reasons for allowance, satisfying the "record as a whole" proviso

of the rule 37 CFR 1.104(e). Specifically, applicant's arguments filed on December 23, 2011 are persuasive, as such the reasons for allowance are in all probability evident from the record and no statement is deemed necessary (see MPEP 1302.14).

Any comments Applicants considers necessary must be submitted no later than the payment of the Issue Fee and to avoid processing delays, should preferable accompany the Issue Fees. Such submission should be clearly labeled "Comments on Statement of Reasons for Allowance". In event of any post-allowance papers (e.g. IDS, 312 amendment, petition, etc.), Applicant is exhorted to mail papers to the Production Control branch in Publications or faxed to post-allowance papers correspondence branch at (703) 308-5864 to expedite issuing process or call PUB's Customer Service if any questions at (703) 305-8497.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to CORDELIA ZECHER whose telephone number is (571)272-7771. The examiner can normally be reached on Monday - Thursday 8:00 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/CORDELIA ZECHER/
Primary Examiner, Art Unit 2432

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
/C.Z./		US -2004/0083408	4/29/2004	Spiegel	
/C.Z./		US -2005/0022014	6/27/2005	Shipman	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]

Examiner Signature	/Cordelia Zecher/	Date Considered	02/17/2012

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 4785

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
11/477,807	06/30/2006	726	2432	339000-000003

APPLICANTS

Melvyn Morris, Turnditch, UNITED KINGDOM;
Paul Stubbs, Wyboston, UNITED KINGDOM;
Markus Hartwig, Milton Keynes, UNITED KINGDOM;
Darren Harter, Hucclecote, UNITED KINGDOM;

**** CONTINUING DATA ***************************

**** FOREIGN APPLICATIONS ***************************

UNITED KINGDOM 0513375.6 06/30/2005

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

07/26/2006

Foreign Priority claimed	35 USC 119(a-d) conditions met	Verified and Acknowledged	Met after Allowance	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
☑ Yes ☐ No	☑ Yes ☐ No	/CORDELIA P K ZECHER/ Examiner's Signature	☐ Initials	UNITED KINGDOM	3	47	8

ADDRESS

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195
UNITED STATES

TITLE

Methods and apparatus for dealing with malware

FILING FEE RECEIVED		
3584	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

BIB (Rev. 05/07).

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	1	"20070016953"	US-PGPUB; USPAT	OR	ON	2010/01/07 08:58
S2	1	("20060026684").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 08:32
S3	1	S2 and malware	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
S4	1164	726/24.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
S5	2821	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:34
S6	6	(("6772346") or ("6944772") or ("20040006704") or ("20040073810") or ("20050021994") or ("20040153644")).PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 09:18
S7	1	("20030177394").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 10:48
S8	0	central near database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
S9	0	database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
S10	115859	database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
S11	2632	central near database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
S12	30	central near database with network with separate	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
S13	17	central near database with distribute with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
S14	199	central near database with distribut$3 with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54
S15	0	central near database with distribute with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54
S16	4	central near database with distribut$3 with network with known	US-PGPUB;	OR	ON	2010/04/28 12:55

			USPAT			
S17	240	database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55
S18	24	database with distribut$3 with network with known and (virus or malware)	US-PGPUB; USPAT	OR	ON	2010/04/28 12:56
S19	4	("20030023857" | "20050027686" | "6338141" | "7013483").PN.	US-PGPUB; USPAT	OR	ON	2011/02/09 08:21
S20	3433	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S21	282	S20 and mask$3	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S22	35	S20 and mask$3 and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S23	328	S20 and control$3 with environment	US-PGPUB; USPAT	OR	ON	2011/02/09 08:43
S24	65	S20 and control$3 with environment and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:43
S25	18	S20 and defin$3 with boundary and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:45
S26	5	S20 and access$3 with rights with object and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:46
S27	2	"20020099952"	US-PGPUB; USPAT	OR	ON	2011/02/09 09:11
S28	24	(updat$3 or modif$4) with token and S20	US-PGPUB; USPAT	OR	ON	2011/02/09 10:54
S29	37	("20010049793" | "20030225698" | "20040068721" | "20050203819" | "20060031686" | "4713753" | "4805222" | "4926476" | "5367624" | "5481700" | "5819091" | "5915025" | "5958051" | "5978484" | "5983349" | "5991519" | "6295605" | "6378069" | "6405202" | "6775779" | "6931540" | "6981281" | "7028180" | "7051366" | "7076557").PN. OR ("7350204").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:23
S30	32	S29 and (modif$4 or updat$3)	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:23
S31	6232	automatic$4 with (updat$3 or chang$3 or modif$4) with access	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:35
S32	29	S20 and S31	US-PGPUB; USPAT;	OR	ON	2011/02/09 11:35

			USOCR			
S33	18	"7350204"	US-PGPUB; USPAT	OR	ON	2011/07/21 09:19
S34	1	"7350204".pn.	US-PGPUB; USPAT	OR	ON	2011/07/21 09:19
S35	42	("20010049793" | "20030225698" | "20040068721" | "20050203819" | "20060031686" | "4713753" | "4805222" | "4926476" | "5367624" | "5481700" | "5819091" | "5915025" | "5958051" | "5978484" | "5983349" | "5991519" | "6295605" | "6378069" | "6405202" | "6775779" | "6931540" | "6981281" | "7028180" | "7051366" | "7076557").PN. OR ("7350204").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:21
S36	106	("2002/0099952").URPN.	USPAT	OR	ON	2011/07/21 09:46
S37	16	("20020026578" | "20020099952" | "20020124170" | "20030033535" | "20040064708" | "20040078604" | "20040117615" | "20050138388" | "5941947" | "6151590" | "6397261" | "6571335" | "6810525" | "7010600" | "7194664").PN. OR ("7509497").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:49
S38	1	(S35 or S36 or S37) and behaviour	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:50
S39	1	"7698713".pn.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:51
S40	2	"20030131152"	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:52
S41	57	(S35 or S36 or S37) and behavior	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:52
S42	2	"20050210035"	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 10:01
S43	2	S42 and behav$5	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 10:04
S44	20	("20040153644" | "6338141" | "20030084323" | "20050027686" | "6772346" | "20050210035" | "6944772" | "20050021994" | "20030023857" | "20040083408" | "20050022014" | "20040006704" | "20010052014" | "20030101381" | "7013483" | "20020099952" | "20040039921" | "20050086500" | "20040073810" | "20050021994" | "20030177394").PN.	US-PGPUB; USPAT	OR	ON	2012/02/17 13:30

S45	5	("20040083408" | "20050022014" | "20030101381" | "20040039921" | "20050086500").PN.	US-PGPUB; USPAT	OR	ON	2012/02/17 13:34
S46	3172	generat$3 with profile with automatic$4	US-PGPUB; USPAT	OR	ON	2012/02/17 14:56
S47	4255	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2012/02/17 14:56
S48	25	generat$3 with profile with automatic$4 and S47	US-PGPUB; USPAT	OR	ON	2012/02/17 14:56

EAST Search History (Interference)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L3	1036	726/24.ccls.	USPAT	OR	ON	2012/03/12 11:36
L4	0	(class$6 and object and malware and automatic$4 and generat$3 and mask$3 and monitor$3).clm.	USPAT	OR	ON	2012/03/12 11:37
L5	0	(class$6 and object and malware and automatic$4 and generat$3 and monitor$3).clm.	USPAT	OR	ON	2012/03/12 11:37
L6	0	5 and 3	USPAT	OR	ON	2012/03/12 11:37
S49	1022	726/24.ccls.	USPAT	OR	ON	2012/02/17 14:59
S50	0	(behavior and malware and classify$3 and automatic$4 and generat$3 and monitor$3 and allow$3).clm.	USPAT	OR	ON	2012/02/17 15:06

3/ 12/ 2012 12:00:57 PM
C:\ Users\ czecher\ Documents\ EAST\ Workspaces\ 11477807.wsp

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2003/101381	5/29/2003	Mateev	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 95/12162	5/4/1995	Sophos PLC		
		EP 1 549 012	6/29/2005	DataCenter Technologies		
		JP 2003-196112	7/11/2003			
		JP 3-233629	10/17/1991	NEC Corp		
		JP 9-504395	4/28/1997			
		JP 6-110718	4/22/1994	Toshiba Corp		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Japanese Office Action issued in JP 2008-518975 on September 13, 2011	
		English Language Translation of Japanese Office Action issued in JP 2008-518975 on September 13, 2011	
		English Language Abstract and Translation of JP 6-110718 published April 22, 1994	
		English Language Abstract of JP 3-233629 published October 17, 1991	
		English Language Abstract of JP 2003-196112 published July 11, 2003	

Examiner Signature	/Cordelia Zecher/	Date Considered	02/17/2012

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010	02/09/2011	07/21/2011	03/12/2012				
1	1	✓	✓	✓	✓	=				
2	2	✓	✓	✓	✓	=				
3	3	✓	✓	✓	✓	=				
4	4	✓	✓	✓	✓	=				
5	5	✓	✓	✓	✓	=				
6	6	✓	✓	✓	✓	=				
7	7	✓	✓	✓	✓	=				
8	8	✓	✓	✓	✓	=				
9	9	✓	✓	✓	✓	=				
10	10	✓	✓	✓	✓	=				
11	11	✓	✓	✓	✓	=				
	12	✓	-	-	-	-				
13	13	✓	✓	✓	✓	=				
14	14	✓	✓	✓	✓	=				
15	15	✓	✓	✓	✓	=				
16	16	✓	✓	✓	✓	=				
17	17	✓	✓	✓	✓	=				
18	18	✓	✓	✓	✓	=				
19	19	✓	✓	✓	✓	=				
20	20	✓	✓	✓	✓	=				
21	21	✓	✓	✓	✓	=				
22	22	✓	✓	✓	✓	=				
23	23	✓	✓	✓	✓	=				
	24	✓	-	-	-	-				
25	25	✓	✓	✓	✓	=				
26	26	✓	✓	✓	✓	=				
27	27	✓	✓	✓	✓	=				
28	28	✓	✓	✓	✓	=				
29	29	✓	✓	✓	✓	=				
30	30	✓	✓	✓	✓	=				
	31	N		N	N	-				
	32	N		N	N	-				
	33	N		N	N	-				
	34	N		N	N	-				
	35	N		N	N	-				
	36	N		N	N	-				

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010	02/09/2011	07/21/2011	03/12/2012				
	37	N		N	N	-				
	38	N		N	N	-				
	39	N		N	N	-				
	40	N		N	N	-				
	41	N		N	N	-				
	42	N		N	N	-				
	43	N		N	N	-				
	44	N		N	N	-				
	45	N		N	N	-				
	46	N		N	N	-				
	47	N		N	N	-				
	48			✓	-	-				
12	49			✓	✓	=				
	50			✓	-	-				
24	51			✓	✓	=				

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2004/039921	2/26/2004	Chuang, Shyne-Song	
		US -2005/086500	4/21/2005	Albornoz, Jordi	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		EP 1 315 066	5/28/2003			

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		International Search Report issued in PCT/GB2006/002439 issued January 15, 2007	
		ZENKIN, D., "Fighting Against the Invisible Enemy - Methods for detecting an unknown virus" Computers & Security, Elsevier Science Publishers; Amsterdam, NL; vol. 20; no. 4; 31 July 2001; pages 316-321, XP004254268	

Examiner Signature	/Cordelia Zecher/	Date Considered	02/17/2012

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

Issue Classification

Application/Control No.	Applicant(s)/Patent Under Reexamination
11477807	MORRIS ET AL.
Examiner	**Art Unit**
CORDELIA ZECHER	2432

ORIGINAL		INTERNATIONAL CLASSIFICATION	
CLASS	SUBCLASS	CLAIMED	NON-CLAIMED
726	24	G 0 6 F 21 / 00 (2006.01)	

CROSS REFERENCE(S)

CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	18	18												
2	2	19	19												
3	3	20	20												
4	4	21	21												
5	5	22	22												
6	6	23	23												
7	7	25	25												
8	8	26	26												
9	9	27	27												
10	10	28	28												
11	11	29	29												
13	13	30	30												
14	14	12	49												
15	15	24	51												
16	16														
17	17														

		Total Claims Allowed:	
(Assistant Examiner)	(Date)	30	
/CORDELIA ZECHER/ Primary Examiner.Art Unit 2432	03/15/2012	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	3

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

SEARCHED			
Class	**Subclass**	**Date**	**Examiner**
726	24	3/12/12	cz

SEARCH NOTES		
Search Notes	**Date**	**Examiner**
Searched EAST (see attached)	3/12/12	cz
Inventor Search	4/28/10	cz

INTERFERENCE SEARCH			
Class	**Subclass**	**Date**	**Examiner**
726	24 (see attached)	3/12/12	cz

PTO/SB/30 (04-05)
Approved for use through 02/29/2008. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number

REQUEST FOR CONTINUED EXAMINATION (RCE) TRANSMITTAL

Address to:
Mail Stop RCE
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	Cordelia P.K. Zecher
Attorney Docket Number	339000-000003

This is a Request for Continued Examination (RCE) under 37 CFR 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. See Instruction Sheet for RCEs (not to be submitted to the USPTO) on page 2.

1. **Submission required under 37 CFR 1.114** Note: If the RCE is proper, any previously filed unentered amendments and amendments enclosed with the RCE will be entered in the order in which they were filed unless applicant instructs otherwise. If applicant does not wish to have any previously filed unentered amendment(s) entered, applicant must request non-entry of such amendment(s).

   a ☐ Previously submitted. If a final Office action is outstanding, any amendments filed after the final Office action may be considered as a submission even if this box is not checked.
   - (i) ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on
   - (ii) ☐ Other

   b ☒ Enclosed
   - (i) ☐ Amendment/Reply
   - (ii) ☒ Information Disclosure Statement (IDS)
   - (iii) ☐ Affidavit(s)/Declarations
   - (iv) ☐ Other _____

2. **Miscellaneous**

   a ☐ Suspension of action on the above-identified application is requested under 37 CFR 1.103(c) for a period of ___ months. (Period of suspension shall not exceed 3 months; Fee under 37 CFR 1.17(i) required)

   b ☐ Other

3. **Fees** The RCE fee under 37 CFR 1.17(e) is required by 37 CFR 1.114 when the RCE is filed.

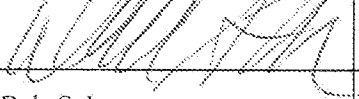
   a ☐ The Director is hereby authorized to charge the following fees, any underpayment of fees, or credit any overpayments, to Deposit Account No. 50-3266. I have enclosed a duplicate copy of this sheet.
   - (i) ☒ RCE fee required under 37 CFR 1.17(e)
   - (ii) ☐ Extension of time fee (37 CFR 1.136 and 1.17)
   - (iii) ☐ Other

   b ☐ Check in the amount of ___ enclosed

   c ☒ Payment by credit card (Form PTO-2038 enclosed)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT, REQUIRED

Signature		Date	June 21, 2012
Name (Print/Type)	Dale S. Lazar	Registration No.	28,872

This collection of information is required by 37 CFR 1.114. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Docket No.: 339000-000003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Melvyn Morris	Conf. No.	4785
SERIAL NUMBER:	11/477,807	Group:	2433
FILED:	June 30, 2006	Examiner:	Cordelia P.K. Zecher
FOR:	METHODS AND APPARATUS FOR DEALING WITH MALWARE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

June 21, 2012

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☐ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☒ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/NP/amd
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	Cordelia P.K. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2009/0271867	10/1/2009	Zhang	
		US -2008/0320595	12/1/2008	van der Made	
		US -2008/0209562	8/1/2008	Szor	
		US -7,093,239	8/1/2006	van der Made	
		US -2004/0255165	12/1/2004	Szor	
		US -2003/0135791	7/1/2003	Natvig	
		US -2004/0068618	4/1/2004	Hooker	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Image File Wrapper of related U.S. Application No. 11/694,261 electronically caputured from PAIR on June 21, 2012.	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Electronic Patent Application Fee Transmittal

Application Number:	11477807
Filing Date:	30-Jun-2006
Title of Invention:	METHODS AND APPARATUS FOR DEALING WITH MALWARE
First Named Inventor/Applicant Name:	Melvyn Morris
Filer:	Dale S. Lazar/Ava M. Dixon
Attorney Docket Number:	339000-000003

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Request for continued examination	1801	1	930	930
		Total in USD ($)		**930**

Electronic Acknowledgement Receipt

EFS ID:	13061027
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	METHODS AND APPARATUS FOR DEALING WITH MALWARE
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Dale S. Lazar/Ava M. Dixon
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000003
Receipt Date:	21-JUN-2012
Filing Date:	30-JUN-2006
Time Stamp:	16:54:34
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$930
RAM confirmation Number	4187
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Non Patent Literature	11694261IFWJUNE212012.pdf	6342518 b165d068743871d3a27606b19237a9c94fad889e	no	143
Warnings:					
Information:					
2		RCE-IDS.pdf	1871639 f4e58b55e45df581f44e70de5674e0eff0fd1a92	yes	5

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Request for Continued Examination (RCE)	1	1
	Transmittal Letter	2	4
	Information Disclosure Statement (IDS) Form (SB08)	5	5

Warnings:					
Information:					
3	Fee Worksheet (SB06)	fee-info.pdf	30380 9332546af9e280fccd4a7893bc830ff6f3223f46	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	8244537		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/694,261	03/30/2007	MIHAI CHIRIAC	339000-000005	1684

47604 7590 06/06/2012

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

EXAMINER	
KANG, INSUN	
ART UNIT	PAPER NUMBER
2198	
MAIL DATE	DELIVERY MODE
06/06/2012	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	11/694,261	CHIRIAC, MIHAI	
	Examiner	Art Unit	
	INSUN KANG	2198	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12/28/2011-3/30/2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

5)☒ Claim(s) *1-47* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-47* is/are rejected.

8)☒ Claim(s) *1-47* is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

10)☒ The specification is objected to by the Examiner.

11)☒ The drawing(s) filed on *3/30/2007* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

12)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *See Continuation Sheet*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Continuation of Attachment(s) 3). Information Disclosure Statement(s) (PTO/SB/08), Paper No(s)/Mail Date :12/28/2011, 8/3/2011, 5/23/2011, 2/3/2011, and 3/30/2007.

DETAILED ACTION

1. This action is responding to application papers 12/28/2011, 8/3/2011, 5/23/2011, 2/3/2011, and 3/30/2007.

2. Claims 1-47 are pending in the application.

Specification

3. The disclosure is objected to because of the following informalities: it appears that "analysing" needs to be changed to "analyzing" and at [0002] "executable co file" to "executable computer file." Appropriate correction is required.

Claim Objections

4. Claims 1-47 are objected to because of the following informalities: Per claims 1-47, "unpacking/decrypting" needs to be changed to unpacking or decrypting. Per claim 37, it appears that "analysing" needs to be changed to "analyzing." Per claim 1, "memory" at line 11 needs to be changed to "a memory." Per claim 2, "memory" at line 25 needs to be changed to the memory. Per claim 13, at line 32, 'a' is missing before "cache memory." At the rest lines after the line 32 of claim 13 and claims 14-17, "cache memory" needs to be changed to "the cache memory." Per claim 19, "memory" at line 32 needs to be changed to "the memory." Per claim 26, at line 12, 'a' is missing before "cache memory." At the rest lines after the line 12 of claim 26 and claim 27, "cache memory" needs to be changed to "the cache memory." Per claim 30, at line 3, "a said host computer" needs to be corrected to "a host computer" and at line 19, 'a' is missing before "cache memory." At the rest lines after the line 19 of claim 30 and claims 31-34, "cache memory" needs to be changed to "the cache memory." Per claim 37, at line 8, 'a' is missing before "cache memory" and "memory" at line 28 needs to be changed to "a

memory." At the rest lines after the line 8 of claim 30 and claims 38-47, "cache memory" needs to be changed to "the cache memory." Per claim 38, "memory" at line 10 needs to be changed to "the memory." Per claims 3-12, 14-17, 19-29, 31-36, and 38-47, these claims are objected for dependency on the above objected parent claims 1, 13, 18, 30, and 37. Appropriate correction is required.

Drawings

5. The drawings are objected to because fig. 2 and 3 are not legible. Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claim 15 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 15 recites the limitation "the host computer at line 2. There is insufficient antecedent basis for this limitation in the claim. Interpretation: a host computer.

Per claims 35, 36, and 47, it is not clear what the "computer" means because the body of the claims does not support the preamble reciting a computer. The body of the claims is directed to programs alone. Therefore, claims 35, 36, and 47 reciting a computer appears inaccurate lacking the necessary structural elements to be a physical system and support, even in its broadest reasonable sense, the preamble. Interpretation: the computer is interpreted as virtual computer software.

Claim Rejections - 35 USC § 101

8. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

9. Claims 18-29 and 30-34 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 18-29 and 30-34 are non-statutory because they are directed to a computer program, that is a merely abstract idea, thus, the computer program does not fit within the definition of the categories of patentable subject matter set forth in § 101. Therefore, the claims are non-statutory.

Per claims 35, 36, and 47, the "computer" can be reasonably interpreted as a virtual computer (vm) having a virtual processor/memory in light of the specification (i.e. 0026, 0058,0059). Such an emulator is a mere computer program which does not fit within the definition of the categories of patentable subject matter set forth in § 101. Therefore, the claims are non-statutory. "A host computer having a real host processor" as defined in the specification is recommended.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 1-47 are rejected under 35 U.S.C. 102(b) as being anticipated by Mateev et al. (US 2003/0101381) hereafter Mateev.

Per claim 1:

Mateev discloses:

A method of unpacking/decrypting an executable computer file using a host computer, the method comprising: partitioning the executable computer file into plural basic blocks of code (i.e. 0053; 0055; 0088; 0095); translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer (i.e. 0053); linking at least some of the translated basic blocks of code in memory of the host computer (i.e. 0091); and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be

analyzed to determine whether the executable computer file is or should be classed as malware (i.e. 0053; 0055).

Per claim 2:

Mateev further discloses:

when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory (i.e. 0091; 0094).

Per claim 3:

Mateev further discloses:

wherein a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block (i.e. 0057; 0058; 0103).

Per claim 4:

Mateev further discloses:

wherein the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated (i.e. 0091).

Per claim 5:

Mateev further discloses:

passing flow of the method to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks (i.e. 0074; 0094).

Per claim 6:

Mateev further discloses:

emulating the executable file by executing the basic blocks of code, wherein the basic blocks of code that are translated are only those that execute more than a predetermined number of times during the emulation (i.e. 0095; 0103).

Per claim 7:

Mateev further discloses:

wherein the value for said predetermined number of times is dynamically variable during the emulation (i.e. 0095; 0103).

Per claim 8:

Mateev further discloses:

wherein the emulation is carried out in virtual memory on the host computer (i.e. 0091).

Per claim 9:

Mateev further discloses:

creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block, and comprising emulating the executable

file by executing the basic blocks of code, and, during the execution of a basic block, checking at least one of the read page and the write page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block (i.e. 0073; 0074).

Per claim 10:

Mateev further discloses:

creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprising, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block (i.e. 0103).

Per claim 11:

Mateev further discloses:

wherein the partitioning is carried out dynamically (i.e. 0065).

Per claim 12:

Mateev further discloses:

wherein the executable file is a Portable Executable File (i.e. 0034; 0062).

Per claim 13:

Mateev discloses:

A method of unpacking/decrypting an executable computer file, the method comprising: partitioning the executable computer file into plural basic blocks of code (i.e. 0053; 0055; 0088; 0095); creating at least a read page of cache memory for at least some of the basic

blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block (i.e. 0073; 0074); and, emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware (i.e. 0053; 0055); wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block (i.e. 0073; 0074).

Per claims 14-17, they are other method versions of claims 8 and 10-12, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 8 and 10-12 above.

Per claims 18-29, they are program versions of claims 1-12, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 1-12 above.

Per claims 30-34, they are program versions of claims 13-17, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 13-17 above.

Per claim 35:

Mateev further discloses:

A computer programmed to carry out a method according to claim 1 (i.e. 0038).

Per claim 36:

Mateev further discloses:

A computer programmed to carry out a method according to claim 13 (i.e. 0038).

Per claim 37:

Mateev discloses:

A method of analyzing an executable computer file using a host computer, the method comprising: partitioning the executable computer file into plural basic blocks of code (i.e. 0053; 0055; 0088; 0095); creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block (i.e. 0073; 0074); emulating the executable file by executing the basic blocks of code, wherein, during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block (i.e. 0053; 0055); translating those basic blocks of code that execute more than a predetermined number of times during the emulation into translated basic blocks of code that can be executed by the host computer (i.e. 0053; 0095; 0103); linking at least some of the translated basic blocks of code in memory of the host computer (i.e. 0091); and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or

decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware (i.e. 0053; 0055).

Per claims 38-47, they are other method versions of claims 2-12 and 35, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 2-12 and 35 above.

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to INSUN KANG whose telephone number is (571)272-3724. The examiner can normally be reached on M-F 8 AM-4 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Don Wong can be reached on 571-272-1834. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Insun Kang/
Primary Examiner, Art Unit 2198

Notice of References Cited	Application/Control No. 11/694,261	Applicant(s)/Patent Under Reexamination CHIRIAC, MIHAI	
	Examiner INSUN KANG	Art Unit 2198	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2009/0271867	10-2009	Zhang, Peng	726/24
*	B	US-2008/0320595	12-2008	van der Made, Peter A.J.	726/24
*	C	US-2008/0209562	08-2008	Szor, Peter	726/24
*	D	US-2003/0101381	05-2003	Mateev et al.	714/38
*	E	US-7,093,239	08-2006	van der Made, Peter A. J	717/135
*	F	US-2004/0255165	12-2004	Szor, Peter	713/201
*	G	US-2003/0135791	07-2003	Natvig, Kurt	714/38
*	H	US-2004/0068618	04-2004	Hooker, Rodney E.	711/141
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L28	459	(virus mal$1ware malicious trojan) and (virtual emulat$4) same (cach$4 pag$4) same (read$3 writ$4 memory$1read$3 read$1pag$4) same address same (real$4 physical)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 18:26
L39	434	(7??/$.ccls.) and (virus mal$1ware malicious trojan) and (virtual emulat$4) same (cach$4 pag$4) same (read$3 writ$4 memory$1read$3 read$1pag$4) same address same (real$4 physical)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 19:57

EAST Search History (Interference)

< This search history is empty>

6/ 3/ 2012 7:59:13 PM
C:\ Users\ ikang\ Documents\ EAST\ Workspaces\ 11694261.wsp

+You Search Images Maps Play YouTube News Gmail Documents Calendar More »

[redacted] Sign in

Search About 1,420,000 results (0.28 seconds)

Web
Images
Maps
Videos
News
Shopping
Books
More

Alexandria, VA
Change location

Show search tools

Polymorphic code - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Polymorphic**_code
Emulation may be used to defeat **polymorphic** obfuscation by letting the **malware** demangle itself in a **virtual** environment before utilising other methods, such as ...

[PDF] DEFEATING **POLYMORPHISM**: BEYOND **EMULATION**
download.microsoft.com/.../Defeating_**Polymorphism**_White_Paper....
File Format: PDF/Adobe Acrobat
by AE Stepan - Cited by 14 - Related articles
unreasonable length of time to **emulate** until it is decrypted. This paper proposes a new method of dealing with **polymorphic malware**. The method relies on ...

Section 11.4: Code **Emulation**
computervirus.uw.hu/ch11ev1sec4.html
Because **polymorphic viruses** decrypt themselves, the virus will readily present itself in the **virtual** machine's memory if **emulated** long enough. The question ...

The Art of Computer **Virus** Research and Defense - Google Books R...
books.google.com/books?isbn=0672333902...
Peter Szor - 2005 - Computers - 744 pages
11.4.1 Encrypted and **Polymorphic Virus** Detection Using **Emulation** Consider ... of the entry-point code will result in a quick decryption of virus code in the **virtual** ...

[PDF] **VIRUS** ANALYSIS 1
pandalabs.pandasecurity.com/blogs/images/.../2007/.../VB200705.pd...
File Format: PDF/Adobe Acrobat - Quick View
Oct 9, 2007 – Being a **polymorphic virus**, **emulation** should be the logical approach to detect and disinfect W32/**Virtu**. Indeed, this seems to be what the virus ...

[PDF] **Virtual** Machine Protection Technology and AV industry
www.f-secure.com/.../Zhenxian_Wang-**Virtual**_machine_protection....
File Format: PDF/Adobe Acrobat - Quick View
Malware authors adopted the **polymorphic** technique to counteract the approach. . **Emulation** technique was used to solve the **polymorphism** issue. . **Malware** ...

[PDF] Hunting For Metamorphic
https://www.symantec.com/avcenter/.../hunting.for.metamorphic.pdf
File Format: PDF/Adobe Acrobat - Quick View
by P Szor - Cited by 228 - Related articles
A virus scanner that used a code **emulator** to detect viruses looked like it was on steroids ... Nowadays, most **polymorphic viruses** are considered boring. Even though ..... section is loaded into this array at the section's relative **virtual** address.

Polymorphic virus detection module
www.google.com/patents/US5696822.pdf
5319776 6/1994 ese viruses, an _a l e ( ) aving an entry or eac. 5321840 6,1994 known **polymorphic virus** (150). Prior to **emulation**, the. 5349655 9/1994 static ...

The Art of Stealthy Viruses.ixi
www.thehackademy.net/.../The%20Art%20of%20Stealthy%20Viruse...
This is all done inside of an **emulator** or a "**virtual** computer" if I may. Anti-Virus engineers got clever and decided to develop a way to trick a **polymorphic virus** ...

[PDF] Kaspersky anti-**virus** engine technology Comprehensive protect...
www.opsec.com/solutions/partners/.../Kaspersky_EngineTech_WP.p...
File Format: PDF/Adobe Acrobat - Quick View
programs] is **emulated** in a **virtual** environment, called a 'sandbox' or **virtual** machine. After this **emulation** process, where the program is a **polymorphic virus**, ...

Ad related to polymorphism (virus OR malware) virtual emulation Why this ad?

Need **Malware** Protection? 1 (888) 632 8796
www.viprebusiness.com/AntiMalware
Protect Your Business Network from Web Threats with VIPRE. Free Trial

Gooooooooooogle >
1 2 3 4 5 6 7 8 9 10 Next

Advanced search Search Help Give us feedback

Google Home Advertising Programs Business Solutions Privacy & Terms
About Google

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known

Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	Mihai Chiriac
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2005/210035	9/1/2005	Kester	
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[2] Number[3] Kind Code[4] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Image File Wrapper of U.S. Appl 11/477,807 from May 24, 2011 to August 3, 2011	

Examiner Signature	/Insun Kang/	Date Considered	06/03/2012

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	18	(self$1modifying "self modifying" polymorphi$4) with (virus mal$1ware malicious trojan) and (virtual emulat$4) and (cach$4 pag$4) with (read$3 writ$4 memory$1read$3 read$1pag$4) same address	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 16:49
L28	459	(virus mal$1ware malicious trojan) and (virtual emulat$4) same (cach$4 pag$4) same (read$3 writ$4 memory$1read$3 read$1pag$4) same address same (real$4 physical)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 18:26
L38	3	(prevx.asn. CHIRIAC.in.) and (virus mal$1ware malicious trojan) same (unpack$4 decrypt$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 19:47
S192	98	polymorphi$4 with (virus mal$1ware malicious) and (virtual emulat$4) and (cach$4 pag$4 memory$1read$3 read$1pag$4) and (unpack$4 de$1crypt$4 de$1compress$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 13:33
S194	10	polymorphi$4 with (virus mal$1ware malicious) and (virtual$7 emulat$4) same (translat$4 link$4) same (partition$4 divid$4 fragment$5 segment$5 block) and (cach$4 pag$4 memory$1read$3 read$1pag$4) and (unpack$4 de$1crypt$4 de$1compress$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 14:15
S199	105	(self$1modifying "self modifying" polymorphi$4) with (virus mal$1ware malicious trojan) and (virtual emulat$4) and (cach$4 pag$4 memory$1read$3 read$1pag$4) and (unpack$4 de$1crypt$4 de$1compress$4)	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 15:10
S208	45	("20040153644" | "6338141" | "20030084323" | "20090271867" | "20030177394" | "20050027686" | "20050210035" | "20080209562" | "20010052014" | "20030023857" | "20040083408" | "20080320595" | "20040006704" | "20030101381" | "7013483" | "20040039921" | "20050086500" | "6772346" |	US-PGPUB; USPAT; USOCR; FPRS; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2012/06/03 15:24

		"20020099952" | "7093239" | "20040073810" | "20050021994" | "6944772").PN.				

EAST Search History (Interference)

<This search history is empty>

6/3/2012 7:57:22 PM
C:\Users\ikang\Documents\EAST\Workspaces\11694261.wsp

Docket No.: 339000-P0004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Mihai CHIRIAC	Conf. No.	Unknown
SERIAL NUMBER:	New Application	Group:	Unknown
FILED:	March 30, 2007	Examiner:	Unknown

FOR: METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

March 30, 2007

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry)not more than three months prior to the filing of this statement.

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application or, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

- ☒ without fee and within three months of the filing date of the application.
- ☐ without fee and within three months of the date of entry of the U.S. national stage.
- ☐ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
- ☐ without fee and with the appropriate certification above.
- ☐ without fee and with a new CPA application.
- ☐ without fee and with a Request for Continued Examination.
- ☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
- ☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

- ☒ Please charge any additional fees for the papers being filed herewith and for which no check is enclosed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER US LLP

Dale S. Lazar
Reg. No. 28,872
Attorney of Record

P.O. Box 9271
Reston, VA 20195
Tel. (703) 773-4000
Fax (703) 773-5200

DSL/jsk

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 2

Complete if Known	
Application Number	New Application 11694261
Filing Date	March 30, 2007
First Named Inventor	Mihai CHIRIAC
Art Unit	Unknown 2198
Examiner Name	Unknown kang
Attorney Docket Number	339000-P0004

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -7013483	03-14-2006	ALADDIN KNOWLEDGE SYSTEMS LTD	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 96/30829	10-03-1996	APPLE COMPUTER, INC.		
		WO 99/15966	04-01-1999	SYMANTEC CORPORATION		
		EP 1536341 A2	06-01-2005	MICROSOFT CORPORATION		

Examiner Signature	Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional). 2 See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. 3 Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). 4 For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. 5 Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. 6 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer,

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.
U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND **TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	2	of	2

Complete if Known	
Application Number	New Application 11694261
Filing Date	March 30, 2007
First Named Inventor	Mihai CHIRIAC
Art Unit	Unknown
Examiner Name	Unknown
Attorney Docket Number	339000-P0004

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		"DEFEATING POLYMORPHISM: BEYOND EMULATION, Adrian E. STEPAN	
		"GHOSTBUSTER 3", Mihai CHIRIAC	

Examiner Signature	/Insun Kang/	Date Considered	06/03/2012

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
[1] Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	Mihai Chiriac
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2002/99952	7/1/2002	Lambert	
		US -2003/84323	5/1/2003	Gales	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	Mihai Chiriac
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

		Image File Wrapper of U.S. Appl 11/477,807 as of February 4, 2011 to May 23, 2011	

/Insun Kang/ 06/03/2012

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	Mihai Chiriac
Art Unit	2193
Examiner Name	I. Kang
Attorney Docket Number	339000-000005

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2003/101381	5/29/2003	Mateev	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 95/12162	5/4/1995	Sophos PLC		
		EP 1 549 012	6/29/2005	DataCenter Technologies		
		JP 2003-196112	7/11/2003			
		JP 3-233629	10/17/1991	NEC Corp		
		JP 9-504395	4/28/1997			
		JP 6-110718	4/22/1994	Toshiba Corp		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Japanese Office Action issued in JP 2008-518975 on September 13, 2011	
		English Language Translation of Japanese Office Action issued in JP 2008-518975 on September 13, 2011	
		English Language Abstract and Translation of JP 6-110718 published April 22, 1994	
		English Language Abstract of JP 3-233629 published October 17, 1991	
		English Language Abstract of JP 2003-196112 published July 11, 2003	
		Image File Wrapper of U.S. Appl 11/477,807 electronically captured on December 28, 2011	

Examiner Signature	/Insun Kang/	Date Considered	06/03/2012

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 1684

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
11/694,261	03/30/2007	717	2198	339000-000005

APPLICANTS
MIHAI CHIRIAC, Bucharest, ROMANIA;

**** CONTINUING DATA ***************************
This appln claims benefit of 60/789,156 04/05/2006 /IK/

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****
04/11/2007

Foreign Priority claimed ☐ Yes ☑ No 35 USC 119(a-d) conditions met ☐ Yes ☐ No Verified and Acknowledged /INSUN KANG/ Examiner's Signature	☐ Met after Allowance Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
		ROMANIA	2	47	5

ADDRESS

DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195
UNITED STATES

TITLE

METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

FILING FEE RECEIVED		
2750	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

BIB (Rev. 05/07).

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694261	CHIRIAC, MIHAI
	Examiner	Art Unit
	INSUN KANG	2198

SEARCHED			
Class	Subclass	Date	Examiner

SEARCH NOTES		
Search Notes	Date	Examiner
east (all DBs)/text only, text+ (717/all; 726/all;714/all;711/all; google search; DP search; IDS considered	6/3/2012	IK
search history included	6/3/2012	IK

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694261	CHIRIAC, MIHAI
	Examiner	**Art Unit**
	INSUN KANG	2198

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	06/03/2012								
	1	✓								
	2	✓								
	3	✓								
	4	✓								
	5	✓								
	6	✓								
	7	✓								
	8	✓								
	9	✓								
	10	✓								
	11	✓								
	12	✓								
	13	✓								
	14	✓								
	15	✓								
	16	✓								
	17	✓								
	18	✓								
	19	✓								
	20	✓								
	21	✓								
	22	✓								
	23	✓								
	24	✓								
	25	✓								
	26	✓								
	27	✓								
	28	✓								
	29	✓								
	30	✓								
	31	✓								
	32	✓								
	33	✓								
	34	✓								
	35	✓								
	36	✓								

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11694261	CHIRIAC, MIHAI
	Examiner	**Art Unit**
	INSUN KANG	2198

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	06/03/2012								
	37	✓								
	38	✓								
	39	✓								
	40	✓								
	41	✓								
	42	✓								
	43	✓								
	44	✓								
	45	✓								
	46	✓								
	47	✓								

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 2

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	CHIRIAC, Mihai
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US –2004/153644	8/5/2004	McCorkendale et al.	
		US -2004/039921	2/26/2004	Chuang, Shyne-Song	
		US -6,772,346	8/1/2004	Chess	
		US -6,944,772	9/1/2005	Dozortsev	
		US -2004/0006704	1/1/2004	Dahlstrom	
		US -2004/0073810	4/15/2004	Dettinger	
		US -2005/0021994	1/1/2005	Barton	
		US -2001/0052014	12/1/2001	Sheymov et al.	
		US -2003/0177394	9/1/2003	Dozortsev, Dmitri	
		US -2005/086500	4/21/2005	Albornoz, Jordi	
		US -60/789,156			
		US -2004/0083408	4/29/2004	Spiegel	
		US -2003/0023857	1/30/2003	Hinchlife	
		US -2005/0027686	2/3/2005	Shipp	
		US -6,338,141	1/8/2002	Wells	
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO02/33525	4/25/2002			
		WO03/021402	3/13/2003			
		WO04/097602	11/11/2004			
		EP 1 280 040	3/24/2004			
		EP 1 549 012	6/29/2005			
		EP 1 315 066	5/28/2003			

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional). 2 See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. 3 Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). 4 For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. 5 Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. 6 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

 ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	2	of	2

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	CHIRIAC, Mihai
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Image File Wrapper of U.S. Appl 11/477,807 as of February 3, 2011	
		International Search Report issued in PCT/GB2006/002439 issued January 15, 2007	
		ZENKIN, D., "Fighting Against the Invisible Enemy - Methods for detecting an unknown virus" Computers & Security, Elsevier Science Publishers; Amsterdam, NL; vol. 20; no. 4; 31 July 2001; pages 316-321, XP004254268	

Examiner Signature	/Insun Kang/	Date Considered	06/03/2012

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
[1] Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /I.K./

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	Mihai Chiriac
Art Unit	2193
Examiner Name	I. Kang
Attorney Docket Number	339000-000005

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2003/101381	5/29/2003	Mateev	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 95/12162	5/4/1995	Sophos PLC		
		EP 1 549 012	6/29/2005	DataCenter Technologies		
		JP 2003-196112	7/11/2003			
		JP 3-233629	10/17/1991	NEC Corp		
		JP 9-504395	4/28/1997			
		JP 6-110718	4/22/1994	Toshiba Corp		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Japanese Office Action issued in JP 2008-518975 on September 13, 2011	
		English Language Translation of Japanese Office Action issued in JP 2008-518975 on September 13, 2011	
		English Language Abstract and Translation of JP 6-110718 published April 22, 1994	
		English Language Abstract of JP 3-233629 published October 17, 1991	
		English Language Abstract of JP 2003-196112 published July 11, 2003	
		Image File Wrapper of U.S. Appl 11/477,807 electronically captured on December 28, 2011	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Electronic Acknowledgement Receipt

EFS ID:	11721420
Application Number:	11694261
International Application Number:	
Confirmation Number:	1684
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Customer Number:	47604
Filer:	Dale S. Lazar/Margaret Filipowicz
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000005
Receipt Date:	28-DEC-2011
Filing Date:	30-MAR-2007
Time Stamp:	16:42:47
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		339000-000005IDS.pdf	1386589 d5dde0c7b2382c60b81104eb26f7cbe6315d8627	yes	4

	Multipart Description/PDF files in .zip description				
	Document Description		Start	End	
	Transmittal Letter		1	3	
	Information Disclosure Statement (IDS) Form (SB08)		4	4	
Warnings:					
Information:					
2	Foreign Reference	D1equivWO9512162A1.pdf	814560 76e316fef9eeb967a09c80cc4f734475d9b52b3a	no	23
Warnings:					
Information:					
3	Foreign Reference	D2EP1549012A1.pdf	802684 74b0525be8de52789acb2ed2d2f609f448dd8629	no	19
Warnings:					
Information:					
4	Foreign Reference	D5JP2003-196112.pdf	1173541 5a8ca3fd9a79aaf502814d6b79f0bdf0c1062db4	no	21
Warnings:					
Information:					
5	Foreign Reference	D1JP9-504395.pdf	486616 5dfc9c398879c28144989c4750759c01b64e16e0	no	20
Warnings:					
Information:					
6	Foreign Reference	D3JP6-110718.pdf	544000 6dc0e22988c4bc748897eed21088efd321852f8d	no	15
Warnings:					
Information:					
7	Non Patent Literature	P10717JP-WJapaneseobjections.pdf	125377 65f00b8dd4484264c751bf13a8455dbd2f0b70af	no	5
Warnings:					
Information:					
8	Non Patent Literature	P10717JP-WOATranslation.pdf	155462 83ac10d3bad7687552b4a56310e6b890399c4b5b	no	7
Warnings:					
Information:					

9	Non Patent Literature	D3JP6-110718AbstractandTranslation.pdf	629386 d50a2d361c6a625b95290e2665b48615d0df6501	no	12
Warnings:					
Information:					
10	Non Patent Literature	D4JP3-233629Abstract.pdf	36645 5e68cefeb1686e0cd5f4bac66081c7948677eb8a	no	1
Warnings:					
Information:					
11	Non Patent Literature	D5JP2003-196112Abstract.pdf	372434 347a9e0ad4d0f769e12b1897b6cda674a1d6f935	no	1
Warnings:					
Information:					
12	Foreign Reference	D4JP3-233629.pdf	192169 7b0ad4827f6e41c20365e32263838db5aede1641	no	7
Warnings:					
Information:					
13	Non Patent Literature	11477807IFWDecember282011.pdf	13877148 c0f6cafb638ca013a2cd40fd65955b8666d7533d	no	355
Warnings:					
Information:					
		Total Files Size (in bytes):	20596611		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Docket No.: 339000-000005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Mihai Chiriac	Conf. No.	1684
SERIAL NUMBER:	11/694,261	Group:	2193
FILED:	March 30, 2007	Examiner:	I. Kang
FOR:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYZING AN EXECUTABLE COMPUTER FILE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

December 28, 2011

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

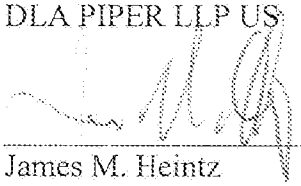
☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☒ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☐ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US



James M. Heintz
Registration No. 41,828
Attorney for Applicant

JMH/maf
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known

Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	Mihai Chiriac
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2005/210035	9/1/2005	Kester	
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Image File Wrapper of U.S. Appl 11/477,807 from May 24, 2011 to August 3, 2011	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Electronic Acknowledgement Receipt

EFS ID:	10660517
Application Number:	11694261
International Application Number:	
Confirmation Number:	1684
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Customer Number:	47604
Filer:	Dale S. Lazar
Filer Authorized By:	
Attorney Docket Number:	339000-000005
Receipt Date:	03-AUG-2011
Filing Date:	30-MAR-2007
Time Stamp:	16:41:27
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	339000-5IDS-080311.pdf	875905 f7397ca5276c2f6890295628c635e36636cd8cc2	no	3

Warnings:	
Information:	

2	Information Disclosure Statement (IDS) Form (SB08)	339000-5PTO1449-080311.pdf	637094 212595103a5bb2563a39750ff3f333b404d808ad	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Non Patent Literature	IFW11477807-080311.pdf	1778795 84ed1545a7abee0922a049f2aceef8932d530e82	no	44
Warnings:					
Information:					
		Total Files Size (in bytes):	3291794		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Docket No.: 339000-000005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Mihai Chiriac	Conf. No.	1684
SERIAL NUMBER:	11/694,261	Group:	2193
FILED:	March 30, 2007	Examiner:	Unknown
FOR:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYZING AN EXECUTABLE COMPUTER FILE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

August 3, 2011

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☒ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☐ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/jcc
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	Mihai Chiriac
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2002/99952	7/1/2002	Lambert	
		US -2003/84323	5/1/2003	Gales	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	Mihai Chiriac
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

		Image File Wrapper of U.S. Appl 11/477,807 as of February 4, 2011 to May 23, 2011	

Electronic Acknowledgement Receipt

EFS ID:	10152673
Application Number:	11694261
International Application Number:	
Confirmation Number:	1684
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Customer Number:	47604
Filer:	Dale S. Lazar/Josefina Cheng
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-000005
Receipt Date:	23-MAY-2011
Filing Date:	30-MAR-2007
Time Stamp:	20:04:33
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	339000-5IDS052311.pdf	93250 73747fffdc0deb758c6b5c07026daac357ed67fe	no	3

Warnings:	
Information:	

2	NPL Documents	IFW11477807-052311.pdf	948149 3a22d85497302ce2ffa456c413cae5a7d32f0ce3	no	21
Warnings:					
Information:					
3	Information Disclosure Statement (IDS) Filed (SB/08)	339000-000005PTO1449.pdf	80077 6808ae7ba7638c0d964921ad897185e3118e7c1c	no	2
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
			Total Files Size (in bytes):	1121476	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Docket No.: 339000-000005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Mihai Chiriac	Conf. No.	1684
SERIAL NUMBER:	11/694,261	Group:	2193
FILED:	March 30, 2007	Examiner:	Unknown
FOR:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYZING AN EXECUTABLE COMPUTER FILE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

May 23, 2011

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☒ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☐ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/jcc
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 11/694,761	Filing Date	☐ To be Mailed

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)	SMALL ENTITY ☐		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT	4/14/11	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	ADDITIONAL FEE ($)		OTHER THAN SMALL ENTITY RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	* 10	Minus	** 20	= 0	X $26 =		OR	X $52=	
	Independent (37 CFR 1.16(h))	* 3	Minus	*** 3	= 0	X $110 =		OR	X $220=	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
	~~1-27~~ ~~28-30~~ (32) (36) ~~37-40-42~~					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

4/15

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))									
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/LISA FULTON/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	1	of	2

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	CHIRIAC, Mihai
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US –2004/153644	8/5/2004	McCorkendale et al.	
		US -2004/039921	2/26/2004	Chuang, Shyne-Song	
		US -6,772,346	8/1/2004	Chess	
		US -6,944,772	9/1/2005	Dozortsev	
		US -2004/0006704	1/1/2004	Dahlstrom	
		US -2004/0073810	4/15/2004	Dettinger	
		US -2005/0021994	1/1/2005	Barton	
		US -2001/0052014	12/1/2001	Sheymov et al.	
		US -2003/0177394	9/1/2003	Dozortsev, Dmitri	
		US -2005/086500	4/21/2005	Albornoz, Jordi	
		US -60/789,156			
		US -2004/0083408	4/29/2004	Spiegel	
		US -2003/0023857	1/30/2003	Hinchlife	
		US -2005/0027686	2/3/2005	Shipp	
		US -6,338,141	1/8/2002	Wells	
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO02/33525	4/25/2002			
		WO03/021402	3/13/2003			
		WO04/097602	11/11/2004			
		EP 1 280 040	3/24/2004			
		EP 1 549 012	6/29/2005			
		EP 1 315 066	5/28/2003			

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional). 2 See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. 3 Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). 4 For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. 5 Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. 6 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 2 of 2

Complete if Known	
Application Number	11/694,261
Filing Date	March 30, 2007
First Named Inventor	CHIRIAC, Mihai
Art Unit	2193
Examiner Name	Unknown
Attorney Docket Number	339000-000005

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Image File Wrapper of U.S. Appl 11/477,807 as of February 3, 2011	
		International Search Report issued in PCT/GB2006/002439 issued January 15, 2007	
		ZENKIN, D., "Fighting Against the Invisible Enemy - Methods for detecting an unknown virus" Computers & Security, Elsevier Science Publishers; Amsterdam, NL; vol. 20; no. 4; 31 July 2001; pages 316-321, XP004254268	
Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
[1] Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Docket No.: 339000-000005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Mihai Chiriac	Conf. No.	1684
SERIAL NUMBER:	11/694,261	Group:	2193
FILED:	March 30, 2007	Examiner:	Unknown
FOR:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYZING AN EXECUTABLE COMPUTER FILE		

LIST OF RELATED CASES

Serial Number	Publication Number	Filing or Publication Date	Status
11/477,807	US 2007-0016953	January 18, 2007	Pending

The cases listed on this Notice of Related Cases include cases which may contain information that is material to patentability. The listing of a case on this Notice should not be taken as an indication or admission that any information contained therein is material. Prior art for each case listed on this Notice may have been cited. **The files corresponding to the listed cases, which are available to the Examiner, may not have been examined to ascertain the materiality of any prior art therein.** Accordingly, the Examiner is requested to review the file for each case listed on this Notice in order to assess the materiality of such prior art.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar

P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

Reg. No. 28,872
Attorney of Record
Date: February 3, 2011

Electronic Acknowledgement Receipt

EFS ID:	9368185
Application Number:	11694261
International Application Number:	
Confirmation Number:	1684
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Customer Number:	47604
Filer:	Dale S. Lazar
Filer Authorized By:	
Attorney Docket Number:	339000-000005
Receipt Date:	03-FEB-2011
Filing Date:	30-MAR-2007
Time Stamp:	13:54:02
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	339000-5IDS.pdf	872934 2309dabfed44a32b82980efa6e6e781a00b1f189	no	3

Warnings:	
Information:	

2	Information Disclosure Statement (IDS) Filed (SB/08)	339000-5PTO1449.pdf	1094980 4eb720c55859a5810658f8276bf232ff6a6e2b98	no	2
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Miscellaneous Incoming Letter	339000-5RCS.pdf	323610 5698180d45ad5055a408595c15e01325593ccfad	no	1
Warnings:					
Information:					
4	Foreign Reference	WO0233525.pdf	806986 ee7f5570c2bbfa238dcf290591e563e0690f8d41	no	18
Warnings:					
Information:					
5	Foreign Reference	WO03021402.pdf	542026 ce8fcdf6194f04f0656dfc773b170d065f1210b2	no	13
Warnings:					
Information:					
6	Foreign Reference	WO2004097602.pdf	423861 471c1b26f116cce6e932e69ebdf466cecdc2e7e8	no	10
Warnings:					
Information:					
7	Foreign Reference	EP1280040.pdf	1047450 9b7f55269555cd96c5080d95dcdf2f21aae0d6b9c	no	17
Warnings:					
Information:					
8	Foreign Reference	EP1549012.pdf	1222535 95a006a49025158d4eb86e8a88d07fee22ce32c6	no	19
Warnings:					
Information:					
9	Foreign Reference	EP1315066.pdf	1589676 ceaa5e6bc415f6c67113676a07a694c628bc6aad	no	28
Warnings:					
Information:					
10	NPL Documents	PCTGB2006002439-11507.pdf	82757 43a5c54223725805d2e8e5b1dbad756ff7f3b284	no	2

Warnings:					
Information:					
11	NPL Documents	ZENKIN.pdf	393991 2b93986bb17e92edd6b9d91b417be94da51c572f	no	6
Warnings:					
Information:					
12	NPL Documents	IFW-11477807-020311.pdf	10330271 4125d00125436b90b56035b3d2293a102567a527	no	270
Warnings:					
Information:					
		Total Files Size (in bytes):	18731077		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Docket No.: 339000-000005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Mihai Chiriac	Conf. No.	1684
SERIAL NUMBER:	11/694,261	Group:	2193
FILED:	March 30, 2007	Examiner:	Unknown
FOR:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYZING AN EXECUTABLE COMPUTER FILE		

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

February 3, 2011

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

The documents listed on the attached Form PTO 1449 were cited in related and corresponding applications. English language abstracts and translations are provided for the convenience of the Examiner where available.

RELATED CASES

☒ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was first cited in any communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry) not more than three months prior to the filing of this statement (See 37 CFR 1.97(e)(1)).

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application, and, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement. (See 37 CFR 1.97(e)(2))

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

☐ without fee and within three months of the filing date of the application.
☐ without fee and within three months of the date of entry of the U.S. national stage.
☒ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
☐ without fee and with the appropriate certification above.
☐ without fee and with a new CPA application.
☐ without fee and with a Request for Continued Examination.
☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

☒ Please charge any additional fees for the papers being filed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER LLP US

Dale S. Lazar
Registration No. 28,872
Attorney for Applicant

DSL/jcc
P.O. Box 2758
Reston, VA 20195
Tel. 703.773.4000
Fax (703) 773-5200

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/694,261	03/30/2007	MIHAI CHIRIAC	339000-P0004

CONFIRMATION NO. 1684

47604
DLA PIPER US LLP
P. O. BOX 9271
RESTON, VA20195

Date Mailed. 02/14/2008

NOTICE OF NEW OR REVISED PROJECTED PUBLICATION DATE

The above-identified application has a new or revised projected publication date. The current projected publication date for this application is CPUB. If this is a new projected publication date (there was no previous projected publication date), the application has been cleared by Licensing & Review or a secrecy order has been rescinded and the application is now in the publication queue.

If this is a revised projected publication date (one that is different from a previously communicated projected publication date), the publication date has been revised due to processing delays in the USPTO or the abandonment and subsequent revival of an application. The application is anticipated to be published on a date that is more than six weeks different from the originally-projected publication date.

More detailed publication information is available through the private side of Patent Application Information Retrieval (PAIR) System. The direct link to access PAIR is currently http://pair.uspto.gov. Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Questions relating to this Notice should be directed to the Office of Patent Publication at 1-888-786-0101.

PART 1 - ATTORNEY/APPLICANT COPY

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/694,261	03/30/2007	MIHAI CHIRIAC	339000-P0004

CONFIRMATION NO. 1684

47604
DLA PIPER US LLP
P. O. BOX 9271
RESTON, VA20195

Title: METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

Publication No. US-2008-0040710-A1
Publication Date: 02/14/2008

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Pre-Grant Publication Division, 703-605-4283

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
11/694,261	03/30/2007	2193	2750	339000-P0004	47	5

CONFIRMATION NO. 1684

47604
DLA PIPER US LLP
P. O. BOX 9271
RESTON, VA20195

FILING RECEIPT

Date Mailed: 04/12/2007

Receipt is acknowledged of this regular Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please mail to the Commissioner for Patents P.O. Box 1450 Alexandria Va 22313-1450. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).**

Applicant(s)

MIHAI CHIRIAC, Bucharest, ROMANIA;

Assignment For Published Patent Application

PREVX LIMITED, PRIDE PARK, UNITED KINGDOM

Power of Attorney: The patent practitioners associated with Customer Number 47604

Domestic Priority data as claimed by applicant

This appln claims benefit of 60/789,156 04/05/2006

Foreign Applications

If Required, Foreign Filing License Granted: 04/11/2007

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US11/694,261**

Projected Publication Date: 10/11/2007

Non-Publication Request: No

Early Publication Request: No

Title

METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the

subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

Electronic Patent Application Fee Transmittal

Application Number:	11694261
Filing Date:	
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Filer:	Dale S. Lazar
Attorney Docket Number:	339000-P0004

Filed as Large Entity

Utility Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Independent claims in excess of 3	1201	1	200	200
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**200**

Electronic Acknowledgement Receipt

EFS ID:	1660024
Application Number:	11694261
International Application Number:	
Confirmation Number:	1684
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Customer Number:	47604
Filer:	Dale S. Lazar
Filer Authorized By:	
Attorney Docket Number:	339000-P0004
Receipt Date:	06-APR-2007
Filing Date:	
Time Stamp:	09:30:29
Application Type:	Utility

Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	$200
RAM confirmation Number	1482
Deposit Account	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)

1	Fee Worksheet (PTO-06)	fee-info.pdf	8167	no	2

Warnings:	
Information:	
Total Files Size (in bytes):	8167

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UTILITY PATENT APPLICATION TRANSMITTAL (Only for new nonprovisional applications under 37 CFR 1.53(b))	Attorney Docket No.	339000-P0004
	First Inventor or Application Identifier	Mihai CHIRIAC
	Title	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
PO Box 1450
Alexandria, VA 22313-1450

1. ☐ Fee Transmittal Form (e.g. PTO/SB/17)
(Submit an original and a duplicate for fee processing)
☐ Applicant claims small entity status.
2. ☒ Specification — Total Pages 44
3. ☒ Drawing(s) *(35 U.S.C. 113)* — Total Sheets 2
4. ☒ Oath or Declaration — Total Pages 1
   a. ☒ Newly executed (original or copy)
   b. ☐ Copy from a prior application *(37 C.F.R. §1.63(d))*
   (for continuation/divisional with box 16 completed)
5. ☐ Incorporation By Reference (*usable if box 4B is checked*)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4B, is considered to be part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

ACCOMPANYING DOCUMENTS
6. ☐ Assignment Papers (cover sheet & documents(s))
7. ☐ 37 C.F.R. §3.73(b) Statement ☐ Power of Attorney
(when there is an assignee)
8. ☐ English Translation Document *(if applicable)*
9. ☒ Information Disclosure Statement (IDS)/PTO-1449 ☒ Copies of IDS Citations
10. ☐ Preliminary Amendment
11. ☐ White Advance Serial No. Postcard
12. ☐ Certified Copy of Priority Document(s)
(if foreign priority is claimed)
13. ☐ Request for Priority
14. ☒ Application Data Sheet
15. ☐ Other:

16. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below:
☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application no.:
Prior application information: Examiner: Group Art Unit:

17.
☐ This application is a ☐ Continuation ☐ Division ☐ Continuation-in-part (CIP)
of application Serial No. Filed on

☒ This application claims priority of provisional application Serial No. 60/789,156, Filed April 5, 2006

18. CORRESPONDENCE ADDRESS

Customer No. 47604

Name	Dale S. Lazar	Registration No.	28,872		
Signature		Date	March 30, 2007	Telephone	703.773.4149

REST1\593685.1

APPLICATION DATA SHEET

Application Information

Application Number::	NEW
Filing Date::	3/30/07
Application Type::	REGULAR
Subject Matter::	UTILITY
Suggested Classification::	
Suggested Group Art Unit::	
CD-ROM or CD-R::	NONE
Number of Disks::	
Number of copies of CDs::	
Sequence Submission::	
Computer Readable Form (CRF)::	NO
Number of Copies of CRF::	
Title::	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
Attorney Docket Number::	339000-P0004
Request for Early Publication::	NO
Request for Non-Publication::	NO
Suggested Drawing Figure::	
Total Drawing Sheets::	2
Small Entity::	NO
Latin Name::	
Variety Denomination Name::	
Petition Included::	NO
Petition Type::	
Licensed US Govt. Agency::	
Contract of Grant Numbers::	
Secrecy Order in Parent Appln.::	NO

INITIAL
3/30/2007

APPLICANT INFORMATION

Applicant Authority Type::	INVENTOR
Primary Citizenship Country::	ROMANIA
Status::	FULL CAPACITY
Given Name::	MIHAI
Middle Name::	
Family Name::	CHIRIAC
Name Suffix::	
City of Residence::	BUCHAREST
State or Province of Residence::	
Country of Residence::	ROMANIA
Street of mailing address::	STR. ALEEA ZORELELOR, NR.3, BI 44B, AP. 122
City of mailing address::	BUCHAREST
State or Province of mailing address::	
Country of mailing address::	ROMANIA
Postal or Zip Code of mailing address::	061184

CORRESPONDENCE INFORMATION

CORRESPONDENCE NUMBER::	47604
PHONE NUMBER::	703-773-4149
FAX NUMBER::	703-773-5200
E-MAIL ADDRESS::	DALE.LAZAR@DLAPIPER.COM

REPRESENTATIVE INFORMATION

INITIAL
3/30/2007

Representative Customer Number:: 47604

DOMESTIC PRIORITY INFORMATION

Application::	Continuity Type::	Parent Application::	Parent Filing Date::
This Application	An application claiming the benefit under 35 USC 119(e)	60/789,156	April 5,2006

FOREIGN PRIORITY INFORMATION

Country::	Application Number::	Filing Date::	Priority Claimed

ASSIGNEE INFORMATION

Assignee Name::	PREVX LIMITED
Street mailing address::	3 NEW PARK PLACE
City mailing address::	PRIDE PARK
State or Province of mailing address::	DERBY
Country of mailing address::	UNITED KINGDOM
Postal or Zip Code of mailing address::	DE24 8DZ

INITIAL
3/30/2007

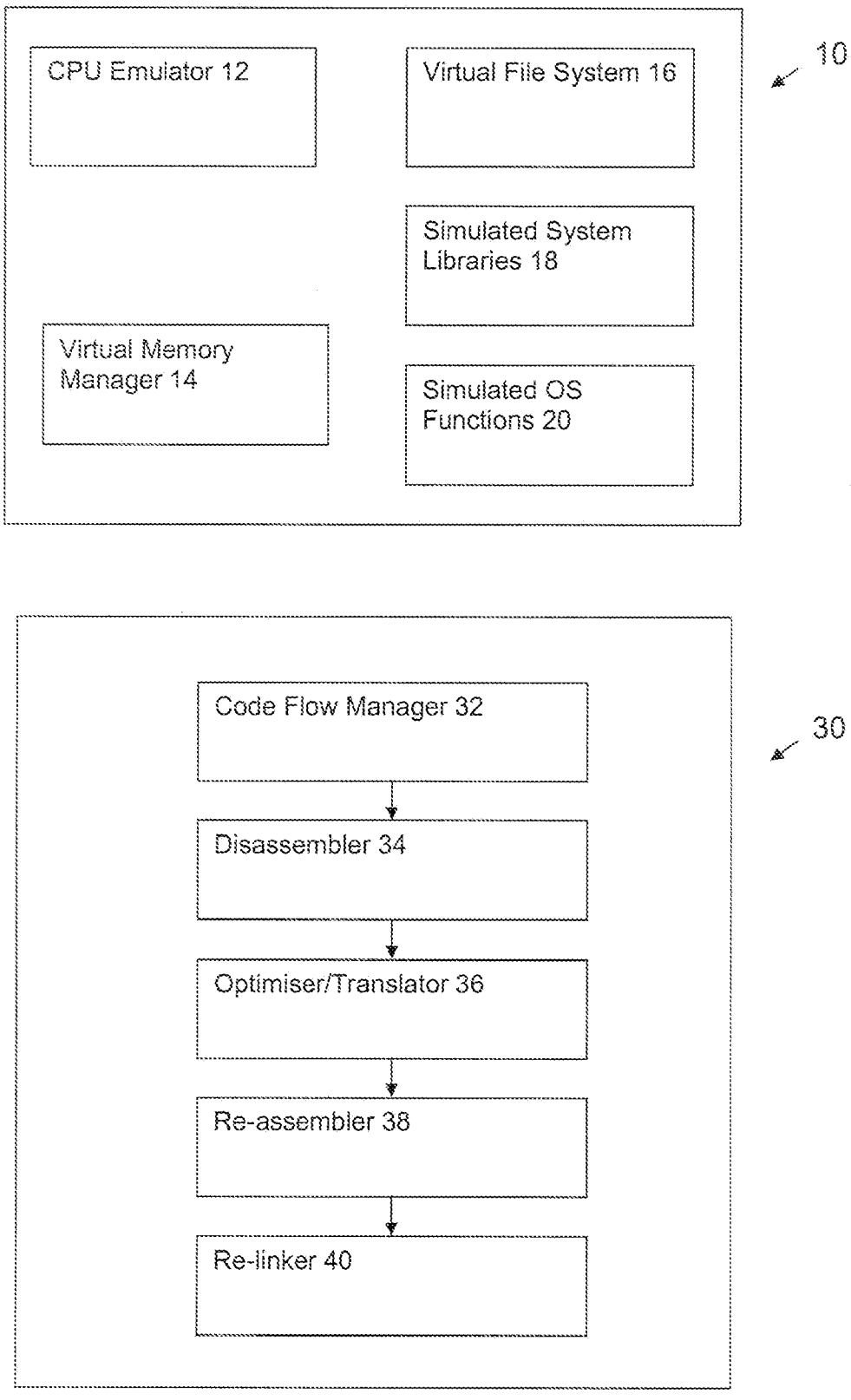

CPU Emulator 12
Virtual File System 16
10
Simulated System Libraries 18
Virtual Memory Manager 14
Simulated OS Functions 20
Code Flow Manager 32
30
Disassembler 34
Optimiser/Translator 36
Re-assembler 38
Re-linker 40

Fig. 1

2/2

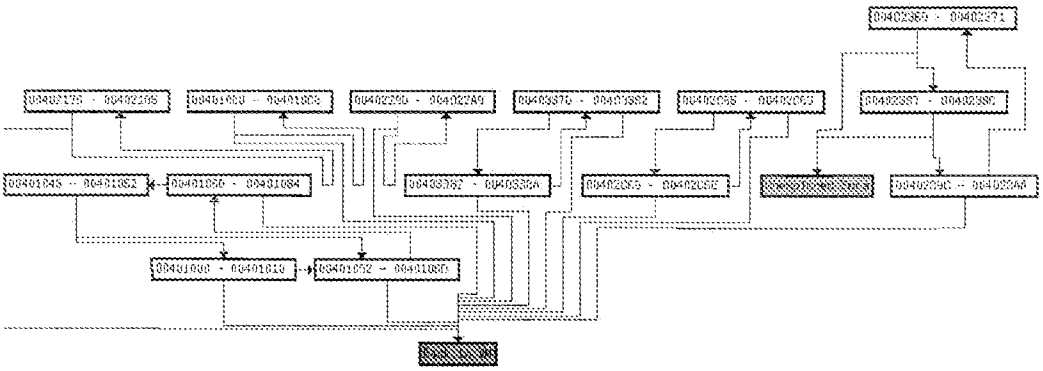


Fig. 2

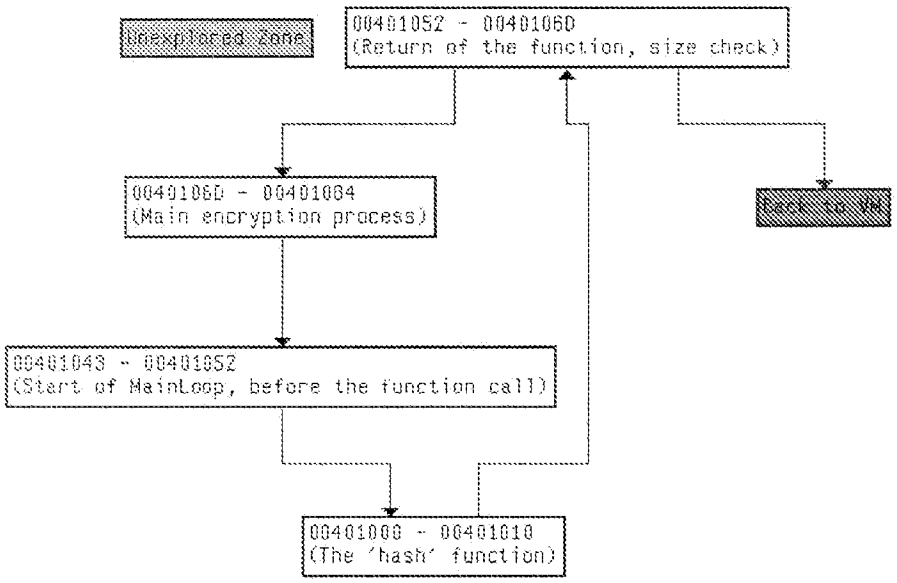

Unexplored zone
00401052 - 0040106D
(Return of the function, size check)
0040106D - 00401084
(Main encryption process)
00401043 - 00401052
(Start of MainLoop, before the function call)
00401000 - 00401010
(The 'hash' function)

Fig. 3

ALL PATENTS, INCLUDING DESIGN FOR APPLICATION BASED ON PCT, PARIS CONVENTION, NON PRIORITY, OR PROVISIONAL APPLICATIONS

COMBINED DECLARATION AND POWER OF ATTORNEY U.S.A.

FOR ATTORNEYS' USE ONLY

ATTORNEYS' DOCKET NO. 339000-P0004

As a below named inventor, I declare that my residence, post office address and citizenship are stated below next to my name, the information given herein is true, that I believe that I am the original, first and sole inventor (if only one name is listed at 101 below), or an original, first and joint inventor (if plural inventors are named below at 101-103, or on additional sheets attached hereto) of the subject matter which is claimed and for which patent is sought on the invention entitled:

Title DYNAMIC CODE TRANSLATOR

which is described and claimed in:

- ☐ PCT International Application No. , filed , (if applicable) and amended on
- ☒ the attached specification
- ☐ the specification in application Serial No. , filed

I (We) hereby state that I (We) have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I (We) acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I (We) hereby claim foreign priority benefits under Title 35, United States Code, §119 (a)-(d) or §365 (b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT international application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate or §365(b) having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s)			Priority Claimed	
(Number)	(Country)	(Day/Month/Year Filed)	☐ Yes	☐ No
(Number)	(Country)	(Day/Month/Year Filed)	☐ Yes	☐ No
(Number)	(Country)	(Day/Month/Year Filed)	☐ Yes	☐ No

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Application No. 60/789,156	Filing Date April 5, 2006	Application No.	Filing Date

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s), or §365 (c) of any PCT international application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT international application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR §1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application.

(Application Serial No.)	(Filing Date)	(Status: patented, pending, abandoned)

POWER OF ATTORNEY: I (We) hereby appoint DLA Piper US LLP, Intellectual Property Group, telephone number (730) 773-4000 (to whom all communications are to be directed), and persons of that firm who as associated with USPTO Customer No.

47604

individually and collectively my attorneys, with full powers of substitution and revocation, to prosecute this application and to transact all business in the USPTO connected therewith.

Inventor(s) name must include at least one unabbreviated first or middle name.

101 FULL NAME OF INVENTOR	FAMILY NAME: CHIRIAC	GIVEN NAME: Mihai	MIDDLE NAME:	
RESIDENCE & CITIZENSHIP	CITY: Bucharest	STATE OR FOREIGN COUNTRY: Romania	COUNTRY OF CITIZENSHIP:	
POST OFFICE ADDRESS	POST OFFICE ADDRESS: Str. Aleea Zorelelor, nr. 3, Bl 44B, Ap. 122	CITY: Bucharest	STATE OR COUNTRY: Romania	ZIP CODE: 061184
102 FULL NAME OF INVENTOR	FAMILY NAME:	GIVEN NAME:	MIDDLE NAME:	
RESIDENCE & CITIZENSHIP	CITY:	STATE OR FOREIGN COUNTRY:	COUNTRY OF CITIZENSHIP:	
POST OFFICE ADDRESS	POST OFFICE ADDRESS:	CITY:	STATE OR COUNTRY:	ZIP CODE:
103 FULL NAME OF INVENTOR	FAMILY NAME:	GIVEN NAME:	MIDDLE NAME:	
RESIDENCE & CITIZENSHIP	CITY:	STATE OR FOREIGN COUNTRY:	COUNTRY OF CITIZENSHIP:	
POST OFFICE ADDRESS	POST OFFICE ADDRESS:	CITY:	STATE OR COUNTRY:	ZIP CODE:

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

SIGNATURE OF INVENTOR 101*	SIGNATURE OF INVENTOR 102*	SIGNATURE OF INVENTOR 103*
DATE 29.03.2007	DATE	DATE

☐ Additional inventors are named on separately numbered sheets attached hereto

DLA Piper US LLP
P.O. Box 9271, Reston, VA 20195

Docket No.: 339000-P0004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION:	Mihai CHIRIAC	Conf. No.	Unknown
SERIAL NUMBER:	New Application	Group:	Unknown
FILED:	March 30, 2007	Examiner:	Unknown

FOR: METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. 1.97

Assistant Commissioner for Patents
PO BOX 1450
ALEXANDRIA, VA 22313-1450

March 30, 2007

Sir:

Applicant(s) wish(es) to disclose the following information.

REFERENCES

☒ Applicant(s) wish(es) to make of record the documents listed on the attached Form PTO-1449. Copies of the listed documents are attached, where required, as are either statements of relevancy or any readily available full or partial English translations of any non-English-language documents.

EXPLANATION OF RELEVANCE

RELATED CASES

☐ Attached is a list of Applicant's(s') pending applications and issued patents which may be related to the present application. Copies of the documents, where required, are attached along with Form PTO-1449.

CERTIFICATION

The undersigned certifies that

☐ each item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application for the first time (to the knowledge of the undersigned, having made reasonable inquiry)not more than three months prior to the filing of this statement.

☐ no item of information contained in this Information Disclosure Statement was cited in a communication from a foreign or international patent office in a counterpart foreign or international application or, to the knowledge of the undersigned, having made reasonable inquiry, was known to any individual designated in 37 C.F.R. 1.56(c) more than three months prior to the filing of this statement.

BASIS FOR CONSIDERATION

This Information Disclosure Statement is filed:

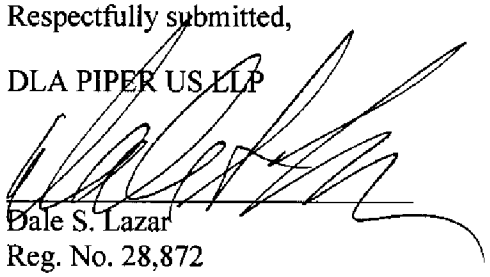
- ☒ without fee and within three months of the filing date of the application.
- ☐ without fee and within three months of the date of entry of the U.S. national stage.
- ☐ without fee and before the mailing date of a first Office Action on the merits (to the knowledge of the undersigned).
- ☐ without fee and with the appropriate certification above.
- ☐ without fee and with a new CPA application.
- ☐ without fee and with a Request for Continued Examination.
- ☐ with fee and before the mailing date of any Final Office Action, Notice of Allowance or an action that otherwise closes prosecution (to the knowledge of the undersigned).
- ☐ with fee, appropriate certification above, and before payment of the Issue Fee.

DEPOSIT ACCOUNT

- ☒ Please charge any additional fees for the papers being filed herewith and for which no check is enclosed herewith, or credit any overpayment to Deposit Account No. 50-3266.

Respectfully submitted,

DLA PIPER US LLP



Dale S. Lazar
Reg. No. 28,872
Attorney of Record

P.O. Box 9271
Reston, VA 20195
Tel. (703) 773-4000
Fax (703) 773-5200

DSL/jsk

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	1	of	2

Complete if Known	
Application Number	New Application
Filing Date	March 30, 2007
First Named Inventor	Mihai CHIRIAC
Art Unit	Unknown
Examiner Name	Unknown
Attorney Docket Number	339000-P0004

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -7013483	03-14-2006	ALADDIN KNOWLEDGE SYSTEMS LTD	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 96/30829	10-03-1996	APPLE COMPUTER, INC.		
		WO 99/15966	04-01-1999	SYMANTEC CORPORATION		
		EP 1536341 A2	06-01-2005	MICROSOFT CORPORATION		

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional). 2 See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. 3 Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). 4 For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. 5 Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. 6 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer,

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.
U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND **TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Substitute for form 1449A/PTO				***Complete if Known***	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT				*Application Number*	New Application
				Filing Date	March 30, 2007
				First Named Inventor	Mihai CHIRIAC
				Art Unit	Unknown
(Use as many sheets as necessary)				*Examiner Name*	Unknown
Sheet	2	of	2	*Attorney Docket Number*	339000-P0004

NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		"DEFEATING POLYMORPHISM: BEYOND EMULATION, Adrian E. STEPAN	
		"GHOSTBUSTER 3", Mihai CHIRIAC	

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
[1] Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

US007013483B2

(12) **United States Patent**
Cohen et al.

(10) **Patent No.: US 7,013,483 B2**
(45) **Date of Patent: Mar. 14, 2006**

(54) **METHOD FOR EMULATING AN EXECUTABLE CODE IN ORDER TO DETECT MALICIOUSNESS**

(75) Inventors: **Oded Cohen**, Tivon (IL); **Inbal Meir**, Haifa (IL); **Yanki Margalit**, Ramat-Gan (IL); **Dany Margalit**, Ramat-Gan (IL)

(73) Assignee: **Aladdin Knowledge Systems Ltd.**, Tel Aviv (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 286 days.

(21) Appl. No.: **10/335,871**

(22) Filed: **Jan. 3, 2003**

(65) **Prior Publication Data**

US 2004/0133796 A1 Jul. 8, 2004

(51) **Int. Cl.**
H04L 713/20 (2006.01)

(52) **U.S. Cl.** **726/25**; 726/23; 726/24; 713/165; 713/167; 713/188

(58) **Field of Classification Search** 713/188, 713/200–201, 167, 165; 380/4, 3
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,398,196	A	* 3/1995	Chambers	714/28
5,842,002	A	* 11/1998	Schnurer et al.	703/21
5,978,917	A	* 11/1999	Chi	713/201
6,035,423	A	3/2000	Hodges et al.	
6,067,410	A	* 5/2000	Nachenberg	703/28
6,269,456	B1	7/2001	Hodges et al.	
6,357,008	B1	* 3/2002	Nachenberg	713/200
2002/0013910	A1	1/2002	Edery et al.	
2002/0073330	A1	6/2002	Chandnani et al.	

FOREIGN PATENT DOCUMENTS

WO	WO01/88673	11/2001
WO	WO02/37740	5/2002

OTHER PUBLICATIONS

http://livedocs.macromedia.com/colfusion/6/Developing_coldFusion_MX_Applications_with_CFML/queryDB6.htm.*
Pleszkoch et al., "Improving Network System with Function Extraction Technology for Automated Calculation of Program Behavior", Proceeding of the 37th Hawaii International Conference on System Science, 2004.*
http://www.cgisecurity.com/owasp/html/ch11s04.*

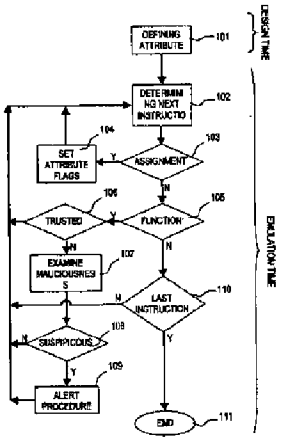
* cited by examiner

Primary Examiner—Gregory Morse
Assistant Examiner—Tongoc Tran
(74) *Attorney, Agent, or Firm*—Mark M. Friedman

(57) **ABSTRACT**

The present invention is directed to a method for emulating an executable code, whether it is a human-readable code (e.g., macro and script) or a compiled code (e.g. Windows executable). At the design time, one or more content attributes are defined for the variables of the code. A content attribute indicates a property with relevance to maliciousness, e.g. Windows directory, a random value, ".EXE" at the right of a string, etc. A content attribute may be implemented, for example, by a flag. Also defined at the design time, is a list of malicious states, where a malicious state comprises at least the combination of a call to a certain system function with certain content, as the calling parameter(s). When emulating an assignment instruction, the attributes of the assigned variable are set according to the assigned content. When emulating a mathematical operator, a content mathematics is also applied. When emulating a function call, the current state (i.e. the function identity and the calling content and values) is compared with the predefined malicious states, and if at least one malicious state corresponds, then the maliciousness of the code is determined.

24 Claims, 1 Drawing Sheet



www.freepatentsonline.com

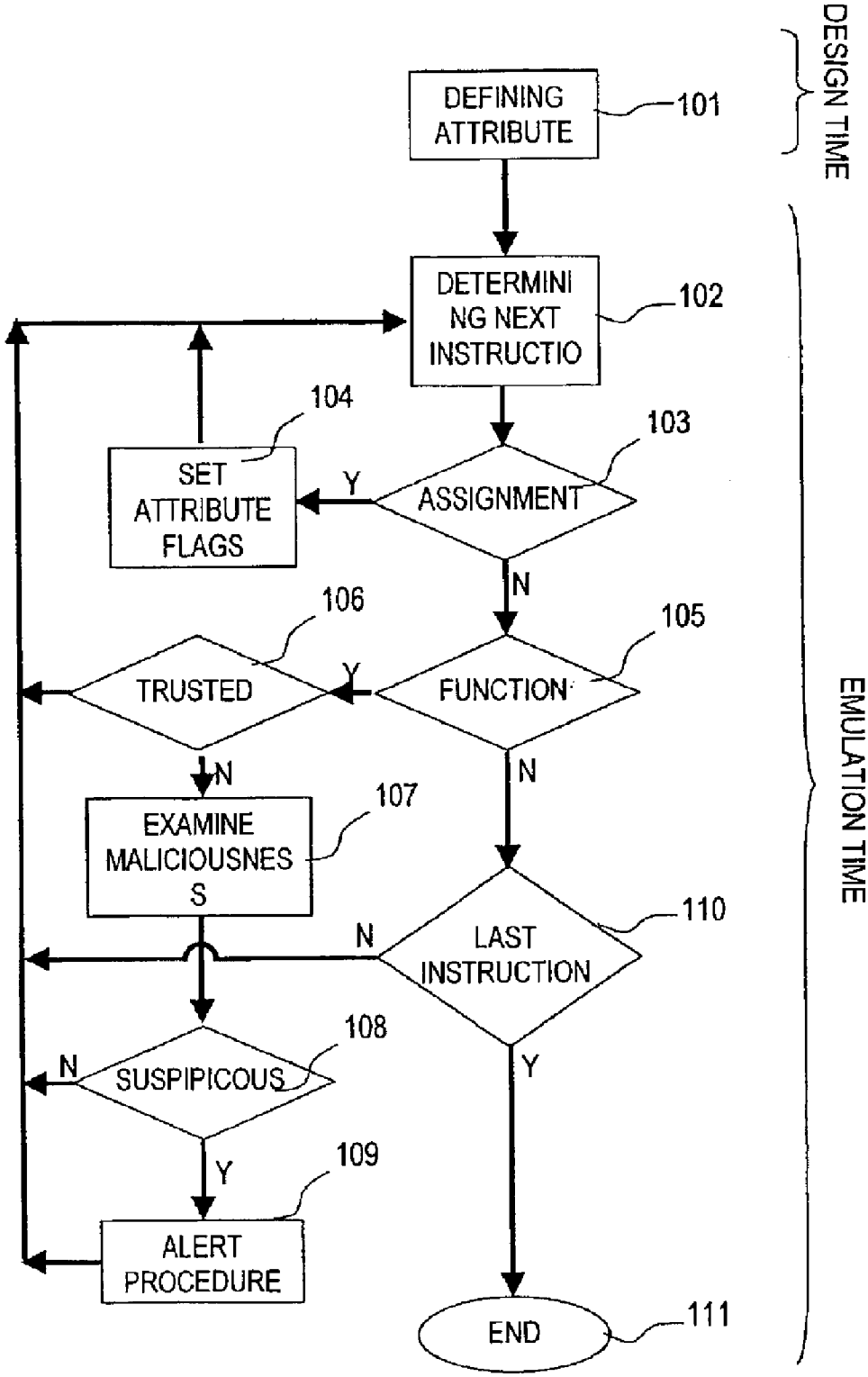


Fig. 1

www.freepatentsonline.com

METHOD FOR EMULATING AN EXECUTABLE CODE IN ORDER TO DETECT MALICIOUSNESS

FIELD OF THE INVENTION

The present invention relates to the field of detecting presence of virus and other kinds of malicious code by emulation.

BACKGROUND OF THE INVENTION

Emulation is one of the methods of detecting viral activity. The emulator creates a virtual environment under which the code of the tested sample is examined instruction-by-instruction. The viral activity is detected by correlating the changes with a viral behavior.

Emulation has several advantages in comparison to other methods of detecting and/or preventing viral damages, since it helps to detect polymorphic viruses, encrypted viruses and unknown viruses, which cannot be detected by other methods.

A programmer can indicate the maliciousness of a source code just by reviewing the code. For example, if he identifies a loop that gets the name of the files of a directory and deletes them, he can assume that the code is malicious. Contrary to a human programmer, an emulator, which is a machine, cannot indicate viral code just by reviewing the code, but it has to execute the code while comparing the caused changes with known patterns. In other words, the emulator, being a machine, is absent of human intelligence.

It is an object of the present invention to provide a method for emulating an executable code in order to detect maliciousness, which is more "intelligent" than known in the prior art.

Other objects and advantages of the invention will become apparent as the description proceeds.

SUMMARY OF THE INVENTION

The present invention is directed to a method for emulating an executable code, whether it is a human-readable code (e.g., macro and script) or a compiled code (e.g. Windows executable). At the design time, one or more content attributes are defined for the variables of the code. A content attribute indicates a property with relevance to maliciousness, e.g. Windows directory, a random value, ".EXE" at the right of a string, etc. A content attribute may be implemented, for example, by a flag. Also defined at the design time, is a list of malicious states, where a malicious state comprises at least the combination of a call to a certain system function with certain content, as the calling parameter(s). When emulating an assignment instruction, the attributes of the assigned variable are set according to the assigned content. When emulating a mathematical operator, a content mathematics is also applied. When emulating a function call, the current state (i.e. the function identity and the calling content and values) is compared with the predefined malicious states, and if at least one malicious state corresponds, then the maliciousness of the code is determined.

The system function can be, for example, a part of the programming language of the executable, a function provided by the operating system under which the code is to be executed, a function provided by a linked package of routines.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood in conjunction with the following figure:

FIG. 1 schematically illustrates a method for emulating an executable, in order to detect malicious code.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The term Macro in conjunction with computers, refers herein to an executable code embedded within another entity, like a document or executable code.

The term Script in conjunction with computers, refers in the art to a program built into the HTML code of a Web page that work on almost all Internet applications. Scripts can also be used with a variety of applications that support their scripting syntax. VBScript and JavaScript are examples of scripting languages.

Macros and scripts are "non-compiled executables", i.e. programs wherein their executable code is the source code, contrary to "compiled executables" which are converted to machine-code before execution. ASP and XML are other examples of non-compiled executables.

Due to their human readability, non-compiled executables are exploited by vandals (malicious executables written into scripts and macros). They can do almost anything—infect a system with Trojans, modify files, cause DoS attacks, or spread malicious code using Microsoft Networks or Microsoft Outlook. Macro-viruses that attack Microsoft Word work by copying themselves into the global environment—usually by modifying NORMAL.DOT. Unfortunately, the enhanced power of software applications also gives virus writers the potential to cause more damage.

It should be noted that the examples herein are presented in a human-readable programming code, although they apply to both compiled and non-compiled code. Moreover, the examples are presented in a schematic programming language, rather than an existing programming language.

There are some programming tools and computer resources commonly used by malicious code. For example, the random generator, which is a programming tool provided by almost any operating system and almost any programming language, is commonly used by viruses to prevent the repetition of the same code when duplicating itself, thereby hardening the detection of the malicious code by bit-pattern (also known as "virus signature") scanning. A human-readable code, as a program written in a programming language, makes use of variables. Since it is plain text, the mechanism that duplicates the malicious code can generate random names for its variables, to add non-effective code comments between its effective code lines, etc. The non-effective code may also be generated randomly, thus using the random generator.

The real time clock, which is a programming-tool provided by any operating system and almost any programming language, is another example of a virus-commonly used programming tool, since it enables a late-trigger of the virus code, and also because it can be used as a substitute to the random generator. Moreover, some of the viruses have a late trigger, thereby enabling them to be widely propagated before causing the damage, and consequently their discovery. The late trigger is tuned to a future date, which involves use of the real time clock of the host machine.

Another example of a content used by malicious code in Windows operating system is the Windows folder, which stores Windows system files. A virus may try to change one of the existing files in the Windows folder, create a file with a random name, etc.

A combination of some of the contents described above may also indicate the presence of malicious code within an

www.freepatentsonline.com

executable. For example, creating a new file having a random name in the Windows directory can be due to a virus. However, a more accurate indication of a malicious code can be obtained if this is an executable file. Thus, if the created file has an EXE extension, which is the extension of Windows executables, or VBS, which is the extension of Visual Basic Script, the executable can be classified as malicious.

According to a preferred embodiment of the present invention, a "malicious state" comprises at least executing a certain system function with certain content as its parameter (s). Thus, at least two elements are involved—a certain system function and certain content as the parameter(s).

It should be noted that the term "content" of a variable refers herein also to its value. Nevertheless, in some cases the value has been also mentioned.

A state can be indicated as malicious if it corresponds to at least one pre-defined malicious state. Two states "correspond" if their elements (i.e. function identity and the calling content) are equal or at least "similar (e.g., at least certain percentage of their content/value is equal). Maliciousness can be defined also by tracking the history of the called functions, e.g., if certain functions have been called with certain content, or even without regard-ness to their content.

Identifying a call to a certain function is quite obvious, and almost any compiler parser performs it. However, the concept of dealing with the content of a parameter rather than its value is novel.

For example, the parameter when calling the following function is a random number:

```
MyVariable=Random( )

MyVariable=MyVariable+1

Call MyFunction(My_Variable)
```

Where Random( ) is a random number generator.

The first time MyFunction is performed, its parameter may be, e.g., 0.443213, and the next time it can be, e.g., 0.332455. Thus, from analyzing the text it is not possible to indicate that the parameter has a random nature.

According to a preferred embodiment of the invention, the content of a variable can be indicated as follows:

Each variable of the executable is characterized by several attributes (e.g. flags), which points out on its content (referred herein as content attributes). For example:

- a flag to indicate if the variable has been assigned with the Windows folder name;
- a flag to indicate if the variable has been assigned with a random value;
- a flag to indicate if the variable has been assigned with the real time;
- a flag to indicate if the variable has been assigned with the value ".EXE", which is the extension of a type of executable file under the Windows operating system;

and so forth.

While emulating the execution of an executable, the following instructions (for example) are performed:

```
File_name=Windows_directory( )+Random( )+0.2+".EXE"

Create_file(File_name)
```

Where: Windows_directory( ) is a system function that returns a string of the Windows directory, and Random returns a numeric random integer.

In this example, the Windows folder flag, the Random flag and the EXE flag of the variable File_name are turned on.

It should be noted that

```
File_name=Windows_directory( )+Random( )+0.2+".EXE"
```

actually comprises several instructions:

Executing the Random function;

Executing the Windows_directory function;

Mathematical addition;

String addition.

It should also be noted that while some programming languages convert a numeric value to a string value and vice versa when it is required, some compilers have no such ability. Thus, assuming the function Str converts a numeric value to a string, in this case the example should look like:

```
File_name=Windows_directory( )+Str(Random( )+0.2)+".EXE"
```

There is no point in considering assigning a value to variables as malicious operation; however without tracking the history of the assignments to a variable, its content cannot be determined. Thus, the examining stage should be at the point where specific system functions which may cause damage to the computer are called, e.g. a function that creates a file, a function that changes the content of a file, and so forth.

The following instruction also creates an executable file with a random name in the Windows directory:

```
Create_file(Windows_directory( )+Random( )+".EXE")
```

However, instead of using a variable as the parameter of the function Create_file, the name of the file is the parameter. Actually, this is equivalent to creating a temporary variable, and assigning a value to the temporary variable just before the execution of said function:

```
Temp_parameter=Windows_directory( )+Random( )+".EXE"

Call Create_file(Temp_parameter)
```

As mentioned above, according to the present invention, the variables comprise content attributes in addition to the value. Thus, there is a question as to how the content is influenced by mathematical operations. For example:

```
MyValue=123+First_var+Second_var
```

Where, for example, the Real time flag of the variable First_var is turned on and the EXE flag of the variable Second_var is turned on. It is obvious that in this case the content result of the assignment is turning on the Real time flag and the EXE flag of the variable MyValue. Thus, the emulator should be provided with "content mathematics" rules. Such rules, which for example comprise a commutative law of addition, an associative law of addition and so forth, should also take into consideration comparison statements and other features supported by the emulated programming language. Of course it is desired to emulate the values as well, in addition to emulating the content.

Another example:

```
Var_A=GetAddressBook( )

Var_B=Var_A->Count( )
```

Where Var_A has the content type "address book object".

When the operator "->Count( )" is operated on an "address book object", it returns not just a numeric value,

www.freepatentsonline.com

but a numeric value with the content attribute "number of entries in address book", thus adding the "number of entries in address book" to the variable Var_B.

Another example:

```
Var_A=GetAddressBook( )

Var_C=Var_A[2]
```

After the assignment to Var_C, its value will be the second entry of the address book which has been assigned to Var_A. The content attributes of Var_A may be something like "address book database", and the content attribute of Var_C may be something like "address book entry".

FIG. **1** schematically illustrates a method for emulating an executable, in order to detect malicious code.

At step **101**, which is the design step, a group of malicious states is defined. A malicious state is the combination of a call to a certain system function with certain content. This is a preliminary stage, which is performed at the anti-virus firm, as a part of the anti-virus emulator design. Of course the list of the content attributes and/or functions can reside on a file, and modified later (whether at the anti-virus firm residence or at the installation site).

At the next step **102**, which is performed at the emulation time (i.e. run time), the next instruction to be performed is determined.

At the next step **103**, the type of the instruction is examined. If it is an assignment instruction, then the flow continues with step **104**, otherwise the flow continues with step **105**.

At step **104**, the instruction is analyzed in order to determine the content attribute(s) of the assigned variable to be set, and the corresponding values are set, and from there, the flow returns to step **102**.

At step **105**, if the type of the instruction is a function call, then the flow continues with step **106**; otherwise the flow continues with step **110**.

From step **106**, if the function is pre-determined as trusted, then the flow continues to step **102**; otherwise flow continues to step **107**.

At step **107** the maliciousness is examined by comparing the current state (i.e. the calling function with the content attributes of the parameters) with the malicious states that have been pre-defined in step **101**. Determining the maliciousness of a code can be indicated if the current state corresponds to at least one of the pre-defined malicious states.

From step **108**, if the instruction seems to be suspicious according to the examination carried out at step **107**, then the flow continues with step **109**, where an alert procedure is performed, and from there to step **102**.

From step **110**, if the last tested instruction is the last instruction (e.g. a Stop instruction), then the emulation stops, otherwise the flow continues with step **102**.

Usually, the emulation continues until the last instruction is performed, and then the results of the emulation are examined. This way, more details are gathered and the maliciousness of the code can be determined more precisely. However, the alert procedure can suspend further execution, thereby shortening the emulation time.

As known to a person of ordinary skill in the art, a function can be declared such that a parameter is transferred by its address or by its value. In case of transferring a value, a temporary parameter is created and its address is transferred as a parameter, like in the other case. In both cases when emulating the code, the variables, whether permanent or temporary, also comprise the content attributes.

It should be noted that the term "function" also refers herein to procedures, software interrupts and so forth.

The emulation can be executed at the user's machine, at the gateway to a network and so forth. For example, while a user may install an emulator to check every incoming HTML file while browsing the Internet/Intranet, an organization may install such an emulator at the gateway to its network, at the mail server and so forth.

Those skilled in the art will appreciate that the invention can be embodied by other forms and ways, without losing the scope of the invention. The embodiments described herein should be considered as illustrative and not restrictive.

What is claimed is:

1. A method for emulating an executable code, said method comprising:

pre-defining at least one content attribute having relevance to maliciousness, for at least one variable of said executable code;

pre-defining at least one malicious state, being a call to a certain system function with certain content and/or value;

upon emulating an assignment instruction, setting the content attributes of the assigned variable according to the assigned content, and the value of the assigned variable;

upon emulating a mathematical operator, employing content mathematics; and upon emulating a function call, checking that the current state, being the combination of the function identity with the content and/or value of the calling parameters, corresponds to at least one of the pre-defined malicious states, thereby indicating the maliciousness of said code.

2. The method according to claim **1**, wherein said at least one content attribute is selected from a group comprising: a programming flag, a Boolean value, a numeric value, and a string value.

3. The method according to claim **1**, wherein said executable includes compiled code.

4. The method according to claim **1**, wherein said executable includes human-readable code.

5. The method according to claim **3**, wherein said compiled code is selected from the group consisting of Windows EXE, Windows DLL, Windows OCX, Linux executables, Solaris executables.

6. The method according to claim **4**, wherein said human-readable code is selected from the group consisting of macro, script, VBScript, JavaScript, BAT, PHP, and ASP.

7. The method according to claim **1**, wherein at least one said checking includes testing an equality of at least one element selected from the group consisting of an identity of said function said content of said calling parameters, and said value of said calling parameters.

8. The method according to claim **1**, wherein at least one said checking includes testing a similarity of at least one element selected from the group consisting of an identity of said function, said content of said calling parameters, and said value of said calling parameters.

9. The method according to claim **1**, wherein at least one said checking includes testing if certain functions have already been called, testing if certain functions have been already called with certain parameters.

10. The method according to claim **1** wherein said system function is a random value.

www.freepatentsonline.com

11. The method according to claim 1 wherein said system function is a tine value.

12. The method according to claim 1 wherein said system function is a certain folder name.

13. The method according to claim 1 wherein said system function is a number of files in a folder.

14. The method according to claim 1 wherein said system function is a certain folder extension.

15. The method according to claim 1, wherein said system function is a function written in a programming language of the executable.

16. The method according to claim 1, wherein said system function is an operating system function of an operating system under which said code is to be executed.

17. The method according to claim 1, wherein at least one said parameter is a permanent parameter.

18. The method according to claim 1, wherein at least one said parameter is a temporary parameter.

19. The method according to claim 1, wherein at least one said parameter is passed by its content.

20. The method according to claim 1, wherein at least one said parameter is passed by its address.

21. A method for emulating, said method comprising:

providing non-compiled code;

pre-defining at least one content attribute having relevance to maliciousness, for at least one variable of said executable code;

pre-defining at least one malicious state, at least one said malicious state being a call to a certain system function with at least one said content attribute;

upon emulating an assignment instruction within said non-compiled code, setting the content attributes of the assigned variable according to the assigned content, and the value of the assigned variable;

upon emulating a mathematical operator within said non-compiled code, employing content mathematics; and

upon emulating a function call within said non-compiled code, checking that the current state, being the combination of the function identity with the content and/or value of the calling parameters, corresponds to at least one of the pre-defined malicious states, thereby indicating the maliciousness of said code.

22. A method for emulating, said method comprising:

providing non-compiled executable code;

pre-defining at least one content attribute having relevance to maliciousness, for at least one variable of said executable code;

pre-defining at least one malicious state, being a call to a certain system function with certain content and/or value;

upon emulating an assignment instruction within said non-compiled executable code, setting the content attributes of the assigned variable according to the assigned content, and the value of the assigned variable;

upon emulating a mathematical operator within said non-compiled executable code, employing content mathematics; and

upon emulating a function call within said non-compiled executable code, checking that the current state, being the combination of the function identity with the content and/or value of the calling parameters, corresponds to at least one of the pre-defined malicious states, thereby indicating the maliciousness of said code.

23. A method for emulating, said method comprising:

pre-defining at least one content attribute having relevance to maliciousness, for at least one variable of said executable code;

pre-defining at least one malicious state, said at least one said malicious state being a call to a certain system function with a variable having a certain content attribute;

upon emulating an assignment instruction, setting the content attributes of the assigned variable according to the assigned content, and the value of the assigned variable;

upon emulating a mathematical operator, employing content mathematics; and

upon emulating a function call, checking that a first current state, being the combination of the function identity with the content attributes of the calling parameters, corresponds to at least one of the pre-defined malicious states, thereby indicating the maliciousness of said code.

24. The method of claim 23 wherein upon emulating a function call a second current state being the combination of the function identity with the content and/or value of the calling parameters is checked.

* * * * *

www.freepatentsonline.com

Electronic Patent Application Fee Transmittal

Application Number:	
Filing Date:	
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Filer:	Dale S. Lazar/Jacqueline Kimvilakani
Attorney Docket Number:	339000-P0004

Filed as Large Entity

Utility Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Utility application filing	1011	1	300	300
Utility Search Fee	1111	1	500	500
Utility Examination Fee	1311	1	200	200
Pages:				
Claims:				
Claims in excess of 20	1202	27	50	1350
Independent claims in excess of 3	1201	1	200	200
Miscellaneous-Filing:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	**2550**

Electronic Acknowledgement Receipt

EFS ID:	1641321
Application Number:	11694261
International Application Number:	
Confirmation Number:	1684
Title of Invention:	METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE
First Named Inventor/Applicant Name:	MIHAI CHIRIAC
Customer Number:	47604
Filer:	Dale S. Lazar/Jacqueline Kimvilakani
Filer Authorized By:	Dale S. Lazar
Attorney Docket Number:	339000-P0004
Receipt Date:	30-MAR-2007
Filing Date:	
Time Stamp:	15:58:52
Application Type:	Utility

Payment information:

Submitted with Payment	yes
Payment was successfully received in RAM	$2550
RAM confirmation Number	489
Deposit Account	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)	Multi Part /.zip	Pages (if appl.)

1	Transmittal of New Application	339000P0004TRANSMITTAL.pdf	609932	no	1
Warnings:					
Information:					
2	Application Data Sheet	339000P0004ADS.pdf	1002466	no	3
Warnings:					
Information:					
This is not an USPTO supplied ADS fillable form					
3	Specification	339000P0004SPECIFICATION.pdf	14949142	no	31
Warnings:					
Information:					
4	Claims	339000P0004CLAIMS.pdf	5496127	no	12
Warnings:					
Information:					
5	Abstract	339000P0004ABSTRACT.pdf	318408	no	1
Warnings:					
Information:					
6	Drawings	339000P0004FIGURES.pdf	573954	no	2
Warnings:					
Information:					
7	Oath or Declaration filed	339000P0004DECLARATION.pdf	2181285	no	1
Warnings:					
Information:					
8	Information Disclosure Statement (IDS) Filed	339000P0004IDS.pdf	950724	no	2
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					

9	Information Disclosure Statement (IDS) Filed	339000P0004PTO1449.pdf	1480893	no	2
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
10	Information Disclosure Statement (IDS) Filed	339000P0004USP7013483.pdf	4589770	no	6
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
11	Foreign Reference	339000P0004WO96-30829.pdf	10090622	no	18
Warnings:					
Information:					
12	Foreign Reference	339000P0004WO99-15966.pdf	21175150	no	46
Warnings:					
Information:					
13	Foreign Reference	339000P0004EP1536341A2.pdf	11848475	no	18
Warnings:					
Information:					
14	NPL Documents	339000P0004NPL.pdf	7463683	no	9
Warnings:					
Information:					
15	NPL Documents	339000P0004NPL2.pdf	4576288	no	7
Warnings:					
Information:					
16	Fee Worksheet (PTO-06)	fee-info.pdf	8593	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	87315512		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (12-04)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

11/694,261

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)		OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))							300
SEARCH FEE (37 CFR 1.16(k), (i), or (m))							500
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))							200
TOTAL CLAIMS (37 CFR 1.16(i))	47 minus 20 =	27	X$25=		OR	X$50	1350
INDEPENDENT CLAIMS (37 CFR 1.16(h))	5 minus 3 =	2	X$100=			X$200=	400
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR						
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))			N/A			N/A	
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	2750

APPLICATION AS AMENDED – PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDI-TIONAL FEE ($)		OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDI-TIONAL FEE ($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDI-TIONAL FEE ($)		RATE ($)	ADDI-TIONAL FEE ($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

3-30-07

Best Available Copy

METHOD, COMPUTER PROGRAM AND COMPUTER FOR ANALYSING AN EXECUTABLE COMPUTER FILE

This application claims the benefit of priority to US application serial no. 60/789156, filed April 5th, 2006, the content of which is hereby incorporated by reference.

The present invention relates to a method, a computer program and a computer for analysing an executable co file.

The term "malware" is used herein to refer generally to any executable computer file that is or contains malicious code, and thus includes viruses, Trojans, etc. For convenience, however, and as is conventional, the term "virus" will also be used generically herein to refer to any of these types of malware.

A typical anti-virus product, such as anti-virus scanning software, scans files to look for signatures in the file that are known to be indicative of the presence of a virus. In order to try to defeat anti-virus products, the malware writers use obfuscation techniques in order to attempt to hide the signature of the virus code from detection. Typically, the obfuscation involves encrypting or packing the viral code. Anti-virus products typically incorporate fixed decryption algorithms to decrypt or unpack files that are encrypted or packed using known encryption algorithms, but this means that the anti-virus product has to be continually updated to include new decryption algorithms as new encryption algorithms appear. Moreover, the same or similar basic viral codes are encrypted using different encryption algorithms, which make it even harder for the anti-virus products to detect the

Best Available Copy

viruses. Furthermore, modern so-called polymorphic viruses make the job of anti-virus products even more difficult because a polymorphic virus changes its encryption algorithm and the corresponding encryption keys each time it replicates from one computer to another. Accordingly, it is now very difficult or almost impossible for the makers of the anti-virus products to use fixed algorithms to decrypt polymorphic viruses.

Accordingly, different techniques were developed for decrypting polymorphic viruses in order to expose the underlying viral code. These include for example emulation and cryptanalysis. However, each of these techniques is rather slow, relatively speaking, and cannot successfully be run in real time on for example a user's personal computer. For example, in the early days of personal computers, say from about 1986 onwards, viruses were typically very small files of a few kilobytes (KB) and such small files can easily be emulated or analyzed in a short period of time (of less than 1 second say). On the other hand, polymorphic viruses are now often written in a high level language and can have a size of 2 megabytes (MB) or more and can contain upwards of 200 million instructions. Known emulation techniques can take 20 seconds to 1 minute or more to emulate such files, which clearly makes such techniques inappropriate and unacceptably slow for real-time "on the fly" use. Indeed, some current anti-virus products simply ignore files having large sizes (say over 1 MB) and/or those files that contain more than a certain number of instructions (say more than 10 or 20 million instructions), which clearly is not safe for the user.

Best Available Copy

Reference is now made to the paper "Defeating Polymorphism: Beyond Emulation" by Adrian Stepan of Microsoft Corporation published in "Virus Bulletin Conference October 2005". The entire content of the paper "Defeating Polymorphism: Beyond Emulation" is hereby incorporated by reference. This paper provides a good overview of current problems with polymorphic viruses. As discussed in the paper by Stepan, conventional emulation is relatively slow. In four examples given in the paper, decryption of a number of files using emulation took around one minute or more. This is clearly unacceptable to users who will expect any virus checking to be carried out almost invisibly, and within a handful of seconds at the most.

A method for decrypting malware to try to overcome this problem is disclosed in the paper by Stepan. In simple terms, the method operates as follows. First, the type of file is checked (whether for example the file is a Portable Executable file as defined by Microsoft or the Executable and Linking Format used in Linux). Then a virtual environment that depends on the file type is created, including for example virtual memory, virtual registers, etc. For example, a Windows-like environment is created for a PE file, and a Linux-like environment is created for an Elf file. The file is then mapped to the virtual environment. Then an instruction is fetched from the Instruction Pointer, the instruction is translated, and emulated. The Instruction Pointer is then updated. Depending on conditions at the time, the virtual environment is exited or the next instruction is translated and emulated.

Best Available Copy

To overcome the speed problems of emulation, the paper by Stepan proposes a technique in which the malware code is partitioned or disassembled into basic blocks of code. Then, a table is created having an entry for each block of code. Searching is carried out in the proposed method for each block of code rather than for each instruction. The use of blocks of code means that the table has significantly fewer entries than if it were to store an entry for each instruction, such that the search process is quicker. Once the basic blocks have been created from the original malware code, the decryption process in the method disclosed by Stepan proceeds as follows. In the virtual environment, a check is first made as to whether the current basic block has already been translated. If yes, then the translated code for that basic block is then executed; the Instruction Pointer is then updated and either the process exited or returned to the next basic block. If the current basic block has not already been translated, then it is translated and the translated code is executed in order to emulate the instructions associated with that basic block. The Instruction Pointer is then updated and either the process exited or returned to the next basic block.

As disclosed in the test examples in the paper by Stepan, this so-called dynamic translation of the malware code speeds up the decryption process by a factor of four to five or so, taking between seven and fifteen seconds to decrypt the test files. This process can handle on average about twenty million instructions per second, though in specific optimized cases, perhaps up to fifty million instructions per second can be handled. This processing time, especially the average processing time, is still

Best Available Copy

nevertheless too slow to be acceptable for real time use, especially given the large file sizes of modern polymorphic viruses.

According to a first aspect of the present invention, there is provided a method of unpacking/decrypting an executable computer file using a host computer, the method comprising: partitioning the executable computer file into plural basic blocks of code; translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer; linking at least some of the translated basic blocks of code in memory of the host computer; and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

In the present context, a basic block is a "straight-line" piece of code that has no jumps (or "branches") or jump targets within its body. If a jump target is present in the block, it is only at the start of the block (and in that case defines the start of the block). A jump is only at the end of the block, and thus defines the end of the block.

The translated code that is obtained in the preferred embodiment of this aspect is functionally equivalent to the corresponding original code but is safe to execute on the host computer. The memory is typically real memory of the host computer.

Best Available Copy

Linking the translated basic blocks makes the execution of those translated basic blocks faster. This is especially the case where the basic blocks are found in loops in the unpacking/decryption process. For example, when unpacking or decrypting a file, some blocks may execute 100,000 times or more, meaning that the unpacking/decryption loops can take more than 90% of the time needed to unpack/decrypt the file in the prior art methods. Linking provides a direct jump between the linked blocks, which can increase the speed of operation by up to a factor of 5 or more allowing on average around 100 million instructions per second to be processed, making the method viable in real-time with minimal interruption to the user even in the case of large and/or polymorphic viruses.

In a preferred embodiment, the method comprises, when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory.

In a preferred embodiment, a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the

Best Available Copy

memory address that follows the first translated basic block.

Preferably, the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated. This allows for the fact that during the translation, the size of some already-translated blocks and their addresses in memory may be changed and therefore linked basic blocks need to be re-linked to their new addresses. It should be borne in mind that a translated basic block may branch to a non-translated basic block. Accordingly, it is preferred that indirect jumps are used and that blocks are re-linked dynamically as they are translated.

In a preferred embodiment, the method comprises passing flow of the method to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks. In such cases, the virtual machine can be used to overcome the cache miss or CPU exception, which might otherwise halt the execution on the host computer, for example by providing updated cache addresses or information to overcome the CPU exception. Flow can then return to the execution on the host computer.

In a preferred embodiment, the method comprises emulating the executable file by executing the basic blocks of code, wherein the basic blocks of code that are translated are only those that execute more than a predetermined number of times during the emulation. Preferably, the value for said predetermined number of times is dynamically variable during the emulation.

Best Available Copy

In a preferred embodiment, the method comprises creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block, and comprising emulating the executable file by executing the basic blocks of code, and, during the execution of a basic block, checking at least one of the read page and the write page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

In practice, during the emulation, if there is a cached real address corresponding to the virtual address that is being accessed, that cached real address is used, else a memory mapper can be used to update the address(es) in the cache(s). The use of caches for basic blocks can significantly increase the speed of operation during emulation because, for example, provided there is present in a cache a cached real address that corresponds to a virtual address to be accessed, a translation between virtual memory and real memory addresses or other similar time-consuming operation can be avoided. In some cases it has been found that there is a high number of cache "hits", i.e. where there is present in a cache a cached real address that corresponds to a virtual address to be accessed, and such cases are particularly quick to analyze and unpack/decrypt. Such cases include those where the

Best Available Copy

file is decrypted using linear decryption, common decompression techniques, etc.

In a preferred embodiment, the method comprises creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprises, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

Some embodiments may use a set of cache pages for every instruction.

According to a second aspect of the present invention, there is provided a method of unpacking/decrypting an executable computer file using a host computer, the method comprising: partitioning the executable computer file into plural basic blocks of code; creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and, emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be

Best Available Copy

analyzed to determine whether the executable computer file is or should be classed as malware; wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

As discussed above, the use of caches for basic blocks can significantly increase the speed of operation during emulation because, for example, provided there is present in a cache a cached real address that corresponds to a virtual address to be accessed, a translation between virtual memory and real memory addresses or other similar time-consuming operation can be avoided.

In a preferred embodiment, the method comprises creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprises, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

The emulation is preferably carried out in virtual memory on the host computer. This keeps the real components of the host computer, including for example the real memory, real input/output devices, real storage devices, etc., safe from attack by the file that is being unpacked.

Best Available Copy

The partitioning is preferably carried out dynamically. For example, new blocks might be discovered or existing blocks might be modified as a result of processing blocks that were previously discovered. This means that already-discovered basic blocks may have to be repartitioned and data about them restored as necessary.

In an embodiment, the executable computer file is a Portable Executable (PE) file. This file type is a file type specified by Microsoft Corporation as part of the original WIN32 specifications. Alternatively, the file might be of the Executable and Linking Format (ELF) used in Linux.

According to a third aspect of the present invention, there is provided a computer program for unpacking/decrypting an executable computer file using a host computer, the computer program comprising program instructions for causing a said host computer to carry out a method of: partitioning the executable computer file into plural basic blocks of code; translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer; linking at least some of the translated basic blocks of code in memory of the host computer; and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

Best Available Copy

According to a fourth aspect of the present invention, there is provided a computer program for unpacking/decrypting an executable computer file using a host computer, the computer program comprising program instructions for causing a said host computer to carry out a method of: partitioning the executable computer file into plural basic blocks of code; creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and, emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware; wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

The preferred embodiments of the computer programs provide computer programs that are arranged to carry out the preferred embodiments of the method described above.

According to a fifth aspect of the present invention, there is provided a method of analysing an executable computer file using a host computer, the method comprising:

Best Available Copy

partitioning the executable computer file into plural basic blocks of code; creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; emulating the executable file by executing the basic blocks of code, wherein, during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block; translating those basic blocks of code that execute more than a predetermined number of times during the emulation into translated basic blocks of code that can be executed by the host computer; linking at least some of the translated basic blocks of code in memory of the host computer; and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

This most preferred method provides the maximum potential speed of operation by using emulation where possible and reasonable, using at least a read cache and a write cache for at least some of the basic blocks to minimise the number of times that a virtual address has to be mapped to a real address and vice versa during the

Best Available Copy

execution of the basic blocks in the emulation, using execution of translated basic blocks when the basic blocks are found to be executed more than a certain number of times during emulation, and using linking of at least some of the translated basic blocks where possible to speed the execution of the translated basic blocks.

Preferred embodiments of this method correspond to the preferred embodiments of the methods described above. Similarly, there is also provided a computer program for causing a computer to carry out the method and a computer programmed to carry out the method.

Embodiments of the present invention will now be described by way of example with reference to the accompanying drawings, in which:

Fig. 1 shows schematically a functional block diagram of the operation of an example of a computer program according to an embodiment of the present invention;

Fig. 2 shows schematically a block diagram of a first example of linked basic blocks of code; and,

Fig. 3 shows schematically a block diagram of a second example of linked basic blocks of code.

The preferred embodiment of an aspect of the present invention is implemented in a computer program which in use is installed and runs on a user's personal computer. The preferred embodiment operates generally as follows. In summary, the file is first inspected for type and "obvious" or known indicators that it is malware. Then, the running

Best Available Copy

of the file on the host computer is emulated in a safe, virtual environment created on the host computer so as to allow the file to be unpacked/decrypted and to collect execution-level flags. Then, the image of the decrypted file is analyzed in combination with the API (application program interface) data obtained during the emulation to determine whether or not it is or should be classed as malware. These three main stages will be discussed in some detail below. Importantly, during the emulation stage, if it is found that any particular basic blocks of code are emulated more than a certain number of times, as can happen for example in loops that arise during the decryption process that occurs as part of the emulation of a packed/encrypted file, then those blocks of code are subject to special treatment as will be discussed below. Also importantly, during the emulation stage, cache memories are used for basic block addresses, which can significantly increase the speed of operation.

In the first main stage, the executable computer file that is to be analyzed to determine whether or not it is malware is firstly inspected to determine its format, including whether for example the file is of the Microsoft Portable Executable type or the Linux Executable and Linking Format type. (Much of the present specification assumes that the file is of the PE type, although the principles described herein are directly applicable and easily converted to be applicable to ELF and files.) As is known per se, the file can then be quickly searched to look for any obvious oddities that are known to be present in malware. Then the file is inspected to see if one or more of the following flags are present:

Best Available Copy

(i) the file is packed/compressed/encrypted, which can be determined by for example entropy analysis of the file. Malware is often packed or compressed or encrypted whereas non-malware files are usually not packed or compressed or encrypted;

(ii) the Entry Point is in the last section of the file;

(iii) the code section is writable, which is an indicator for self-modifying code;

(iv) there is a mismatch in the DataDirectory[x].VirtualAddress/Size and the [Last]SectionAdress/Size. Because for example the Microsoft PE loader does not check for these abnormalities, this is the place where many viruses reside;

(v) the file has undeclared internal overlays; and,

(vi) differences in the section flags/name and the DataDirectory flags (e.g. a file with a .reloc section, but no relocations).

Some of the flags are only found in malware. However, some flags may also be present in legitimate files, though such cases are very, very low in number.

In the second main stage, after these flags have been checked, the potentially malicious file is run in a virtual environment that emulates the real host computer. Details of the setting up and operation of the virtual environment for this emulation will be discussed further below. The emulation of the running of the file allows the detection

Best Available Copy

of suspicious instructions, memory accesses, RTC (real time clock) accesses, internal windows structures accesses, exceptions, etc. A list of some of the execution-level flags that can be obtained is as follows:

(i) during the decryption/decompression, it is detected if the file is packed twice, which is common in malware but never used in real applications;

(ii) specific packers (such as Morphine, MoleBox, Mew) used by malware writers are detected;

(iii) common packers (such as UPX, Aspack) with slight modifications are detected. The slight modifications are used by malware writers as an anti-antivirus trick so that standard decompression does not work;

(iv) anti-debugging/anti-tracing tricks are detected;

(v) APIs (application program interfaces) are monitored. Packers/cryptors compress/encrypt the Import Table, so in a normal analysis session it is very hard to tell which functions are being used by a program. Some packers spend a great amount of code doing what is called "API redirection", that is they hijack the original API functions to a set of stubs that are created at runtime by the packer. Also, even if a memory dump of the program is available, the list of APIs may not be available since some packers use hashes and patch the ".idata" section directly. Since the preferred embodiment controls the entire decompression process, it can be detected when the packer's code parses the export table of the DLLs in the virtual environment; and,

Best Available Copy

(vi) exported functions are monitored. Since many Trojans/backdoors use plug-ins (a famous example being Back Orifice), it is possible to say whether the file is a backdoor plug-in or not by simply analyzing the export data, as plug-ins need specific names/structures to work together correctly with the host.

After the (emulated) decompression has finished, not only is the above list of flags and APIs used by the program available, but also the real, unpacked/decrypted data is available, which is used in the subsequent analysis of the file.

In the third main stage, a detailed scan is made in the file's original, decrypted image, and the results combined with the API data to allow a determination to be made as to whether or not the file is or should be treated as malware.

Referring now to Figure 1, the preferred computer program when running can be thought of as providing two main components, a virtual machine engine 10 and an optimisation engine 30. The virtual machine engine 10 is responsible for setting up the virtual machine environment which is used for the emulation in the second main stage discussed above. The optimisation engine 30 is responsible for ensuring that the process can be carried out sufficiently quickly.

The virtual machine engine 10 provides inter alia a CPU emulator 12, a virtual memory manager 14, a virtual file system 16, a set of simulated system libraries 18, and

Best Available Copy

a set of simulated Windows-like operating system functions 20 (such as a memory mapper, a thread scheduler, a structured exception handler, etc.), which together make up an IA32 (32 bit Intel Architecture) emulator. The virtual machine engine 10 provides a module loader function which is responsible for checking the consistency of the main module ("module" being the name given to the in-memory version of a PE file when loaded into memory), i.e. for example whether the file is a valid PE file; for initialising the CPU/virtual memory; mapping the main module inside the virtual memory; setting pointers to the specific "Swap-In" routines; setting up the necessary structures (Process Environment Block, Thread Environment Block, Structured Exception Handler); and setting up a default stack and a default register set.

The CPU emulator 12 provides the necessary emulation of the host CPU, memory, input/output ports, etc. such that the virtual machine is, preferably, a complete clone of the host computer. The virtual CPU works the same as a normal, real CPU, using for example fetch/decode/emulate functions.

The virtual memory manager 14 is capable of virtualising a full 4GB address space (even if the real, host memory is less than 4GB). The virtual memory manager 14 enables analysis of any kind of memory access, thus allowing the detection of potentially malicious files (by, for example, detecting an illegal access to system variables).

The virtual file system 16 enables monitoring of all accesses to the (virtual) file system by a potentially malicious file, which enables analysis through heuristics

Best Available Copy

(i.e. patterns of behaviour). The virtual file system 16 may provide a number of "goat" files, which are files that are intended to lure malware into attacking or otherwise compromising them. Each goat file may have a "dirty" bit to indicate its status, the "dirty" bit initially being cleared and then being set whenever it is detected that the program file being analyzed is attempting to modify or overwrite the goat file.

The simulated system libraries 18, which are loaded on demand during emulation, are set depending on the import table or specific requirements of the program file being analyzed. Each time a simulated system library function gets called during the emulation, the virtual machine engine 10 checks its parameters and logs any unusual activity or results.

The simulated operating system functions 20 enable control over any low-level system functions, thus enhancing the detection capabilities.

The virtual machine engine 10 may also provide a set of virtual processes with antiviral names. These do not have any function other than to try to lure malware into attempting to attack them, typically by ending the process, which is again evidence that the program file being analyzed is malware.

The optimisation engine 30 provides inter alia a code flow manager 32, a disassembler 34, an optimiser/translator 36, a re-assembler 38 and a re-linker 40.

Best Available Copy

During the emulation, the code flow manager 32 creates "basic blocks" of code from the program file instructions that are being emulated by appropriately partitioning the program file instructions. As mentioned above, in the present context, a basic block is a "straight-line" piece of code that has no jumps (or "branches") or jump targets within its body. If a jump target is present in the block, it is only at the start of the block (and in that case defines the start of the block). A jump is only at the end of the block, and thus defines the end of the block. The code flow manager 32 also detects "hot spots", which will be discussed further below.

The code flow manager 32 also detects unusual changes in the code flow. For example, a buffer overflow is considered an unusual change in the code flow. By way of another example, the code flow manager 32 verifies whether "ret"-like instructions link correctly with "call"-like instructions. A return to an exported function, a return to an existing code block, generated by a CALL instruction, and a return to the heap or stack can certainly be regarded as suspicious operations, and are reported accordingly.

Some basic blocks execute more often than others. In an encrypted program file for example, the decryption loops can take more than 90% of the time needed to decrypt the file. Those basic blocks and loops are often called "hot spots". What makes a basic block a "hot spot" depends on the virtual machine implementation. If N is taken as a threshold, then it can be said that "every basic block that executes for N times or more is a hot spot". By way of example only, N may be 40 or 400 or 1,000 or 10,000 or 20,000, or any integer in between. N may be varied

Best Available Copy

dynamically during the emulation, depending on for example the presence and number of the flags present as found during the inspection of the file in the first main stage briefly mentioned above. In general, the value of N can be safely decreased, and the maximum number of emulated instructions safely increased, to provide a large improvement in the speed of analysis of malicious files whilst having only a small impact on the speed of analysis of files that are not malicious. The code flow manager 32 maintains a flag for every basic block and a "basic block execution count" to help in detecting time-consuming, possibly infinite loops, and to allow detection of hot spots by determining whenever a basic block of code is executed more than N times during the emulation.

Returning now to the description of the optimisation engine 30, the disassembler 34 takes the basic blocks of code provided by the code flow manager 32 and translates them into a set of internal instructions for the subsequent stages in the optimisation engine 30.

The optimiser/translator 36 removes unnecessary instructions from the translated instructions provided by the disassembler 34. Most polymorphic decryptors add garbage code to confuse analysis tools, and this is removed by the optimiser/translator 36. In practice, given that most of the variations between polymorphic or metamorphic viruses is in the garbage code and the translated/normalized code for the different generations of a particular virus are often very similar, this removal of unnecessary instructions makes detection of the virus much easier. The optimiser/translator 36 also changes the instruction(s) provided by the disassembler 34 by adapting

Best Available Copy

them to the requirements of the virtual machine in which the emulation is run (e.g. by ensuring that memory accesses by the program file being analyzed are rerouted to the virtual memory), whilst preserving the logic of the program file.

Whenever the code flow manager 32 detects that a particular basic block of code has been emulated more than N times during the emulation process, the basic block is "translated". In other words, functionally equivalent code that is completely safe to execute directly on the real host processor is generated from the basic block. This translation of the basic blocks of code is done by the re-assembler 38.

The "hot spot" basic blocks of code are passed to the real host processor and memory for execution. Not only are the "hot spot" basic blocks executed on the real processor rather than in the virtual machine, but also the "hot spot" basic blocks are "linked" (or, in a sense, "re-linked") in the real memory by the re-linker 40. Linking is a significant contributor to the speed of operation of the preferred embodiment compared to the prior art. Linking provides a direct jump between the linked blocks and thus avoids the need to look up jump addresses with the associated laborious memory accesses, etc. during execution on the real host processor.

Linking of blocks can be achieved in a number of ways. For example, a first translated basic block can be linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an

Best Available Copy

unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block. One or more of these techniques can be used to link any particular pair of blocks.

When a basic block of code is newly translated by the re-assembler 38, a check is made whether an already translated basic block of code has a jump instruction to the newly translated basic block of code. If so, the newly translated basic block of code is linked to the already translated basic block of code in memory.

It should be borne in mind that during the translation, the size of some already-translated blocks and their addresses in memory may be changed and therefore linked basic blocks need to be re-linked to their new addresses. However, a translated basic block may branch to a non-translated basic block. Accordingly, it is preferred that indirect jumps are used and that blocks are re-linked dynamically as they are translated.

This linking of basic blocks for execution in the real memory of the real host processor speeds the processing of the basic blocks. In the case especially of long decryption loops that span across many basic blocks, there can be a speed boost of a factor of 5 or more to 100 million instructions per second, which compares to the typical maximum of 20 million instructions per second that can be achieved in the prior art. In other words, a

Best Available Copy

process that took say 5 seconds in the prior art, which is probably unacceptably long to most users, can be carried out in 1 second, which is more likely to be acceptable to most users. Practically speaking, a potentially malicious program file can be translated and re-linked and then allowed to decrypt practically at near real-speed. This makes the method viable in real time with minimal interruption to the user even for large and/or polymorphic viruses.

Figure 2 shows schematically translated code blocks for a simple example when N (the threshold) is 40. Figure 3 shows schematically the case where only the basic blocks that get emulated 400 times or more are linked. This provides a much clearer clear view of the decryption loop.

Whilst the translated basic blocks of code are executed on the real host processor in real memory, the virtual machine can be used when needed, for example in the case of a cache miss, CPU exception, etc.

An example of a "cache miss" is as follows. When a basic block is being translated prior to execution on the real host processor, information about the memory access(es) made by the basic block during the emulation is gathered by the virtual machine engine 10 and used to create cached pages of memory addresses, which are used during execution of the translated basic block on the real host processor. It can happen during this execution, for example after a number of iterations of the translated basic block, that an address being accessed is outside the range of these cached pages of memory addresses. In that case, the virtual machine engine 10 can take control of the

Best Available Copy

process, check the type of memory access being attempted by the execution of the translated basic block, and load a new cache page with a new address for the address that is being accessed.

An example of a CPU exception such that the virtual machine engine 10 can take control of the process is as follows. Assume that a decompression loop consists of four basic blocks X, Y, Z and T. Assume that block X jumps to block Y, block Y jumps to block Z, block Z jumps to block T and block T jumps back to block X. Further assume that the re-assembler 38 cannot translate block T. In that case, during execution of the translated basic blocks on the real host processor, the flow would be as follows:

(i) execute X, which is linked to Y and so jumps to Y;
(ii) execute Y, which is linked to Z and so jumps to Z;
(iii) execute Z, which is linked to T and so would normally jump to T. However, T has not been translated. Accordingly, flow returns to the virtual machine;
(iv) in the virtual machine, emulate T, notice that it would jump to X, and then return to (i).

In this way, if there is a CPU exception (on the real host processor), such as flow passing to a basic block that has not been translated, the virtual machine can take control to deal with the exception.

A separate important contributor to improving the speed of operation of the preferred embodiment is the use of cache pages of memory for each basic block of code for use during emulation. The use of cache pages during emulation and the linking method for the translated basic

Best Available Copy

blocks for execution on the real host processor discussed above can be used individually or together.

Consider the following example code:

```
mov ecx, 10; the ECX register is set to 10
_begin: mov eax, [0x401000 + ecx * 4]; the 32 bit EAX register gets the value from the address 0x401000 + ECX * 4
mov [0x402000 + ecx * 4], eax; the value of EAX is written to the address 0x402000 + ECX * 4
loop _begin; decrement ECX and jump to _begin if ECX is not 0
```

For the two memory accesses, the method disclosed in the paper by Stepan discussed above calls a function called a "memory mapper". Since the virtual memory is, of course, virtual, a translation is required between the virtual and the real memory in order to be able to access the real memory where data is to be stored or fetched. Since in this method the memory mapper is invoked for every instruction that accesses memory, its simple presence is a problem for the execution speed of the emulator.

The preferred embodiment of the present method tries to minimise the number of times that the memory mapper is invoked during emulation by using at least two separate pages of cache for every basic block, these being a "read" page and a "write" page. More preferably, there is a third page of cache for every basic block, a "stack" page. The cache pages are preferably created as the blocks are translated. This set up of the cache pages is based on the assumption that a single basic block accesses ONE "read" page, ONE "write" page and, sometimes, ONE stack page.

Best Available Copy

This is true for most decryption engines. Some embodiments, however, may use a set of cache pages for every instruction.

In simple terms, the cache pages provide pairs of correspondences between real and virtual addresses, i.e. (a) a "read" cached virtual address - a "read" cached real address; (b) a "write" cached virtual address - a "write" cached real address; and (c) a "stack" cached virtual address - a "stack" cached real address.

Consider the case where a virus wants to access address 0xBAADF00D, with an instruction like mov eax, [0xBAADF00D]. This instruction reads a 32 bit value from the address 0xBAADF00D into the EAX register. Assume that the virtual address 0xBAADF00D is mapped to the real address 0xDEADBABE.

When emulating, the process is like this:

a) decode the instruction, see that it is necessary to read a 32 bit word from the virtual address 0xBAADF00D;

b) parse the internal structures stored by the virtual machine engine 10; discover that virtual address 0xBAADF00D is mapped to the real address 0xDEADBABE;

c) read a 32 bit word from 0xDEADBABE, and put it into the EAX register.

Now imagine that the mov eax, [0xBAADF00D] instruction is inside a loop such that it is a hot spot and has to be translated.

As mentioned above, the method proposed by Stepan generates code like this:

Best Available Copy

a) call a Memory_Mapper function, for every memory access, "translate" the virtual address 0xBAADF00D to 0xDEADBABE;

b) access the memory from 0xDEADBABE.

This "translation", which implies parsing the internal structures of the virtual machine, takes time.

The caches of the present method are used like this:

a) see if address 0xBAADF00D is in the cache. If not, call a Memory_Mapper, and update the caches.

b) access the memory from 0xDEADBABE.

In the best case (linear decryption, common decompression, etc.), this provides a very good rate of cache hits and this greatly increases the speed of emulation of the basic blocks. In a very worst case, it is necessary to invoke the memory mapper, for example when an out-of-bound page access is requested, but then in that eventuality the present method is the same as the method proposed by Stefan.

For the example code given above, the pseudo-code for the generated code obtained in the present method is like this:

```
mov ecx, 10
_begin:

if (is_address_in_read_cache (0x401000 + ecx * 4) =
FALSE)  // happens rarely {
call Memory Mapper
put_address_in_read_cache (0x401000 + ecx * 4)
```

Best Available Copy


```
}

    mov eax, [read_cache_address + offset]

    if (is_address_in_write_cache (0x402000 + ecx * 4) =
FALSE) // happens rarely {
    call Memory Mapper
    put_address_in_write_cache (0x402000 + ecx * 4)
}

    mov [write_cache_address + offset], eax

    loop _begin
```

One problem when dealing with memory accesses is inter-page-boundary accesses. Since the present method preferably uses an "on-demand" paging scheme, some addresses that are linear in the virtual memory may actually not be linear in the real memory manager. This is just a consequence of memory pages being loaded on demand and unloaded. For example, some pages that were not accessed for a long period of time may be unmapped as a consequence of memory needs. In extreme cases, the performance penalty is too high when it is necessary to generate two memory accesses (read and write) over two consecutive pages. So, preferably consecutive pages are kept linear in memory if needed (e.g. because an application does cross-page memory accesses).

Although the embodiments of the invention described with reference to the drawings comprise computer processes performed in computer apparatus and computer apparatus

Best Available Copy

itself, the invention also extends to computer programs, particularly computer programs on or in a carrier, adapted for putting the invention into practice. The program may be in the form of source code, object code, a code intermediate source and object code such as in partially compiled form, or in any other form suitable for use in the implementation of the processes according to the invention. The carrier be any entity or device capable of carrying the program. For example, the carrier may comprise a storage medium, such as a ROM, for example a CD ROM or a semiconductor ROM, or a magnetic recording medium, for example a floppy disk or hard disk. Further, the carrier may be a transmissible carrier such as an electrical or optical signal which may be conveyed via electrical or optical cable or by radio or other means.

Embodiments of the present invention have been described with particular reference to the examples illustrated. However, it will be appreciated that variations and modifications may be made to the examples described within the scope of the present invention.

Best Available Copy

CLAIMS

1. A method of unpacking/decrypting an executable computer file using a host computer, the method comprising:

partitioning the executable computer file into plural basic blocks of code;

translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer;

linking at least some of the translated basic blocks of code in memory of the host computer; and,

executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

2. A method according to claim 1, comprising, when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory.

3. A method according to claim 1, wherein a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any

Best Available Copy

existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block.

4. A method according to claim 1, wherein the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated.

5. A method according to claim 1, comprising passing flow of the method to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks.

6. A method according to claim 1, comprising emulating the executable file by executing the basic blocks of code, wherein the basic blocks of code that are translated are only those that execute more than a predetermined number of times during the emulation.

7. A method according to claim 6, wherein the value for said predetermined number of times is dynamically variable during the emulation.

8. A method according to claim 6, wherein the emulation is carried out in virtual memory on the host computer.

9. A method according to claim 1, comprising creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating

Best Available Copy

at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block, and comprising emulating the executable file by executing the basic blocks of code, and, during the execution of a basic block, checking at least one of the read page and the write page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

10. A method according to claim 9, comprising creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprising, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

11. A method according to claim 1, wherein the partitioning is carried out dynamically.

12. A method according to claim 1, wherein the executable file is a Portable Executable File.

13. A method of unpacking/decrypting an executable computer file, the method comprising:

partitioning the executable computer file into plural basic blocks of code;

creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache

Best Available Copy

memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and,

emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware;

wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

14. A method according to claim 13, comprising creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprising, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

15. A method according to claim 13, wherein the emulation is carried out in virtual memory on the host computer.

Best Available Copy

16. A method according to claim 13, wherein the partitioning is carried out dynamically.

17. A method according to claim 13, wherein the executable computer file is a Portable Executable file.

18. A computer program for unpacking/decrypting an executable computer file using a host computer, the computer program comprising program instructions for causing a said host computer to carry out a method of:

partitioning the executable computer file into plural basic blocks of code;

translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer;

linking at least some of the translated basic blocks of code in memory of the host computer; and,

executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware.

19. A computer program according to claim 18, wherein the computer program is arranged so that the method comprises, when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory.

Best Available Copy

20. A computer program according to claim 18, wherein the computer program is arranged so that a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block.

21. A computer program according to claim 18, wherein the computer program is arranged so that the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated.

22. A computer program according to claim 18, wherein the computer program is arranged so that flow of the method passes to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks.

23. A computer program according to claim 18, wherein the computer program is arranged so that the method comprises emulating the executable file by executing the basic blocks of code, wherein the basic blocks of code that are translated are only those that execute more than a predetermined number of times during the emulation.

Best Available Copy

24. A computer program according to claim 23, wherein the computer program is arranged so that the value for said predetermined number of times is dynamically variable during the emulation.

25. A computer program according to claim 23, wherein the computer program is arranged such that the emulation is carried out in virtual memory on the host computer.

26. A computer program according to claim 18, wherein the computer program is arranged so that the method comprises creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block, and comprises emulating the executable file by executing the basic blocks of code, and, during the execution of a basic block, checking at least one of the read page and the write page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

27. A computer program according to claim 26, wherein the computer program is arranged so that the method comprises creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprises, during the execution of a basic

Best Available Copy

block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

28. A computer program according to claim 18, wherein the computer program is arranged so that the partitioning is carried out dynamically.

29. A computer program according to claim 18, wherein the computer program is arranged so that the executable computer file is a Portable Executable File.

30. A computer program for unpacking/decrypting an executable computer file, the computer program comprising program instructions for causing a said host computer to carry out a method of:

partitioning the executable computer file into plural basic blocks of code;

creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block; and,

emulating the executable file by executing at least some of the basic blocks of code so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware;

Best Available Copy

wherein during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

31. A computer program according to claim 30, wherein the computer program is arranged so that the method comprises creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprises, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

32. A computer program according to claim 30, wherein the computer program is arranged so that the emulation is carried out in virtual memory on the host computer.

33. A computer program according to claim 30, wherein the computer program is arranged so that the partitioning is carried out dynamically.

34. A computer program according to claim 30, wherein the computer program is arranged so that the executable computer file is a Portable Executable file.

35. A computer programmed to carry out a method according to claim 1.

Best Available Copy

36. A computer programmed to carry out a method according to claim 13.

37. A method of analysing an executable computer file using a host computer, the method comprising:

partitioning the executable computer file into plural basic blocks of code;

creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block, and creating at least a write page of cache memory for at least some of the basic blocks, the write page of cache memory storing a write cached real address corresponding to a write cached virtual memory address for the respective basic block;

emulating the executable file by executing the basic blocks of code, wherein, during the execution of a basic block, at least one of the read page and the write page of cache memory is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block;

translating those basic blocks of code that execute more than a predetermined number of times during the emulation into translated basic blocks of code that can be executed by the host computer;

linking at least some of the translated basic blocks of code in memory of the host computer; and,

executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be

Best Available Copy

analyzed to determine whether the executable computer file is or should be classed as malware.

38. A method according to claim 37, comprising, when a basic block of code is newly translated during the translating step, checking whether an already translated basic block of code has a jump instruction to the newly translated basic block of code and, if so, linking the newly translated basic block of code and the already translated basic block of code in memory.

39. A method according to claim 37, wherein a first translated basic block is linked to a second translated basic block by updating a branch instruction of the first translated basic block to point to the second translated basic block, or by inserting an unconditional branch instruction into the first translated basic block to jump to the second translated basic block, or by removing any existing branch instructions from the first translated basic block and physically moving the second translated basic block to the memory address that follows the first translated basic block.

40. A method according to claim 37, wherein the linking of the translated basic blocks of code is carried out dynamically as the blocks are translated.

41. A method according to claim 37, comprising passing flow of the method to a virtual machine in the case of one or more of (i) a cache miss and (ii) a CPU exception arising during execution of said at least some of the translated basic blocks.

Best Available Copy

42. A method according to claim 37, comprising creating at least a stack page of cache memory for at least some of the basic blocks, the stack page of cache memory storing a stack cached real address corresponding to a stack cached virtual memory address for the respective basic block, and comprising, during the execution of a basic block, checking the stack page of cache memory for a cached real address corresponding to the virtual address that is being accessed for said basic block.

43. A method according to claim 37, wherein the value for said predetermined number of times is dynamically variable during the emulation.

44. A method according to claim 37, wherein the emulation is carried out in virtual memory on the host computer.

45. A method according to claim 37, wherein the partitioning is carried out dynamically.

46. A method according to claim 37, wherein the executable computer file is a Portable Executable file.

47. A computer programmed to carry out a method according to claim 37.

Best Available Copy

ABSTRACT

In one aspect, an executable computer file is partitioned into basic blocks of code. At least some basic blocks are translated into translated basic blocks. At least some translated basic blocks are linked in memory of a computer. At least some translated basic blocks on the computer are executed so as to enable the file to be unpacked or decrypted. In this way, the file can be analyzed to determine whether the file is or should be classed as malware. In another aspect, at least a read page of cache memory is created for at least some basic blocks and at least a write page of cache memory is created for at least some basic blocks. During the execution of a basic block, at least one of the read page and the write page is checked for a cached real address corresponding to the virtual address that is being accessed for said basic block.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ❑ **BLACK BORDERS**
- ❑ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ❑ **FADED TEXT OR DRAWING**
- ❑ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ❑ **SKEWED/SLANTED IMAGES**
- ❑ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ❑ **GRAY SCALE DOCUMENTS**
- ☒ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ❑ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ❑ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

PTO/AIA/80 (07-12)
Approved for use through 11/30/2014. OMB 0651-0035
U.S. Patent and Trademark Office; U.S DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

POWER OF ATTORNEY TO PROSECUTE APPLICATIONS BEFORE THE USPTO

I hereby revoke all previous powers of attorney given in the application identified in the attached statement under 37 CFR 3.73(c).

I hereby appoint:

☒ Practitioners associated with Customer Number: 22442

OR

☐ Practitioner(s) named below (if more than ten patent practitioners are to be named, then a customer number must be used):

Name	Registration Number

Name	Registration Number

As attorney(s) or agent(s) to represent the undersigned before the United States Patent and Trademark Office (USPTO) in connection with any and all patent applications assigned only to the undersigned according to the USPTO assignment records or assignments documents attached to this form in accordance with 37 CFR 3.73(c).

Please change the correspondence address for the application identified in the attached statement under 37 CFR 3.73(c) to:

☒ The address associated with Customer Number: 22442

OR

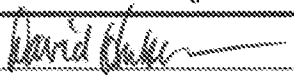
☐	Firm or Individual Name		
	Address		
	City	State	Zip
	Country		
	Telephone	Email	

Assignee Name and Address: Prevx Limited
3 New Park Place, Pride Park
Derby, United Kingdom DE24 8DZ

A copy of this form, together with a statement under 37 CFR 3.73(c) (Form PTO/AIA/96 or equivalent) is required to be filed in each application in which this form is used. The statement under 37 CFR 3.73(c) may be completed by one of the practitioners appointed in this form, and must identify the application in which this Power of Attorney is to be filed.

SIGNATURE of Assignee of Record

The individual whose signature and title is supplied below is authorized to act on behalf of the assignee

Signature		Date	October 26, 2012
Name	David Huberman, Esq.	Telephone	720-842-3627
Title	Director		

This collection of information is required by 37 CFR 1.31, 1.32 and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/AIA/96 (08-12)
Approved for use through 01/31/2013. OMB 0651-0031
U.S. Patent and Trademark Office;U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of1995, no persons arerequired to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

Applicant/Patent Owner: Prevx Limited

Application No./Patent No.: 11/477,807 Filed/Issue Date: 2006-06-30

Titled: METHODS AND APPARATUS FOR DEALING WITH MALWARE

Prevx Limited, a Corporation

(Name of Assignee) (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.)

states that, for the patent application/patent identified above, it is (choose **one** of options 1, 2, 3 or 4 below):

1. ☑ The assignee of the entire right, title, and interest.

2. ☐ An assignee of less than the entire right, title, and interest (check applicable box):

   ☐ The extent (by percentage) of its ownership interest is _____%. Additional Statement(s) by the owners holding the balance of the interest must be submitted to account for 100% of the ownership interest.

   ☐ There are unspecified percentages of ownership. The other parties, including inventors, who together own the entire right, title and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

3. ☐ The assignee of an undivided interest in the entirety (a complete assignment from one of the joint inventors was made). The other parties, including inventors, who together own the entire right, title, and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

4. ☐ The recipient, via a court proceeding or the like (*e.g.*, bankruptcy, probate), of an undivided interest in the entirety (a complete transfer of ownership interest was made). The certified document(s) showing the transfer is attached.

The interest identified in option 1, 2 or 3 above (not option 4) is evidenced by either (choose **one** of options A or B below):

A. ☑ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel 018348, Frame 0840, or for which a copy thereof is attached.

B. ☐ A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

   1. From: _____ To: _____

      The document was recorded in the United States Patent and Trademark Office at

      Reel _____, Frame _____, or for which a copy thereof is attached.

   2. From: _____ To: _____

      The document was recorded in the United States Patent and Trademark Office at

      Reel _____, Frame _____, or for which a copy thereof is attached.

This collection of information is required by37 CFR3.73(b). The information is required toobtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentialityis governed by35 U.S.C. 122and 37 CFR1.11 and1.14. Thiscollection is estimated to take 12 minutes to complete, including gathering, preparing, and submittingthe completed application form to the USPTO.Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent tothe Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS.**SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

STATEMENT UNDER 37 CFR 3.73(c)

3. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

4. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

5. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

6. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at

Reel _____, Frame _____, or for which a copy thereof is attached.

☐ Additional documents in the chain of title are listed on a supplemental sheet(s).

☑ As required by 37 CFR 3.73(c)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[NOTE: A separate copy (i.e., a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

/Bradley M. Knepper/	2012-10-30
Signature	Date
Bradley M. Knepper	44,189
Printed or Typed Name	Title or Registration Number

[Page 2 of 2]

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that yoube given certain informationin connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, pleasebe advised that: (1) the general authority forthe collection of thisinformation is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and(3) the principal purpose forwhich the information isused by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent applicationor patent. If you do not furnish the requested information,the U.S. Patent and Trademark Office may not be able to process and/or examineyour submission,which may result in termination of proceedings or abandonment of the applicationor expiration of the patent.

The informationprovided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the informationin order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an InternationalApplication filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, arecord may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from thissystem of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Acknowledgement Receipt

EFS ID:	14107697
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	METHODS AND APPARATUS FOR DEALING WITH MALWARE
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	47604
Filer:	Bradley M. Knepper/Leslie Roberts
Filer Authorized By:	Bradley M. Knepper
Attorney Docket Number:	339000-000003
Receipt Date:	30-OCT-2012
Filing Date:	30-JUN-2006
Time Stamp:	18:09:14
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	POA_Executed_Prevx_AIA.pdf	912728 db9c376ddb33b6c00063bcf3d63fb2b6debbbc85	no	1

Warnings:	
Information:	

2	Assignee showing of ownership per 37 CFR 3.73.	STATEMENT.pdf	120051 f0bd0957c20603e852900731f1e41d190ad9a34b	no	3

Warnings:	
Information:	
Total Files Size (in bytes):	1032779

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/477,807	06/30/2006	Melvyn Morris	339000-000003

CONFIRMATION NO. 4785

POA ACCEPTANCE LETTER

22442
Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

OC000000057472399

Date Mailed: 11/08/2012

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 10/30/2012.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

/hchristian/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/477,807	06/30/2006	Melvyn Morris	339000-000003

CONFIRMATION NO. 4785

47604
DLA PIPER LLP US
P. O. BOX 2758
RESTON, VA 20195

POWER OF ATTORNEY NOTICE

OC000000057472396

Date Mailed: 11/08/2012

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 10/30/2012.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

/hchristian/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

22442 7590 12/13/2012

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

EXAMINER
ZECHER, CORDELIA P K

ART UNIT	PAPER NUMBER
2432	

DATE MAILED: 12/13/2012

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	6890-93	4785

TITLE OF INVENTION: METHODS AND APPARATUS FOR DEALING WITH MALWARE

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1770	$300	$0	$2070	03/13/2013

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22442 7590 12/13/2012

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	6890-93	4785

TITLE OF INVENTION: METHODS AND APPARATUS FOR DEALING WITH MALWARE

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1770	$300	$0	$2070	03/13/2013

EXAMINER	ART UNIT	CLASS-SUBCLASS
ZECHER, CORDELIA P K	2432	726-024000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____

2 _____

3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:

❑ Issue Fee

❑ Publication Fee (No small entity discount permitted)

❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

❑ A check is enclosed.

❑ Payment by credit card. Form PTO-2038 is attached.

❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

❑ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

❑ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	6890-93	4785

22442 7590 12/13/2012

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

EXAMINER	
ZECHER, CORDELIA P K	
ART UNIT	PAPER NUMBER
2432	

DATE MAILED: 12/13/2012

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 765 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 765 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Notice of Allowability	Application No. 11/477,807	Applicant(s) MORRIS ET AL.	
	Examiner CORDELIA ZECHER	Art Unit 2432	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *IDS Filed 6/21/12*.

2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

3. ☒ The allowed claim(s) is/are *1-11,13-23,25-30,49 and 51*. As a result of the allowed claim(s), you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov .

4. ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☒ All b) ☐ Some* c) ☐ None of the:
      1. ☒ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☒ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
3. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
4. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
5. ☒ Examiner's Amendment/Comment
6. ☐ Examiner's Statement of Reasons for Allowance
7. ☐ Other _____.

/CORDELIA ZECHER/
Primary Examiner, Art Unit 2432

Information Disclosure Statement

The information disclosure statement (IDS) submitted on June 21, 2012 was filed after the mailing date of the Notice of Allowance on March 21, 2012. The submission is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

EXAMINER'S COMMENTS

This communication warrants No Examiner's Reason for Allowance, applicant's reply make evident the reasons for allowance, satisfying the "record as a whole" proviso of the rule 37 CFR 1.104(e). Specifically, applicant's arguments filed on December 23, 2011 are persuasive, as such the reasons for allowance are in all probability evident from the record and no statement is deemed necessary (see MPEP 1302.14).

Any comments Applicants considers necessary must be submitted no later than the payment of the Issue Fee and to avoid processing delays, should preferable accompany the Issue Fees. Such submission should be clearly labeled "Comments on Statement of Reasons for Allowance". In event of any post-allowance papers (e.g. IDS, 312 amendment, petition, etc.), Applicant is exhorted to mail papers to the Production Control branch in Publications or faxed to post-allowance papers correspondence branch at (703) 308-5864 to expedite issuing process or call PUB's Customer Service if any questions at (703) 305-8497.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to CORDELIA ZECHER whose telephone number is (571)272-7771. The examiner can normally be reached on Monday - Thursday 8:00 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Gilberto Barron can be reached on 571-272-3799. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/CORDELIA ZECHER/
Primary Examiner, Art Unit 2432

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	Cordelia P.K. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -2009/0271867	10/1/2009	Zhang	
		US -2008/0320595	12/1/2008	van der Made	
		US -2008/0209562	8/1/2008	Szor	
		US -7,093,239	8/1/2006	van der Made	
		US -2004/0255165	12/1/2004	Szor	
		US -2003/0135791	7/1/2003	Natvig	
		US -2004/0068618	4/1/2004	Hooker	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Image File Wrapper of related U.S. Application No. 11/694,261 electronically caputured from PAIR on June 21, 2012.	

Examiner Signature	/Cordelia Zecher/	Date Considered	12/11/2012

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

EAST\48824215.1

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	Cordelia Zecher	2432

SEARCHED			
Class	**Subclass**	**Date**	**Examiner**
726	24	3/12/12	cz

SEARCH NOTES		
Search Notes	**Date**	**Examiner**
Searched EAST (see attached)	12/11/12	cz
Inventor Search	4/28/10	cz

INTERFERENCE SEARCH			
Class	**Subclass**	**Date**	**Examiner**
726	24 (see attached)	12/11/12	cz

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 4785

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
11/477,807	06/30/2006	726	2432	6890-93

APPLICANTS

Melvyn Morris, Turnditch, UNITED KINGDOM;
Paul Stubbs, Wyboston, UNITED KINGDOM;
Markus Hartwig, Milton Keynes, UNITED KINGDOM;
Darren Harter, Hucclecote, UNITED KINGDOM;

**** CONTINUING DATA ***************************

**** FOREIGN APPLICATIONS ***************************

UNITED KINGDOM 0513375.6 06/30/2005

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

07/26/2006

Foreign Priority claimed	Met after Allowance	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
☑ Yes ☐ No	☐	UNITED KINGDOM	3	47	8
35 USC 119(a-d) conditions met ☑ Yes ☐ No					
Verified and Acknowledged /CORDELIA P K ZECHER/ Examiner's Signature	Initials				

ADDRESS

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202
UNITED STATES

TITLE

METHODS AND APPARATUS FOR DEALING WITH MALWARE

FILING FEE RECEIVED		
3584	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

BIB (Rev. 05/07).

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	CORDELIA ZECHER	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010	02/09/2011	07/21/2011	03/12/2012	12/11/2012			
1	1	✓	✓	✓	✓	=	=			
2	2	✓	✓	✓	✓	=	=			
3	3	✓	✓	✓	✓	=	=			
4	4	✓	✓	✓	✓	=	=			
5	5	✓	✓	✓	✓	=	=			
6	6	✓	✓	✓	✓	=	=			
7	7	✓	✓	✓	✓	=	=			
8	8	✓	✓	✓	✓	=	=			
9	9	✓	✓	✓	✓	=	=			
10	10	✓	✓	✓	✓	=	=			
11	11	✓	✓	✓	✓	=	=			
	12	✓	-	-	-	-	-			
13	13	✓	✓	✓	✓	=	=			
14	14	✓	✓	✓	✓	=	=			
15	15	✓	✓	✓	✓	=	=			
16	16	✓	✓	✓	✓	=	=			
17	17	✓	✓	✓	✓	=	=			
18	18	✓	✓	✓	✓	=	=			
19	19	✓	✓	✓	✓	=	=			
20	20	✓	✓	✓	✓	=	=			
21	21	✓	✓	✓	✓	=	=			
22	22	✓	✓	✓	✓	=	=			
23	23	✓	✓	✓	✓	=	=			
	24	✓	-	-	-	-	-			
25	25	✓	✓	✓	✓	=	=			
26	26	✓	✓	✓	✓	=	=			
27	27	✓	✓	✓	✓	=	=			
28	28	✓	✓	✓	✓	=	=			
29	29	✓	✓	✓	✓	=	=			
30	30	✓	✓	✓	✓	=	=			
	31	N		N	N	-	-			
	32	N		N	N	-	-			
	33	N		N	N	-	-			
	34	N		N	N	-	-			
	35	N		N	N	-	-			
	36	N		N	N	-	-			

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	11477807	MORRIS ET AL.
	Examiner	**Art Unit**
	CORDELIA ZECHER	2432

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	04/28/2010	09/23/2010	02/09/2011	07/21/2011	03/12/2012	12/11/2012			
	37	N		N	N	-	-			
	38	N		N	N	-	-			
	39	N		N	N	-	-			
	40	N		N	N	-	-			
	41	N		N	N	-	-			
	42	N		N	N	-	-			
	43	N		N	N	-	-			
	44	N		N	N	-	-			
	45	N		N	N	-	-			
	46	N		N	N	-	-			
	47	N		N	N	-	-			
	48			✓	-	-	-			
12	49			✓	✓	=	=			
	50			✓	-	-	-			
24	51			✓	✓	=	=			

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	7	("20030135791" | "20040068618" | "20040255165" | "20080209562" | "20080320595" | "20090271867" | "7093239").PN.	US-PGPUB; USPAT	OR	ON	2012/12/11 08:38
S1	1	"20070016953"	US-PGPUB; USPAT	OR	ON	2010/01/07 08:58
S2	1	("20060026684").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 08:32
S3	1	S2 and malware	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
S4	1164	726/24.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:33
S5	2821	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2010/04/28 08:34
S6	6	(("6772346") or ("6944772") or ("20040006704") or ("20040073810") or ("20050021994") or ("20040153644")).PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 09:18
S7	1	("20030177394").PN.	US-PGPUB; USPAT	OR	OFF	2010/04/28 10:48
S8	0	central near database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
S9	0	database with each with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:51
S10	115859	database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
S11	2632	central near database with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:52
S12	30	central near database with network with separate	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
S13	17	central near database with distribute with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:53
S14	199	central near database with distribut$3 with network	US-PGPUB; USPAT	OR	ON	2010/04/28 12:54
S15	0	central near database with distribute with	US-	OR	ON	2010/04/28

		network with known	PGPUB; USPAT			12:54
S16	4	central near database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55
S17	240	database with distribut$3 with network with known	US-PGPUB; USPAT	OR	ON	2010/04/28 12:55
S18	24	database with distribut$3 with network with known and (virus or malware)	US-PGPUB; USPAT	OR	ON	2010/04/28 12:56
S19	4	("20030023857" | "20050027686" | "6338141" | "7013483").PN.	US-PGPUB; USPAT	OR	ON	2011/02/09 08:21
S20	3433	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S21	282	S20 and mask$3	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S22	35	S20 and mask$3 and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:35
S23	328	S20 and control$3 with environment	US-PGPUB; USPAT	OR	ON	2011/02/09 08:43
S24	65	S20 and control$3 with environment and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:43
S25	18	S20 and defin$3 with boundary and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:45
S26	5	S20 and access$3 with rights with object and malware	US-PGPUB; USPAT	OR	ON	2011/02/09 08:46
S27	2	"20020099952"	US-PGPUB; USPAT	OR	ON	2011/02/09 09:11
S28	24	(updat$3 or modif$4) with token and S20	US-PGPUB; USPAT	OR	ON	2011/02/09 10:54
S29	37	("20010049793" | "20030225698" | "20040068721" | "20050203819" | "20060031686" | "4713753" | "4805222" | "4926476" | "5367624" | "5481700" | "5819091" | "5915025" | "5958051" | "5978484" | "5983349" | "5991519" | "6295605" | "6378069" | "6405202" | "6775779" | "6931540" | "6981281" | "7028180" | "7051366" | "7076557").PN. OR ("7350204").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:23
S30	32	S29 and (modif$4 or updat$3)	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:23
S31	6232	automatic$4 with (updat$3 or chang$3 or modif$4) with access	US-PGPUB; USPAT;	OR	ON	2011/02/09 11:35

			USOCR			
S32	29	S20 and S31	US-PGPUB; USPAT; USOCR	OR	ON	2011/02/09 11:35
S33	18	"7350204"	US-PGPUB; USPAT	OR	ON	2011/07/21 09:19
S34	1	"7350204".pn.	US-PGPUB; USPAT	OR	ON	2011/07/21 09:19
S35	42	("20010049793" | "20030225698" | "20040068721" | "20050203819" | "20060031686" | "4713753" | "4805222" | "4926476" | "5367624" | "5481700" | "5819091" | "5915025" | "5958051" | "5978484" | "5983349" | "5991519" | "6295605" | "6378069" | "6405202" | "6775779" | "6931540" | "6981281" | "7028180" | "7051366" | "7076557").PN. OR ("7350204").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:21
S36	106	("2002/0099952").URPN.	USPAT	OR	ON	2011/07/21 09:46
S37	16	("20020026578" | "20020099952" | "20020124170" | "20030033535" | "20040064708" | "20040078604" | "20040117615" | "20050138388" | "5941947" | "6151590" | "6397261" | "6571335" | "6810525" | "7010600" | "7194664").PN. OR ("7509497").URPN.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:49
S38	1	(S35 or S36 or S37) and behaviour	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:50
S39	1	"7698713".pn.	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:51
S40	2	"20030131152"	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:52
S41	57	(S35 or S36 or S37) and behavior	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 09:52
S42	2	"20050210035"	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 10:01
S43	2	S42 and behav$5	US-PGPUB; USPAT; USOCR	OR	ON	2011/07/21 10:04
S44	20	("20040153644" | "6338141" | "20030084323" | "20050027686" | "6772346" | "20050210035" | "6944772" | "20050021994" | "20030023857" | "20040083408" | "20050022014" |	US-PGPUB; USPAT	OR	ON	2012/02/17 13:30

		"20040006704" | "20010052014" | "20030101381" | "7013483" | "20020099952" | "20040039921" | "20050086500" | "20040073810" | "20050021994" | "20030177394").PN.				
S45	5	("20040083408" | "20050022014" | "20030101381" | "20040039921" | "20050086500").PN.	US-PGPUB; USPAT	OR	ON	2012/02/17 13:34
S46	3172	generat$3 with profile with automatic$4	US-PGPUB; USPAT	OR	ON	2012/02/17 14:56
S47	4255	726/23-25.ccls.	US-PGPUB; USPAT	OR	ON	2012/02/17 14:56
S48	25	generat$3 with profile with automatic$4 and S47	US-PGPUB; USPAT	OR	ON	2012/02/17 14:56

EAST Search History (Interference)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L2	1319	726/24.ccls.	USPAT	OR	ON	2012/12/11 08:43
L3	0	(class$6 and object and malware and automatic$4 and generat$3 and monitor$3).clm.	USPAT	OR	ON	2012/12/11 08:43
L4	0	L3 and L2	USPAT	OR	ON	2012/12/11 08:43
S49	1022	726/24.ccls.	USPAT	OR	ON	2012/02/17 14:59
S50	0	(behavior and malware and classify$3 and automatic$4 and generat$3 and monitor$3 and allow$3).clm.	USPAT	OR	ON	2012/02/17 15:06
S51	1036	726/24.ccls.	USPAT	OR	ON	2012/03/12 11:36
S52	0	(class$6 and object and malware and automatic$4 and generat$3 and mask$3 and monitor$3).clm.	USPAT	OR	ON	2012/03/12 11:37
S53	0	(class$6 and object and malware and automatic$4 and generat$3 and monitor$3).clm.	USPAT	OR	ON	2012/03/12 11:37
S54	0	S53 and S51	USPAT	OR	ON	2012/03/12 11:37

12/ 11/ 2012 8:43:49 AM
C:\ Users\ czecher\ Documents\ EAST\ Workspaces\ 11477807.wsp

Issue Classification

Application/Control No.	Applicant(s)/Patent Under Reexamination
11477807	MORRIS ET AL.
Examiner	**Art Unit**
CORDELIA ZECHER	2432

ORIGINAL		INTERNATIONAL CLASSIFICATION	
CLASS	SUBCLASS	CLAIMED	NON-CLAIMED
726	24	G 0 6 F 21 / 00 (2006.01)	

CROSS REFERENCE(S)

CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	18	18												
2	2	19	19												
3	3	20	20												
4	4	21	21												
5	5	22	22												
6	6	23	23												
7	7	25	25												
8	8	26	26												
9	9	27	27												
10	10	28	28												
11	11	29	29												
13	13	30	30												
14	14	12	49												
15	15	24	51												
16	16														
17	17														

		Total Claims Allowed:	
(Assistant Examiner)	(Date)	30	
/CORDELIA ZECHER/ Primary Examiner.Art Unit 2432	12/11/2012	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	3

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	1	of	1

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	Cordelia P.K. Zecher
Attorney Docket Number	6890-93

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]; Number[4]; Kind Code[5] *(if known)*	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON-PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.
	1	English Translation of Official Action for China Patent Application No. 201110036121.8, dated Sep. 24, 2012, 8 pages
	2	Official Action with English translation for China Patent Application No. 201110036121.8, dated Oct. 10, 2011, 14 pages

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

THE PATENT OFFICE OF THE PEOPLE'S REPUBLIC OF CHINA

Address: 6 Xi Tu Cheng Lu, Haidian, Beijing 100088

Applicant:	**PREVX LIMITED**	
Attorney:	**Maohua Wang**	Date of Notification:
Application No.:	**201110036121.8**	Date: 24 Month: 09 Year: 2012
Title of the Invention:	**METHODS AND APPARATUS FOR DEALING WITH MALWARE**	

Rejection Decision

1.The above-identified application is rejected under Article 38 of the Patent Law and Rule 53 of the Implementing Regulations thereof. The rejection is made based on the ground that:

☐ The application does not comply with Rule 2 paragraph 1 of the Implementing Regulations of the Patent Law.

☒ The subject matter contained in the application is not patentable under Article 22 of the Patent Law.

☐ The subject matter contained in the application is not patentable under Article 25 of the Patent Law.

☐ The application does not comply with Article 26 paragraph 3 or Article 26 paragraph 4 of the Patent Law.

☐ The application does not comply with the provisions of Rule 12 paragraph 1 of the Implementing Regulations of the Patent Law.

☐ The applicant is not qualified for obtaining the patent right under Article 9 of the Patent Law.

☐ The amendment of the application does not comply with Article 33 of the Patent Law.

2.☒ Details of the reasons for the rejection are recited in the text portion of the Rejection Decision (Total 4 pages).

3.The Rejection Decision is made based on

☐ Chinese translation of the international application text as originally filed.

☒ the documents as specified below.

Paragraphs 1-149 of the specification, the drawings, the abstract and the abridged figure filed on the filling date of the divisional application; and

claims 1-12 filed on April 25, 2012.

3. The applicant may, within 3 months counting from the date of receipt of this Notification, request the Patent Reexamination Board to make a reexamination under Article 41 of the Patent Law and Rule 60 of the Implementation Rule of the Patent Law, if the Applicant is not satisfied with the decision.

Examination Dept. _____ Examiner: _____ Seal of the Examination Department

Text of the Rejection Decision

Application No. 201110036121.8

This Rejection Decision relates to a divisional application for a patent for invention entitled "Methods and apparatus for dealing with malware" with an application number of 201110036121.8, which was filed with the State Intellectual Property Office by Applicant, PREVX LTD., with on February 9, 2011. The parent application of the present divisional application is No. 200680023707.8, which was filed on June 30, 2006 and claimed the priority date of June 30, 2005.

I. Brief of the Case

On February 9, 2011, Applicant filed with the State Intellectual Property Office the application document including: Claims 1-9, paragraphs 1-149 of the specification, the accompanying drawings, the abstract and the abridged figure.

As per Applicant's request for substantive examination made on February 9, 2011, the Examiner examined the present application as to its substance and issued on October 10, 2011 the first Office Action, wherein reference was made to D1 (US2004/0039921A1, published on February 26, 2004). The Examiner pointed out that Claims 1-9 lack inventiveness under Article 22.3 of the Chinese Patent Law.

In view of the first Office Action, Applicant submitted a response on April 25, 2012. Briefly, Applicant added dependent Claims 4, 8 and 12 according to the disclosure of the specification, adjusted the numbers and dependencies of original Claims 1-9 (made no amendment to the substance of original Claims 1-9), and submitted that Claims 1-12 have inventiveness.

Based on the above work, the Examiner opines that facts of the present application are now clear, and hence issues this Rejection Decision against paragraphs 1-149 of the specification, the accompanying drawings, the abstract and the abridged figure as filed on the divisional application filing date, as well as Claims 1-12 as filed on April 25, 2012.

II. Grounds for Rejection

(I) Claims 1-3, 5-7 and 9-11 lack inventiveness under Article 22.3 of the Chinese Patent

Law.

Claim 1 recites a method of determining the protection that a remote computer has from malware. D1 (US2004/0039921A) discloses a method for detecting rogue software and specifically reveals the following technical features (see paras.[0033]-[0036] of the description and Figs.1-2): a client 12 (corresponding to "a remote computer") is connected to a server 14 and a database of acceptable file fingerprints via a network (the server and the database are together equivalent to the base computer receiving details of a program product); first, the system calculates and retrieves file fingerprints of application software it already has , at the client 12 (corresponding to "the program product operating on the remote computer, receiving similar information such as details of the program product from remote computers connected to the base computer"); upon installation of software, new files and their fingerprints and file locations are added to the database; when application software is run on the client, the calculated Hash result and collected client information are forwarded to the server 14 for verification; on the server 14, the received Hash value of the file of the client is compared with certain desired client information stored in the database 16, such as a version of an operating system and parameters of a software patch; if they do not match each other, it will be judged that an application program having rogue software is running on the client 12 (corresponding to "identifying any malware processes that were not identified by said other remote computers having that particular combination of program products").

Claim 1 differs from D1 in that 1) Claim 1 recites what is run and identified on the computer is security products whereas D1 defines application programs; 2) Claim 1 recites that the security products operate at a point in time on the computer, whereas D1 does not disclose this technical feature. However, in the art, it is common technical means in the art to set a client to run antivirus/Trojan virus searching and killing software at a certain point in time, to judge whether the client has unidentified viruses or security vulnerabilities via a network virus/Trojan virus library, and notify the client of taking defensive measures such as updating. Therefore, it is obvious for those skilled in the art to obtain the claimed technical solution of Claim 1 on the basis of D1 in conjunction with the common technical means in the art. Accordingly, Claim 1 neither has any prominent substantive features nor represents any notable progress and therefore lacks an inventive step under Article 22.3 of the Chinese Patent Law.

2. Claim 2 further defines Claim 1. It is common technical means in the art to set a client to judge whether the client has unidentified viruses or security vulnerabilities in a way that the

client is connected to a network virus/Trojan virus library, and notify the client of such information. Therefore, when Claim 1 lacks an inventive step, Claim 2 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

3. Claim 3 further defines Claim 1 or 2. In the art, digital signature is a conventional security technology for ensuring security of information, so it is common technical means in the art to use signature files to identify malware. Name and versions are common attributes of application programs, so use thereof as details of security products is common technical means in the art. Therefore, when Claim 1 or 2 lacks an inventive step, Claim 3 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

4. Claim 5 recites an apparatus for determining the protection that a remote computer has from malware, the apparatus. D1 discloses a system for detecting rogue software and specifically reveals the following technical features (see paras.[0033]-[0036] of the description and Figs.1-2): a client 12 (corresponding to "a remote computer") is connected to a server 14 and a database of acceptable file fingerprints via a network (the server and the database are together equivalent to the base computer constructed and arranged to receive computer details of all or selected security products); first, the system calculates and retrieves file fingerprints of application software it already has , at the client 12 (corresponding to "the program product operating on the remote computer, the base computer being constructed and arranged to receive similar information such as details of program products from other remote computers connected to the base computer"); upon installation of software, new files and their fingerprints and file locations are added to the database; when application software is run on the client, the calculated Hash result and collected client information are forwarded to the server 14 for verification; on the server 14, the received Hash value of the file of the client is compared with certain desired client information stored in the database 16, such as a version of an operating system and parameters of a software patch; if they do not match each other, it will be judged that an application program having rogue software is running on the client 12 (corresponding to "the base computer being constructed and arranged to identify any malware processes that were not identified by said other remote computers having that particular combination of program products").

Claim 5 differs from D1 in that 1) Claim 5 recites what is run and identified on the computer is security products whereas D1 defines application programs; 2) Claim 5 recites that the security products operate at a point in time on the computer, whereas D1 does not disclose

this technical feature. However, in the art, it is common technical means in the art to set a client to run antivirus/Trojan virus searching and killing software at a certain point in time, to judge whether the client has unidentified viruses or security vulnerabilities via a network virus/Trojan virus library, and notify the client of taking defensive measures such as updating. Therefore, it is obvious for those skilled in the art to obtain the claimed technical solution of Claim 5 on the basis of D1 in conjunction with the common technical means in the art. Accordingly, Claim 5 neither has any prominent substantive features nor represents any notable progress and therefore lacks an inventive step under Article 22.3 of the Chinese Patent Law.

5. Claim 6 further defines Claim 5. It is common technical means in the art to set a client to judge whether the client has unidentified viruses or security vulnerabilities in a way that the client is connected to a network virus/Trojan virus library, and notify the client of such information. Therefore, when Claim 5 lacks an inventive step, Claim 6 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

6. Claim 7 further defines Claim 5 or 6. In the art, digital signature is a conventional security technology for ensuring security of information, so it is common technical means in the art to use signature files to identify malware. Name and versions are common attributes of application programs, so use thereof as details of security products is common technical means in the art. Therefore, when Claim 5 or 6 lacks an inventive step, Claim 7 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

7. Claims 9-11 are product claims having a function module architecture exactly corresponding to steps of the method recited in Claims 1-3. For the same reasons, Claims 9-11 lack an inventive step under Article 22.3 of the Chinese Patent Law.

(II) Regarding Applicant's observations

1. Applicant has submitted: Claim 1 does not necessarily include a step in which it is determined whether or not the remote computer is infected by a malware program, rather, the present invention determines if the remote computer is vulnerable to being infected by the malware program based on the security products it has; however, there is no mention in D1 of determining which security products a remote computer is running let alone determining what security "gaps" the security products may have.

The Examiner opines: first of all, Claim 1 recites "identifying any malware processes"

rather than determining if the remote computer is vulnerable to being infected by the malware program, that is, nowhere in Claim 1 does it define a possibility of infection in place of real infection; as seen from Applicant's observations, although Claim 1 does not necessarily include a step of determining whether or not the remote computer is infected, this claim may include such a step provided that malware can be identified through authenticating a calculation result and client information from the client by the server as disclosed in D1. In addition, as seen from the comments on Claim 1, D1 discloses "when application software is run on the client, the calculated Hash result and collected client information are forwarded to the server 14 for verification"; such a feature as delivering the client's running result to the server for verification corresponds to running some security products on a client, so D1 discloses the feature of a remote computer running security products. Moreover, for what security "gaps" the security products may have, the Examiner regards it as common technical means in the art to set a client to run antivirus/Trojan virus searching and killing software at a certain point in time, to judge whether the client has unidentified viruses or security vulnerabilities. This has been notified to Applicant during the first Office Action.

2. Applicant has submitted: Claim requires the steps of "receiving at a base computer details of all or selected security products operating at a point in time on a remote computer" and "receiving similar information from other remote computers connected to the base computer." D1 does not teach or suggest receiving details of security products at the base computer at all. On the contrary, D1 teaches that the security product (i.e. the system's client product 12) transmits details of other programs to the base computer (server 14).

The Examiner opines: as seen from the comments on Claim 1, D1 discloses "client 12 (corresponding to "a remote computer") is connected to a server 14 and a database of acceptable file fingerprints via a network (the server and the database are together equivalent to the base computer receiving details of a program product); first, the system calculates and retrieves file fingerprints of application software it already has , at the client 12 (corresponding to "the program product operating on the remote computer, receiving similar information such as details of the program product from remote computers connected to the base computer"); upon installation of software, new files and their fingerprints and file locations are added to the database."

3. Applicant has submitted: Claim 1 requires "identifying any malware processes that were not identified by said other remote computers having that particular combination of security

products." However, D1 does not consider at all what malware processes have been undetected on remote computers with a particular set of security products.

The Examiner opines: D1 discloses "when application software is run on the client, the calculated Hash result and collected client information are forwarded to the server 14 for verification; on the server 14, the received Hash value of the file of the client is compared with certain desired client information stored in the database 16, such as a version of an operating system and parameters of a software patch; if they do not match each other, it will be judged that an application program having rogue software is running on the client 12." Apparently, D1 teaches identifying all malware processes on remote computers having that particular combination of program products. Since Claim 1 merely defines detecting any malware processes that were not identified, D1 discloses the above feature.

III. Decision

To sum up, the present divisional application meets the requirements on inventiveness as set forth in Article 22.3 of the Chinese Patent Law, which falls within the circumstances as set forth in Rule 53(2) of the Implementing Regulations thereof. Therefore, the present application is rejected Article 38 of the Chinese Patent Law.

Pursuant to Article 41.1 of the Chinese Patent Law, if Applicant is dissatisfied with the present Rejection Decision, he may file a reexamination request with the patent reexamination board within three months after the date of receipt of the present Rejection Decision.

IV. Other Explanations

1. Claim 4 further limits any of Claims 1-3. D1 discloses: upon installation of software, new files and their fingerprints and file locations are added to the database; when application software is run on the client, the calculated Hash result (corresponding to receiving from the base computer process details of respective clients, namely other remote computers) and collected client information are forwarded to the server 14 for verification; on the server 14, the received Hash value of the file of the client is compared with certain desired client information stored in the database 16; if they do not match each other, it will be judged that an application program having rogue software is running on the client 12 (corresponding to "the step of identifying comprises searching said details of processes for remote computers having said particular combination of security products to find any processes that are known to be malware"). It can

be seen that the additional technical features of Claim 4 are disclosed by D1. Therefore, when the base claim lacks an inventive step, Claim 4 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

2. Claims 8 and 12 each define the same additional technical features as Claim 4. Based on the same reasoning, Claims 8 and 12 also lack an inventive step under Article 22.3 of the Chinese Patent Law.

THE STATE INTELLECTUAL PROPERTY OFFICE OF THE PEOPLE'S REPUBLIC OF CHINA

Address: 6 Xi Tu Cheng Lu, Haidian, Beijing Post Code: 100088

Applicant:	**PREVX LIMITED**	
Attorney:	**Maohua Wang**	Date of Notification:
Application No.:	**201110036121.8**	Date: 10 Month: 10 Year: 2011
Title of the Invention:	**METHODS AND APPARATUS FOR DEALING WITH MALWARE**	

Notification of the First Office Action

1. ☒ The applicant requested examination as to substance on and examination has been carried out on the above-identified patent application for invention under Article 35(1) of the Patent Law of the People's Republic of China(hereinafter referred to as "the Patent Law").
   ☐ The Chinese Patent Office has decided to examine the application on its own initiative under Article 35(2) of the Patent Law.

2. ☒ The applicant claimed priority/priorities based on the application(s):
   filed in GB on June 30, 2005, filed in _____ on _____,
   filed in _____ on _____, filed in _____ on _____,
   filed in _____ on _____, filed in _____ on _____,
   ☒ The applicant has provided the priority documents certified by the Patent Office where the priority application(s) was/were filed.
   ☐ The applicant has not provided the priority documents certified by the Patent Office where the priority application(s) was/were filed and therefore the priority claim(s) is/are deemed not to have been made under Article 30 of the Patent Law.
   ☐ The application is a PCT continuation.

3. ☐ The applicant submitted amendments to the application on _____ and on _____, wherein the amended _____ submitted on _____ and the amended _____ submitted on _____ are not acceptable, because said amendments do not comply with ☐Article 33 of the Patent Law.
   ☐Rule 51 of the Implementing Regulations of the Patent Law.
   The specific reasons why the amendments are not allowable are set forth in the text portion of this Notification.

4. ☐ Examination as to substance was directed to the initial application documents as filed.
   ☒ Examination as to substance was directed to the divisional application documents as filed.

5. ☐ This Notification is issued without search reports.
   ☒ This Notification is issued with consideration of the search results.
   ☒ Below is/are the reference document(s) cited in this Office Action(the reference number(s) will be used throughout the examination procedure):

No.	Number(s) or Title(s) of Reference(s)	Date of Publication (or the filing date of conflicting application)
1	US 2004/0039921A1	Date: 26 Month: 02 Year: 2004

6. Conclusions of the Action:
   ☐ On the Specification:
   ☐ The subject matter contained in the application is not patentable under Article 5 of the Patent Law.
   ☐ The description does not comply with Article 26 paragraph 3 of the Patent Law.
   ☐ The description does not comply with Article 33 of the Patent Law.
   ☐ The draft of the description does not comply with Rule 18 of the Implementing Regulations.

☐ The draft of the description does not comply with Rule 19 of the Implementing Regulations

☒ On the Claims:

☐ Claim(s) _____ does/do not possess the novelty as required by Article 22 paragraph 2 of the Patent Law.

☒ Claim(s) 1-9 does/do not possess the inventiveness as required by Article 22 paragraph 3 of the Patent Law.

☐ Claim(s) _____ does/do not possess the practical applicability as required by Article 22 paragraph 4 of the Patent Law.

☐ Claim(s) _____ is/are not patentable under Article 25 of the Patent Law.

☐ Claim(s) _____ does/do not comply with Article 26 paragraph 4 of the Patent Law.

☐ Claim(s) _____ does/do not comply with Article 31 paragraph 1 of the Patent Law.

☐ Claim(s) _____ does/do not comply with Article 33 of the Patent Law.

☐ Claim(s) _____ does/do not comply with the definition of inventions prescribed by Rule 2 paragraph 1 of the Implementing Regulations.

☐ Claim(s) _____ does/do not comply with the provisions of Rule 13 paragraph 1 of the Implementing Regulations.

☐ Claim(s) _____ does/do not comply with the provisions of Rule 20 of the Implementing Regulations.

☐ Claim(s) _____ does/do not comply with the provisions of Rule 21 of the Implementing Regulations.

☐ Claim(s) _____ does/do not comply with the provisions of Rule 22 of the Implementing Regulations.

☐ Claim(s) _____ does/do not comply with the provisions of Rule 23 of the Implementing Regulations.

☐ Claim(s) _____ does not comply with the provisions of Rule 43 paragraph 1 of the Implementing Regulations.

☐ The divisional application does not comply with the provisions of Rule 43 paragraph 1 of the Implementing Regulations.

7. In view of the conclusions set forth above, the Examiner is of the opinion that:

☐ The applicant should make amendments as directed in the text portion of the Notification.

☐ The applicant should expound in the response reasons why the application is patentable and make amendments to the application where there are deficiencies as pointed out in the text portion of the Notification, otherwise, the application will not be allowed.

☒ The application contains no allowable invention, and therefore, if the applicant fails to submit sufficient reasons to prove that the application does have merits, it will be rejected.

☐

8. The followings should be taken into consideration by the applicant in making the response:

(1) Under Article 37 of the Patent Law, the applicant should respond to the office action within 4 months counting from the date of receipt of the Notification. If, without any justified reason, the time limit is not met, the application shall be deemed to have been withdrawn.

(2) Any amendments to the application should be in conformity with the provisions of Article 33 of the Patent Law. Substitution pages should be in duplicate and the format of the substitution should be in conformity with the relevant provision contained in "The Examination Guidelines".

(3) The response to the Notification and/or revision of the application should be mailed to or handed over to the "Reception Division" of the Patent Office, and documents not mailed or handed over to the Reception Divisions have no legal effect.

(4) Without an appointment, the applicant and/or his agent shall not interview with the Examiner in the Patent Office.

9. This Notification contains a text portion of 2 pages and the following attachments:

☐ __ cited reference(s), totaling __ pages.

☐

Examination Dept. _____ Examiner:_____ Seal of the Examination Department

Text of the First Office Action

1. Claim 1 recites a method of determining the protection that a remote computer has from malware. D1 (US2004/0039921A) discloses a method for detecting rogue software and specifically reveals the following technical features (see paras.[0033]-[0036] of the description and Figs.1-2): a client 12 (corresponding to "a remote computer") is connected to a server 14 and a database of acceptable file fingerprints via a network (the server and the database are together equivalent to the base computer receiving details of a program product); first, the system calculates and retrieves file fingerprints of application software it already has , at the client 12 (corresponding to "the program product operating on the remote computer, receiving similar information such as details of the program product from remote computers connected to the base computer"); upon installation of software, new files and their fingerprints and file locations are added to thc database; when application software is run on the client, the calculated Hash result and collected client information are forwarded to the server 14 for verification; on the server 14, the received Hash value of the file of the client is compared with certain desired client information stored in the database 16, such as a version of an operating system and parameters of a software patch; if they do not match each other, it will be judged that an application program having rogue software is running on the client 12 (corresponding to "identifying any malware processes that were not identified by said other remote computers having that particular combination of program products").

Claim 1 differs from D1 in that 1) claim 1 recites what is run and identified on the computer is security products whereas D1 defines application programs; 2) claim 1 recites that the security products operate at a point in time on the computer, whereas D1 does not disclose this technical feature. However, in the art, it is common technical means in the art to set a client to run antivirus/Trojan virus searching and killing software at a certain point in time, to judge whether the client has unidentified viruses or security vulnerabilities via a network virus/Trojan virus library, and notify the client of taking defensive measures such as updating. Therefore, it is obvious for those skilled in the art to obtain the claimed technical solution of claim 1 on the basis of D1 in conjunction with the common technical means in the art. Accordingly, claim 1 neither has any prominent substantive features nor represents any notable progress and

therefore lacks an inventive step under Article 22.3 of the Chinese Patent Law.

2. Claim 2 further defines claim 1. It is common technical means in the art to set a client to judge whether the client has unidentified viruses or security vulnerabilities in a way that the client is connected to a network virus/Trojan virus library, and notify the client of such information. Therefore, when claim 1 lacks an inventive step, claim 2 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

3. Claim 3 further defines claim 1 or 2. In the art, digital signature is a conventional security technology for ensuring security of information, so it is common technical means in the art to use signature files to identify malware. Name and versions are common attributes of application programs, so use thereof as details of security products is common technical means in the art. Therefore, when claim 1 or 2 lacks an inventive step, claim 3 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

4. Claim 4 recites an apparatus for determining the protection that a remote computer has from malware, the apparatus. D1 discloses a system for detecting rogue software and specifically reveals the following technical features (see paras.[0033]-[0036] of the description and Figs.1-2): a client 12 (corresponding to "a remote computer") is connected to a server 14 and a database of acceptable file fingerprints via a network (the server and the database are together equivalent to the base computer constructed and arranged to receive computer details of all or selected security products); first, the system calculates and retrieves file fingerprints of application software it already has , at the client 12 (corresponding to "the program product operating on the remote computer, the base computer being constructed and arranged to receive similar information such as details of program products from other remote computers connected to the base computer"); upon installation of software, new files and their fingerprints and file locations are added to the database; when application software is run on the client, the calculated Hash result and collected client information are forwarded to the server 14 for verification; on the server 14, the received Hash value of the file of the client is compared with certain desired client information stored in the database 16, such as a version of an operating system and parameters of a software patch; if they do not match each other, it will be judged that an application program having rogue software is running on the client 12 (corresponding to "the base computer being constructed and arranged to identify any malware processes that were not

identified by said other remote computers having that particular combination of program products").

Claim 4 differs from D1 in that 1) claim 4 recites what is run and identified on the computer is security products whereas D1 defines application programs; 2) claim 4 recites that the security products operate at a point in time on the computer, whereas D1 does not disclose this technical feature. However, in the art, it is common technical means in the art to set a client to run antivirus/Trojan virus searching and killing software at a certain point in time, to judge whether the client has unidentified viruses or security vulnerabilities via a network virus/Trojan virus library, and notify the client of taking defensive measures such as updating. Therefore, it is obvious for those skilled in the art to obtain the claimed technical solution of claim 4 on the basis of D1 in conjunction with the common technical means in the art. Accordingly, claim 4 neither has any prominent substantive features nor represents any notable progress and therefore lacks an inventive step under Article 22.3 of the Chinese Patent Law.

5. Claim 5 further defines claim 4. It is common technical means in the art to set a client to judge whether the client has unidentified viruses or security vulnerabilities in a way that the client is connected to a network virus/Trojan virus library, and notify the client of such information. Therefore, when claim 4 lacks an inventive step, claim 5 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

6. Claim 6 further defines claim 4 or 5. In the art, digital signature is a conventional security technology for ensuring security of information, so it is common technical means in the art to use signature files to identify malware. Name and versions are common attributes of application programs, so use thereof as details of security products is common technical means in the art. Therefore, when claim 4 or 5 lacks an inventive step, claim 6 dependent thereon lacks an inventive step under Article 22.3 of the Chinese Patent Law.

7. Claims 7-9 are product claims having a function module architecture exactly corresponding to steps of the method recited in claims 1-3. For the same reasons, claims 7-9 lack an inventive step under Article 22.3 of the Chinese Patent Law.

For the above reasons, all claims of the present application lack an inventive step. Meanwhile, the description does not contain any other patentable substantive content, so even if the Applicant reorganizes the claims and/or further defines the claims according to the disclosure of the description, the present application does not have the prospect of

being granted a patent right. If the Applicant cannot furnish sufficient reasons to prove the present application involves an inventive step within the four-month time limit for response prescribed in the Office Action herein, the present application will be rejected.

中华人民共和国国家知识产权局

100020
北京市朝阳区东三环中路1号环球金融中心办公楼东楼20层 北京市
金杜律师事务所
王茂华 黄倩

发文日：
2011年10月10日

2CK

申请号或专利号：201110036121.8　　发文序号：2011100200036220

申请人或专利权人：普瑞维克斯有限公司

发明创造名称：用于处理恶意软件的方法和装置

第一次审查意见通知书

1. ☒应申请人提出的实质审查请求，根据专利法第35条第1款的规定，国家知识产权局对上述发明专利申请进行实质审查。
☐根据专利法第35条第2款的规定，国家知识产权局决定自行对上述发明专利申请进行审查。

2. ☒申请人要求以其在：
GB专利局的申请日2005年06月30日为优先权日。
☒申请人已经提交了经原受理机构证明的第一次提出的在先申请文件的副本。
☐申请人尚未提交经原受理机构证明的第一次提出的在先申请文件的副本，根据专利法第30条的规定视为未要求优先权要求。

3. ☐经审查，申请人于____提交的修改文件，不符合专利法实施细则第51条第1款的规定，不予接受。

4. 审查针对的申请文件：
☐原始申请文件。☒分案申请递交日提交的文件。☐下列申请文件：

5. ☐本通知书是在未进行检索的情况下作出的。
☒本通知书是在进行了检索的情况下作出的。
☒本通知书引用下列对比文件(其编号在今后的审查过程中继续沿用)：

编号	文件号或名称	公开日期（或抵触申请的申请日）
1	US 2004/0039921A1	20040226

6. 审查的结论性意见：
关于说明书：
☐申请的内容属于专利法第5条规定的不授予专利权的范围。
☐说明书不符合专利法第26条第3款的规定。
☐说明书不符合专利法第33条的规定。
☐说明书的撰写不符合专利法实施细则第17条的规定。
☐____

中华人民共和国国家知识产权局 专利审查业务章

ODK

关于权利要求书：

☐权利要求_____不符合专利法第 2 条第 2 款的规定。
☐权利要求_____不符合专利法第 9 条第 1 款的规定。
☐权利要求_____不具备专利法第 22 条第 2 款规定的新颖性。
☒权利要求 1-9 不具备专利法第 22 条第 3 款规定的创造性。
☐权利要求_____不具备专利法第 22 条第 4 款规定的实用性。
☐权利要求_____属于专利法第 25 条规定的不授予专利权的范围。
☐权利要求_____不符合专利法第 26 条第 4 款的规定。
☐权利要求_____不符合专利法第 31 条第 1 款的规定。
☐权利要求_____不符合专利法第 33 条的规定。
☐权利要求_____不符合专利法实施细则第 19 条的规定。
☐权利要求_____不符合专利法实施细则第 20 条的规定。
☐权利要求_____不符合专利法实施细则第 21 条的规定。
☐权利要求_____不符合专利法实施细则第 22 条的规定。
☐_____

☐申请不符合专利法第 26 条第 5 款或者实施细则第 26 条的规定。
☐申请不符合专利法第 20 条第 1 款的规定。
☐分案申请不符合专利法实施细则第 43 条第 1 款的规定。

上述结论性意见的具体分析见本通知书的正文部分。

7. 基于上述结论性意见，审查员认为：

☐申请人应当按照通知书正文部分提出的要求，对申请文件进行修改。
☐申请人应当在意见陈述书中论述其专利申请可以被授予专利权的理由，并对通知书正文部分中指出的不符合规定之处进行修改，否则将不能授予专利权。
☒专利申请中没有可以被授予专利权的实质性内容，如果申请人没有陈述理由或者陈述理由不充分，其申请将被驳回。
☐_____

8. 申请人应注意下列事项：

(1) 根据专利法第 37 条的规定，申请人应在收到本通知之日起的 4 个月内陈述意见，如果申请人无正当理由逾期不答复，其申请将被视为撤回。

(2) 申请人对其申请的修改应当符合专利法第 33 条的规定，不得超出原说明书和权利要求书记载的范围，同时申请人对专利申请文件进行的修改应当符合专利法实施细则第 51 条第 3 款的规定，按照本通知书的要求进行修改。

(3) 申请人的意见陈述书和/或修改文本应邮寄或递交国家知识产权局专利局受理处，凡未邮寄或递交给受理处的文件不具备法律效力。

(4) 未经预约，申请人和/或代理人不得前来国家知识产权局专利局与审查员举行会晤。

9. 本通知书正文部分共有 2 页，并附有下述附件：

☐引用的对比文件的复印件共_____份_____页。
☐_____

中华人民共和国国家知识产权局 专利申查业务章

审查员：何明伦　　联系电话：01062412811

纸件申请，回函请寄：100088 北京市海淀区蓟门桥西土城路 6 号　国家知识产权局专利局受理处收
电子申请，应当通过电子专利申请系统以电子文件形式提交相关文件。除另有规定外，以纸件等其他形式提交的文件视为未提交。

第一次审查意见通知书

申请号:2011100361218

1.权利要求 1 要求保护一种确定远程计算机具有防止恶意软件的保护的方法。对比文件 1（US2004/0039921A1）公开了一种检测流氓软件的方法，其具体（参见[0033]-[0046]、图 1-2）披露了如下技术特征：客户机 12（即远程计算机）通过网络连接到服务器 14 及接收文件指纹的数据库（服务器和数据库一起相当于基计算机，接收程序产品的细节）；系统首先在客户机 12 计算和检索其已有的应用软件的文件指纹（即程序产品在远程计算机上操作，并从连接到基计算机的远程计算机接收程序产品的细节等类似信息）；软件安装时新的文件及其指纹、文件位置被添加到数据库；客户机上运行应用软件时，计算的哈希结果和收集的客户机信息被传送到服务器 14 校验；服务器 14 上将收到的客户机的文件的哈希值与数据库 16 中存储的期望的某种客户机信息，如操作系统版本、软件补丁包的参数进行比较，如果不匹配则判定客户机 12 上运行了带有流氓软件的应用程序（相当于识别具有程序产品特定组合的其他远程计算机没有识别出来的恶意软件）。

权利要求 1 与对比文件 1 的区别在于：1.权利要求 1 限定了计算机上运行和识别的是安全产品，而对比文件 1 中是应用程序；2.权利要求 1 限定了安全产品在计算机上按时间点操作，而对比文件 1 没有公开该技术特征。但在本领域中，设定客户机在一定的时间点运行反病毒/查杀木马软件，通过网络病毒/木马库判定客户机是否存在未识别的病毒或安全漏洞，并通知客户机执行更新等防护措施，属于本领域的常用技术手段。故在对比文件 1 上结合本领域的常用技术手段得到权利要求 1 要求保护的技术方案，对本领域技术人员而言是显而易见的。因此，权利要求 1 不具备突出的实质性特点和显著的进步，不具备专利法第 22 条第 3 款规定的创造性。

2.权利要求 2 对权利要求 1 进行了进一步限定。通过客户机连接网络病毒/木马库来判定客户机是否存在未识别的病毒或安全漏洞，并通知客户机这些信息，属于本领域的常用技术手段。因此，当其引用的权利要求不具备创造性时，该权利要求也不具备专利法第 22 条第 3 款规定的创造性。

3.权利要求 3 对权利要求 1 或 2 进行了进一步限定。在本领域中数字签名是一种常规的确保信息的安全技术，故使用签名文件来识别恶意软件，是本领域的常用技术手段。而使用名称、版本是常见的应用程序属性，即使用它们来作为安全产品的细节，属于本领域的常用技术手段。因此，当其引用的权利要求不具备创造性时，该权利要求也不具备专利法第 22 条第 3 款规定的创造性。

4. 权利要求 4 要求保护一种确定远程计算机具有防止恶意软件的保护的装置。对比文件 1 公开了一种检测流氓软件的系统，其具体（参见[0033]-[0046]、图 1-2）披露了如下技术特征：客户机 12（即远程计算机）通过网络连接到服务器 14 及接收文件指纹的数据库（服务器和数据库一起相当于基计算机，被构建和安排为接收所有或选择的程序产品的细节）；系统首先在客户机 12 计算和检索其已有的应用软件的文件指纹（即程序产品在远程计算机上操作，且基计算机被构建和安排为从连接到基计算机的其他远程计算机接收程序产品的细节等类似信息）；软件安装时新的文件及其指纹、文件位置被添加到数据库；客户机上运行应用软件时，计算的哈希结果和收集的客户机信息被传送到服务器 14 校验；服务器 14 上将收到的客户机的文件的哈希值与数据库 16 中存储的期望的某种客户机信息，如操作系统版本、软件补丁包的参数进行比较，如果不匹配则

专利审查业务章

判定客户机 12 上运行了带有流氓软件的应用程序(相当于基计算机被构建和安排为识别具有程序产品特定组合的其他远程计算机没有识别出来的恶意软件)。

权利要求 4 与对比文件 1 的区别在于：1. 权利要求 4 限定了计算机上运行和识别的是安全产品，而对比文件 1 中是应用程序；2. 权利要求 4 限定了安全产品在计算机上按时间点操作，而对比文件 1 没有公开该技术特征。但在本领域中，设定客户机在一定的时间点运行反病毒/查杀木马软件，通过网络病毒/木马库判定客户机是否存在未识别的病毒或安全漏洞，并通知客户机执行更新等防护措施，属于本领域的常用技术手段。故在对比文件 1 上结合本领域的常用技术手段得到权利要求 4 要求保护的技术方案，对本领域技术人员而言是显而易见的。因此，权利要求 4 不具备突出的实质性特点和显著的进步，不具备专利法第 22 条第 3 款规定的创造性。

5. 权利要求 5 对权利要求 1 进行了进一步限定。通过客户机连接网络病毒/木马库来判定客户机是否存在未识别的病毒或安全漏洞，并通知客户机这些信息，属于本领域的常用技术手段。因此，当其引用的权利要求不具备创造性时，该权利要求也不具备专利法第 22 条第 3 款规定的创造性。

6. 权利要求 6 对权利要求 4 或 5 进行了进一步限定。在本领域中数字签名是一种常规的确保信息的安全技术，故使用签名文件来识别恶意软件，是本领域的常用技术手段。而使用名称、版本是常见的应用程序属性，即使用它们来作为安全产品的细节，属于本领域的常用技术手段。因此，当其引用的权利要求不具备创造性时，该权利要求也不具备专利法第 22 条第 3 款规定的创造性。

7. 权利要求 7-9 是与权利要求 1-3 所述的方法各步骤完全一一对应的功能模块架构的产品权利要求，故基于相同的理由，权利要求 7-9 也不具备专利法第 22 条第 3 款规定的创造性。

基于上述理由，该申请的全部权利要求都不具备创造性，同时说明书中也没有记载其他任何可以授予专利权的实质性内容，因而即使申请人对权利要求进行重新组合和/或根据说明书记载的内容作进一步的限定，该申请也不具备被授权专利权的前景，除非申请人能够在本通知书指定的四个月答复期限内提出表明该申请具有创造性的充分理由，否则该申请将被驳回。

1FK

中华人民共和国国家知识产权局 专利审查业务章

审查员姓名：[illegible]明伦
审查员电话：1[illegible]515

100020
北京市朝阳区东三环中路1号环球金融中心办公楼东楼20层 北京市金杜律师事务所
王茂华 黄倩

发文日：
2011年10月10日

2CK

申请号或专利号：201110036121.8　　发文序号：2011100200036220

申请人或专利权人：普瑞维克斯有限公司

发明创造名称：用于处理恶意软件的方法和装置

第一次审查意见通知书

1. ☒应申请人提出的实质审查请求，根据专利法第35条第1款的规定，国家知识产权局对上述发明专利申请进行实质审查。
☐根据专利法第35条第2款的规定，国家知识产权局决定自行对上述发明专利申请进行审查。
2. ☒申请人要求以其在：
GB专利局的申请日2005年06月30日为优先权日。
☒申请人已经提交了经原受理机构证明的第一次提出的在先申请文件的副本。
☐申请人尚未提交经原受理机构证明的第一次提出的在先申请文件的副本，根据专利法第30条的规定视为未要求优先权要求。
3. ☐经审查，申请人于____提交的修改文件，不符合专利法实施细则第51条第1款的规定，不予接受。
4. 审查针对的申请文件：
☐原始申请文件。☒分案申请递交日提交的文件。☐下列申请文件：

5. ☐本通知书是在未进行检索的情况下作出的。
☒本通知书是在进行了检索的情况下作出的。
☒本通知书引用下列对比文件(其编号在今后的审查过程中继续沿用)：

编号	文件号或名称	公开日期（或抵触申请的申请日）
1	US 2004/0039921A1	20040226

6. 审查的结论性意见：
关于说明书：
☐申请的内容属于专利法第5条规定的不授予专利权的范围。
☐说明书不符合专利法第26条第3款的规定。
☐说明书不符合专利法第33条的规定。
☐说明书的撰写不符合专利法实施细则第17条的规定。
☐____

中华人民共和国国家知识产权局 专利审查业务章

关于权利要求书：

☐权利要求_____不符合专利法第 2 条第 2 款的规定。
☐权利要求_____不符合专利法第 9 条第 1 款的规定。
☐权利要求_____不具备专利法第 22 条第 2 款规定的新颖性。
☒权利要求 1-9 不具备专利法第 22 条第 3 款规定的创造性。
☐权利要求_____不具备专利法第 22 条第 4 款规定的实用性。
☐权利要求_____属于专利法第 25 条规定的不授予专利权的范围。
☐权利要求_____不符合专利法第 26 条第 4 款的规定。
☐权利要求_____不符合专利法第 31 条第 1 款的规定。
☐权利要求_____不符合专利法第 33 条的规定。
☐权利要求_____不符合专利法实施细则第 19 条的规定。
☐权利要求_____不符合专利法实施细则第 20 条的规定。
☐权利要求_____不符合专利法实施细则第 21 条的规定。
☐权利要求_____不符合专利法实施细则第 22 条的规定。
☐_____

☐申请不符合专利法第 26 条第 5 款或者实施细则第 26 条的规定。
☐申请不符合专利法第 20 条第 1 款的规定。
☐分案申请不符合专利法实施细则第 43 条第 1 款的规定。

上述结论性意见的具体分析见本通知书的正文部分。

7. 基于上述结论性意见，审查员认为：

☐申请人应当按照通知书正文部分提出的要求，对申请文件进行修改。
☐申请人应当在意见陈述书中论述其专利申请可以被授予专利权的理由，并对通知书正文部分中指出的不符合规定之处进行修改，否则将不能授予专利权。
☒专利申请中没有可以被授予专利权的实质性内容，如果申请人没有陈述理由或者陈述理由不充分，其申请将被驳回。
☐_____

8. 申请人应注意下列事项：

(1) 根据专利法第 37 条的规定，申请人应在收到本通知之日起的 4 个月内陈述意见，如果申请人无正当理由逾期不答复，其申请将被视为撤回。

(2) 申请人对其申请的修改应当符合专利法第 33 条的规定，不得超出原说明书和权利要求书记载的范围，同时申请人对专利申请文件进行的修改应当符合专利法实施细则第 51 条第 3 款的规定，按照本通知书的要求进行修改。

(3) 申请人的意见陈述书和/或修改文本应邮寄或递交国家知识产权局专利局受理处，凡未邮寄或递交给受理处的文件不具备法律效力。

(4) 未经预约，申请人和/或代理人不得前来国家知识产权局专利局与审查员举行会晤。

9. 本通知书正文部分共有 2 页，并附有下述附件：

☐引用的对比文件的复印件共_____份_____页。
☐_____

审查员：何明伦　　联系电话：01062412811

专利审查协作中心

210401
2010.2

纸件申请，回函请寄：100088 北京市海淀区蓟门桥西土城路 6 号　国家知识产权局专利局受理处收
电子申请，应当通过电子专利申请系统以电子文件形式提交相关文件。除另有规定外，以纸件等其他形式提交的文件视为未提交。

第一次审查意见通知书

申请号:2011100361218

1. 权利要求 1 要求保护一种确定远程计算机具有防止恶意软件的保护的方法。对比文件 1（US2004/0039921A1）公开了一种检测流氓软件的方法，其具体（参见[0033]-[0046]、图 1-2）披露了如下技术特征：客户机 12（即远程计算机）通过网络连接到服务器 14 及接收文件指纹的数据库（服务器和数据库一起相当于基计算机，接收程序产品的细节）；系统首先在客户机 12 计算和检索其已有的应用软件的文件指纹（即程序产品在远程计算机上操作，并从连接到基计算机的远程计算机接收程序产品的细节等类似信息）；软件安装时新的文件及其指纹、文件位置被添加到数据库；客户机上运行应用软件时，计算的哈希结果和收集的客户机信息被传送到服务器 14 校验；服务器 14 上将收到的客户机的文件的哈希值与数据库 16 中存储的期望的某种客户机信息，如操作系统版本、软件补丁包的参数进行比较，如果不匹配则判定客户机 12 上运行了带有流氓软件的应用程序（相当于识别具有程序产品特定组合的其他远程计算机没有识别出来的恶意软件）。

权利要求 1 与对比文件 1 的区别在于：1. 权利要求 1 限定了计算机上运行和识别的是安全产品，而对比文件 1 中是应用程序；2. 权利要求 1 限定了安全产品在计算机上按时间点操作，而对比文件 1 没有公开该技术特征。但在本领域中，设定客户机在一定的时间点运行反病毒/查杀木马软件，通过网络病毒/木马库判定客户机是否存在未识别的病毒或安全漏洞，并通知客户机执行更新等防护措施，属于本领域的常用技术手段。故在对比文件 1 上结合本领域的常用技术手段得到权利要求 1 要求保护的技术方案，对本领域技术人员而言是显而易见的。因此，权利要求 1 不具备突出的实质性特点和显著的进步，不具备专利法第 22 条第 3 款规定的创造性。

2. 权利要求 2 对权利要求 1 进行了进一步限定。通过客户机连接网络病毒/木马库来判定客户机是否存在未识别的病毒或安全漏洞，并通知客户机这些信息，属于本领域的常用技术手段。因此，当其引用的权利要求不具备创造性时，该权利要求也不具备专利法第 22 条第 3 款规定的创造性。

3. 权利要求 3 对权利要求 1 或 2 进行了进一步限定。在本领域中数字签名是一种常规的确保信息的安全技术，故使用签名文件来识别恶意软件，是本领域的常用技术手段。而使用名称、版本是常见的应用程序属性，即使用它们来作为安全产品的细节，属于本领域的常用技术手段。因此，当其引用的权利要求不具备创造性时，该权利要求也不具备专利法第 22 条第 3 款规定的创造性。

4. 权利要求 4 要求保护一种确定远程计算机具有防止恶意软件的保护的装置。对比文件 1 公开了一种检测流氓软件的系统，其具体（参见[0033]-[0046]、图 1-2）披露了如下技术特征：客户机 12（即远程计算机）通过网络连接到服务器 14 及接收文件指纹的数据库（服务器和数据库一起相当于基计算机，被构建和安排为接收所有或选择的程序产品的细节）；系统首先在客户机 12 计算和检索其已有的应用软件的文件指纹（即程序产品在远程计算机上操作，且基计算机被构建和安排为从连接到基计算机的其他远程计算机接收程序产品的细节等类似信息）；软件安装时新的文件及其指纹、文件位置被添加到数据库；客户机上运行应用软件时，计算的哈希结果和收集的客户机信息被传送到服务器 14 校验；服务器 14 上将收到的客户机的文件的哈希值与数据库 16 中存储的期望的某种客户机信息，如操作系统版本、软件补丁包的参数进行比较，如果不匹配则

专利审查业务章

判定客户机 12 上运行了带有流氓软件的应用程序(相当于基计算机被构建和安排为识别具有程序产品特定组合的其他远程计算机没有识别出来的恶意软件)。

权利要求 4 与对比文件 1 的区别在于：1.权利要求 4 限定了计算机上运行和识别的是安全产品，而对比文件 1 中是应用程序；2.权利要求 4 限定了安全产品在计算机上按时间点操作，而对比文件 1 没有公开该技术特征。但在本领域中，设定客户机在一定的时间点运行反病毒/查杀木马软件，通过网络病毒/木马库判定客户机是否存在未识别的病毒或安全漏洞，并通知客户机执行更新等防护措施，属于本领域的常用技术手段。故在对比文件 1 上结合本领域的常用技术手段得到权利要求 4 要求保护的技术方案，对本领域技术人员而言是显而易见的。因此，权利要求 4 不具备突出的实质性特点和显著的进步，不具备专利法第 22 条第 3 款规定的创造性。

5.权利要求 5 对权利要求 1 进行了进一步限定。通过客户机连接网络病毒/木马库来判定客户机是否存在未识别的病毒或安全漏洞，并通知客户机这些信息，属于本领域的常用技术手段。因此，当其引用的权利要求不具备创造性时，该权利要求也不具备专利法第 22 条第 3 款规定的创造性。

6.权利要求 6 对权利要求 4 或 5 进行了进一步限定。在本领域中数字签名是一种常规的确保信息的安全技术，故使用签名文件来识别恶意软件，是本领域的常用技术手段。而使用名称、版本是常见的应用程序属性，即使用它们来作为安全产品的细节，属于本领域的常用技术手段。因此，当其引用的权利要求不具备创造性时，该权利要求也不具备专利法第 22 条第 3 款规定的创造性。

7.权利要求 7-9 是与权利要求 1-3 所述的方法各步骤完全一一对应的功能模块架构的产品权利要求，故基于相同的理由，权利要求 7-9 也不具备专利法第 22 条第 3 款规定的创造性。

基于上述理由，该申请的全部权利要求都不具备创造性，同时说明书中也没有记载其他任何可以授予专利权的实质性内容，因而即使申请人对权利要求进行重新组合和/或根据说明书记载的内容作进一步的限定，该申请也不具备被授权专利权的前景，除非申请人能够在本通知书指定的四个月答复期限内提出表明该申请具有创造性的充分理由，否则该申请将被驳回。

审查员姓名：[illegible]
[illegible]：19[illegible]515

Electronic Acknowledgement Receipt

EFS ID:	14793160
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	METHODS AND APPARATUS FOR DEALING WITH MALWARE
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	22442
Filer:	Bradley M. Knepper/Robert Roe-Pachirat
Filer Authorized By:	Bradley M. Knepper
Attorney Docket Number:	6890-93
Receipt Date:	25-JAN-2013
Filing Date:	30-JUN-2006
Time Stamp:	14:07:47
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		IDS_07.pdf	242532 6b03d67d391d639abee874a6eccd76d398185fe9	yes	3

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Transmittal Letter	1	2
	Information Disclosure Statement (IDS) Form (SB08)	3	3

Warnings:					
Information:					
2	Non Patent Literature	CN2011100361218_OA_09-24-2012.pdf	450553 052bc8d93f9a752f83fb171c5d3dc627fb600e2e	no	8
Warnings:					
Information:					
3	Non Patent Literature	CN2011100361218_OA_10-10-2011.pdf	834595 777e6078d28d50f39b6b50f8075c57d204938e58	no	14
Warnings:					
Information:					
		Total Files Size (in bytes):	1527680		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re the Application of:	)	Group Art Unit: 2432
Melvyn Morris	)	Confirmation No.: 4785
Serial No.: 11/477,807	)	Examiner: Cordelia P.K. Zecher
Filed: June 30, 2006	)	SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT
Atty. File No.: 6890-93	)	
Entitled: "Methods and Apparatus for Dealing With Malware"	)	*Electronically Submitted*

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

The references cited on attached Form PTO/SB08 are being called to the attention of the Examiner.

☒ Copies of the cited non-patent and/or foreign references are enclosed herewith.

☐ Copies of the cited U.S. patents and/or patent applications are enclosed herewith.

☐ Copies of the cited U.S. patents/patent application publications are not enclosed in accordance with 37 C.F.R. § 1.98(a).

☐ Copies of the cited references are not enclosed, in accordance with 37 C.F.R. § 1.98(d), because the references were cited by or submitted to the U.S. Patent and Trademark Office in prior application Serial No. _____ filed _____, which is relied upon for an earlier filing date under 35 U.S.C. § 120.

☒ To the best of applicants' belief, the pertinence of the foreign-language references is believed to be summarized in the attached English abstracts and/or in the figures, although applicants do not necessarily vouch for the accuracy of the translation.

☒ Examiner's attention is drawn to the following related applications:

- Serial No. 13/543,865 filed 07/08/12 (Attorney's Ref. No. 6890-93-DIV)
- Serial No. 13/543,866 filed 07/08/12 (Attorney's Ref. No. 6890-93-CON)

☐ Other:_____

Submission of the above information is not intended as an admission that any item is citable under the statutes or rules to support a rejection, that any item disclosed represents analogous art, or that those skilled in the art would refer to or recognize the pertinence of any reference without the benefit of hindsight, nor should an inference be drawn as to the pertinence

of the references based on the order in which they are presented. Submission of this statement should not be taken as an indication that a search has been conducted, or that no better art exists.

It is respectfully requested that the cited information be expressly considered during the prosecution of this application and the references made of record therein.

Respectfully submitted,

SHERIDAN ROSS P.C.

By: _____

Bradley M. Knepper
Registration No. 44189
1560 Broadway, Suite 1200
Denver, Colorado 80202-5141
(303) 863-9700

Date: June 24, 2013

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	6890-93	4785

7590 01/30/2013

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

EXAMINER	
ZECHER, CORDELIA P K	
ART UNIT	**PAPER NUMBER**
2432	
MAIL DATE	**DELIVERY MODE**
01/30/2013	PAPER

NOTICE OF NON-COMPLIANT INFORMATION DISCLOSURE STATEMENT

An Information Disclosure Statement (IDS) filed 01-25-13 in the above-identified application fails to meet the requirements of 37 CFR 1.97(d) for the reason(s) specified below. Accordingly, the IDS will be placed in the file, but the information referred to therein has not been considered.

The IDS is not compliant with 37 CFR 1.97(d) because:

☒ The IDS lacks a statement as specified in 37 CFR 1.97(e).

☐ The IDS lacks the fee set forth in 37 CFR 1.17(p).

☐ The IDS was filed after the issue fee was paid. Applicant may wish to consider filing a petition to withdraw the application from issue under 37 CFR 1.313(c) to have the IDS considered. See MPEP 1308.

Alma Cedeño

571-272-4200 or 1-888-786-0101
Application Assistance Unit
Office of Data Management

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

22442 7590 12/13/2012

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

Leslie M. Roberts	(Depositor's name)
/Leslie M. Roberts/	(Signature)
March 8, 2013	(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	06/30/2006	Melvyn Morris	6890-93	4785

TITLE OF INVENTION: METHODS AND APPARATUS FOR DEALING WITH MALWARE

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1770	$300	$0	$2070	03/13/2013

EXAMINER	ART UNIT	CLASS-SUBCLASS
ZECHER, CORDELIA P K	2432	726-024000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 Sheridan Ross P.C.
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: PREVX LIMITED

(B) RESIDENCE: (CITY and STATE OR COUNTRY): UNITED KINGDOM

Please check the appropriate assignee category or categories (will not be printed on the patent) : ☐ Individual ☒ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:

☒ Issue Fee

☒ Publication Fee (No small entity discount permitted)

☐ Advance Order - # of Copies _____

4b. Payment of Fee(s): (**Please first reapply any previously paid issue fee shown above**)

☐ A check is enclosed.

☐ Payment by credit card. Form PTO-2038 is attached.

☐ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

X Payment made via EFS-Web

5. **Change in Entity Status** (from status indicated above)

☒ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature /Bradley M. Knepper/ Date March 8, 2013

Typed or printed name Bradley M. Knepper Registration No. 44,189

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re the Application of:	)	Group Art Unit: 2432
	)	
MORRIS et al.	)	Examiner: ZECHER, CORDELIA P K
	)	
Serial No.: 11/477,807	)	Confirmation No.: 4785
	)	
Filed: June 30, 2006	)	SUBMISSION OF FORMAL DRAWINGS
	)	
Atty. File No.: 6890-93	)	***Filed Electronically***
	)	

For: "METHODS AND APPARATUS FOR DEALING WITH MALWARE"

Assistant Commissioner for Patents
Attn: Drawing Review Branch
Washington, D.C. 20231

Dear Sir:

In response to the Notice of Allowance mailed December 13, 2012, in connection with the above-identified patent application, enclosed for filing with said application are three sheets of formal drawings illustrating Figs. 1-4 of the application. Figs. 1-4 constitute all of the drawings in the application.

Respectfully submitted,

SHERIDAN ROSS P.C.

Date: March 8, 2013

By: /Bradley M. Knepper/
Bradley M. Knepper
Reg. No. 44,189
1560 Broadway, Suite 1200
Denver, CO 80202
Telephone: 303-863-9700

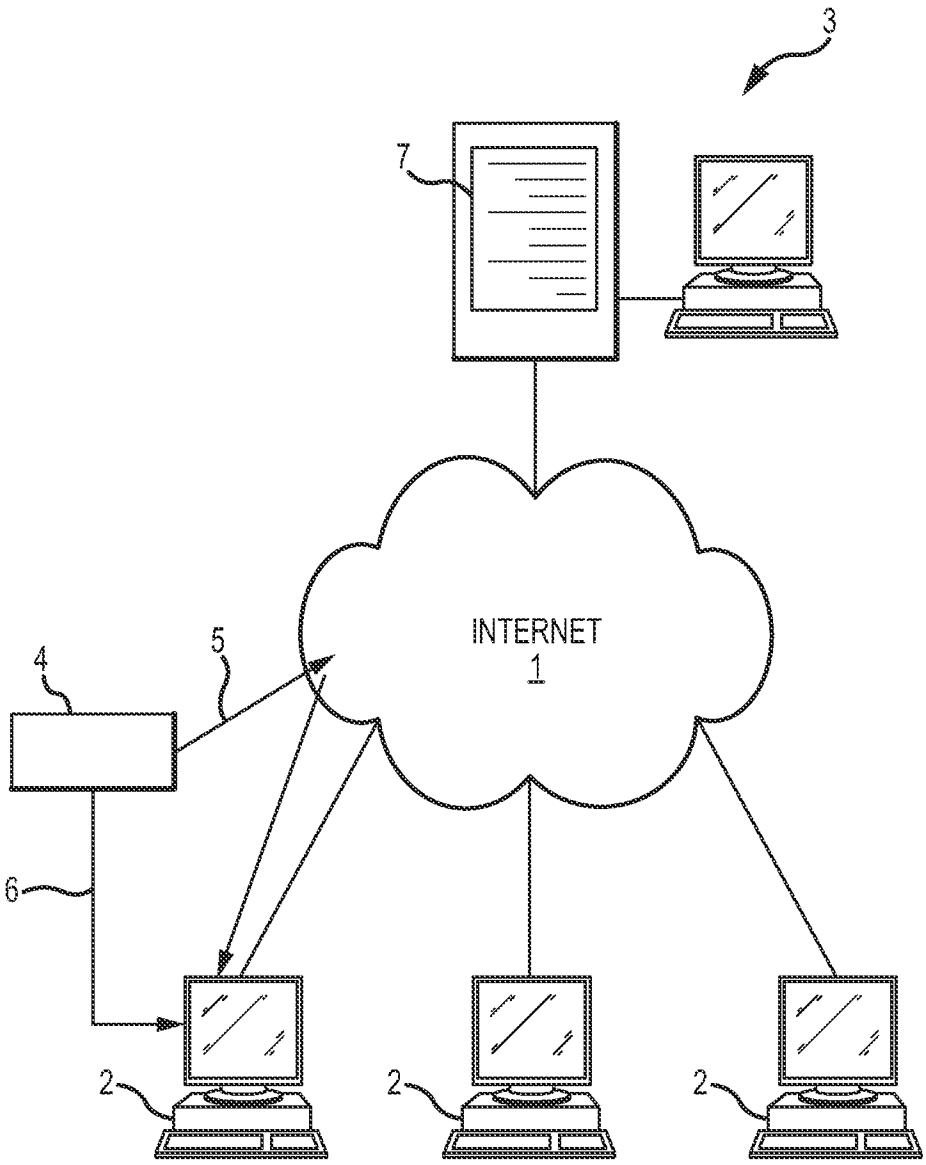


FIG.1

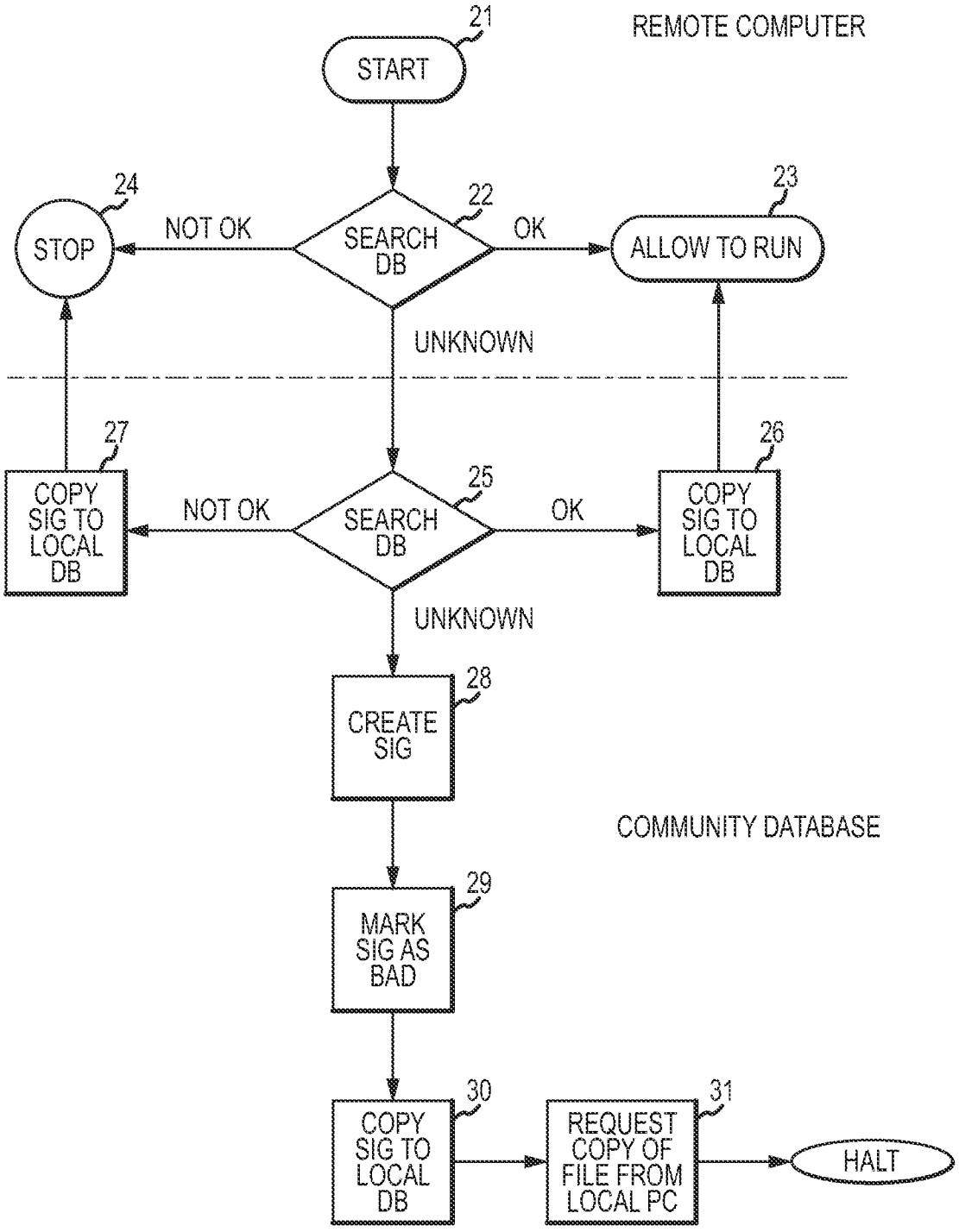

21
START
REMOTE COMPUTER
24
STOP
NOT OK
22
SEARCH DB
OK
23
ALLOW TO RUN
UNKNOWN
27
COPY SIG TO LOCAL DB
NOT OK
25
SEARCH DB
OK
26
COPY SIG TO LOCAL DB
UNKNOWN
28
CREATE SIG
COMMUNITY DATABASE
29
MARK SIG AS BAD
30
COPY SIG TO LOCAL DB
31
REQUEST COPY OF FILE FROM LOCAL PC
HALT

FIG.2

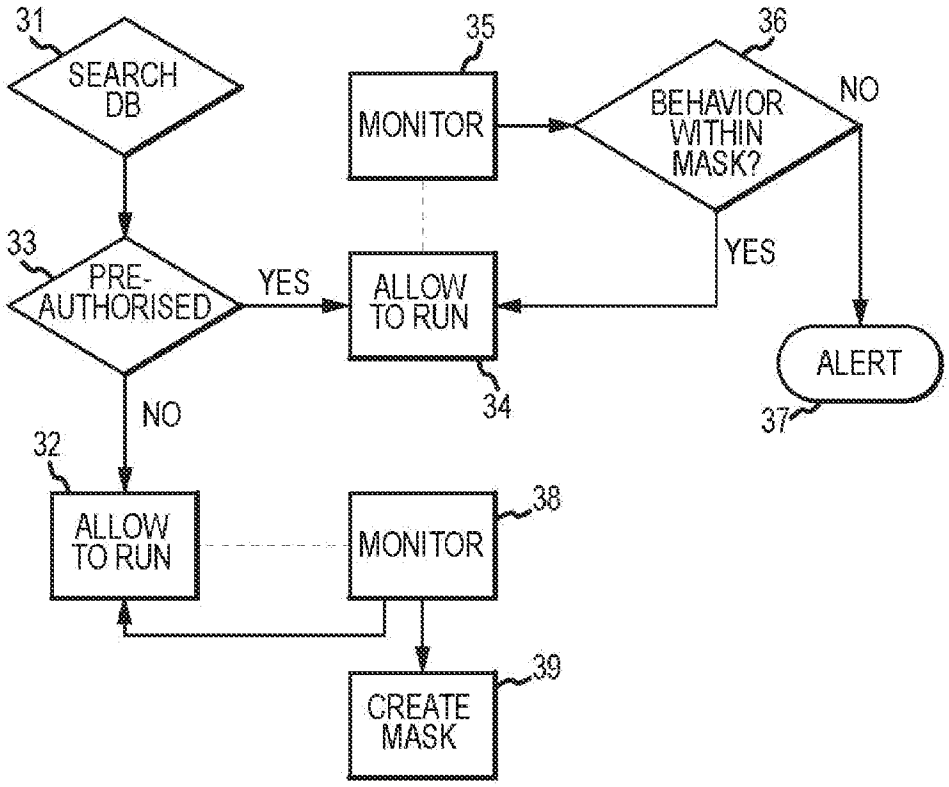

31
SEARCH DB
35
MONITOR
36
BEHAVIOR WITHIN MASK?
NO
YES
33
PRE-AUTHORISED
YES
ALLOW TO RUN
34
ALERT
37
NO
32
ALLOW TO RUN
MONITOR
38
CREATE MASK
39

FIG.3

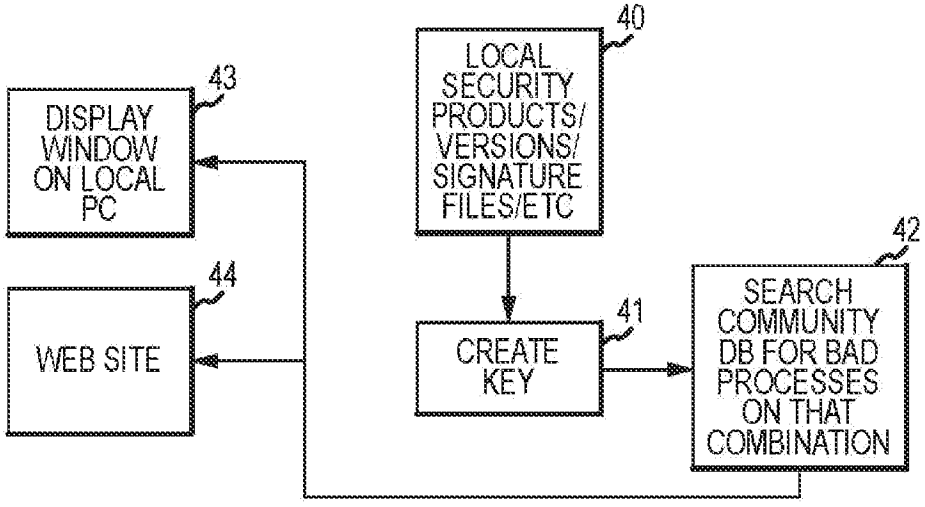

40
LOCAL SECURITY PRODUCTS/ VERSIONS/ SIGNATURE FILES/ETC
43
DISPLAY WINDOW ON LOCAL PC
42
SEARCH COMMUNITY DB FOR BAD PROCESSES ON THAT COMBINATION
44
WEB SITE
41
CREATE KEY

FIG.4

Electronic Patent Application Fee Transmittal

Application Number:	11477807
Filing Date:	30-Jun-2006
Title of Invention:	METHODS AND APPARATUS FOR DEALING WITH MALWARE
First Named Inventor/Applicant Name:	Melvyn Morris
Filer:	Bradley M. Knepper/Leslie Roberts
Attorney Docket Number:	6890-93

Filed as Small Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Utility Appl Issue Fee	2501	1	885	885
Publ. Fee- Early, Voluntary, or Normal	1504	1	300	300

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	**1185**

Electronic Acknowledgement Receipt

EFS ID:	15159720
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	METHODS AND APPARATUS FOR DEALING WITH MALWARE
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	22442
Filer:	Bradley M. Knepper/Leslie Roberts
Filer Authorized By:	Bradley M. Knepper
Attorney Docket Number:	6890-93
Receipt Date:	08-MAR-2013
Filing Date:	30-JUN-2006
Time Stamp:	16:13:08
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$1185
RAM confirmation Number	3883
Deposit Account	191970
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Issue Fee Payment (PTO-85B)	ISSUE_FEE.pdf	90696 f5921b560342095438a948aeb8cb459dc9c6b096	no	1
Warnings:					
Information:					
2	Drawings-only black and white line drawings	SUBMISSION_FORMAL_DRAWINGS.pdf	62199 5db86477c63cc79f12dd6de761b420a0e25b709b	no	1
Warnings:					
Information:					
3	Drawings-only black and white line drawings	FORMAL_FIGS.pdf	262130 103d7cbe07f455f662a25ddad5a3edfcb1d7b90c	no	3
Warnings:					
Information:					
4	Fee Worksheet (SB06)	fee-info.pdf	31822 dff5e6ffd37a6a7c8afc325bdb0b4ba5f7eb1ed7	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	446847		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 2

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	MORRIS, Melvyn et al.
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US -60/789,156			
		US -7,013,483	3/14/2006	Cohen, et al.	
		US -2003/0023857	1/30/2003	Hinchlife	
		US -2005/0027686	2/3/2005	Shipp	
		US -6,338,141	1/8/2002	Wells	
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			
		US -			

Change(s) applied to document, /D.HP./ 4/24/2012

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
		WO 96/30829	10/3/1996	APPLE COMPUTER, INC.		
		WO 99/15966	4/1/1999	SYMANTEC CORPORATION		
		EP 1536341 A2	6/1/2005	MICROSOFT CORPORATION		

Examiner Signature		Date Considered	

•EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional). 2 See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. 3 Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). 4 For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. 5 Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. 6 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

 ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

PTO/SB/08a (08-03)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	11/477,807
Filing Date	June 30, 2006
First Named Inventor	Melvyn Morris
Art Unit	2432
Examiner Name	C. Zecher
Attorney Docket Number	339000-000003

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number-Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
/C.Z./		US -2004/0083408	4/29/2004	Spiegel	
/C.Z./		US -2005/0022014	~~6/27/2005~~	Shipman	
		US -	01/2005		
		US -			
		US -			
		US -			
		US -			
		US -			

Change(s) applied to document, /A.J.P./ 5/3/2012

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]

Examiner Signature	/Cordelia Zecher/	Date Considered	02/17/2012

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

Receipt date: 01/18/2007

PTO JAN 18 2007 PATENT & TRADEMARK OFFICE

11477807 - GAU: 2432

FORM PTO-1449 (modified)
To: U.S. Department of Commerce
(PW FORM PAT-1449) Patent and Trademark Office

Atty. Dkt. No.	M#	Client Ref.
	11765-0355986	

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Applicant: MORRIS et al.

Appln. No.: 11/477,807

Filing Date: June 30, 2006

Date: January 18, 2007 Page 1 of 1

Examiner: Unassigned | Group Art Unit: 2131

U.S. PATENT DOCUMENTS

Examiner's Initials*		Document Number	Date MM/YYYY	Name (Family Name of First Inventor)	Class	Sub Class	Filing Date (if appropriate)
	AR	6,772,346	08/2004	CHESS			
	BR	6,944,772	09/2005	DOZORTSEV			
	CR	2004/0006704	01/2004	DAHLSTROM			
	DR	~~2004/0073810~~	05/2004	RACINE 20040083810			
Change(s) applied	ER	2005/0021994	01/2005	BARTON			
to document,	FR	2004/0153644	08/2004	McCORKENDALE			
/A.J.P./	GR						
5/3/2012	HR						
	IR						
	JR						
	KR						
	LR						
	MR						
	NR						

FOREIGN PATENT DOCUMENTS

		Document Number	Date MM/YYYY	Country	Inventor Name		English Abstract		Translation Readily Available	
							Enclosed	No	Enclose	No
	OR	1 280 040	01/2003	EP						
	PR	1 549 012	12/2003	EP						
	QR	03/021402	03/2003	WO						
	RR	2004/097602	11/2004	WO						
	SR	02/33525	04/2002	WO						

OTHER (Including in this order Author, Title, Periodical Name, Date, Pertinent Pages, etc.)

	TR				
	UR				
	VR				
	WR				

Examiner /Cordelia Zecher/ Date Considered: 04/28/2010

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP § 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to Applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /C.Z./

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
11/477,807	04/09/2013	8418250	6890-93	4785

22442 7590 03/20/2013

Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 1111 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site http://pair.uspto.gov for additional applicants):

Melvyn Morris, Turnditch, UNITED KINGDOM;
Paul Stubbs, Wyboston, UNITED KINGDOM;
Markus Hartwig, Milton Keynes, UNITED KINGDOM;
Darren Harter, Hucclecote, UNITED KINGDOM;

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The USA offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to encourage and facilitate business investment. To learn more about why the USA is the best country in the world to develop technology, manufacture products, and grow your business, visit SelectUSA.gov.

Best Available Copy

Large Entity Declaration

Schedule A – Patents assigned to *Assignee*

APPLICATION NUMBER	FILING DATE	PATENT NUMBER	ISSUE DATE
11/334317	18 JANUARY 2006	8418245	09 APRIL 2013
11/477807	30 JUNE 2006	8418250	09 APRIL 2013

PAGE 15/19 * RCVD AT 6/23/2016 6:04:10 AM [Eastern Daylight Time] * SVR:WPTOFAX-002/7 * DNIS:2736500 * CSID:+441534811210 * DURATION (mm-ss):03-58

Best Available Copy

Large Entity Declaration

PATENT
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

ASSERTION OF LARGE ENTITY STATUS
PURSUANT TO 37 C.F.R. §1.27 (g)(2)

Commissioner for Patents
Mail Stop M Correspondence
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This communication hereby asserts that the patents identified in the attached Schedule A are entitled to large entity status.

Respectfully submitted,

COMPANY or FIRM NAME AND ADDRESS:

Webroot Inc.
385 Interlocken Crescent, Suite 800
Broomfield, CO 80021
USA

Signature
Ryan Howe
Printed Name
General Counsel
Title

OR
Reg. # if US Attorney_____

PAGE 14/19 * RCVD AT 6/23/2016 6:04:10 AM [Eastern Daylight Time] * SVR:WFTOFAX-002/7 * DNIS:2736500 * CSID:+441534811210 * DURATION (mm-ss):03-58

PTO/AIA/82A (07-13)
Approved for use through 11/30/2014. OMB 0651-0051
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FOR POWER OF ATTORNEY TO ONE OR MORE REGISTERED PRACTITIONERS

NOTE: This form is to be submitted with the Power of Attorney by Applicant form (PTO/AIA/82B) to identify the application to which the Power of Attorney is directed, in accordance with 37 CFR 1.5, unless the application number and filing date are identified in the Power of Attorney by Applicant form. If neither form PTO/AIA/82A nor form PTO/AIA82B identifies the application to which the Power of Attorney is directed, the Power of Attorney will not be recognized in the application.

Application Number	11/477,807
Filing Date	06-30-2006
First Named Inventor	Melvyn Morris
Title	METHODS AND APPARATUS FOR DEALING WITH MALWARE
Art Unit	2432
Examiner Name	ZECHER, CORDELIA P K
Attorney Docket Number	04584.0116US01

SIGNATURE of Applicant or Patent Practitioner

Signature	/Andrew T. Pouzeshi/	Date (Optional)	2017-01-11
Name	Andrew T. Pouzeshi	Registration Number	67672
Title (if Applicant is a juristic entity)	Attorney for Applicant		
Applicant Name (if Applicant is a juristic entity)	Webroot Inc.		

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. If more than one applicant, use multiple forms.

☑ *Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Doc Code: PA..
Document Description: Power of Attorney

PTO/AIA/82B (07-13)
Approved for use through 11/30/2014. OMB 0651-0051
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

POWER OF ATTORNEY BY APPLICANT

I hereby revoke all previous powers of attorney given in the application identified in either the attached transmittal letter or the boxes below.

Application Number	Filing Date

(Note: The boxes above may be left blank if information is provided on form PTO/AIA/82A.)

☑ I hereby appoint the Patent Practitioner(s) associated with the following Customer Number as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above: 23552

OR

☐ I hereby appoint Practitioner(s) named in the attached list (form PTO/AIA/82C) as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the patent application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above. (Note: Complete form PTO/AIA/82C.)

Please recognize or change the correspondence address for the application identified in the attached transmittal letter or the boxes above to:

☑ The address associated with the above-mentioned Customer Number

OR

☐ The address associated with Customer Number:

OR

Firm or Individual Name					
Address					
City		State		Zip	
Country					
Telephone		Email			

I am the Applicant (if the Applicant is a juristic entity, list the Applicant name in the box):

WEBROOT INC.

☐ Inventor or Joint Inventor (title not required below)
☐ Legal Representative of a Deceased or Legally Incapacitated Inventor (title not required below)
☑ Assignee or Person to Whom the Inventor is Under an Obligation to Assign (provide signer's title if applicant is a juristic entity)
☐ Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a petition under 37 CFR 1.46(b)(2) was granted in the application or is concurrently being filed with this document) (provide signer's title if applicant is a juristic entity)

SIGNATURE of Applicant for Patent

The undersigned (whose title is supplied below) is authorized to act on behalf of the applicant (e.g., where the applicant is a juristic entity).

Signature		Date (Optional)	May 12, 2014
Name	Ryan Howe		
Title	Associate General Counsel		

NOTE: Signature - This form must be signed by the applicant in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. If more than one applicant, use multiple forms.

☐ Total of forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/14 (11-08)
Approved for use through 04/30/2017. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	011765-0355986
	Application Number	11/477,807
Title of Invention	METHODS AND APPARATUS FOR DEALING WITH MALWARE	

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.
This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Applicant Information:

Applicant 1

Applicant Authority ◉ Inventor ○ Legal Representative under 35 U.S.C. 117 ○ Party of Interest under 35 U.S.C. 118

Prefix	Given Name	Middle Name	Family Name	Suffix
	Melvyn		Morris	

Residence Information (Select One) ○ US Residency ◉ Non US Residency ○ Active US Military Service

City	Derbyshire	Country Of Residence	UK
Citizenship under 37 CFR 1.41(b)	GB		

Mailing Address of Applicant:

Address 1	Redmire Gap, Intake Lane		
Address 2	Turnditch		
City	Derbyshire	State/Province	
Postal Code	DE56 2LU	Country	UK

Applicant 2

Applicant Authority ◉ Inventor ○ Legal Representative under 35 U.S.C. 117 ○ Party of Interest under 35 U.S.C. 118

Prefix	Given Name	Middle Name	Family Name	Suffix
	Paul		Stubbs	

Residence Information (Select One) ○ US Residency ◉ Non US Residency ○ Active US Military Service

City	Bedfordshire	Country Of Residence	UK
Citizenship under 37 CFR 1.41(b)	GB		

Mailing Address of Applicant:

Address 1	7 Rookery Road		
Address 2	Wyboston		
City	Bedfordshire	State/Province	
Postal Code	MK44 3AX	Country	UK

Applicant 3

Applicant Authority ◉ Inventor ○ Legal Representative under 35 U.S.C. 117 ○ Party of Interest under 35 U.S.C. 118

Prefix	Given Name	Middle Name	Family Name	Suffix
	Markus		Hartwig	

Residence Information (Select One) ○ US Residency ◉ Non US Residency ○ Active US Military Service

City	Buckinghamshire	Country Of Residence	UK

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	011765-0355986
	Application Number	11/477,807
Title of Invention	METHODS AND APPARATUS FOR DEALING WITH MALWARE	

Citizenship under 37 CFR 1.41(b)	DE			
Mailing Address of Applicant:				
Address 1	Exchange House, 494 Midsummer Boulevard			
Address 2	Milton Keynes			
City	Buckinghamshire	**State/Province**		
Postal Code	MK9 2EA	**Country**	UK	

Applicant 4

Applicant Authority (•) Inventor ( ) Legal Representative under 35 U.S.C. 117 ( ) Party of Interest under 35 U.S.C. 118

Prefix	Given Name	Middle Name	Family Name	Suffix
	Darren		Harter	

Residence Information (Select One) ( ) US Residency (•) Non US Residency ( ) Active US Military Service

City	Gloucestershire	**Country Of Residence**	UK
Citizenship under 37 CFR 1.41(b)	GB		
Mailing Address of Applicant:			
Address 1	178 Barnwood Road, Hucclecote		
Address 2	Gloucester		
City	Gloucestershire	**State/Province**	
Postal Code	GL4 3JZ	**Country**	UK

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button. Add

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below.
For further information see 37 CFR 1.33(a).

☐ **An Address is being provided for the correspondence Information of this application.**

Customer Number	~~00000~~ 23552		
Email Address		Add Email	Remove Email

Application Information:

Title of the Invention	METHODS AND APPARATUS FOR DEALING WITH MALWARE		
Attorney Docket Number	011765-0355986	**Small Entity Status Claimed**	☐
Application Type	Nonprovisional		
Subject Matter	Utility		
Suggested Class (if any)		**Sub Class (if any)**	
Suggested Technology Center (if any)			
Total Number of Drawing Sheets (if any)	3	**Suggested Figure for Publication (if any)**	1

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	011765-0355986
	Application Number	11/477,807
Title of Invention	METHODS AND APPARATUS FOR DEALING WITH MALWARE	

Publication Information:

☐	Request Early Publication (Fee required at time of Request 37 CFR 1.219)
☐	**Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32).
Enter either Customer Number or complete the Representative Name section below. If both sections are completed the Customer Number will be used for the Representative Information during processing.

Please Select One:	◉ Customer Number	○ US Patent Practitioner	○ Limited Recognition (37 CFR 11.9)
Customer Number	~~00909~~ 23552		

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, or 365(c) or indicate National Stage entry from a PCT application. Providing this information in the application data sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78(a)(2) or CFR 1.78(a)(4), and need not otherwise be made part of the specification.

Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)

Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button.

Foreign Priority Information:

This section allows for the applicant to claim benefit of foreign priority and to identify any prior foreign application for which priority is not claimed. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55(a).

Remove

Application Number	Country	Parent Filing Date (YYYY-MM-DD)	Priority Claimed
0513375.6	GB	2005-06-30	◉ Yes ○ No

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button.

Assignee Information:

Providing this information in the application data sheet does not substitute for compliance with any requirement of part 3 of Title 37 of the CFR to have an assignment recorded in the Office.

Assignee 1

PTO/SB/14 (11-08)
Approved for use through 04/30/2017. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	011765-0355986
	Application Number	11/477,807
Title of Invention	METHODS AND APPARATUS FOR DEALING WITH MALWARE	

If the Assignee is an Organization check here. ☒			
Organization Name	~~Prevx Limited~~ Webroot Inc.		
Mailing Address Information:			
Address 1	~~3 New Park Place~~ 385 Interlocken Crescent		
Address 2	~~Pride Park~~		
City	~~Derby~~ Broomfield	**State/Province**	CO
Country	~~UK~~ USA	Postal Code	~~DE24 8DZ~~ 80021
Phone Number		Fax Number	
Email Address			
Additional Assignee Data may be generated within this form by selecting the **Add** button.			

Signature:

A signature of the applicant or representative is required in accordance with 37 CFR 1.33 and 10.18. Please see 37 CFR 1.4(d) for the form of the signature.					
Signature	/Andrew T. Pouzeshi/			Date (YYYY-MM-DD)	2017-01-11
First Name	Andrew T.	Last Name	Pouzeshi	Registration Number	67672

This collection of information is required by 37 CFR 1.76. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

PTO/SB/96 (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(b)

Applicant/Patent Owner: WEBROOT INC.

Application No./Patent No.: 8,418,250 Filed/Issue Date: 04-09-2013

Titled: METHODS AND APPARATUS FOR DEALING WITH MALWARE

WEBROOT INC. , a Corporation

(Name of Assignee) (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.

states that it is:

1. ☒ the assignee of the entire right, title, and interest in;

2. ☐ an assignee of less than the entire right, title, and interest in (The extent (by percentage) of its ownership interest is _____ %); or

3. ☐ the assignee of an undivided interest in the entirety of (a complete assignment from one of the joint inventors was made)

the patent application/patent identified above, by virtue of either:

A. ☐ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel _____, Frame _____, or for which a copy therefore is attached.

OR

B. ☒ A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

1. From: Morris et al. To: PREVX LIMITED

The document was recorded in the United States Patent and Trademark Office at
Reel 018348 , Frame 0840 , or for which a copy thereof is attached.

2. From: PREVX LIMITED To: WEBROOT SOLUTIONS LTD

The document was recorded in the United States Patent and Trademark Office at
Reel 032971 , Frame 0778 , or for which a copy thereof is attached.

3. From: WEBROOT SOLUTIONS LTD To: WEBROOT INC.

The document was recorded in the United States Patent and Trademark Office at
Reel 037886 , Frame 0085 , or for which a copy thereof is attached.

☐ Additional documents in the chain of title are listed on a supplemental sheet(s).

☒ As required by 37 CFR 3.73(b)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[NOTE: A separate copy (*i.e.*, a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

/Andrew T. Pouzeshi/ 2017-01-11

Signature Date

Andrew T. Pouzeshi Attorney for Applicant

Printed or Typed Name Title

This collection of information is required by 37 CFR 3.73(b). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Acknowledgement Receipt

EFS ID:	28031317
Application Number:	11477807
International Application Number:	
Confirmation Number:	4785
Title of Invention:	METHODS AND APPARATUS FOR DEALING WITH MALWARE
First Named Inventor/Applicant Name:	Melvyn Morris
Customer Number:	22442
Filer:	Andrew T. Pouzeshi/Elizabeth Pallardy
Filer Authorized By:	Andrew T. Pouzeshi
Attorney Docket Number:	6890-93
Receipt Date:	11-JAN-2017
Filing Date:	30-JUN-2006
Time Stamp:	11:48:20
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	04584_0116US01_POA.PDF	352758 ee3e3f7cb57a9a4c39c6d819f5a2d8d2e4405145	no	2

Warnings:

Information:					
2	Application Data Sheet	04584_0116US01_SuppADS.PDF	964889 9ff9fa32f414f53b71bb5166559cb374318ee325	no	5
Warnings:					
Information:					
This is not an USPTO supplied ADS fillable form					
3	Assignee showing of ownership per 37 CFR 3.73	04584_0116US01_373.PDF	429378 62d28cf4c8f16d1f41d63d0d89da8005345782ac	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	1747025		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/477,807	06/30/2006	Melvyn Morris	6890-93

CONFIRMATION NO. 4785

POWER OF ATTORNEY NOTICE

22442
Sheridan Ross PC
1560 Broadway
Suite 1200
Denver, CO 80202

OC000000088575425

Date Mailed: 01/23/2017

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 01/11/2017.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/ldvan/

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
11/477,807	06/30/2006	Melvyn Morris	04584.0116US01

CONFIRMATION NO. 4785

23552
MERCHANT & GOULD P.C.
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

POA ACCEPTANCE LETTER

OC000000088575457

Date Mailed: 01/23/2017

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 01/11/2017.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/ldvan/