01/04/03

10336299.010303

A

JC836 U.S. PTO 01/03/03

JC996 U.S. PTO 10/336299 01/03/03

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new non-provisional applications under 37 CFR 1.53(b))

Title of Invention	**System and Method for the Managed Security Control of Processes on a Computer System**
Named Inventor(s)	**Thomas James Satterlee and William Franko Hackenberger**
Attorney Docket	**05456.105038 US**
Express Mail Label No.	**EU 468794582 US**

APPLICATION ELEMENTS

1. ☒ Fee Transmittal Form
   (Submit an original, and a duplicate for fee processing)
2. ☐ Applicant claims small entity status. See 37 CFR 1.27.
3. ☒ Specification, Claims, and Abstract Total Pages **28**
4. ☒ Drawings Total Sheets **8**
5. Oath or Declaration Total Pages **4**
   a. ☒ 1 Newly executed; 1 unexecuted (original)
   b. ☐ Copy from prior application (37 CFR 1.63(d))
   (for continuation/divisional with Box 17 completed)
   (i) ☐ DELETION OF INVENTOR(S)
   Signed statement attached deleting inventor(s) named 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)*
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. ☐ Statement verifying identity of above copies

ADDRESS TO: Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

ACCOMPANYING APPLICATION PARTS

9. ☐ Assignment Papers (cover sheet & document(s))
10. ☐ 37 CFR 3.73(b) Statement *(when there is an assignee)*
    ☐ Power of Attorney by assignee
11. ☐ English Translation Document *(if applicable)*
12. ☒ Information Disclosure Statement (IDS) PTO-1449
    ☒ Copies of IDS Citations
13. ☐ Preliminary Amendment
14. ☒ Return Receipt Postcard (MPEP 503)
    (Should be specifically itemized)
15. ☐ Certified Copy of Priority Document(s)
16. ☒ Assignee: Internet Security Systems
    6303 Barfield Road
    Atlanta, GA 30328
19. ☐ Other: _____

17. If a **CONTINUING APPLICATION**, *check appropriate box and supply the requisite information*:
☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No:
Prior application information: Examiner: Group/Art Unit:

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 5b, is considered as being part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

18. CORRESPONDENCE ADDRESS:

Robert T. Neufeld, Esq.
KING & SPALDING LLP
45th Floor
191 Peachtree Street, N.E.
Atlanta, Georgia 30303

By: Robert T. Neufeld Reg. No. 48,394
Date: January 3, 2003
Telephone: 404.572.4600
Facsimile: 404.572.5145
Customer No. 20786

[redacted]

10336299.010303

FEE TRANSMITTAL

Attorney Docket No. **05456.105038 US**
Express Mail Label No. **EU 468794582 US**

This sheet accompanies a patent application transmittal for the following application:

Inventor(s): **Thomas James Satterlee and William Frank Hackenberger**
Filing Date: **January 3, 2003**
Title: **System and Method for the Managed Security Control of Processes on a Computer System**

The filing fee is calculated as shown below:

1. **FILING FEE:**

	SMALL ENTITY		LARGE ENTITY	
FOR:	FEE	FEE PAID	FEE	FEE PAID
☒ UTILITY FILING FEE	**$370**		**$740**	**740.00**
☐ DESIGN FILING FEE	**$165**		**$330**	
☐ PLANT FILING FEE	**$255**		**$510**	
☐ REISSUE FILING FEE	**$370**		**$740**	
☐ PROVISIONAL FILING FEE	**$80**		**$160**	
SUBTOTAL (1)	**$xxx**			**$740.00**

2. **CLAIMS:**

			SMALL ENTITY		LARGE ENTITY	
FOR:	NO. FILED	NO. EXTRA	RATE	FEE	RATE	FEE
TOTAL CLAIMS	40 - 20 =	20	x 9 =		**20 x 18 =**	**360.00**
INDEP. CLAIMS	5 - 3 =	2	**x 42 =**		**2 x 84 =**	**168.00**
☐ MULTIPLE DEPENDENT CLAIM PRESENTED			**+140 =**		**+280 =**	
		SUBTOTAL (2)	**$xxx**			**$528.00**

3. **ADDITIONAL FEES:**

	SMALL ENTITY		LARGE ENTITY	
FOR:	FEE	FEE PAID	FEE	FEE PAID
☐ LATE FILING, FEE OR OATH	**$65**		**$130**	
☐ NON-ENGLISH SPECIFICATION	**$130**		**$130**	
☐ OTHER				
SUBTOTAL (3)	**$xxx**			**$xxx**

TOTAL FILING FEES: $1,268.00

A check is enclosed for the total amount: **$1,178.00**

☒ Charge any additional fees required under 37 C.F.R. 1.16 or 1.17 to Deposit Account 11-0980.

KING & SPALDING LLP
45th Floor, 191 Peachtree Street, N.E.
Atlanta, Georgia 30303
Telephone: 404.572.4600

By: _____ Date: January 3, 2003
Robert T. Neufeld, Attorney for Applicant
Reg. No. 48,394

[redacted]

SYSTEM AND METHOD FOR THE MANAGED SECURITY CONTROL OF PROCESSES ON A COMPUTER SYSTEM

PRIORITY AND RELATED APPLICATIONS

The present application claims priority to and incorporates herein provisional patent application entitled, "System and Method for the Managed Security Control of Processes on a Computer System," filed on January 4, 2002 and assigned U.S. Application Serial Number 60/345,432.

TECHNICAL FIELD

The present invention is generally directed to managing the security of a network. More specifically, the present invention provides kernel-level protection of a computer system from rogue or malicious computer programs.

BACKGROUND OF THE INVENTION

The security of computing networks is an increasingly important issue. With the growth of wide area networks (WANs), such as the Internet and the World Wide Web, people rely on computing networks to locate, transfer, and store an increasing amount of valuable information. This is also true of local area networks (LANs) used by companies, schools, organizations, and other enterprises. LANs generally are used by a bounded group of people in an organization to communicate and store electronic documents and information. LANs typically are coupled to or provide access to other local or wide area networks. Greater use and availability of computing networks produces a corresponding increase in the size and complexity of computing networks.

With the growth of networks and the importance of information available on the networks, there is also a need for better and more intelligent security. One approach to securing larger and more complex computer networks is to use a greater number and variety of security assessment and intrusion detection devices. Security assessment devices can be used to evaluate elements in the network such as desktop computers, servers, and routers, and determine their respective

vulnerability to attack from hackers. Intrusion detection devices, on the other hand, identify and prevent entry of foreign or malicious computer programs and can notify a network manager of the presence or attempted entry of such a computer program. Security assessment and intrusion detection devices can also be used more frequently to monitor the activity or status of the elements in a computing network.

However, simply adding devices or filters to a network is not always the only and best solution to maintaining network security. Adding security devices can complicate the network and inundate the network manager with security data. Threats to the security of a device or network can take a variety of forms including the introduction of harmful computer code, unauthorized attempts to gain access, and misuse by people with authority to use a device or network. The various types of harmful computer code that can threaten a computing device or distributed computing system can generally be categorized as either a virus or some form of "malware". Computer viruses harm computing devices and systems by entering and then propagating. In some respects, the propagating nature of computer viruses makes them easier to detect and there are many commercially available products that detect and exclude viruses from computing devices and systems.

In contrast, malware is a general description for other types of programs and computer code that are designed to harm a computing device or system in ways other than simply propagating as a virus does. Malware presents a more sophisticated challenge for network security and traditional anti-virus software is not designed to prevent malware from harming computing devices and networks. Malware can take a variety of forms including corrupted applications and applications that retrieve corrupted code that is modified to harm a computing device or network.

There are generally two different approaches to protecting against malware. The first approach involves virtual execution of a computer code to attempt to identify harmful code before it is allowed to actually execute. Virtual execution is limited in its ability to detect harmful code because it does not

IPR2024-00027
CrowdStrike Exhibit 1002 Page 4

actually execute every process of the code. Instead, the virtual execution technique performs a quick and high-level "walk through" of the processes in the code to attempt to detect suspicious patterns in the code. By its nature, the virtual execution technique is limited in its ability to detect suspicious activities embedded in a piece of code. As a result, when the virtual execution technique is implemented, it must be used conservatively which produces a high number of false positive alerts. In other words, because the virtual execution security technique is not as accurate as actually running the code, it is implemented to identify a broader scope of potentially suspicious code and produces a greater number of alerts to the user. A high percentage of false positive security alerts is undesirable because it translates into a greater number of security interruptions for the user.

The second approach involves controlling and monitoring a computing device in real time while it is actually running a program and attempting to anticipate any harmful activity the program may try to initiate. One example of a real-time solution is set forth in U.S. Patent No. 5,987,611, which describes a client-based monitoring system for filtering network access in conjunction with a centralized enforcement supervisor. The supervisor maintains access rules for the client-based filtering and verifies the existence and proper operation of the client-based filter application. Access rules specify network access criteria for a client, such as (1) total time a user can be connected to the Internet (*e.g.*, per day, week, month, or the like), (2) time a user can interactively use the Internet (*e.g.*, per day, week, month, or the like), (3) a list of applications or application versions that a user can or cannot use in order to access the Internet, (4) a list of URLs (or WAN addresses) that a user application can (or cannot) access, (5) a list of protocols or protocol components that a user application can or cannot use, and (6) rules to determine what events should be logged (including how long are logs to be kept).

By intercepting process loading and unloading and keeping a list of currently-active processes, each client process can be checked for various characteristics, including checking executable names, version numbers, executable file checksums, version header details, configuration settings, and the like. With

IPR2024-00027
CrowdStrike Exhibit 1002 Page 5

this information, a determination can be made whether a particular process in question should have access to the Internet and what kind of access (*i.e.*, protocols, Internet addresses, time limitations, and the like) is permissible for the given specific user.

The limitation with the solution presented in U.S. Patent No. 5,987,611 and other similar real-time prior art solutions is that they are packet based. In other words, the security decisions are based on the data packets that are passing between the computing device that is being monitored and external networks or computing resources. When security decisions are based on the traffic of data packets, it is more likely the security systems will not detect harmful activities until after the harm has already begun. Accordingly, the second approach is not satisfactory because conventional real-time security monitoring solutions do not detect security problems early enough and allow time for a response before the malicious program does harm.

In view of the foregoing, there is a need in the art for a security system which will provide early detection of security threats to a computing device or network before any harm can be done. Specifically, a need exists to be able to quickly and efficiently examine code in real time, but before it is able to harm a computing device or system. A further need exists for a computer security system that can accurately identify security threats contained in software programs so that users are not interrupted frequently to address potential security questions. Finally, a security system is needed that can efficiently and effectively respond to security threats detected in software programs.

SUMMARY OF THE INVENTION

The present invention satisfies the above-described needs by evaluating and monitoring software programs running on a computing device. The invention uses a protector system that comprises several different software modules for performing evaluation, detection, monitoring, and response functions. Implementing a two-phased approach at the kernel level of the computing device's operating system, the present invention provides fast and efficient

IPR2024-00027
CrowdStrike Exhibit 1002 Page 6

security that minimizes interruptions for the user. The first phase, the pre-execution process, performs a rapid validation check to determine whether the program has been approved for the computing device or network. If the program is validated, it can be run without further monitoring or interruptions for the user. If the program is not validated, it can be monitored at the kernel level of the operating system during the second phase, while the program is executing. Detection and monitoring modules of the present invention can identify suspicious activities triggered by the program and monitor the activities before they are able to cause harm to the computing device or network. In the event the program is initiating harmful activities, the protector software module can respond by taking remedial action to address the threat.

In one aspect, the present invention comprises a method for determining whether a program is approved to execute by comparing it to a predetermined list of approved programs. The operating system kernel notifies a pre-execution monitoring module when a new program begins to load so that it can be validated before execution. A validation module can compare the new program to the predetermined list of approved programs. If the new program is validated, the pre-execution monitoring module allows the operating system to continue loading and executing the program. Once a program is validated in the pre-execution phase, little or no additional security monitoring needs to be performed on the new program while it is executing. If the new program is not validated, the program can continue to load and execute, but other execution security modules are responsible for detecting, monitoring, and responding to suspicious activities. For example, the execution security modules can control access to certain files or registry settings, or limit network access. The execution security modules can also consider whether a new program was previously permitted to execute on the computing device.

In another aspect, the present invention provides a protector system for improving and expediting security on a computing device or network. The protector system comprises several software modules coupled to the operating system kernel of the computing device that manage and control activities at the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 7

kernel level. The protector system comprises a pre-execution monitoring component that can suspend a new program as it is loading into memory and before it can execute. The pre-execution monitoring component can operate with a validation module to determine whether the user has already validated the new program. If the pre-execution component validates the new program, it can continue to load and execute with minimal security concerns. However, if the pre-execution component is unable to validate the new program, execution security modules can perform additional monitoring while the new program is executing. Execution security modules can intercept various operating system triggers and calls before they are executed to determine if the activity is suspicious. If the program's activities are deemed suspicious or malicious, the execution security modules can respond by terminating the activities or taking other responsive measures to protect the security of the computing device or network.

These and other aspects of the invention will be described below in connection with the drawing set and the appended specification and claim set.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating an exemplary architecture for operating an embodiment of the present invention.

FIG. 2 is a block diagram illustrating components of a database implemented in accordance with an exemplary embodiment of the present invention.

FIG. 3 is a logic flow diagram illustrating a setup process for implementing the protector system in accordance with an exemplary embodiment of the present invention.

FIGs. 4A and 4B are logic flow diagrams illustrating a pre-execution process using the protector system in accordance with an exemplary embodiment of the present invention. FIG. 5 is a logic flow diagram illustrating a validation process using the protector system in accordance with an exemplary embodiment of the present invention.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 8

FIG. 6 is a logic flow diagram illustrating a non-validated execution process using the protector system in accordance with an exemplary embodiment of the present invention.

FIG. 7 is a logic flow diagram illustrating a file protection process using the protector system in accordance with an exemplary embodiment of the present invention.

DETAILED DESCRIPTION OF THE EXEMPLARY EMBODIMENTS

The present invention employs a protector system comprising several software modules to support the protection of computing devices and computing networks from malicious software programs. Specifically, the present invention employs a two-step process to validate software programs and monitor non-validated software programs. The two-step process provides an efficient and effective means for protecting a computing device or network while minimizing disruptions for the user. In the first phase of the process, the protector system validates authorized programs to ensure that they have not been corrupted before running them. For programs that cannot be validated, the protector system can monitor the programs as they execute during the second phase. If during the monitoring step, the program initiates any suspicious activities, the protector system can respond by taking one or more remedial actions.

Although the exemplary embodiments will be generally described in the context of software modules running in a distributed computing environment, those skilled in the art will recognize that the present invention also can be implemented in conjunction with other program modules for other types of computers. In a distributed computing environment, program modules may be physically located in different local and remote memory storage devices. Execution of the program modules may occur locally in a stand-alone manner or remotely in a client/server manner. Examples of such distributed computing environments include local area networks of an office, enterprise-wide computer networks, and the global Internet.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 9

The detailed description that follows is represented largely in terms of processes and symbolic representations of operations in a distributed computing environment by conventional computer components, such as database servers, application servers, routers, security devices, firewalls, clients, workstations, memory storage devices, display devices and input devices. Each of these conventional distributed computing components is accessible via a communications network, such as a wide area network or local area network.

The processes and operations performed by the computer include the manipulation of signals by a client or server and the maintenance of these signals within data structures resident in one or more of the local or remote memory storage devices. Such data structures impose a physical organization upon the collection of data stored within a memory storage device and represent specific electrical or magnetic elements. These symbolic representations are the means used by those skilled in the art of computer programming and computer construction to most effectively convey teachings and discoveries to others skilled in the art.

The present invention also includes computer programs that embody the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the invention in computer programming, and the invention should not be construed as limited to any one set of computer program instructions. Further, a skilled programmer would be able to write such a computer program to implement the disclosed invention based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions is not considered necessary for an adequate understanding of how to make and use the invention. The inventive functionality of the claimed computer program will be explained in more detail in the following description in conjunction with figures illustrating the program flow.

Referring now to the drawings, in which like numerals represent like elements throughout the several figures, aspects of the present invention and the preferred operating environment will be described.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 10

Figure 1 illustrates various aspects of an exemplary computing environment in which an embodiment of the present invention is designed to operate. Those skilled in the art will appreciate that Figure 1 and the associated discussion are intended to provide a representative description of the computer components in an exemplary protector system.

Referring to Figure 1, an exemplary architecture 101 is illustrated for computing device 103. The exemplary computing device 103 is divided into two general regions referred to as the user space 105 and the kernel space 107. The kernel space 107 refers to the central part of the operating system 180. The kernel space 107 typically represents that portion of the operating system 180 that directly accesses the hardware of the computing device 103. In contrast, the user space 105 represents portions of the computing device 103 that interact with software and data received from outside the computing device 103. The exemplary architecture 101 shown in Figure 1, illustrates the components of an exemplary protector system 104 that operates to protect the computing device 103 from rogue or malicious software.

Beginning with the user space 105, the system management module 125 manages security operations on the computing device 103 and can coordinate the functions of the protector system with other security devices that may be coupled to or operating on the computing device 103. If the computing device is coupled to a network 102, as shown in the exemplary architecture in Figure 1, the system management module 125 can also be used to coordinate security settings and responses with other components on the network 102. The system management module 125 can also comprise a list of programs and processes that are allowed to run on the computing device 103.

In the user space 105, the command line interface 120 and the protector application 115 are coupled to the system management module 125. The command line interface 120 is typically implemented as a wrapper around the protector API library 117. The primary purpose of the command line interface 120 is to configure various security settings, such as how to respond to a certain threat, and to load the settings into the protector driver management interface 145.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 11

The protector application 115 communicates with the protector driver management interface 145 via the API library 117 and provides user-level services in the protector system 104. The protector application 115 can provide the initial configuration load when the protector system 104 is initialized and can interact with a user on certain security decisions. The protector application 115 also interacts with database 110. The database 110 can comprise various data used in performing protector security functions, which will be described in greater detail in connection with Figure 2.

Communication between the user space 105 and the kernel space 107 is performed with the API library 117 and the protector driver management interface 145. The API library is preferably implemented as a C level application programming interface that allows various processes to maintain the data in database 110 and provide instructions to the drivers in the kernel space 107.

The binary execution monitor 125 implements the primary functions of the protector system 104. The binary execution monitor 125 is an in-kernel driver that monitors the loading of binary executable files and other executable libraries in real time by recognizing and validating any executable file that is being loaded. Validation of an executable file is implemented with the validity module 108 and can be performed using a variety of techniques described in greater detail herein. When the binary execution monitor 125 is installed, it establishes the presence of its processing at the system process-creation hooks 170 within the kernel where it can observe all process and creation activities. When a program is initially loaded into memory in anticipation of execution, the binary execution monitor 125 works with the validity module 108 to validate the program. Using the system process-creation hooks 170, the binary execution monitor 125 can recognize the initial loading of executable files, and any subsequent loading of executable libraries that can occur while a program is executing. The binary execution monitor's 125 functions performed prior to execution of an executable file can also be generally described as being performed by a pre-execution module of the protector system 104.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 12

If the binary execution monitor 125 is unable to validate a program, the detect drivers 153 can monitor the non-validated program when it is executing and identify potential threats to the computing device 103 before they are executed. The detect drivers 153 are plug-in modules linked to the system call hooks 175 within the kernel. The detect drivers 153 communicate system call hooks, using component interface 150, to associated behavior monitoring modules 128 that can react to the suspicious activities. The binary execution monitor 125 also works in conjunction with the behavior monitoring modules 128 to analyze and respond to system call hooks communicated from the detect drivers 153. The behavior monitoring modules 128 are a collection of in-kernel modules that are associated with the detect drivers 153. For example, the privacy monitor 130 reacts to a program's attempt to use a network connection to other computer systems. The file protection monitor 135 can react to an attempt to alter a specified file. The registry protection monitor 140 protects against unauthorized changes to registry settings. Generally, the behavior monitors 128 can take direct action to address a security threat or instruct the protector application 115 to query the user for instructions on how to handle the threat. Other behavior monitors 128 and their associated detect drivers 153 can be plugged into the protector system 104 to implement different security functions. Furthermore, those skilled in the art will understand that the architecture 101 of the protector system 104 shown in Figure 1 is an example and that various components can be located external to the computing device 103 or on the network 102 in other embodiments. The functions of the binary execution monitor 125, the detect drivers 153, and the behavior monitors 128 performed during execution of an executable file can generally be referred to as being performed by an execution module of the protector system 104.

Figure 2 illustrates components of the database 110 in accordance with an exemplary embodiment of the protector system 104. The exemplary database 110 comprises checksum data 205, user action data 210, configuration settings 215, and file lock data 220. The checksum data 205 comprises data for each executable that is permitted to run on the computing device 103. The checksum

IPR2024-00027
CrowdStrike Exhibit 1002 Page 13

data 205 is used to validate executable files before they are run on the computing device 103. The user action data 210 comprises a record of the user's responses to the introduction of a new executable on the computing device 103. For example, an executable being loaded onto the computing device 103 may not be on a list of allowed programs. If this executable has previously been loaded onto the computing device 103 and the user has been queried as to whether or not the executable is allowed to run, the user's response to this query can be stored in the user action data 210. The configuration settings 215 comprise settings that can be controlled by the user for determining how the protector system 104 will respond to new or suspicious programs being loaded onto the computing device 103. The file lock data 220 represents files selected by the user or network administrator that are to be restricted from access by programs running on the computing device 103. As illustrated in greater detail in the discussion associated with Figure 7, the file protection monitor 140 can use the file lock data 220 to protect certain files from being accessed.

Referring to Figure 3, an exemplary setup process 300 is illustrated. The setup process 300 is typically performed prior to the protector system 104 operating on the computing device 103. In step 305 of exemplary process 300, the user stores a list of allowed executable files in the system management module 125. This list of allowed executable files is associated with programs that are approved to run on the computing device 103. The list of allowed executable files may be defined by the user or by a network manager if the computing device 103 is coupled to a network, such as in a workplace environment. In step 310, the user can set access rights for each of the allowed executable files. The access rights for the allowed executable files define which components of the computing device 103, or the network to which the computing device is coupled, may be accessed by that executable. Defining access rights can also include defining which files are restricted from access.

Steps 315 and 320 provide specific examples of validation steps conducted during the setup process using the validity module 108. The binary execution monitor 125 works with the validity module 108 to validate each of the allowed

IPR2024-00027
CrowdStrike Exhibit 1002 Page 14

executable files. The validation module 108 can be any one of a variety of pieces of software that are used to verify that a program has not been tampered with. For example, the validation module can represent the MD5 software module commonly known to those in the art. The MD5 software module calculates a checksum for each allowed program and that checksum is stored for later comparison. As described in connection with Figure 2, the checksum data 205 can be stored in database 110 in step 320.

Finally, in step 325 of the setup process 300, the configuration settings chosen by the user are stored in the database 110. The configuration settings can be chosen by the user of the computing device 103 or by a network administrator if the computing device 103 is coupled to a network. The configuration settings can include predetermined responses to particular threats and decision rules as to when the user should be queried about a security threat.

The subject matter of the remaining drawings, Figures 4A, 4B, 5, 6 and 7, generally can be categorized in two phases of execution of a program. The first phase, called pre-execution, occurs as a program is being loaded into memory, but before it can execute. The second phase is the execution process for the program. One advantage of the protector system 104 is that it performs the majority of the security decision-making in the pre-execution process illustrated in Figures 4A, 4B and 5. By moving much of the security decision-making to the pre-execution process, the protector system 104 enables the execution process, represented by Figures 6 and 7, to run more smoothly and with fewer interruptions for the user.

Referring to Figure 4A, an exemplary process 400 is illustrated for performing the steps in the pre-execution phase for an executable file. Beginning with step 405, the user, or another software module, may attempt to load a new executable file for running on the computing device 103. In step 410, the kernel begins loading the new executable file into memory in anticipation of running the program. As the executable file is loading, a system process-creation hook 170 notifies the binary execution monitor 125 of the loading process. By initiating the monitoring process at the kernel level, the protector system 104 is able to begin performing its functions before the program can execute and cause possible harm.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 15

In step 420, the kernel will determine whether this executable is already running on the computing device 103. If in fact the executable is running, the "yes" branch is followed to step 425 and the executable will not be loaded. If the executable is already running, it has been approved previously and the protector system 104 can skip the validation process described in connection with Figure 4B. If however, the executable has not already been loaded, the binary execution monitor 125 will respond to the system process-creation hook 170 by suspending execution of the executable file, in step 430, until the program can be validated.

The pre-execution process performed by the binary execution monitor 125 and the other associated components of the protector system 104 supports an initial determination of whether the new executable is safe for loading onto the computing device 103. Continuing with Figure 4B, step 435 illustrates a representative step for validating the new executable. Step 435 will be described in greater detail in one exemplary embodiment in the discussion in connection with Figure 5 below. In step 440, if the binary execution monitor 125 is able to validate the new executable, the suspended state will be released in step 445 and loading of the executable will continue in step 450. However, if the binary execution monitor 125 is unable to validate the executable in the pre-execution process, additional precautionary steps will have to be taken in the execution phase as described in greater detail in connection with Figures 6 and 7.

Referring to Figure 5, the exemplary process referred to in step 435 is illustrated in greater detail. As mentioned earlier, the checksum technique illustrated in Figure 5 is only one example of a method for validating an executable. Other validation techniques can use various software modules to analyze the behavior or characteristics of certain executable files in order to validate them. In step 510, the binary execution monitor 125 retrieves the checksum data that was previously calculated and stored in database 110. In step 515, the binary execution monitor 125 calculates a checksum for the new executable which is being loaded on the computing device 103. If the executable is associated with an allowed program, and the program has not been corrupted, the binary execution monitor 125 should find a match between the data contained

IPR2024-00027
CrowdStrike Exhibit 1002 Page 16

in the checksum data 205 and the checksum calculated for the new executable. If the binary execution monitor 125 finds a match with the checksum data, the new executable will be found to be valid in step 525. If the program associated with the new executable is not one of the allowed programs designated during the setup process 300, then the executable is found to be not valid in step 530. Additionally, if the executable corresponds to an allowed program but the program has in someway been corrupted, the checksum calculated for the new executable will not match the previously calculated checksum and the new executable is found to be not valid in step 530.

As mentioned above, the exemplary processes illustrated in Figure 6 and Figure 7 occur after the pre-execution process and concern executable files that the binary execution monitor 125 could not not validate in the pre-execution phase. Referring to Figure 6, an exemplary process 600 is illustrated for executing an executable file that has not been validated. Once the pre-execution process 400 terminates and the suspended state is lifted, in step 605 the protector application 115 will consult the database 110 to determine if the user has been previously queried about the new non-validated executable file. If the user has been previously queried about this executable and the previous decision was to not allow this executable file to load, the executable will be terminated in step 620 without interrupting the user for a decision. If the user has previously approved the loading of this executable file in step 610, or, if the user approves the new executable in this instance in step 615, then execution of the executable file will proceed.

Although the user has allowed the non-validated executable file to proceed, the protector system 104 will take steps to protect the computing device 103 and the network 102 that it may be connected to. Steps 625 through 645 illustrate exemplary processes that may be performed in allowing the non-validated executable file to proceed. In step 625, the protector application 115 will notify the system management module 125 that the non-validated executable file is allowed to execute. This notification will serve to allow the system management module 125 to take any precautions, in step 630, to protect other

IPR2024-00027
CrowdStrike Exhibit 1002 Page 17

components of the computing device 103 or other network components coupled to the computing device. In step 635, the binary execution monitor 125 will release the suspended state for the new executable and the program will continue to load and execute in step 640.

As the program is executing, the other components of the protector system 104, such as the detect drivers 153 and the behavior monitoring modules 128, operate to prevent the non-validated program from performing any malicious activities on the computing device 103 or on the network 102. The detect drivers 153 are linked to the kernel activities through system call hooks 175. In step 645, certain activities performed by the program will trigger system call hooks that, in turn, trigger the detect drivers 153. The detect drivers 153 are then coupled to the behavior monitoring modules 128, which can observe the program's behavior and respond to any malicious activity. An exemplary process 645 for triggering a detect driver 153 is illustrated in greater detail in Figure 7.

Referring to Figure 7, an exemplary detection process 645 is illustrated that employs the file input/output detector 160. The exemplary process 645 is illustrative of the operations of detect drivers 153 and their associated behavior monitors 128. In step 705, the executable file attempts to open a file in connection with a process or activity it is performing. As the kernel attempts to follow the instruction of the executable and open the file, the system call hook 175 linked to this activity triggers the file input/output detector 160 in step 710. The file input/output detector 160, in turn, notifies the file protection monitor 135 in step 715.

Any files that have restricted access, as determined by the user or network administrator in the setup process 300, will be identified in the file lock data in database 210. The file protection monitor 135 consults the file lock data in step 720 to determine whether the subject file has been restricted. If the file is not restricted, the file protection monitor 135 permits the system call hook 175 to proceed with opening the file. However, if the file does appear in the file lock data 220 in step 735, the file protection monitor can limit access to the file. For

IPR2024-00027
CrowdStrike Exhibit 1002 Page 18

instance, the file protection monitor can provide read-only access to a file or can prohibit access entirely.

In other examples of how the behavior monitors function, the type of response can depend on the type of activity that is detected as well as the configuration settings 215 selected by the user or the network administrator. For instance, if the program is attempting to perform functions that may seriously impair the computing device 103 or the network 102, the protector system 104 may immediately terminate execution of the program. Alternatively, if the file protection monitor 135 determines that the threat is less severe, the protector application 115 may simply query the user to insure that it is safe to continue executing the program.

In conclusion, the present invention enables and supports security from malicious software programs for a computing device or computing network. The two-step process of the protector system provides an effective and efficient method for implementing security while minimizing the burdens and interruptions for the user. The pre-execution process provides an efficient method for determining whether an uncorrupted program is allowed to execute. By validating certain programs during the pre-execution process, the protector system minimizes the amount of work that must be done in monitoring and controlling programs during the execution phase. The validation step also reduces the number of false positive alarms, thereby reducing security interruptions for the user.

It will be appreciated that the present invention fulfills the needs of the prior art described herein and meets the above-stated objects. While there has been shown and described the preferred embodiment of the invention, it will be evident to those skilled in the art that various modifications and changes may be made thereto without departing from the spirit and the scope of the invention as set forth in the appended claims and equivalence thereof. Although the present invention has been described as operating on a computing device coupled to a network, it should be understood that the invention can be applied to other types of distributed computing environments. Furthermore, it should be readily

IPR2024-00027
CrowdStrike Exhibit 1002 Page 19

apparent that the components of the protector system can be located in various local and remote locations of a distributed computing environment.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 20

CLAIMS

What is claimed is:

1. A computer-implemented method for implementing security for a computing device, comprising the steps of:

identifying an allowed program that is permitted to execute on the computing device;

receiving a signal that a new program is going to be executed on the computing device;

suspending the execution of the new program on the computing device;

determining whether the new program is the same as the allowed program;

if the new program is the same as the allowed program, permitting the new program to execute on the computing device; and

if the new program is not the same as the allowed program, monitoring the new program while allowing it to execute on the computing device.

2. The method of Claim 1, wherein the step of receiving a signal that a new program is going to be executed is performed at the operating system kernel of the computing device.

3. The method of Claim 1, wherein the step of monitoring the new program while allowing it to execute is performed at the operating system kernel of the computing device.

4. The method of Claim1, wherein the step of determining whether the new program is the same as the allowed program comprises:

computing a checksum for the new program, and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 21

comparing the new program's checksum to a checksum for the allowed program.

5. The method of Claim 1, wherein the step of determining whether the new program is the same as the allowed program comprises determining whether the new program has been modified.

6. The method of Claim 1, wherein the step of monitoring the new program while allowing it to execute comprises controlling the files the program attempts to access during execution of the new program on the computing device.

7. The method of Claim 1, wherein the step of monitoring the new program while allowing it to execute comprises controlling the registry settings the new program attempts to access during execution of the new program on the computing device.

8. The method of Claim 1, wherein the step of monitoring the new program while allowing it to execute comprises controlling the network activity of the new program.

9. The method of Claim 1, wherein the step of monitoring the new program while allowing it to execute comprises determining whether a user has previously approved operation of the new program.

10. The method of Claim 1, further comprising the step of terminating the execution of the new program if the program performs suspicious activities.

11. The method of Claim 1, further comprising the step of taking remedial measures to protect the security of other computing device components if the new program is not the same as the allowed program.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 22

12. A computer-readable medium having computer-executable instructions for performing the steps recited in Claim 1.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 23

13. A computer-implemented method for implementing security for a computing device comprising the steps of:

interrupting the loading of a new program for operation with the computing device;

validating the new program;

if the new program is validated, permitting the new program to continue loading and to execute in connection with the computing device;

if the new program is not validated, monitoring the new program while it loads and executes in connection with the computing device.

14. The method of Claim 13, wherein the step of interrupting the loading of a new program comprises:

intercepting a signal from the computing device's operating system that the new program is loading, and

suspending the loading of the new program.

15. The method of Claim 13, wherein the step of monitoring the new program comprises intercepting a signal from the computing device's operating system.

16. The method of Claim 13, wherein the step of validating the new program comprises determining whether the new program corresponds with an approved program.

17. The method of Claim 13, wherein the step of validating the new program comprises comparing a checksum for the new program with a previously determined checksum.

18. The method of Claim 13, wherein the step of validating the new program comprises analyzing characteristics of the new program.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 24

19. The method of Claim 13, wherein the step of monitoring the new program comprises controlling the files the new program attempts to access during execution of the new program.

20. A computer-readable medium having computer-executable instructions for performing the steps recited in Claim 13.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 25

21. A computer-implemented method for performing security for a computing device during a pre-execution phase comprising the steps of:

identifying an allowed program that is permitted to execute with the computing device;

receiving a signal that a new program is being loaded for execution with the computing device;

suspending the loading of the new program;

comparing the new program to the allowed program; and

determining whether the new program is valid.

22. The method of Claim 21, wherein the step of receiving a signal comprises receiving a signal from the computing device's operating system kernel.

23. The method of Claim 21, wherein the step of comparing the new program to the allowed program comprises comparing a checksum of the new program to a checksum of the allowed program.

24. The method of Claim 21, wherein the step of determining whether the new program is valid comprises determining whether a checksum of the new program corresponds to a checksum of the allowed program.

25. A computer-readable medium having computer-executable instructions for performing the steps recited in Claim 21.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 26

26. A computer-implemented method for performing security for a computing device during an execution phase comprising the steps of:

intercepting a call to open a file from the computing device's operating system;

determining whether the file is restricted;

if the file is restricted, taking remedial action to protect the computing device; and

if the file is not restricted, permitting the call to open the file.

27. The method of Claim 26, further comprising the step of receiving an instruction to monitor a call from an executing program.

28. The method of Claim 26, further comprising the step of defining files that are restricted from access.

29. The method of Claim 28, wherein the step of defining files that are restricted comprises designating a file that cannot be accessed in response to the call.

30. The method of Claim 28, wherein the step of defining files that are restricted comprises designating a file that cannot be modified in response to the call.

31. The method of Claim 26, further comprising the step of alerting a security management module of the presence of a malicious program.

32. The method of Claim 26, wherein the step of taking remedial action comprises terminating the call.

33. The method of Claim 26, wherein the step of taking remedial action comprises alerting another security module of the existence of the call.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 27

34. The method of Claim 26, wherein the step of taking remedial action comprises notifying the user of the existence of the call.

35. A computer-readable medium having computer-executable instructions for performing the steps recited in Claim 26.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 28

36. A system for managing security of a computing device comprising:

a pre-execution module operable for receiving notice from the computing device's operating system that a program is being loaded onto the computing device;

a validation module coupled to the pre-execution monitor operable for determining whether the program is valid;

a detection module coupled to the pre-execution monitor operable for intercepting a trigger from the computing device's operating system; and

an execution module coupled to the detection module and operable for monitoring the program in response to the trigger intercepted by the detection module.

37. The system of Claim 36, wherein the pre-execution module is further operable for suspending loading of the program onto the computing device.

38. The system of Claim 36, wherein the pre-execution module is further operable for retrieving validation data.

39. The system of Claim 36, wherein the execution module is further operable for deciding whether to terminate the trigger intercepted by the detection module.

40. The system of Claim 36, wherein the execution module is further operable for retrieving authorization data and deciding how to respond to the trigger intercepted by the detection module.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 29

SYSTEM AND METHOD FOR THE MANAGED SECURITY CONTROL OF PROCESSES ON A COMPUTER SYSTEM

Abstract of the Disclosure

Managing and controlling the execution of software programs with a computing device to protect the computing device from malicious activities. A protector system implements a two-step process to ensure that software programs do not perform malicious activities which may damage the computing device or other computing resources to which the device is coupled. In the first phase, the protector system determines whether a software program has been previously approved and validates that the software program has not been altered. If the software program is validated during the first phase, this will minimize or eliminate security monitoring operations while the software program is executing during the second phase. If the software program cannot be validated, the protector system enters the second phase and detects and observes executing activities at the kernel level of the operating system so that suspicious actions can be anticipated and addressed before they are able to do harm to the computing device.

K&S Docket: 05456-105038

IPR2024-00027
CrowdStrike Exhibit 1002 Page 30

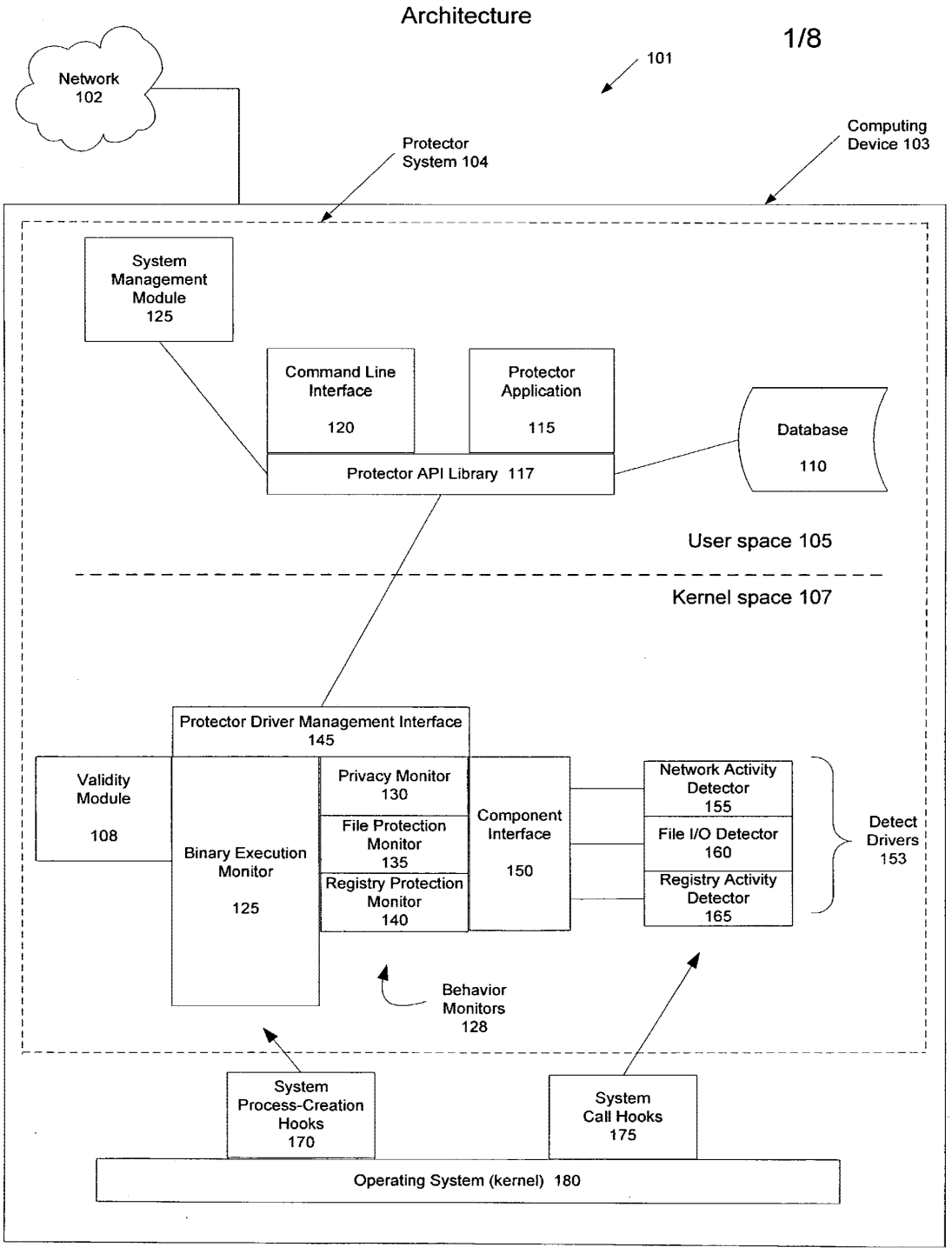

Architecture
1/8
101
Network
102
Protector
System 104
Computing
Device 103
System
Management
Module
125
Command Line
Interface
120
Protector
Application
115
Database
110
Protector API Library 117
User space 105
Kernel space 107
Protector Driver Management Interface
145
Validity
Module
108
Binary Execution
Monitor
125
Privacy Monitor
130
File Protection
Monitor
135
Registry Protection
Monitor
140
Component
Interface
150
Network Activity
Detector
155
File I/O Detector
160
Registry Activity
Detector
165
Detect
Drivers
153
Behavior
Monitors
128
System
Process-Creation
Hooks
170
System
Call Hooks
175
Operating System (kernel) 180

FIG. 1

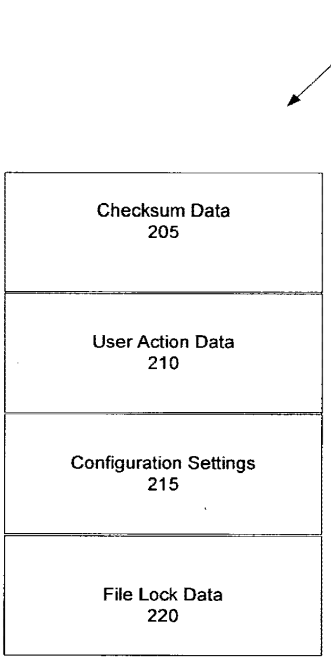

Database
110
Checksum Data
205
User Action Data
210
Configuration Settings
215
File Lock Data
220

FIG. 2

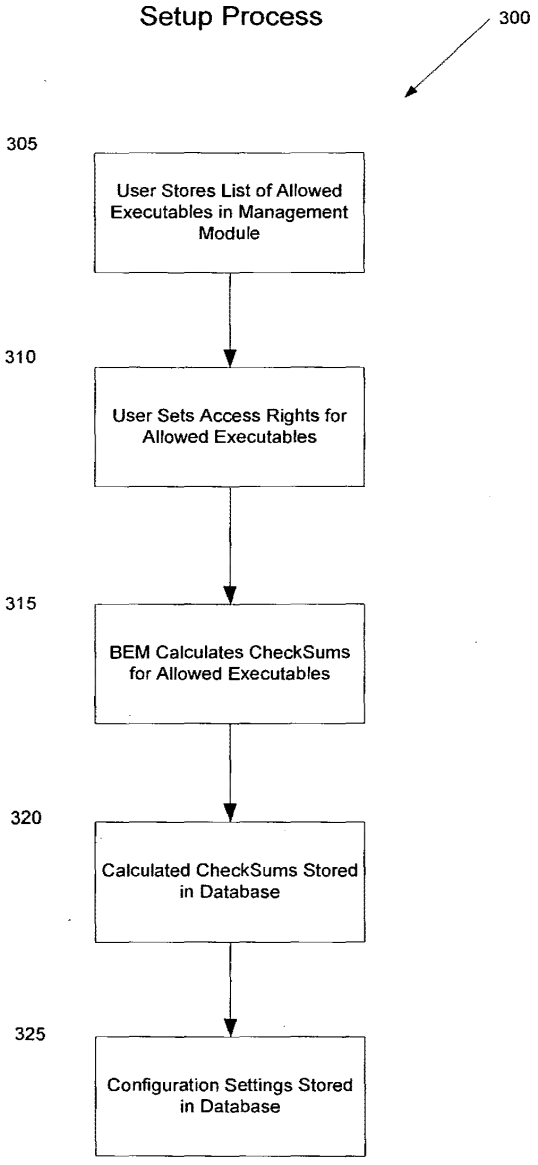

Setup Process
300
305
User Stores List of Allowed Executables in Management Module
310
User Sets Access Rights for Allowed Executables
315
BEM Calculates CheckSums for Allowed Executables
320
Calculated CheckSums Stored in Database
325
Configuration Settings Stored in Database

FIG. 3

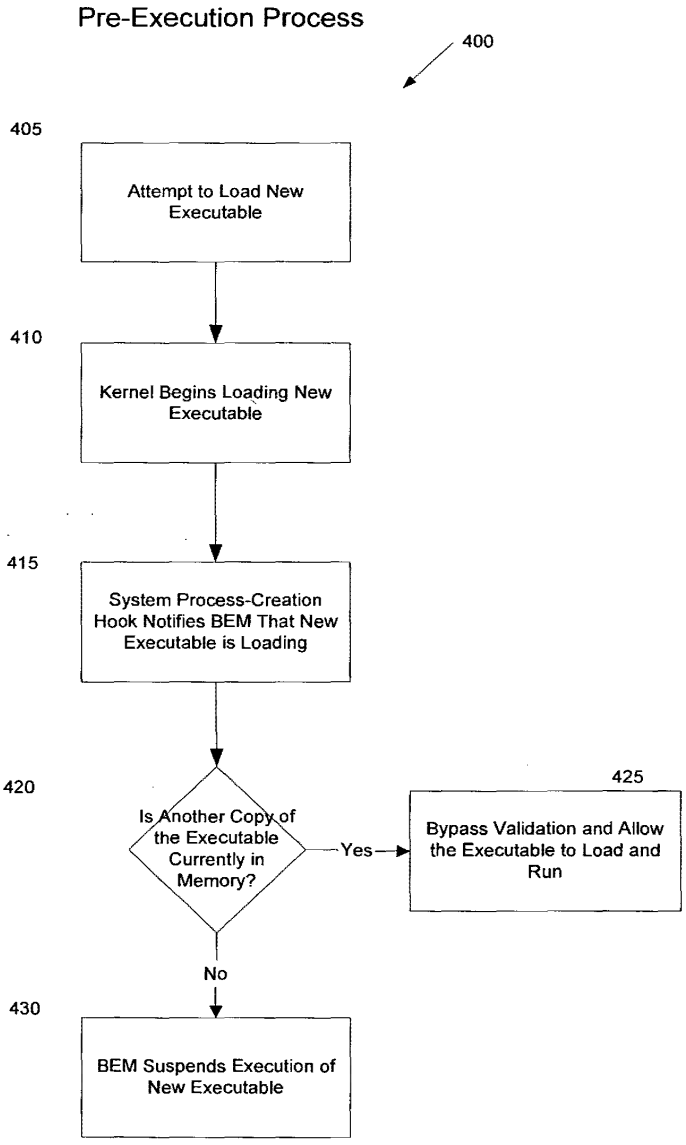

Pre-Execution Process
400
405
Attempt to Load New Executable
410
Kernel Begins Loading New Executable
415
System Process-Creation Hook Notifies BEM That New Executable is Loading
420
Is Another Copy of the Executable Currently in Memory?
Yes
425
Bypass Validation and Allow the Executable to Load and Run
No
430
BEM Suspends Execution of New Executable

FIG. 4A

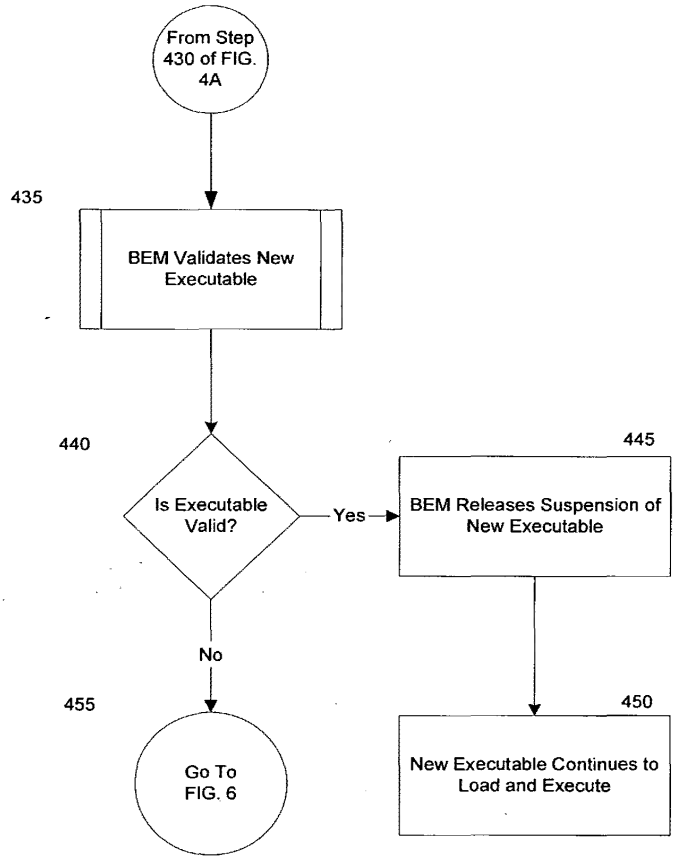

From Step 430 of FIG. 4A
435
BEM Validates New Executable
440
Is Executable Valid?
Yes
445
BEM Releases Suspension of New Executable
No
455
Go To FIG. 6
450
New Executable Continues to Load and Execute

FIG. 4B

Validating the Executable

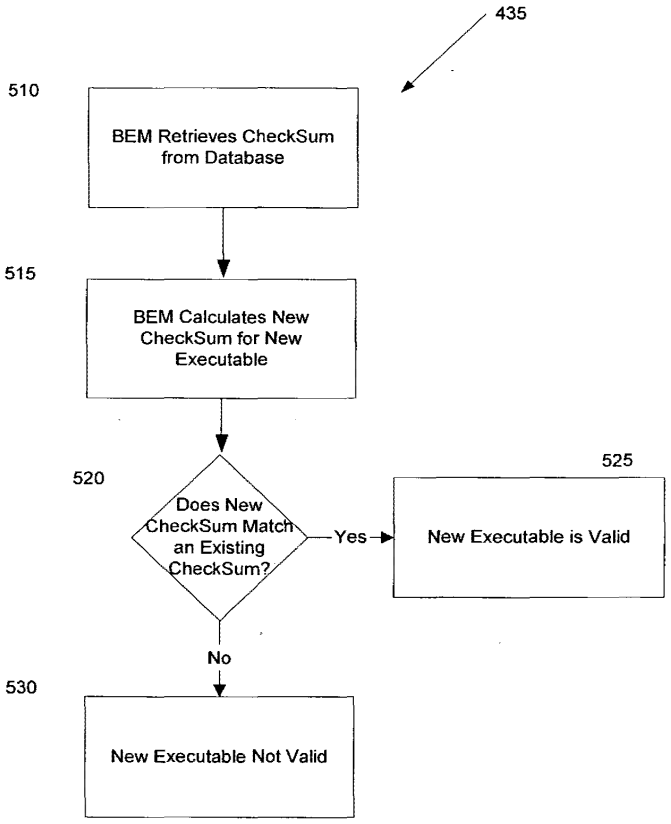


FIG. 5

Non-Validated Execution Process

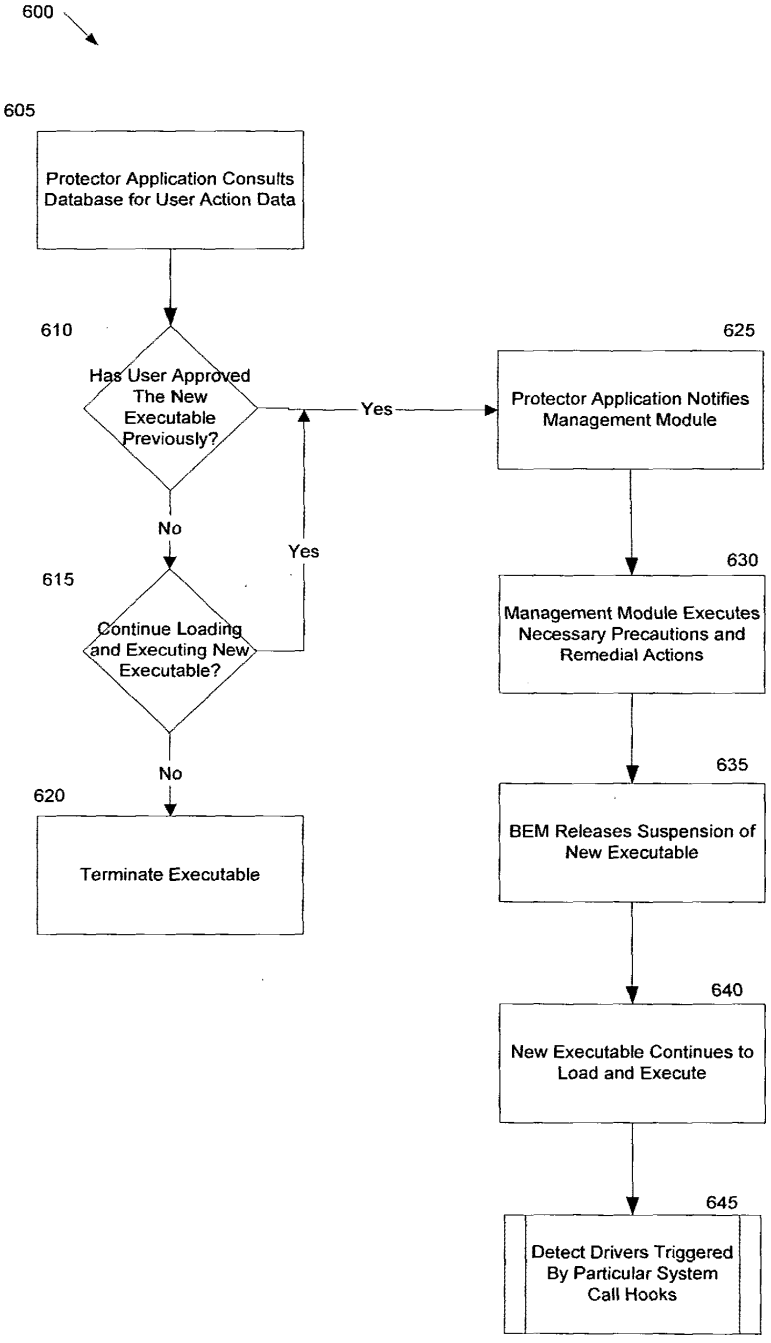


FIG. 6

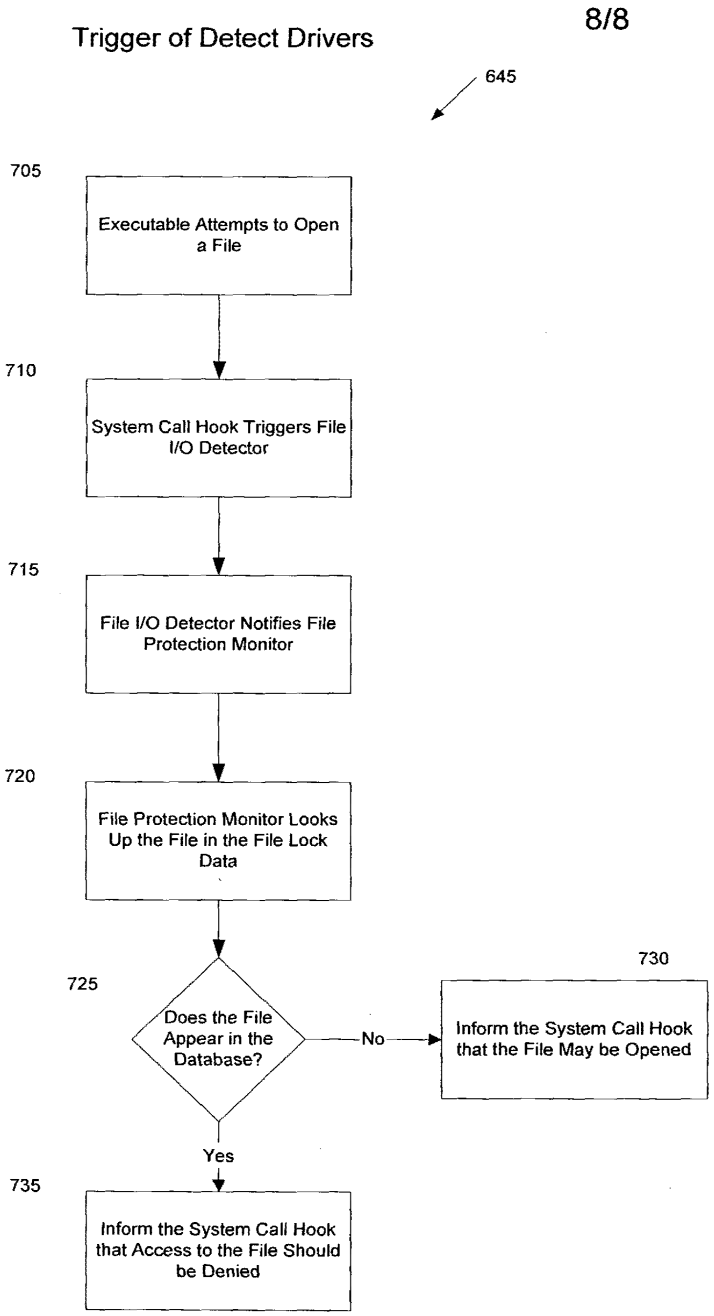

Trigger of Detect Drivers
645
705
Executable Attempts to Open a File
710
System Call Hook Triggers File I/O Detector
715
File I/O Detector Notifies File Protection Monitor
720
File Protection Monitor Looks Up the File in the File Lock Data
725
Does the File Appear in the Database?
No
730
Inform the System Call Hook that the File May be Opened
Yes
735
Inform the System Call Hook that Access to the File Should be Denied

FIG. 7

DECLARATION AND POWER OF ATTORNEY Attorney's Docket No. **05456.105038 US**

In re Application of: **Thomas James Satterlee and William Frank Hackenberger**

As a below named inventor, I hereby declare that my residence, post office address and citizenship are as stated below next to my name. 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: **System and Method for the Managed Security Control of Processes on a Computer System**, the specification of which is attached hereto.

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 do not know and do not believe that the same was ever known or used by others in the United States of America before my or our invention thereof, or patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to the date of this application. I further state that the invention was not in public use or on sale in the United States of America more than one year prior to the date of this application. *I understand that I have a duty of candor and good faith toward the Patent and Trademark Office,* and I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 (a)-(d) of the foreign application(s) for patent or inventor's certificate listed below, and have also identified below any foreign application for patent or inventor's certificate disclosing subject matter in common with the above-identified specification and having a filing date before that of the application on which priority is claimed:

Application No.	Country	Filing Date	Priority Claimed Under 35 USC §119 (a)-(d)
			Yes_____ No_____

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

60/345,432	January 4, 2002		
(Application No.)	(Filing Date)	(Application No.)	(Filing Date)

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter disclosed and claimed in the present application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §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

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 statement 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 patents issuing thereon.

POWER OF ATTORNEY: The following are hereby appointed to prosecute this application and transact all business in the Patent and Trademark Office connected therewith: Sherry M. Knowles-33,052; W. Scott Petty-35,645; Clark G. Sullivan-36,942; Holmes J. Hawkins-38,913; Steven P. Wigmore-40,447; Charles E. Peeler-45,004; Joseph M. Bennett-Paris-47,226; A. Shane Nichols-43,836; William O. Isaacs II-44,165; Robert T. Neufeld-48,394; Stephanie D. Adams-47,378; Kristin Chrudimsky-51,561; Hoang Vo-47,158; James M. Hannon-48,565; Charles Thorpe, Jr.-48,782

[redacted]

IPR2024-00027
CrowdStrike Exhibit 1002 Page 39

Send correspondence to: **King & Spalding, LLP** Customer No. 20786 Direct telephone calls at **(404) 572-4600**
191 Peachtree Street, N.E., 45th Floor
Atlanta, Georgia 30303 Robert T. Neufeld

Full name of inventor: Thomas James Satterlee	Citizenship: United States of America
Inventor's signature	Date: 1-2-03
Residence and Post Office Address: 1885 Nugget Drive, Felton, CA 95018	

[redacted]

IPR2024-00027
CrowdStrike Exhibit 1002 Page 40

DECLARATION AND POWER OF ATTORNEY Attorney's Docket No. **05456.105038 US**

In re Application of: **Thomas James Satterlee and William Frank Hackenberger**

As a below named inventor, I hereby declare that my residence, post office address and citizenship are as stated below next to my name. 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: **System and Method for the Managed Security Control of Processes on a Computer System**, the specification of which is attached hereto.

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 do not know and do not believe that the same was ever known or used by others in the United States of America before my or our invention thereof, or patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to the date of this application. I further state that the invention was not in public use or on sale in the United States of America more than one year prior to the date of this application. *I understand that I have a duty of candor and good faith toward the Patent and Trademark Office,* and I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 (a)-(d) of the foreign application(s) for patent or inventor's certificate listed below, and have also identified below any foreign application for patent or inventor's certificate disclosing subject matter in common with the above-identified specification and having a filing date before that of the application on which priority is claimed:

Application No.	Country	Filing Date	Priority Claimed Under 35 USC §119 (a)-(d)
			Yes_____ No_____

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

60/345,432	January 4, 2002		
(Application No.)	(Filing Date)	(Application No.)	(Filing Date)

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter disclosed and claimed in the present application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §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

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 statement 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 patents issuing thereon.

POWER OF ATTORNEY: The following are hereby appointed to prosecute this application and transact all business in the Patent and Trademark Office connected therewith: Sherry M. Knowles-33,052; W. Scott Petty-35,645; Clark G. Sullivan-36,942; Holmes J. Hawkins-38,913; Steven P. Wigmore-40,447; Charles E. Peeler-45,004; Joseph M. Bennett-Paris-47,226; A. Shane Nichols-43,836; William O. Isaacs II-44,165; Robert T. Neufeld-48,394; Stephanie D. Adams-47,378; Kristin Chrudimsky-51,561; Hoang Vo-47,158; James M. Hannon-48,565; Charles Thorpe, Jr.-48,782

[redacted]

IPR2024-00027
CrowdStrike Exhibit 1002 Page 41

Send correspondence to: **King & Spalding, LLP** Customer No. 20786 Direct telephone calls at **(404) 572-4600**
191 Peachtree Street, N.E., 45th Floor
Atlanta, Georgia 30303 Robert T. Neufeld

Full name of inventor: William Frank Hackenberger	Citizenship: United States of America
Inventor's signature	Date:
Residence and Post Office Address: 707 University Avenue, Los Altos, CA 94022	

[redacted]

01/03/03 JC836 U.S. PTO

01/04/03

10336299.010303

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new non-provisional applications under 37 CFR 1.53(b))

Title of Invention	**System and Method for the Managed Security Control of Processes on a Computer System**
Named Inventor(s)	**Thomas James Satterlee and William Franko Hackenberger**
Attorney Docket	**05456.105038 US**
Express Mail Label No.	**EU 468794582 US**

JC996 U.S. PTO 10/336299 01/03/03

APPLICATION ELEMENTS

1. ☒ Fee Transmittal Form
   (Submit an original, and a duplicate for fee processing)
2. ☐ Applicant claims small entity status. See 37 CFR 1.27.
3. ☒ Specification, Claims, and Abstract Total Pages **28**
4. ☒ Drawings Total Sheets **8**
5. Oath or Declaration Total Pages **4**
   a. ☒ 1 Newly executed; 1 unexecuted (original)
   b. ☐ Copy from prior application (37 CFR 1.63(d))
   (for continuation/divisional with Box 17 completed)
   (i) ☐ DELETION OF INVENTOR(S)
   Signed statement attached deleting inventor(s) named 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)*
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. ☐ Statement verifying identity of above copies

ADDRESS TO: Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

ACCOMPANYING APPLICATION PARTS

9. ☐ Assignment Papers (cover sheet & document(s))
10. ☐ 37 CFR 3.73(b) Statement *(when there is an assignee)*
    ☐ Power of Attorney by assignee
11. ☐ English Translation Document *(if applicable)*
12. ☒ Information Disclosure Statement (IDS) PTO-1449
    ☒ Copies of IDS Citations
13. ☐ Preliminary Amendment
14. ☒ Return Receipt Postcard (MPEP 503)
    (Should be specifically itemized)
15. ☐ Certified Copy of Priority Document(s)
16. ☒ Assignee: Internet Security Systems
    6303 Barfield Road
    Atlanta, GA 30328
19. ☐ Other: _____

17. If a **CONTINUING APPLICATION**, *check appropriate box and supply the requisite information*:
☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No:
Prior application information: Examiner: Group/Art Unit:

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 5b, is considered as being part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

18. CORRESPONDENCE ADDRESS:

Robert T. Neufeld, Esq.
KING & SPALDING LLP
45th Floor
191 Peachtree Street, N.E.
Atlanta, Georgia 30303

By: Robert T. Neufeld Reg. No. 48,394
Date: January 3, 2003
Telephone: 404.572.4600
Facsimile: 404.572.5145
Customer No. 20786

[redacted]

10336299.010303

FEE TRANSMITTAL

Attorney Docket No. **05456.105038 US**
Express Mail Label No. **EU 468794582 US**

This sheet accompanies a patent application transmittal for the following application:

Inventor(s): **Thomas James Satterlee and William Frank Hackenberger**
Filing Date: **January 3, 2003**
Title: **System and Method for the Managed Security Control of Processes on a Computer System**

The filing fee is calculated as shown below:

1. **FILING FEE:**

	SMALL ENTITY		LARGE ENTITY	
FOR:	FEE	FEE PAID	FEE	FEE PAID
☒ UTILITY FILING FEE	**$370**		**$740**	**740.00**
☐ DESIGN FILING FEE	**$165**		**$330**	
☐ PLANT FILING FEE	**$255**		**$510**	
☐ REISSUE FILING FEE	**$370**		**$740**	
☐ PROVISIONAL FILING FEE	**$80**		**$160**	
SUBTOTAL (1)	**$xxx**			**$740.00**

2. **CLAIMS:**

			SMALL ENTITY		LARGE ENTITY	
FOR:	NO. FILED	NO. EXTRA	RATE	FEE	RATE	FEE
TOTAL CLAIMS	40 - 20 =	20	x 9 =		**20 x 18 =**	**360.00**
INDEP. CLAIMS	5 - 3 =	2	**x 42 =**		**2 x 84 =**	**168.00**
☐ MULTIPLE DEPENDENT CLAIM PRESENTED			**+140 =**		**+280 =**	
		SUBTOTAL (2)	**$xxx**			**$528.00**

3. **ADDITIONAL FEES:**

	SMALL ENTITY		LARGE ENTITY	
FOR:	FEE	FEE PAID	FEE	FEE PAID
☐ LATE FILING, FEE OR OATH	**$65**		**$130**	
☐ NON-ENGLISH SPECIFICATION	**$130**		**$130**	
☐ OTHER				
SUBTOTAL (3)	**$xxx**			**$xxx**

TOTAL FILING FEES: $1,268.00

A check is enclosed for the total amount: **$1,178.00**

☒ Charge any additional fees required under 37 C.F.R. 1.16 or 1.17 to Deposit Account 11-0980.

KING & SPALDING LLP
45th Floor, 191 Peachtree Street, N.E.
Atlanta, Georgia 30303
Telephone: 404.572.4600

By: _____ Date: January 3, 2003
Robert T. Neufeld, Attorney for Applicant
Reg. No. 48,394

[redacted]

PATENT APPLICATION SERIAL NO. _____

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

01/07/2003 MBERHE 00000025 110980 10336299

01 FC:1202		360.00 OP
02 FC:1201		168.00 OP
03 FC:1001	100.00 CH	650.00 OP

PTO-1556
(5/87)

*U.S. Government Printing Office: 2001 — 481-697/59173

PATENT APPLICATION FEE DETERMINATION RECORD
Effective January 1, 2003

Application or Docket Number: 05056-105038 US

CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)
TOTAL CLAIMS	40	
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	40 minus 20=	* 20
INDEPENDENT CLAIMS	1 minus 3 =	* 2
MULTIPLE DEPENDENT CLAIM PRESENT		☐

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

SMALL ENTITY TYPE ☐		OR	OTHER THAN SMALL ENTITY	
RATE	FEE		RATE	FEE
BASIC FEE	$375	OR	BASIC FEE	$750
X$ 9=		OR	X$18=	360
X42=		OR	X84=	168
+140=		OR	+280=	
TOTAL		OR	TOTAL	1278

CLAIMS AS AMENDED - PART II

AMENDMENT A

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X$ 9=		OR	X$18=	
X42=		OR	X84=	
+140=		OR	+280=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT B

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X$ 9=		OR	X$18=	
X42=		OR	X84=	
+140=		OR	+280=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT C

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X$ 9=		OR	X$18=	
X42=		OR	X84=	
+140=		OR	+280=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. 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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 46

FORM PTO-1449 Adapted

MAY 14 2003

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Satterlee et al.		

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	CA		Mounji et al., "Distributed Audit Trail Analysis," Proceedings of the Symposium on Network and Distributed System Security, San Diego, California, February 16-17, 1995, pp. 102-112.
	CB		Wobber et al., "Authentication in the Taos Operating System," ACM Transactions on Computer Systems, Vol. 12, No. 1, February 1994, pp. 3-32.
	CC		Mayer et al., "The Design of the Trusted Workstation: A True 'Infosec' Product," 13th National Computer Security Conference, Washington, DC, October 1-4, 1990, pp. 827-839.
	CD		Dawson, J. Bruce, "Intrusion Protection for Networks," Byte, April 1995, pp. 171-172.
	CE		Buhkan, Michael, "CheckPoint Charlie," PC Week Netweek, November 27, 1995, pp. N1, N6-N7.
	CF		Process Software Technical Support page , found on http://www.process.com/techsupport/whitesec.html, printed off of the Process Software website on February 26, 2003, pp. 1-5.
	CG		Ganesan, R., "BAfirewall: a modern firewall design," Proceedings Internet Society Symposium on Network and Distributed System Security 1994, Internet Soc. 1994, pp. 99-108, Reston, VA.
	CH		Lee, Theodore M.P., "Trusted Systems," Chapter II-1-6 of Handbook of Information Security Management, Ed. Zella G. Ruthberg and Harold F. Tipton, Auerbach, Boston and New York, 1993, pp. 345-362.
	CI		Lunt, Teresa F., "Automated Intrusion Detection," Chapter II-4-4 of Handbook of Information Security Management, Ed. Zella G. Ruthberg and Harold F. Tipton, Auerbach, Boston and New York, 1993, pp. 551-563.

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.
	CJ	WO 01/84285	11/08/01	PCT	Internet Security Systems, Inc.		
	CK	WO 02/06928	01/24/02	PCT	vCIS, Inc.		
	CL	WO 00/54458	09/14/00	PCT	Psionic Software, Inc.		
	CM	WO 02/056152	07/18/02	PCT	Psionic Software, Inc.		

EXAMINER	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.

OIPE
MAY 1 4 2003
PATENT & TRADEMARK OFFICE

#3 0300

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Satterlee et al.

Serial No.: **10/336,299**

Filed: **January 3, 2003**

For: **System and Method for the Managed Security Control of Processes on a Computer System**

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT

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

Sir:

The citation of information on the attached Form PTO-1449, "List of Art Cited by Applicant" is made pursuant to 37 C.F.R. §§ 1.56, 1.97, and 1.98. A copy of each cited item is enclosed.

The citation of this information does not constitute an admission of priority or that any cited item is available as a reference, or a waiver of any right the applicant may have under applicable statutes, Rules of Practice in patent cases, or otherwise.

Respectfully submitted,

W. Scott Petty
Reg. No. 35,645

King & Spalding LLP
45th Floor, 191 Peachtree Street, N.E.
Atlanta, GA 30303
404.572.4600
K&S Docket: 05456.105038

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on May 8, 2003.

W. Scott Petty, Reg. No. 35,645

OIPE
MAY 1 4 2003

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003

APPLICANT	GROUP
Satterlee et al.	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	AA	5,857,191	01/05/99	Blackwell, Jr. et al.			
	AB	6,035,423	03/07/00	Hodges et al.			
/	AC	5,345,595	09/06/94	Johnson et al.			
/	AD	6,266,774	07/24/01	Sampath et al.			
/	AE	6,026,442	02/15/00	Lewis et al.			
/	AF	6,016,553	01/18/00	Schneider et al.			
/	AG	6,014,645	01/11/00	Cunningham			
/	AH	5,881,236	03/09/99	Dickey			
/	AI	5,764,890	06/09/98	Glasser et al.			
/	AJ	5,630,061	05/13/97	Richter et al.			
/	AK	5,414,833	05/09/95	Hershey et al.			
/	AL	4,819,234	04/04/89	Huber			
/	AM	6,275,938	08/14/01	Bond et al.			
/	AN	5,950,012	09/07/99	Shiell et al.			
/	AO	5,278,901	01/11/94	Shieh et al.			
/	AP	6,405,318	06/11/02	Rowland			
	AQ	5,475,839	12/12/95	Watson et al.			
/	AR	5,590,331	12/31/96	Lewis et al.			
/	AS	5,761,504	06/02/98	Corrigan et al.			

EXAMINER	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.

OIPE
MAY 1 4 2003

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		GROUP
Satterlee et al.		

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	BA		Sammons, Nathaniel, "Multi-platform Interrogation and Reporting with Rscan," The Ninth Systems Administration Conference, LISA 1995, Monterrey, California, September 17-22, 1995, pp. 75-87.
	BB		Dean et al., "Java Security: From HotJava to Netscape and Beyond," Proceedings of the 1996 IEEE Symposium on Security and Privacy, May 6-8, 1996, Oakland, California, pp. 190-200.
	BC		Fisch et al., "The Design of an Audit Trail Analysis Tool," Proceedings of the 10th Annual Computer Security Applications Conference, December 5-9, 1994, Orlando, Florida, pp. 126-132.
	BD		Safford et al., "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment," USENIX Symposium Proceedings, UNIX Security IV, October 4-6, 1993, Santa Clara, California, pp. 91-118.
	BE		Sugawara, Toshiharu, "A Cooperative LAN Diagnostic and Observation Expert System," Ninth Annual Phoenix Conference on Computers and Communications, 1990 Conference Proceedings, March 21-23, 1990, Scottsdale, Arizona, pp. 667-674.
	BF		Casella, Karen A., "Security Administration in an Open Networking Environment," The Ninth Systems Administration Conference, LISA 1995, Monterrey, California, September 17-22, 1995, pp. 67-73.
	BG		Burchell, Jonathan, "Vi-SPY: Universal NIM?" Virus Bulletin, January 1995, pp. 20-22.
	BH		Benzel et al., "Identification of Subjects and Objects in a Trusted Extensible Client Server Architecture," 18th National Information Systems Security Conference, October 10-13, 1995, Baltimore, Maryland, pp. 83-99.
	BI		Epstein et al., "Component Architectures for Trusted Netware," 18th National Information Systems Security Conference, October 10-13, 1995, Baltimore, Maryland, pp. 455-463.
	BJ		Varadharajan, Vijay, "Design and Management of a Secure Networked Administration System: A Practical Approach," 19th National Information Systems Security Conference, October 22-25, 1996, Baltimore, Maryland, pp. 570-580.
	BK		Snapp et al., "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype," 14th National Computer Security Conference, October 1-4, 1991, Washington, DC, pp. 167-176.
	BL		Broner et al., "IntelligentI/O Rule-Based Input/Output Processing for Operating Systems," Operating Systems Review, Volume 25, Number 3, July 1991, pp. 10-26.
	BM		Drews et al., "Special Delivery - Automatic Software Distribution Can Make You A Hero," Network Computing, August 1, 1994, pp. 80, 82-86, 89, 91-95.
	BN		Morrissey, Peter, "Walls," Network Computing, February 15, 1996, pp. 55-59, 65-67.
	BO		Harlander, Dr. Magnus, "Central System Administration in a Heterogenous Unix Environment: GeNUAdmin," Proceedings of the Eighth Systems Administration Conference (LISA VIII), September 19-23, 1994, San Diego, California, pp. 1-8.
	BP		Shaddock et al., "How to Upgrade 1500 Workstations on Saturday, and Still Have Time to Mow the Yard on Sunday," The Ninth Systems Administration Conference LISA '95, September 17-22, 1995, Monterrey, California, pp. 59-65.
	BQ		Anderson, Paul, "Towards a High-Level Machine Configuration System," Proceedings of the Eighth Systems Administration Conference (LISA VIII), September 19-23, 1994, San Diego, California, pp. 19-26.
	BR		Cooper, Michael A., "Overhauling Rdist for the '90s," Proceedings of the Sixth Systems Administration Conference (LISA VI), October 19-23, 1992, Long Beach, California, pp. 175-188.
	BS		Vangala et al., "Software Distribution and Management in a Networked Environment," Proceedings of the Sixth Systems Administration Conference, October 19-23, 1992, Long Beach, California, pp. 163-170.
	BT		Kim et al., "The Design and Implementation of Tripwire: A File System Integrity Checker," 2nd ACM Conference on Computer and Communications Security, November 2-4, 1994, Fairfax, Virginia, pp. 18-29.

EXAMINER	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.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification: **H04L 9/32**	**A1**	(11) International Publication Number:	**WO 00/54458**
		(43) International Publication Date:	14 September 2000 (14.09.2000)

(21) International Application Number: PCT/US00/06313

(22) International Filing Date: 10 March 2000 (10.03.2000)

(30) Priority Data:
09/268,084 12 March 1999 (12.03.1999) US

(60) Parent Application or Grant
PSIONIC SOFTWARE, INC. [/]; (). ROWLAND, Craig [/]; () . ROWLAND, Craig [/]; (). TAYLOR RUSSELL, Gail ; ().

Published

(54) Title: INTRUSION DETECTION SYSTEM
(54) Titre: SYSTEME DE DETECTION D'INTRUSION

(57) Abstract

A computer-implemented intrusion detection system and method (1) that monitors a computer system in real-time for activity indicative of attempted or actual access by unauthorized persons or computers. The system detects unauthorized users (20) attempting to enter into a computer system by comparing user behavior to a user profile (22), detects events that indicate an unauthorized entry into the computer system (90), notifies a control (37, 97) function about the unauthorized users and events that indicate unauthorized entry into the computer system and has a control function (125) that automatically takes action in response to the event (127). The user profiles are dynamically constructed for each computer user when the computer user first attempts to log into the computer system (24) and upon subsequent logins (25), the user's profile is dynamically updated (25). By comparing user behavior to the dynamically built user profile (3-5), false alarms are reduced. The system also includes a log auditing function (10, a port scan detector (75) and a session monitor function (90).

(57) Abrégé

L'invention concerne un procédé (1) et un système de détection d'intrusion informatisé, qui surveille un système informatique en temps réel et signale une tentative d'accès ou un accès réel par des personnes ou des ordinateurs non autorisés. Le système détecte les utilisateurs non autorisés (20) qui tentent d'entrer dans un système informatique par comparaison d'un comportement d'utilisateur avec un profil d'utilisateur (22), détecte les événements indiquant une entrée non autorisée dans le système informatique (90), notifie une fonction de contrôle (37, 97) stockant des informations associées aux utilisateurs et aux événements non autorisés indiquant une entrée non autorisée dans le système informatique, et exécute une fonction de contrôle (a25) qui réagit automatiquement à l'événement (127). Les profils d'utilisateur sont construits en dynamique pour chaque utilisateur d'ordinateur lorsque ledit utilisateur tente, pour la première fois, d'entrer en communication avec le système informatique (24), et ces profils sont mis à jour (25) en dynamique lors d'entrées en communication (25) ultérieures. Les fausses alarmes sont réduites par comparaison du comportement d'utilisateur avec le profil d'utilisateur (3-5) construit en dynamique. Le système comprend également une fonction de vérification (10) d'entrée en communication, un détecteur de balayage (75) de port, et une fonction de surveillance (90) de session.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H04L 9/32 — A1

(11) International Publication Number: WO 00/54458

(43) International Publication Date: 14 September 2000 (14.09.00)

(21) International Application Number: PCT/US00/06313

(22) International Filing Date: 10 March 2000 (10.03.00)

(30) Priority Data:
09/268,084 12 March 1999 (12.03.99) US

(71) Applicant (*for all designated States except US*): PSIONIC SOFTWARE, INC. [US/US]; 6908 Dogwood Hollow, Austin, TX 78750 (US).

(72) Inventor; and

(75) Inventor/Applicant (*for US only*): ROWLAND, Craig [US/US]; 6908 Dogwood Hollow, Austin, TX 78750 (US).

(74) Agent: TAYLOR RUSSELL, Gail; Taylor Russell & Russell, P.C., Building One, Suite 1200, 4807 Spicewood Springs Road, Austin, TX 78759 (US).

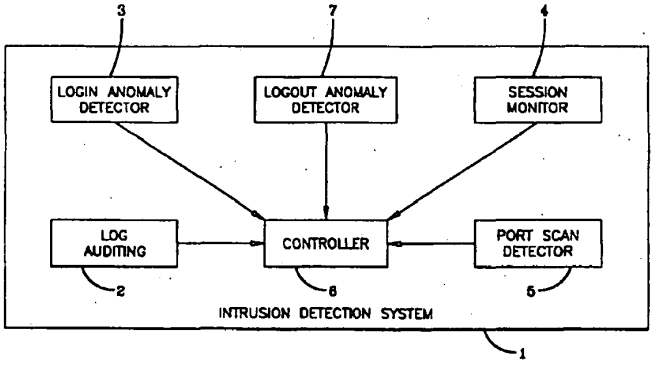
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, 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), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, 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: INTRUSION DETECTION SYSTEM



(57) Abstract

A computer-implemented intrusion detection system and method (1) that monitors a computer system in real-time for activity indicative of attempted or actual access by unauthorized persons or computers. The system detects unauthorized users (20) attempting to enter into a computer system by comparing user behavior to a user profile (22), detects events that indicate an unauthorized entry into the computer system (90), notifies a control (37, 97) function about the unauthorized users and events that indicate unauthorized entry into the computer system and has a control function (125) that automatically takes action in response to the event (127). The user profiles are dynamically constructed for each computer user when the computer user first attempts to log into the computer system (24) and upon subsequent logins (25), the user's profile is dynamically updated (25). By comparing user behavior to the dynamically built user profile (3–5), false alarms are reduced. The system also includes a log auditing function (10, a port scan detector (75) and a session monitor function (90).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 52

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		

Descripti n

INTRUSION DETECTION SYSTEM

TECHNICAL FIELD OF THE INVENTION

The present invention relates generally to intrusion detection for a computer system. More particularly, the invention is a computer-implemented intrusion detection system and method that monitors a computer system for activity indicative of attempted or actual access by unauthorized persons or computers.

BACKGROUND

Because of the increasing reliance on Internet, Intranet and extranet network computer access, intrusion into computer systems by unauthorized users is a growing problem. An intrusion is unauthorized access or attempted access into or unauthorized activity in a computer or information system. Intrusion detection technologies are therefore becoming extremely important to improve the overall security of computer systems. Intrusion detection is the process of identifying that an intrusion has been attempted, is occurring or has occurred.

In most intrusion detection systems, data may be automatically collected and reduced but the analysis of that data usually remains manual. Profiling and pattern recognition techniques also have been used to analyze the data collected and presented to an intrusion detection system. The off-line analysis involves determining normal behavior for a user, application or system. The normal behavior is then used to develop sets of rules. Significant deviations from the rules, referred to as anomalous behavior, may then be flagged as potential intrusions. Some intrusion detection systems, based on anomaly detection techniques, look for statistically anomalous behavior, that is, behavior that appears unusual when compared to other user behavior. One drawback of anomaly detection systems is that they are prone to both false positive and false negative alerts because the rules are general in nature and not specific for the behavior of each user. False positives occur when the intrusion detection system identifies an event as an intrusion when none has occurred. False positives may divert the attention and time of the system administrator and security staff and if frequent enough, may cause a lack of confidence in the intrusion detection system. False negatives are instances where the intrusion detection system fails to detect an intrusion while it is occurring or after it has occurred. The result may be slow or no response to the intrusion that can result in financial loss and system damage. False negatives often occur because the models used to profile the anomalous behavior do not adequately predict the intruder behavior and its result within the computer system.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 55

Some intrusion detection systems use xpert systems, which are driven from an encoded rule base to monitor policy compliance. The expert system applies the rules to assure all users are operating within their privileged rights. Even in the expert system, the encoded rules are usually generated by profiling the anomalous behavior and then building a rule based system. This means that the expert system intrusion detection system at present suffers from the same problems such as false positives and false negatives as the anomalous detection systems. Other systems have passive monitor functions that continually analyze data presented to them. They are similar to antivirus functions in that they can only detect what has been defined to them. Another type of intrusion detection system is a scanner. Unlike other intrusion detection tools that report when a threshold has been exceeded, scanners actively attempt to find security holes (called vulnerabilities) and unauthorized hardware and software.

The above mentioned intrusion detection methods have many drawbacks. One is that the systems can only detect and monitor what has been previously defined to them, either using expert system rules or rules developed through data collection reduction and analysis or through profiling. This can result in false negatives because unknown attacks have not been previously defined. In addition, most systems only analyze and develop profiles and patterns after the fact. These profiles and patterns of behavior are subsequently incorporated into rule-based systems to recognize future attacks. Even in those instances where alerts are issued in near real-time, valuable time and the intruder's trail can be lost. In addition, many of these systems require human intervention, both in the initial analysis of data and profile and pattern recognition building steps and when an anomalous event has occurred, to determine the action to be taken. Relying on human intervention can delay the identification of the intrusion and may not prevent network damage or exploitation.

To be able to detect intrusions as they are occurring or soon after, there is a need for the intrusion detection system to be a real-time system. There is a need to automatically build profiling data specific for each user or class of users that can be used to determine normal actions for a user to reduce the occurrence of false alarms and to improve detection. There is a need for a system that can detect suspicious actions, determine the source and institute autonomous responses. There is also a need for the intrusion detection system to take automatic action, without waiting for a human administrator to intervene and act, to mitigate the effects of an intrusion and to prevent future actions. There is also a need to coordinate information transfer within host, multi-host and network environments so responses to intrusions can be coordinated. In addition, there is a need to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 56

combine the above listed capabilities with real-time monitoring of log audit files, port scan detection capability and session monitoring.

The present invention is a computer implemented method for detecting intruders in a computer system. The method comprising the steps of detecting an unauthorized user attempting to enter into a computer system by comparing actions of the user to a dynamically built profile for the user, and if the action is out of range of the user profile, notifying a control function. If events are detected that indicate an unauthorized entry into the computer system has occurred by comparing and if an event occurs that indicates unauthorized entry, a control function is notified, and automatically executes a specific action in response to the event.

The dynamically built user profile comprises dynamically constructing a user profile for each computer user when the computer user first attempts to log into the computer system, dynamically updating the user profile for the user for each attempt by the user to log into the system after the first attempt, and updating the user profile when the user logs out of the computer system.

Dynamically monitoring computer system log files comprises monitoring for events that indicate an unauthorized attempted entry into the computer system. Dynamically monitoring system log files comprises comparing the system log files to events to ignore and ignoring the event if the system log file indicates a match with the event to ignore and comparing the system log files to events known to indicate an unauthorized entry into the computer system and notifying a control function about the unauthorized entry and automatically executing a specific action in response to the event by the control function.

The method further comprises dynamically monitoring user actions after the user has logged into a computer system for unauthorized access by the user to system information, and if unauthorized access occurs, notifying a control function about the unauthorized access and automatically executing a specific action in response to the event by the control function. The method dynamically monitors user actions after the user has logged into a computer system for corruption of system information by the user and if corruption of system information occurs, a control function is notified and automatically executes a specific action in response by the control function.

The method further comprises scanning network ports to determine if a user has connected to more than a selected number of network ports. If the user has exceeded the selected number of network ports, the control function is notified and automatically executes a specific action in response to the. The selected number of network ports may be set by the system administrator.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 57

The detecting vents that indicate an unauthorized entry into the computer system comprise detecting anomalous events when a user logs out of the computer system. This comprises monitoring a user's file history to determine if the user's file history has been altered, monitoring computer system files to determine if a modification has been made that indicates an unauthorized intrusion into the computer system, monitoring a user's computer files to determine if a modification has been made that indicates an unauthorized intrusion into the computer system, determining if a program has been left running that should have stopped running when the user logs out of the computer system. If an anomalous event has been a control function is notified about the anomalous event and the control function takes user specified action in response to the anomalous event.

Detecting unauthorized users comprises if the user has attempted to log in from a computer host that is not allowed access to the computer system, notifying a control function about the attempted login and allowing the control function to take a user specified action in response. Detecting unauthorized users also comprises if the user attempts to log into the computer system and has an active login, checking to determine if the user is allowed to have more than one login active simultaneously, and if not, notifying a control function about the attempted login and automatically executing a specific action in response to the event by the control function.

The dynamically constructed user profile for each computer user comprises storing user name, login terminal, time of creation of initial user profile, time of user's first login, time history of the user's logins, time periods that the user is allowed to log into the system and total number of logins when the computer user. The user profile may be stored in a user profile database.

Dynamically updating the user profile for the user comprises, for each user, entering the current login time, login terminal, updating the time history of the user's login and incrementing the total number of logins.

Detecting unauthorized users comprises if the user has attempted to log in at a time different from the time periods that the user is allowed to log into the computer system, notifying a control function about the attempted login and allowing the control function to take user specified action in response.

The method further comprises dynamically constructing a list of active users logged into the computer system and dynamically updating the list of active users when a user logs into the system and logs out of the system. The list of active users may comprise user name, user terminal and time of user login.

The control function comprises storing information about unauthorized users and events that indicate an unauthorized entry into the computer system, taking action in response to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 58

the unauthorized users and events. The action is selected from the group consisting of logging the information in a local controller, sending the information to a network controller, disabling the unauthorized user's account, blocking access to the computer system for the user, notifying a system administrator and ignoring the unauthorized user and unauthorized entry. The action taken may be defined by the system administrator prior to initialization of the intrusion detection system.

The control function may be located in a local computer where the unauthorized user and unauthorized entry occurred. The control function in the local computer sends information about unauthorized users and events to a central computer connected to the local computer. Alternatively, the control function may be located in a central computer connected to the local computer. Multiple local computers may be connected to the central computer.

The central computer comprises performing centralized analysis of unauthorized users and events, performing correlation of unauthorized users and events from the multiple local computers, alerting a central computer system administrator, and sending the analysis and correlation results to the multiple local computers.

The method further comprises, for each user, continuously monitoring user activity for a threat to the computer system. Continuously monitoring comprises analyzing user command entries and comparing the entries to known threat events and known attack patterns indicating a computer intrusion and if a match occurs, notifying the control function and allowing the control function to take user specified action in response. Continuously monitoring the system process accounting records comprises comparing the entries to known threat events and known attack patterns indicating a computer intrusion and if a match occurs, notifying the control function and allowing the control function to take user specified action in response.

The method further comprises continuously monitoring commands entered by the user and comparing the commands to known threat events and known attack patterns indicating a computer intrusion and if a match occurs, notifying the control function and allowing the control function to take user specified action in response. The method further comprises continuously monitoring network port activity and comparing the activity to known threat events and known attack patterns indicating a computer intrusion and if a match occurs, notifying the control function and allowing the control function to take user specified action in response. The action taken may be selected from the group consisting of logging the event, removing the user from the computer system and executing a selected command.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 59

The computer-implemented methods are embodied in software programs that may be stored on a computer-readable medium.

SUMMARY

The present invention provides a real-time intrusion detection method and system. The intrusion detection system automatically and dynamically builds user profile data (known as a signature) for each user (or alternatively, a class of users) that can be used to determine normal actions for each user to reduce the occurrence of false alarms and to improve detection. The user profile data (signature) is saved and updated every time the user logs on and off the system. The advantage of dynamically building user profile data based on past user behavior and comparing it to that user's current behavior is that the number of false alarms is reduced. In addition, there is no need to enter sets of rules prior to system initialization. The system detects suspicious actions, determines the source and institutes autonomous responses. The system acts to mitigate the effects of an intrusion and to prevent future actions without waiting for human action. The automatic actions to be taken can be specified by the system administrator prior to initialization of the system. The automatic actions can be tailored to address the specific anomaly detected by the intrusion detection system. For example, through a local or system controller, the system can log the events, disable user accounts and block access to the system. In one embodiment, the system coordinates information transfer within host, multi-host and network environments to coordinate intrusion response. The system combines the above listed capabilities with real-time monitoring of log audit files, port scan detection capability and session monitoring. Throughout this document, use of the terms dynamic or dynamically in relation to a process means that the process is operating in real-time or close to real-time.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features, aspects and advantages of the present invention will become better understood with regard to the following description, appended claims and accompanying drawings where:

Fig. 1 shows a functional block diagram of the host based intrusion detection system.

Fig. 2 is a block diagram of the log file auditing function.

Fig. 3 is a flow diagram of the login anomaly detection function.

Fig. 4 is a flow diagram of the user profile database and active user database update function.

Figs. 5A and 5B are flow diagrams of the logout anomaly detection function.

Fig. 6 is a flow diagram of the port scan detector function.

Fig. 7 is a flow diagram of the session monitor function.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 60

Fig. 8 is a flow diagram of the controller function.

Fig. 9 is a block diagram of an alternate embodiment of a host based intrusion detection system having a central system controller.

Fig. 10 is a flow diagram of program setup for the intrusion detection systems.

DETAILED DESCRIPTION OF THE DRAWINGS

Fig. 1 shows a functional block diagram of the intrusion detection system. The system is comprised of a log audit function 2, a login anomaly detection function 3, a logout anomaly detection 7, a session monitor function 4 and a port scan detector function 5 interfacing with a local controller function 6. The log audit function 2, login anomaly detection function 3, logout anomaly detection function 7, session monitoring function 4 and port scan detector function 5 all operate in real-time to detect activity indicative of an attack by unauthorized users or systems. The log audit function continuously monitors system log files for anomalous activity which can include known suspicious activity and unknown system anomalies. When anomalous behavior occurs, the log audit function 2 notifies the controller 6 and sends information about the activity to the controller 6 for further processing. The log auditing function 2 is described in Fig. 2. The login anomaly detection function 3 monitors system login activity and when anomalous behavior is detected, notifies the controller and sends information about the activity to the controller 6 for further processing. The login anomaly detection function 3 is described in Fig. 3. The logout anomaly detection function 7 monitors system logout activity and if anomalous behavior is detected, notifies the controller and sends information about the activity to the controller 6 for further processing. The logout anomaly detection function 7 is described in Figs. 5A and 5B. The session monitoring function 4 watches user activity after a login has been established. The function continuously watches keystrokes for known attack signatures and suspicious activity. Signatures are kept in a user-editable database on the local machine. Once suspicious or known attack activity is detected, the session monitor 4 will send information about the activity to the controller 6 for further processing. The session monitoring function 4 is described in Fig. 7. The port scan detector function 5 monitors Internet ports (such as TCP and UDP) for port scanning activity which is a method used by attackers to determine the vulnerabilities of a target host and to run a series of attacks to gain entry on the vulnerable target host. When the port scan detection function 5 detects port scanning activity, it sends information about the activity to the controller for further processing. The port scan detector function 5 is described in Fig. 6. The controller function 6 controls all actions that the host-based intrusion detection system may perform upon being notified from the log audit function 2, the login anomaly detection function 3, the logout anomaly detection function 7, the session monitor 4 or the port scan detector

IPR2024-00027
CrowdStrike Exhibit 1002 Page 61

function 5 that an anomalous activity has occurred, the controller takes an appropriate action based on that activity. The controller function is described in Fig. 8.

Turning now to Fig. 2, a block diagram of the log file auditing function is shown. The log auditing function 10 monitors the system login auditing files 11 by comparing the log file activity known attack events 12, known security violations 13, and events to ignore 14. If the log file activity indicates a known attack event 12 or a known security violation 13 indicating a suspicious event or unknown event has occurred or is in the process of occurring, then the log auditing function 10 constructs a message containing the log file information and signature identification information and forwards it to the controller for action. The log auditing function can run on a periodic basis with the period selected by the user or it can run continuously in real-time. The user has the flexibility to add or remove functions within the login anomaly detection to customize the system.

Turning now to Fig. 3, a flow diagram of the login anomaly detection function 20 is shown. The system monitors login and logout audit files and logs (records) all logins and logouts for the target host 21. The target host is the computer that the user is logging into or logging out of. The system login auditing files may be login records (such as wtmp and utmp records) for a Unix® based operating system or may be event logs for a Windows NT® operating system. The system checks to determine if the user should be ignored 38. Certain users are not checked for login or logout anomalies. If the user is to be ignored processing continues at step 35 where the user is logged into the system. If the user is not to be ignored and if the user is logging in to the system, the monitor builds/updates the user profile database 22 and updates the active user database as shown in Fig. 4. The system administrator has the flexibility to add or remove functions within the logout anomaly detection to customize the system.

Turning now to Fig. 4, a flow diagram is shown of the user profile database and active user database update function. If the user is not in the user profile database 23, then the user is a new user and process first login function is executed 24. A new user profile entry is created 24 which contains the user name, the login host, the login terminal (sometimes called the TTY), the time of creating the initial user profile, the time of the user's first login, the set days and hours the user is allowed login access, the version of the database record type and sets the initial number of logins to one. In addition, the system administrator notified whenever a user logs into a host for the first time. If the user is already in the user profile database 23, then a user profile entry already exists for this user and that profile is updated 25. The updates to the user profile include appending the login time, login host and incrementing the total number of logins. The system also checks to determine that the user's login account is still valid, that is that it has not been disabled by

IPR2024-00027
CrowdStrike Exhibit 1002 Page 62

a system administrator. An entry is created in the active user database 36 which contains the user's name, the terminal the user is logged in on, the time of login for this entry and the version of the database record.

Turning back to Fig. 3, the next step is to check to determine if the login is from a foreign domain 26. A foreign domain is one that is not contained within or allowed access to the host where the login is attempted. The list of allowed domains within the system is accessed 27 and if the login domain is not listed, it is considered foreign and the control function is notified 37.

The user login is checked to determine if there are multiple concurrent logins for the same user 28 A multiple concurrent login means that a user is logged into the system more than once from one or more different hosts concurrently. This type of behavior may indicate an intrusion. The log file is checked to determine if a user is logged in from one or more different hosts concurrently. If so and the user is not allowed to have multiple logins 29, then this login entry is denied and the multiple users are logged off from the system 30.

The next step is to determine if the user is logged in at an unusual time 31. For each user, a profile is automatically built of the days, times and length of time that the user has logged in. Once a certain threshold number of user logins have occurred for this user to allow for accurate user profiling (usually approximately ten logins, but this can be adjusted by the user), the day and time of the current user's attempted login is compared to that profile. If the current login time differs from the user's login profile, the control function is notified 37.

The next step is to compare the login activity with known attack patterns 34. If the login activity is similar to a known attack pattern, then the control function is notified 37. Next the history file is checked for suspicious command entries 39.

If these steps are successfully completed, the user is logged in 35 and the user's profile database entry is updated and the active user database is updated to track the login state of the user.

Turning now to Figs. 5A and 5B, a flow diagram of the logout anomaly detection function is shown. When a user attempts to logout, the logout anomaly detector 49 goes through a series of steps to process the logout to determine if something has occurred during the user's login time that may indicate a system anomaly. The logout entry for the user is updated in the user profile and the active user database is updated 50. If the user is to be ignored 65, then no other checking is done and the user is successfully logged out 70. The next step is to determine if the user's file history has been compromised 51. If the history file no longer exists 52, the history file has been truncated 53 or the history file is a symbolic link 54, the event is logged and information about the event is sent to the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 63

controller 55. The system examines the rhost file and other system authentication files to determine if dangerous security modifications to the host file have occurred 56. For example, entering a wildcard symbol, allows the host to allow anyone to log in without a password. If the host has been altered to allow anyone to login without a password or if other activity has occurred that may compromise security, the event is logged and information about the event is sent to the controller. The next step is to determine if the user's home directory contains one or more suspicious directories 59. Intruders will sometimes name a local directory in an odd way to hide their work. The system checks for known suspicious directories 60 and if it finds any, it may log the event and send information about the event to the controller 55. If a network computer process (sometimes called a "daemon") is left operating after logout 63 this could indicate a suspicious login and event is logged and information about the event is sent to the controller 55. The system checks to determine if the system audit records have been altered or are missing 66. If so, the control function is notified 55. Next the program checks an administrator generated list of generic files to see if one or more of them exists in the user's home directory 67. If so, the control function is notified. Next, if a suspicious directory name is found 68, the control function is notified 55. If an rhost file exists, the control function is notified 69.

Turning now to Fig. 6, a flow diagram of the port scan detector is shown. Port scanning is a method used by attackers to determine the vulnerabilities of a target host. Once vulnerabilities are found, a series of attacks are usually run to gain entry. Port scanning makes use of the TCP/IP protocol, which is the core communication protocol of the Internet. It allows machines to communicate throughout the world in a reliable manner. One of its features is the use of protocol "ports" on remote and originating systems to establish connections between hosts. The ports available on a host are usually between the ranges of 1 to 65535, with ports 1 to 1024 being what is commonly referred to as "reserved" for use by critical Internet services. Each port that presents a service to a remote user is usually registered with the Internet Assigned Numbers Authority registry (IANA). This registration ensures that programs know what ports to avoid or specifically connect to depending on the services being requested. Examples of commonly used ports are:

21 – File Transfer Protocol (FTP) services.

25 – Simple Mail Transfer Protocol (SMTP) services.

80 – HTTP services (WWW servers)

When an attacker is looking for a new host to penetrate, they will often begin by looking for Internet programs that have known exploitable problems. These programs (called

IPR2024-00027
CrowdStrike Exhibit 1002 Page 64

"daemons") vary in number and degree of susceptibility to problems. As new problems are found the hacker community quickly makes use of them to penetrate more hosts. To facilitate looking for new victims, the attacker will use a program that may either: connect to all ports on the remote machine or deliberately pick one or more ports to search for a particular problem. Some of the ports may not answer, in which case the attacker moves on. Other ports will answer and the attacker can then glimpse at what problems they can take advantage of. Often attackers will go from host to host on the Internet looking for the same problem to exploit. An example port scan of a host may return the following information:

```
localhost   telnet   23/tcp
localhost   smtp     25/tcp
localhost   finger   79/tcp
localhost   http     80/tcp
localhost   pop      110/tcp
localhost   imap     143/tcp
```

The port scan detector of the present invention alerts administrators that a person is actively looking for services on their host in a manner that indicates a hostile action. In the above port scan example our detector could present "fake" ports that an attacker will likely scan for. This could change the above port scan into the following:

```
localhost   fake   23/tcp  (Fake port)
localhost   smtp   25/tcp
localhost   fake   79/tcp (Fake port)
localhost   http   80/tcp
localhost   fake   110/tcp (Fake port)
localhost   fake   143/tcp (Fake port)
```

So even though in our example system there are only ports 25 and 80 active, the other ports will be tripwired by the port scan detector waiting for an attacker to unwittingly try to connect to them. When this occurs, the administrator or program can then take action to prevent this activity. In Figure 6, a flowchart of the port scan detector function 75 is shown. Internet ports (such as TCP and UCP) are monitored 76. If the port is in a list indicating that the port is not to be monitored 77, processing ends and no action is taken 78. If the port is in a list indicating it is to be monitored 77, the next step is to determine if the port is being used locally 79. If the port is being used locally it is temporarily removed from the monitored list until it is no longer used locally 80. If the port is not being used locally, the port is placed in the list of ports to be monitored 81. If the terminal or host computer where

IPR2024-00027
CrowdStrike Exhibit 1002 Page 65

the user is attempting to log in from is a terminal or host to be ignored for port scanning 82, no action is taken 78. If the terminal or host computer is not to be ignored 82, then if the number of ports that are being scanned is less than a minimum number of ports 83, no action is taken 78. If the number of ports that are potentially being scanned is greater than or equal to the minimum number of ports 83, the next step is to determine if the terminal or host computer where the user is attempting to log in from is already blocked from the system 84. If so, no action is taken 78. If not, information about the apparent port scan is sent to the controller 85 and the controller then takes the appropriate action as discussed in Figs. 8 and 9 below. The appropriate actions can vary from logging the event, blocking access to the computer system from the attacking host or executing a user-supplied command.

Turning now to Fig. 7, a flow diagram of the session monitoring function 90 is shown. For each user, the session monitor continuously monitors user activity for a threat to the computer system 91. It continuously monitors the user command entries 92, the system process accounting records 93, and commands entered by the user as stored in the user's command history file 94. It compares the command entries 92, system process accounting records 93 and commands in the user's command history file to known threat events and known attack patterns indicating a computer intrusion 95. If a match occurs 96, information and notification is sent to the control function 97. In either case the continuous session monitoring process continues its dynamic monitoring at step 91.

Turning now to Fig. 8, a flow diagram of the control function is shown. The controller 125 receives information about events and receives signature information to identify the user and type of event 126. Because the controller may be local to the system, the system can function in real time for suspicious events. In addition, if the controller is local, the intrusion detection system can be located entirely within the local host computer. The controller then determines the action to be taken and takes appropriate action 127. The action may be to log the event to the local system log 128, log the event to a remote system log 129, disable the user's account 130, block access to the attacking host system address 131, trigger a user defined event 132, drop the route to the offending system 133, block network access from the offending system 134, notify the system administrator 135, to ignore the event 136 or any combination of these actions. If the controller is a local 137, the information can be sent to a local system controller 138. If the controller is not local 137, the information can be sent to the central system controller 139, which then takes the appropriate actions (127-136) instead of the local controller.

Fig. 9 shows a block diagram of an alternate embodiment of a host based intrusion detection system having a central system controller. The central system controller 150

IPR2024-00027
CrowdStrike Exhibit 1002 Page 66

may be part of a network that contains multiple host computers (1 through N) 151-153. Each host 151-153 comprises a local controller that sends information about log auditing, login anomaly detection, logout anomaly detection, session monitoring and port scan detector functions to the central controller. The central controller can perform centralized auditing of events 154, data analysis 155, cross correlation of intrusion activity throughout the network 156 and can alert the network system administrator 157 if anomalous activity if found. In addition, the central controller 150 can send information about anomalous activity found within the system back to the multiple hosts 151-153 so as to alert the hosts.

Fig. 10 is a flow diagram of a program set up for the intrusion detection system. Prior to initialization of the intrusion detection system, the system administrator 161 may select program functions to run in the intrusion detection system. For example, the system administrator may select the log auditing function 162, login anomaly detection 163, logout anomaly detection 164, session monitor 165 and port scan detector 166. The system administrator may also select the actions to be taken by the control function if an unauthorized user or event occurs 167. If the system administrator chooses not to select functions, the preprogrammed default functions will run. If the system administrator chooses not to select the actions to be taken by the control function or only changes some of the actions, the preprogrammed default actions will used. The system administrator may also alter the alarm thresholds or use preprogrammed alarm thresholds 168. The system administrator may select whether a warning is to be displayed on the system administrators graphical user interface 169. The system administrator may also select whether a local or central controller will be used for reporting and for taking action 170.

Fig. 11 is a block diagram of the software modules of the Login Anomaly Detector. The Login Anomaly Detector 3 comprises a login audit module 180, first login warning module 181, a foreign domain warning module 182, a multiple concurrent logins module 183 and an odd login time module 184.

The login audit module 180 logs all user logins into the target host computer. This information is recorded in the system audit records and the intrusion detection system also records this information. The login audit module 180 provides secondary audit trail of user activity in case the system audit files are damaged or altered.

The first login warning module 181 notifies the control function and or administrators whenever a user logs into a host for the first time. After the first login the module will no longer activate. The first login warning module 181 detects a first time login by noting whether the user has more than one login in the dynamic user database. It is used to spot users who are not authorized to connect to the computer system.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 67

The foreign domain warning module 182 notifies administrators whenever a "foreign" domain login is detected by a user. The foreign domain warning module 182 checks the Internet domain the user is logging in from as indicated in their user database login record. If the domain is not listed in an "allowed" file, it assumes the domain is "foreign" and notifies the administrator. The foreign domain warning module 182 allows an administrator to spot logins from odd places that are not allowed to connect to the target host computer.

The multiple concurrent logins module 183 watches user logins and looks for concurrent logins from multiple domains. When a user is logged in from multiple domains at the same time, this may be an indication of suspicious activity. The multiple concurrent logins module 183 module will monitor logins and if it spots a user that is logged in more than once or is logged in from two or more separate domains/hosts, it will notify a control function and or the system administrator. Hackers often will log in from multiple hosts across the Internet or within a local network. The multiple concurrent logins module 183 attempts to detect this condition.

The odd login time module 184 monitors user logins and attempts to spot "unusual" login times based on past data collected for this user. Odd login times are one of the primary indicators of unauthorized system intrusion. The odd login time module 184 runs only after a predetermined amount of user logins have been collected by the user database. This amount defaults to ten logins, but can be adjusted by the user or system administrator to begin comparing login times after any amount has passed, although sufficient time should be granted to allow accurate profiling. The theory of operation is to take the average login hours from the login tracking field for a particular user. This average is used to draw conclusions about the user's login habits including the days they log into the computer, the times they log into the computer and how long they stay logged into the computer.

This data can be obtained because the login stamp for each login tracking entry is dynamically maintained by the login monitoring process. Because the database is dynamically generated the signature can be built with intelligence to take advantage of this fact to reduce false alarms. This relieves the administrator of having to setup predefined login profiles for users. This is a great benefit if you have an eclectic user base who work strange hours or login from multiple time zones. The values derived are obtained by calculating the days/hours/minutes. These values can be combined in a number of ways to determine normal patterns of behavior such as days of the week the user is active and for how long they use the system at a time. Further pieces of information can be derived such as average hours online per day and the hours they are normally working between.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 68

Fig. 12 is a block diagram of the software modules of the Logout Anomaly Detector. The Logout Anomaly Detector 7 comprises a logout audit module 190, suspicious entries in user's home directory 191, generic file exists module 192, history file truncated/altered module 193, suspicious directory name module 14, altered/missing audit record module 195, network process active module 196, suspicious history file commands module 197, and rhost file exists module 198.

The logout audit module 190 logs all user logouts from the target host computer. This information is recorded in the system audit records and the intrusion detection system also records this information. The logout audit module 190 provides secondary audit trail of user activity in case the system audit files are damaged or altered.

The suspicious entries in user's home directory module 191 checks the for a ".rhost" file in the user's home directory with a dangerous entry. Dangerous entries include wildcard characters. If a wildcard character is found, the suspicious entries in user's home directory module 191 alerts the administrator that a dangerous .rhost file exists. Dangerous entries indicate suspicious activity for most users and may allow the host system to be easily compromised by remote attackers.

The generic file exists module 192 module checks an administrator-generated list of files to see if one or more of them exist in the user's home directory. This module allows an administrator to flag certain files for monitoring (password files, etc.) and generate custom alerts. A file list is used to parse against the user's directory listing. If a matching file name is found the event if flagged and the control function and or system administrator alerted.

The history file truncated/altered module 193 module checks a user's command history file for alterations or truncations. Hackers often alter the history file to conceal activity on a host. The module checks to determine if the history file is truncated to zero bytes long, is missing or deleted and if the history file is a "symbolic link" to another file or device. If the history file indicates that any of these conditions have occurred, this may that unauthorized activity is being hidden on many attacked hosts. When an altered history file is found it is reported to central controller and system the administrator may be notified.

The suspicious directory name module 194 detects suspicious directory names. Hackers will often employ odd directory names in order to hide activity on a host. This module searches for common directory name hiding tactics. For example, this module will check a user's home directory for odd directory names such as: ".. ", "...", etc. and notify the control function and report them to the administrator if one is found. The directory names that can be searched for are configurable by the administrator.

The altered/missing audit record module 195 checks to determine if an entry for the user's session is missing from the systems audit records (such as utmp, wtmp, event logs

IPR2024-00027
CrowdStrike Exhibit 1002 Page 69

or the like). Intruders will use commonly available utilities to purge audit records of their tracks. An altered or missing entry may indicate that the user has deleted or altered their system audit record entry to avoid detection. The altered/missing audit record module 195 checks the pertinent system accounting records (utmp, wtmp, event logs or the like) and ensures that a matching record exists for the session that the user is logging out of. A missing record indicates that the user may be trying to conceal their activity on the host. This module attempts to analyze the login/logout records of the host to spot this condition and notify the administrator.

The network process active module 196 checks to determine if a user has logged out of the system and left a process running with a listening network. A listening socket on a network host is a program that is offering services to the Internet. Typically a socket is a number in the range of about 1-65535 that a user can connect to using one of the Internet protocols. An example of common socket numbers include: 23 for Telnet services, 25 for mail services and 80 for worldwide web services. If a user executes a rogue program with a listening socket they can connect back to the system from anywhere on the Internet and bypass normal authentication and audit procedures of the target host. Therefore, programs with listening network sockets run by ordinary users are a security threat and should not be allowed on any host. This could indicate that an unauthorized program is running. The network process active module 196 checks the system process table for network socket programs and alerts the controller and system administrator to this fact.

The suspicious history file commands module 197 checks to determine if the history files contain commands that could be considered "suspicious". Most Unix systems have the ability to store a "command history" of all logged in users. This command history is used by the interactive shells to allow users to quickly recall commands either manually or with a script. As part of this feature the interactive shells (depending on their type) will write a "history" file in the user's home directory. Usually this file is called ".history" but other variations exist. The basic operation occurs when a user logs out of the host system. During this time the login file is flushed to the disk and the login monitor process can begin reading the most current command history. This module then takes the .history file and compares the command contents to a known database of unusual or suspicious commands. Such commands may include commands that indicate the user has viewed the master password list on the system, the user has added a global wildcard to the system which will allow all hosts access to the system, the user has tried to copy the password file list and the user has tried to switch to system administrator credentials. When suspicious commands are spotted, they are flagged and the appropriate actions can be taken by the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 70

monitor program to alert the controller and system administrator or disable the user account.

The .rhost file exists module 198 checks for the existence of a ".rhost" file in the user's home directory. The .rhost files contains a list of hosts that are trusted by the system when a login is detected originating from them. If a user is originating form a host that is listed in the .rhost file, then they are allowed access to the system as long as their account name matches the owner of the .rhost file. If an entry such as "++" is placed in the .rhost file, it signifies a wildcard and any host can log into the system as the owner of the rhost file. A wildcard entry is always suspicious and is virtually always a dangerous modification. If an .rhost file exists in the user's directory, and is greater than zero bytes, then the controller is notified and the system administrator may be alerted.

Although the present invention has been described in detail with reference to certain preferred embodiments, other embodiments are possible. Therefore, the spirit and scope of the appended claims should not be limited to the description of the preferred embodiments herein.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 71

Claims

What is claimed is:

1. A computer implemented method for detecting intruders in a computer system (1), the method comprising the steps of:
   a. detecting an unauthorized user attempting to enter into a computer system (20) by comparing actions of the user to a dynamically built profile for the user (22), and if the action is out of range of the user profile, notifying a control function (37);
   b. detecting events that indicate an unauthorized entry into the computer system (49, 75, 90) has occurred and if an event occurs that indicates unauthorized entry, notifying a control function (55, 85, 97); and
   c. executing an action (127) by the control function (125).
2. The method of claim 1 wherein the dynamically built user profile comprises:
   a. dynamically constructing a user profile (22) for each computer user when the computer user first attempts to log into the computer system (24, 36);
   b. dynamically updating the user profile for the user for each attempt by the user to log into the system after the first attempt (25, 36); and
   c. updating the user profile when the user logs out of the computer system (50).
3. The method of claim 1 further comprising dynamically monitoring computer system log files (10) for events that indicate an unauthorized attempted entry into the computer system.
4. The method of claim 3 wherein the dynamically monitoring system log files comprises:
   a. comparing the system log files to events to ignore and ignoring the event if the system log file indicates a match with an event to ignore (14); and
   b. comparing the system log files to events known to indicate an unauthorized entry event into the computer system (12) and notifying a control function about the unauthorized entry event;
   c. executing the action in response to the event by the control function (17).
5. The method of claim 1 further comprising:
   a. dynamically monitoring user actions after the user has logged into a computer system for unauthorized access by the user to system information (92), and if unauthorized access event occurs, notifying a control function (97) about the unauthorized access and automatically executing a specific action in response to the event by the control function (127); and
   b. dynamically monitoring user actions after the user has logged into a computer system for corruption of system information by the user (56, 59) and if a corruption of system information occurs, notifying a control function of the corruption of system information and executing the action in response by the control function (127).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 73

6. The method of claim 1 further comprising:
   a. scanning network ports (76) to determine if a user has connected to more than a selected number of network ports (83);
   b. if the user has exceeded the selected number of network ports (83), notifying the control function and executing an action in response by the control function (85).

7. The method of claim 6 wherein the selected number of network ports is set by the system administrator (161).

8. The method of claim 1 wherein the detecting events (49, 75, 90) that indicate an unauthorized entry into the computer system comprises:
   a. detecting anomalous events when a user logs out of the computer system (49) comprising:
      i. monitoring a user's file history to determine if the user's file history has been altered (51);
      ii. monitoring computer system files to determine if a modification has been made that indicates an unauthorized intrusion into the computer system (56);
      iii. monitoring a user's computer files to determine if a modification has been made that indicates an unauthorized intrusion into the computer system (59);
      iv. determining if a program has been left running that should have stopped running when the user logs out of the computer system (63); and
   b. if an anomalous event has been detected:
      i. notifying the control function about the anomalous event (55); and
      ii. allowing the control function to take action in response to the anomalous event (127).

9. The method of claim 1 wherein the detecting unauthorized users (20) comprises:
   a. if the user has attempted to log in from a computer host that is not allowed access to the computer system, notifying the control function about the attempted login (26); and
   b. allowing the control function to take action in response (37).

10. The method of claim 1 wherein the detecting unauthorized users (20) comprises:
   a. if the user attempts to log into the computer system and has an active login (28), checking to determine if the user is allowed to have more than one login active simultaneously (29), and if not notifying a control function about the attempted login (37); and
   b. executing an action by the control function (127).

11. The method of claim 2 wherein the dynamically constructed user profile for each computer user (22) is selected from the group consisting of storing user name, login

IPR2024-00027
CrowdStrike Exhibit 1002 Page 74

terminal, time of creation of initial user profile, time of user's first login, time history of the user's logins, time periods that the user is allowed to log into the system and total number of logins for the computer user (24).

12. The method of claim 11 wherein the user profile is stored in a user profile database (36).

13. The method of claim 11 wherein dynamically updating the user profile for the user (22) comprises, for each user selecting from the group consisting of entering a current login time, login terminal, updating a time history of a user's login and incrementing the total number of logins (24, 25).

14. The method of claim 11 wherein the detecting unauthorized users (20) comprises if the user has attempted to log in at a time different from the time periods (30) that the user is allowed to log into the computer system, notifying a control function (37) about the attempted login and allowing the control function to take action in response (127).

15. The method of claim 1 further comprising:

a. dynamically constructing a list of active users logged into the computer system (36); and

b. dynamically updating the list of active users when a user logs into the system and logs out of the system (50).

16. The method of claim 15 wherein the list of active users (36) comprises information selected from the group consisting of user name, user terminal and time of user login (24).

17. The method of claim 1 wherein the control function (125) comprises:

a. storing information about unauthorized users and events that indicate an unauthorized entry into the computer system (126);

b. taking action in response to the unauthorized users and events, the action is selected from the group consisting of:

i. logging the information in a local controller (128);

ii. sending the information to a network controller (139);

iii. disabling the unauthorized user's account (130);

iv. blocking access to the computer system for the user (131);

v. notifying a system administrator (135); and

vi. ignoring the unauthorized user and unauthorized entry (136).

18. The method of claim 17 wherein the action taken is defined by the system administrator prior to initialization of the intrusion detection system (161).

19. The method of claim 1 wherein the control function (125) is located in a local computer (137) where the unauthorized user and unauthorized entry occurred.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 75

20. The method of claim 19 further comprising the control function (125) in the local computer sends information about unauthorized users and anomalous events to a central computer (139) connected to the local computer (137).

21. The method of claim 1 wherein the control function (125) is located in a central computer (139) connected to the local computer (137).

22. The method according to claim 21 further comprising multiple local computers (151-153) connected to the central computer (150).

23. The method of claim 21 wherein the control function (125) in the central computer (150) comprises:

a. performing centralized analysis of unauthorized users and events (154, 155);
b. performing correlation of unauthorized users and events from the multiple local computers (156);
c. alerting a central computer system administrator (157); and
d. sending the analysis and correlation results to the multiple local computers (151-153).

24. The method of claim 1 further comprising:

a. for each user, continuously monitoring user activity for a threat to the computer system (90); and
b. the continuously monitoring comprises analyzing user command entries (92) and comparing the entries to known threat events and known attack patterns (95) indicating a computer intrusion and if a match occurs (96), notifying the control function (97) and allowing the control function to take action in response (127).

25. The method of claim 24 further comprising continuously monitoring the system process accounting records (93) and comparing the entries to known threat events and known attack patterns (95) indicating a computer intrusion and if a match occurs (96), notifying the control function (97) and allowing the control function to take action in response (127).

26. The method of claim 24 further comprising continuously monitoring commands (94) entered by the user and comparing the commands to known threat events and known attack patterns (95) indicating a computer intrusion and if a match occurs (96), notifying the control function (97) and allowing the control function to take action in response (127).

27. The method of claim 1 further comprising continuously monitoring network port activity (76) and comparing the activity to known threat events and known attack patterns indicating a computer intrusion (83, 84) and if a match occurs, notifying the control function (85) and allowing the control function to take action in response (127).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 76

28. The method of claim 22 wherein the action taken is selected from the group consisting of logging the event (128, 129), disabling a user account (130), blocking access to the system (131), initiating a user defined action (132), dropping a route to an attacking system (133), dropping a route to an attacking user (133), blocking access from an offending system (134), notifying a system administrator (135) and ignoring the event (136).

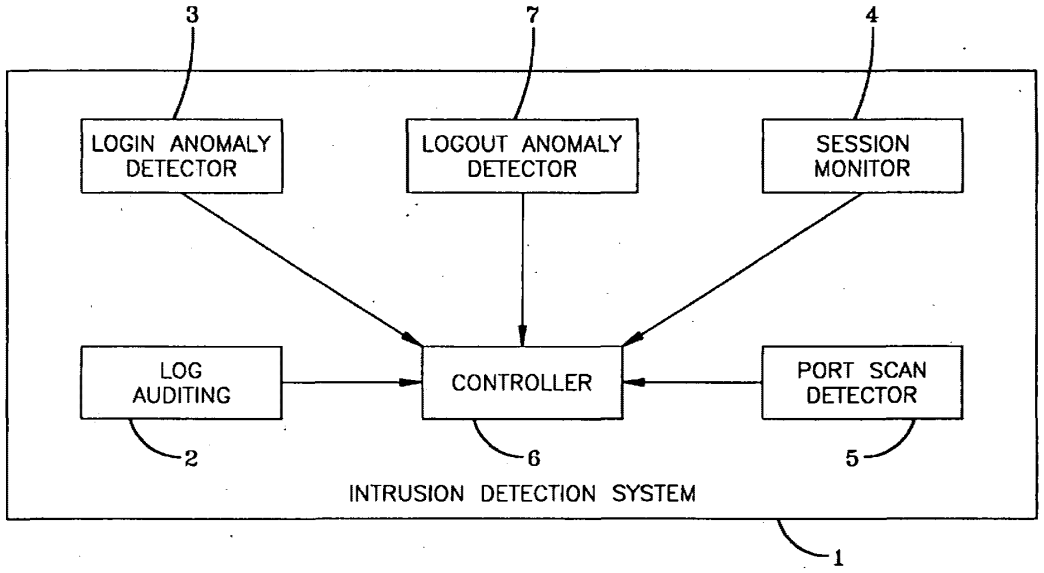
29. The method as in any of claims 1, 4-6, 8, 9, 14, 17, 24-27 wherein the action comprises a user specified action (167).

30. The method of claim 29 wherein the user specified action is entered by a system administrator (161).

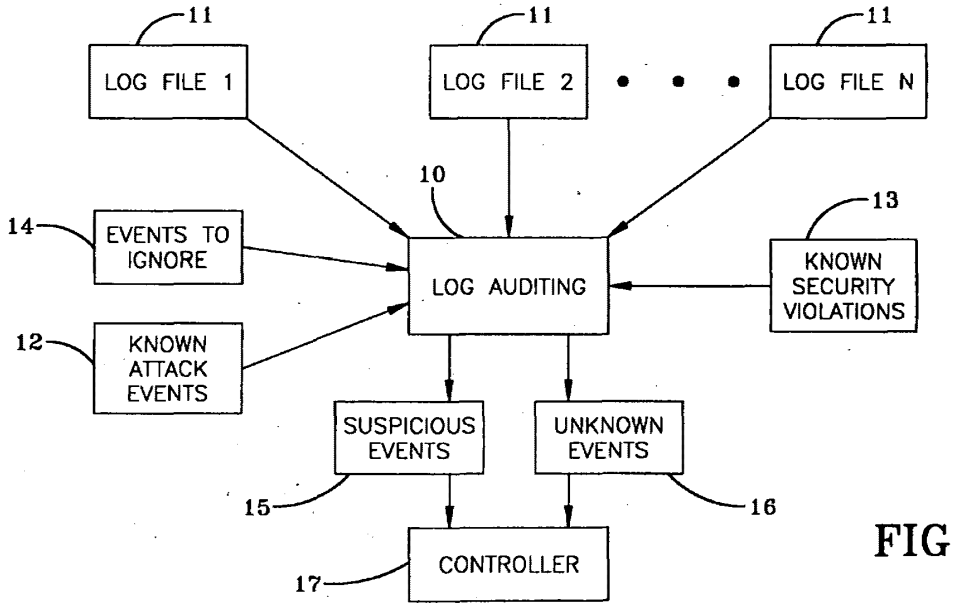
31. The method as in any of claims 1, 4-6, 8, 9, 14, 17, 24-27 wherein the action is automatically executed by the control function (127).

32. Computer executable software code stored on a computer readable medium incorporating the method as recited in any of claims 1 through 31.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 77


3
7
4
LOGIN ANOMALY DETECTOR
LOGOUT ANOMALY DETECTOR
SESSION MONITOR
LOG AUDITING
CONTROLLER
PORT SCAN DETECTOR
2
6
5
INTRUSION DETECTION SYSTEM
1

FIG–1


11
LOG FILE 1
11
LOG FILE 2
11
LOG FILE N
10
14
EVENTS TO IGNORE
LOG AUDITING
13
KNOWN SECURITY VIOLATIONS
12
KNOWN ATTACK EVENTS
SUSPICIOUS EVENTS
UNKNOWN EVENTS
15
16
CONTROLLER
17

FIG–2

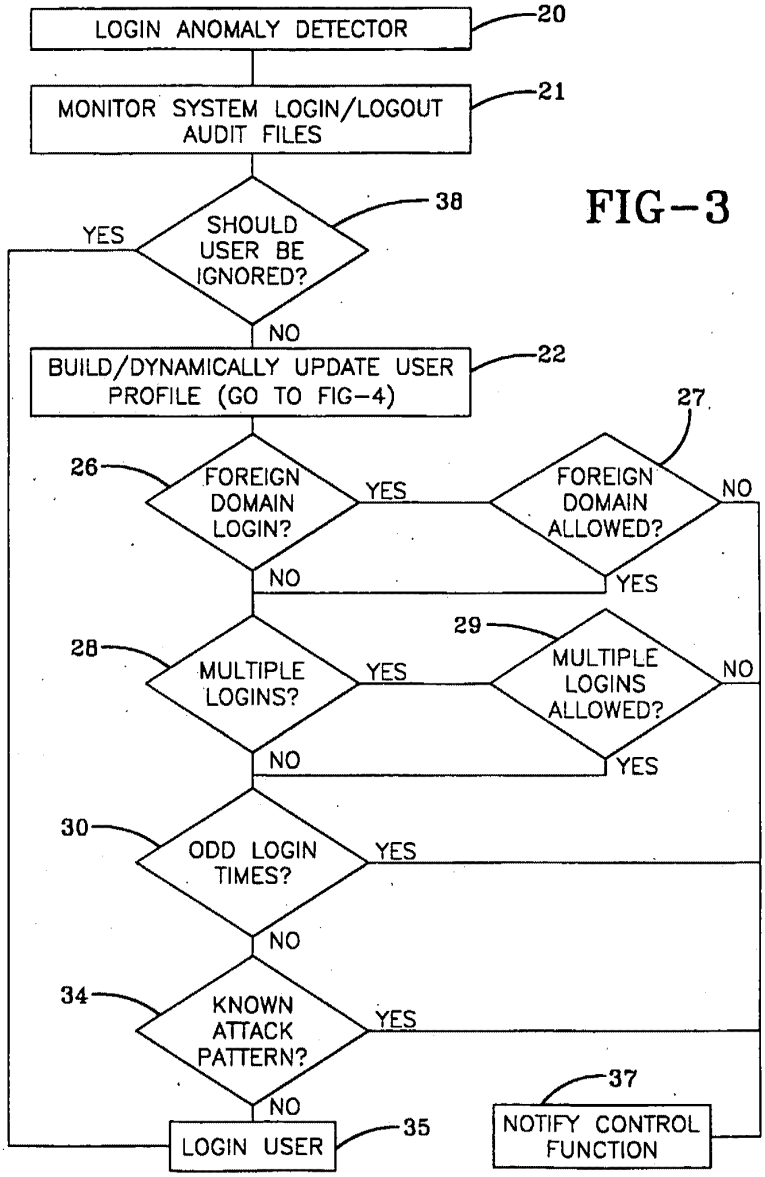

LOGIN ANOMALY DETECTOR
20
MONITOR SYSTEM LOGIN/LOGOUT AUDIT FILES
21
SHOULD USER BE IGNORED?
38
YES
NO
BUILD/DYNAMICALLY UPDATE USER PROFILE (GO TO FIG-4)
22
27
26
FOREIGN DOMAIN LOGIN?
YES
FOREIGN DOMAIN ALLOWED?
NO
NO
YES
29
28
MULTIPLE LOGINS?
YES
MULTIPLE LOGINS ALLOWED?
NO
NO
YES
30
ODD LOGIN TIMES?
YES
NO
34
KNOWN ATTACK PATTERN?
YES
NO
37
LOGIN USER
35
NOTIFY CONTROL FUNCTION

FIG-3

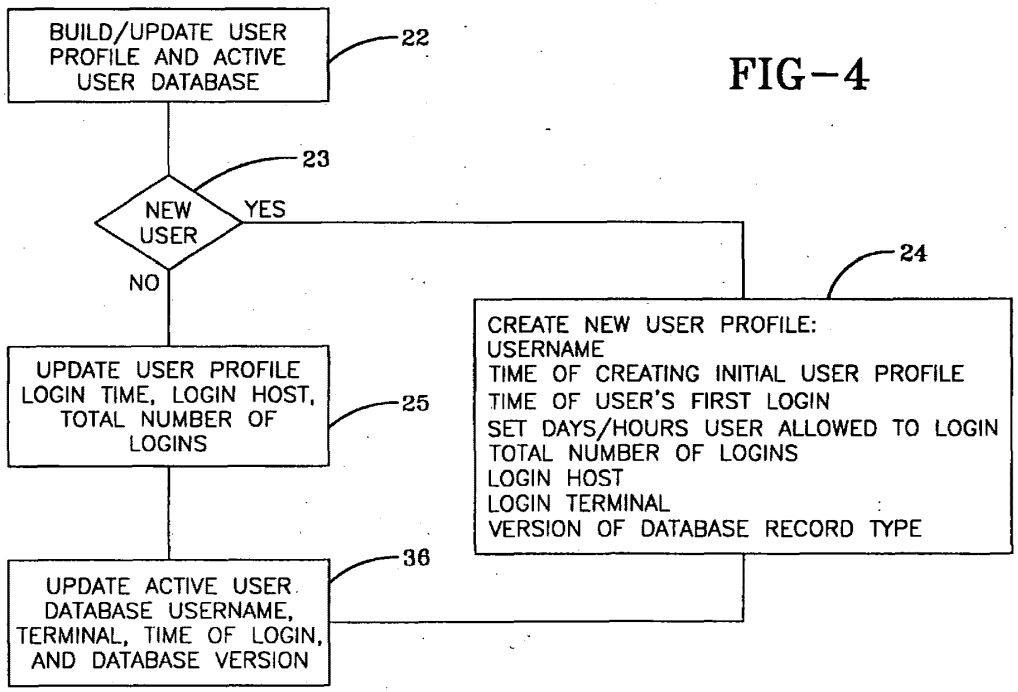

BUILD/UPDATE USER PROFILE AND ACTIVE USER DATABASE
22
NEW USER
23
YES
NO
24
CREATE NEW USER PROFILE:
USERNAME
TIME OF CREATING INITIAL USER PROFILE
TIME OF USER'S FIRST LOGIN
SET DAYS/HOURS USER ALLOWED TO LOGIN
TOTAL NUMBER OF LOGINS
LOGIN HOST
LOGIN TERMINAL
VERSION OF DATABASE RECORD TYPE
UPDATE USER PROFILE LOGIN TIME, LOGIN HOST, TOTAL NUMBER OF LOGINS
25
UPDATE ACTIVE USER DATABASE USERNAME, TERMINAL, TIME OF LOGIN, AND DATABASE VERSION
36

FIG-4

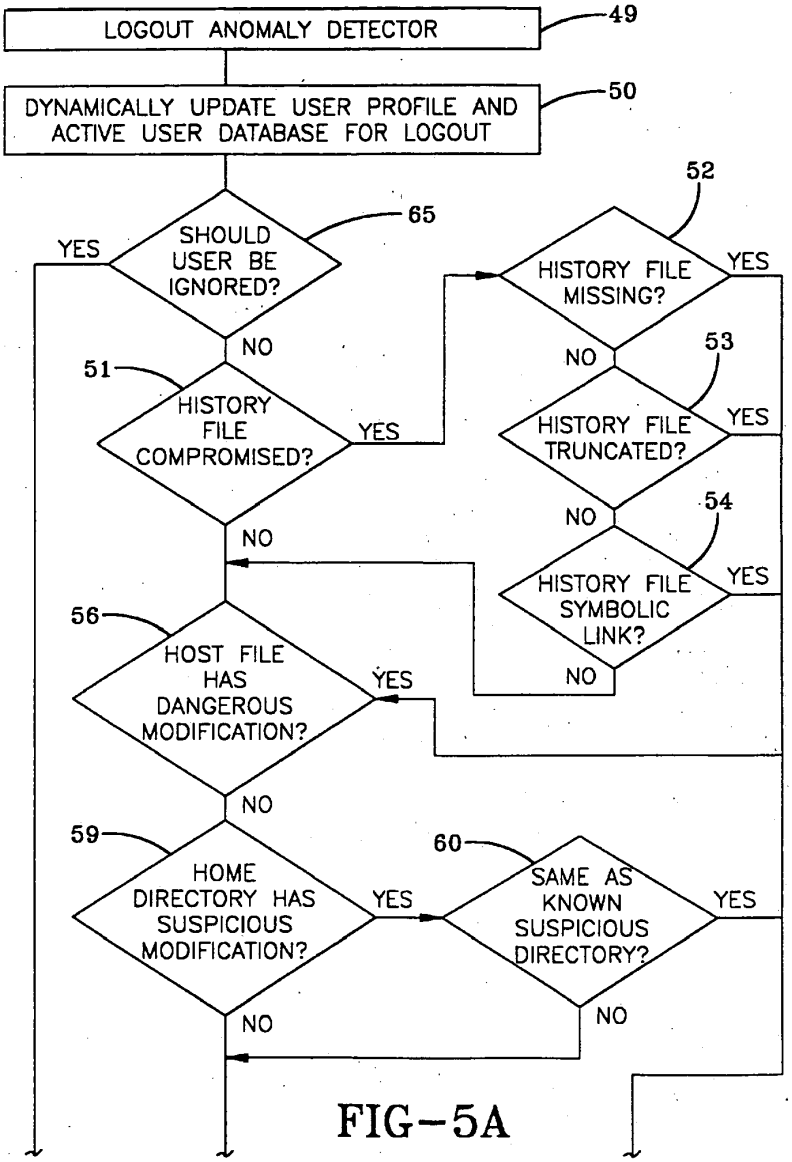
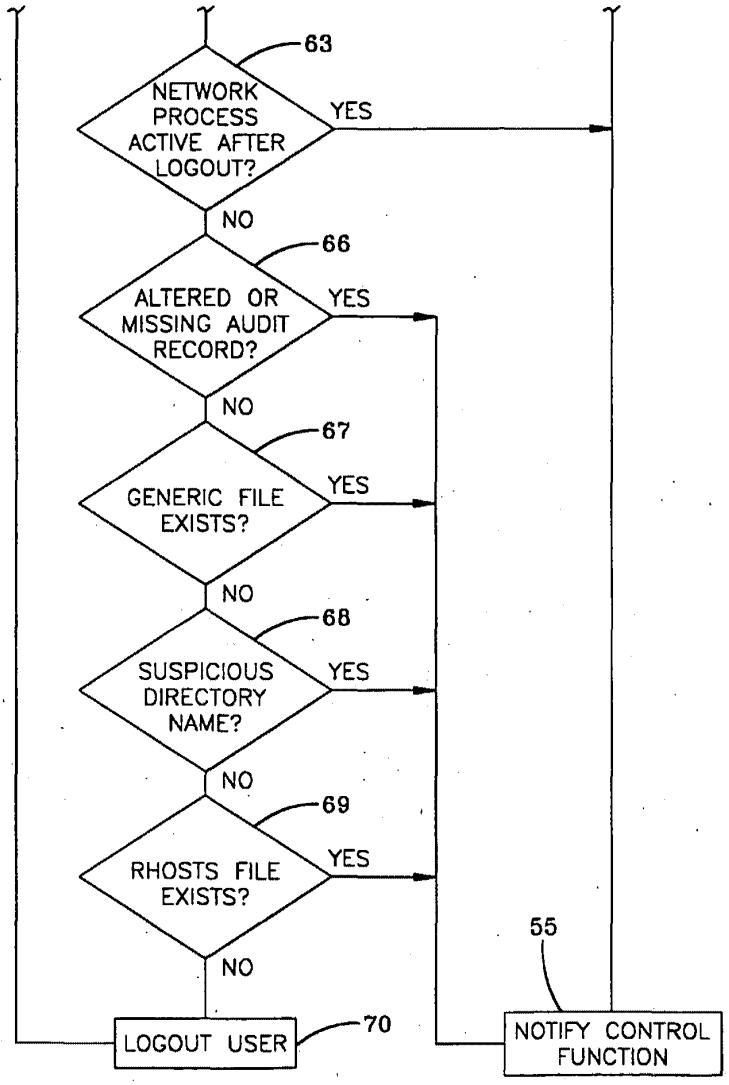

LOGOUT ANOMALY DETECTOR
49
DYNAMICALLY UPDATE USER PROFILE AND ACTIVE USER DATABASE FOR LOGOUT
50
52
SHOULD USER BE IGNORED?
65
YES
NO
HISTORY FILE MISSING?
YES
NO
51
HISTORY FILE COMPROMISED?
YES
53
HISTORY FILE TRUNCATED?
YES
NO
54
HISTORY FILE SYMBOLIC LINK?
YES
NO
NO
56
HOST FILE HAS DANGEROUS MODIFICATION?
YES
NO
59
HOME DIRECTORY HAS SUSPICIOUS MODIFICATION?
YES
60
SAME AS KNOWN SUSPICIOUS DIRECTORY?
YES
NO
NO

FIG-5A


63
NETWORK PROCESS ACTIVE AFTER LOGOUT?
YES
NO
66
ALTERED OR MISSING AUDIT RECORD?
YES
NO
67
GENERIC FILE EXISTS?
YES
NO
68
SUSPICIOUS DIRECTORY NAME?
YES
NO
69
RHOSTS FILE EXISTS?
YES
NO
55
NOTIFY CONTROL FUNCTION
70
LOGOUT USER

FIG–5B

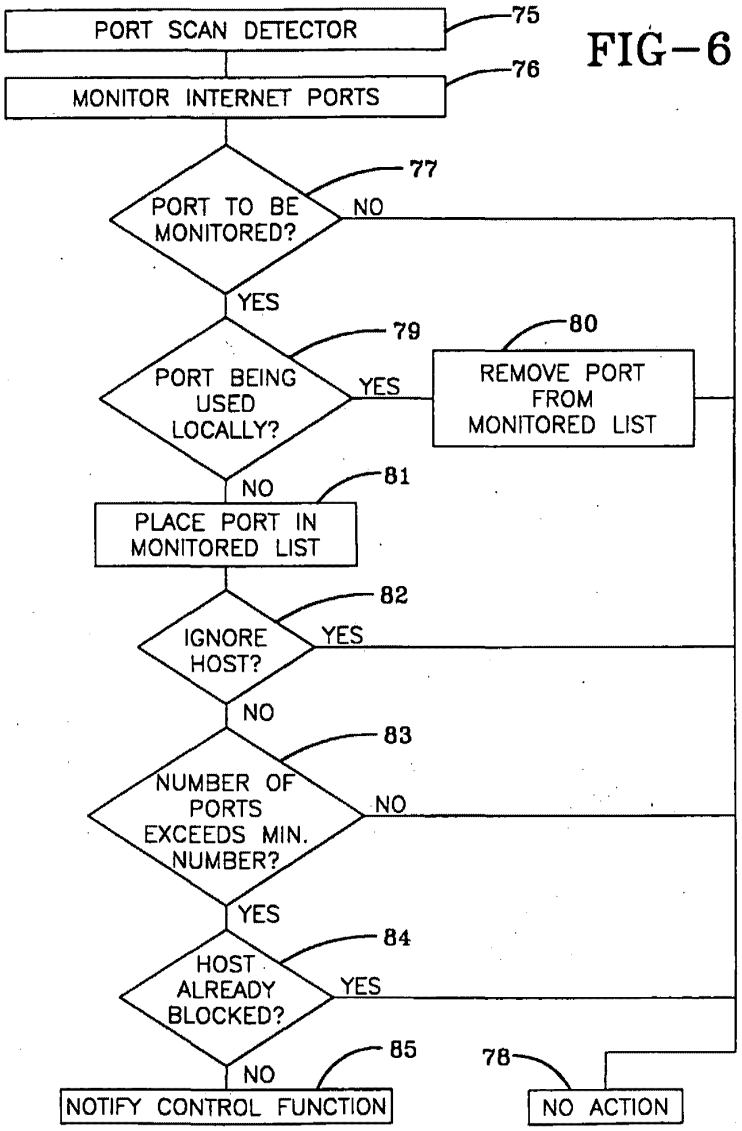

PORT SCAN DETECTOR
75
MONITOR INTERNET PORTS
76
PORT TO BE MONITORED?
77
NO
YES
PORT BEING USED LOCALLY?
79
YES
REMOVE PORT FROM MONITORED LIST
80
NO
PLACE PORT IN MONITORED LIST
81
IGNORE HOST?
82
YES
NO
NUMBER OF PORTS EXCEEDS MIN. NUMBER?
83
NO
YES
HOST ALREADY BLOCKED?
84
YES
NO
NOTIFY CONTROL FUNCTION
85
NO ACTION
78

FIG-6

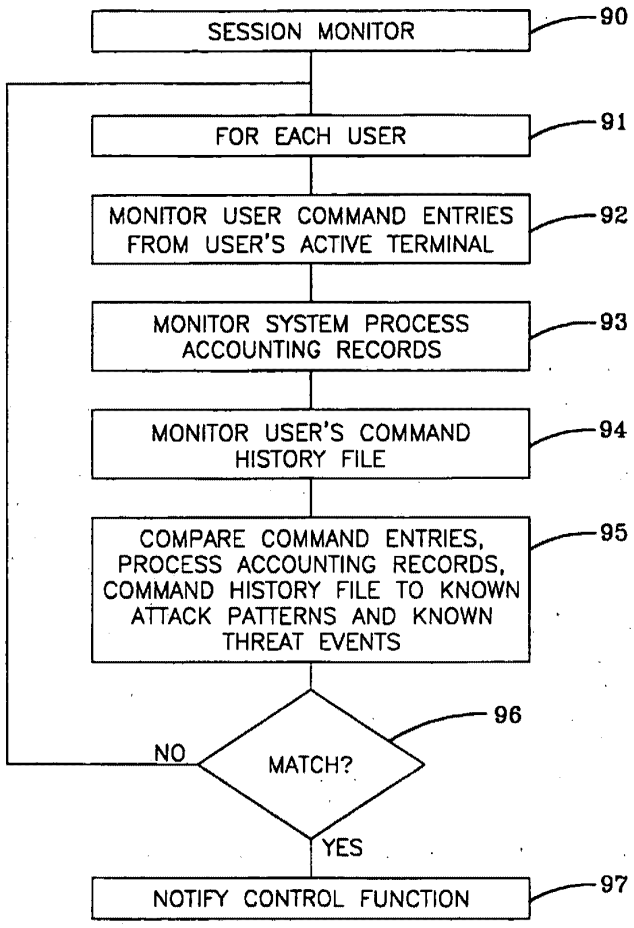

SESSION MONITOR
90
FOR EACH USER
91
MONITOR USER COMMAND ENTRIES FROM USER'S ACTIVE TERMINAL
92
MONITOR SYSTEM PROCESS ACCOUNTING RECORDS
93
MONITOR USER'S COMMAND HISTORY FILE
94
COMPARE COMMAND ENTRIES, PROCESS ACCOUNTING RECORDS, COMMAND HISTORY FILE TO KNOWN ATTACK PATTERNS AND KNOWN THREAT EVENTS
95
MATCH?
96
NO
YES
NOTIFY CONTROL FUNCTION
97

FIG-7

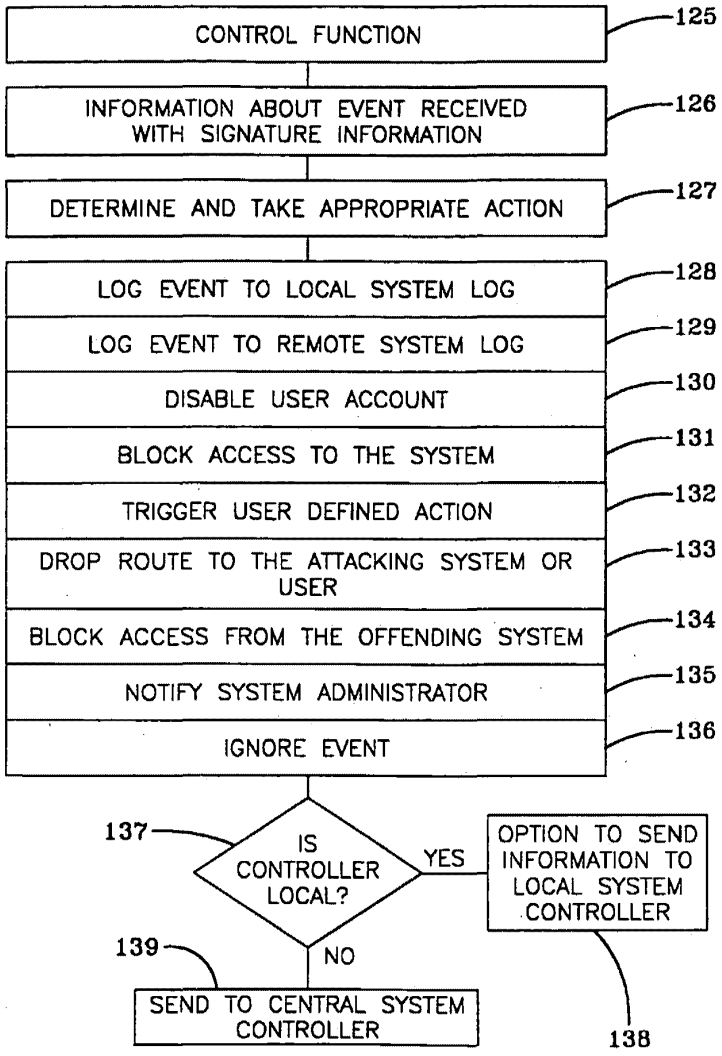
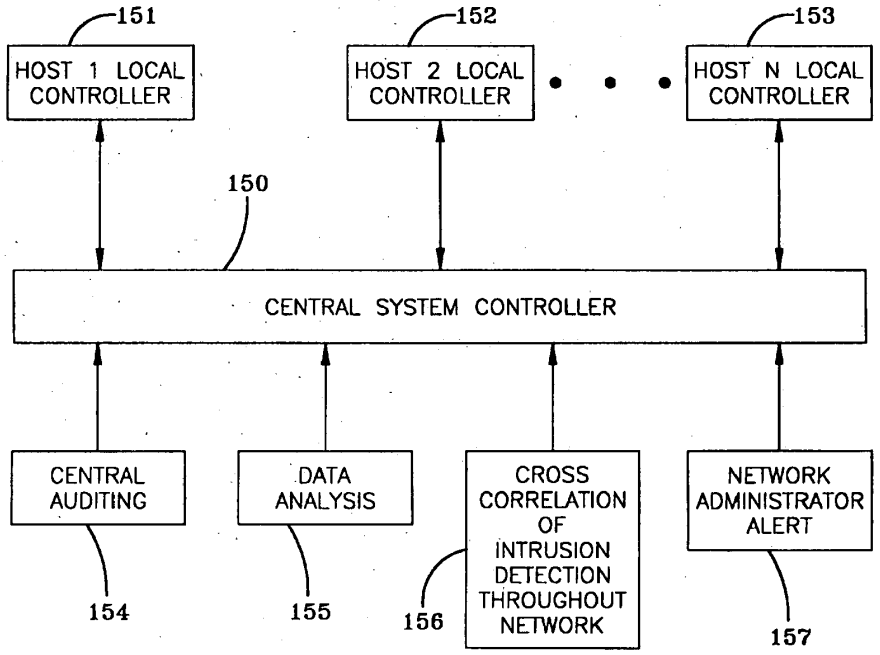

CONTROL FUNCTION
125
INFORMATION ABOUT EVENT RECEIVED WITH SIGNATURE INFORMATION
126
DETERMINE AND TAKE APPROPRIATE ACTION
127
LOG EVENT TO LOCAL SYSTEM LOG
128
LOG EVENT TO REMOTE SYSTEM LOG
129
DISABLE USER ACCOUNT
130
BLOCK ACCESS TO THE SYSTEM
131
TRIGGER USER DEFINED ACTION
132
DROP ROUTE TO THE ATTACKING SYSTEM OR USER
133
BLOCK ACCESS FROM THE OFFENDING SYSTEM
134
NOTIFY SYSTEM ADMINISTRATOR
135
IGNORE EVENT
136
137
IS CONTROLLER LOCAL?
YES
OPTION TO SEND INFORMATION TO LOCAL SYSTEM CONTROLLER
138
NO
139
SEND TO CENTRAL SYSTEM CONTROLLER

FIG-8



FIG–9

IPR2024-00027
CrowdStrike Exhibit 1002 Page 87

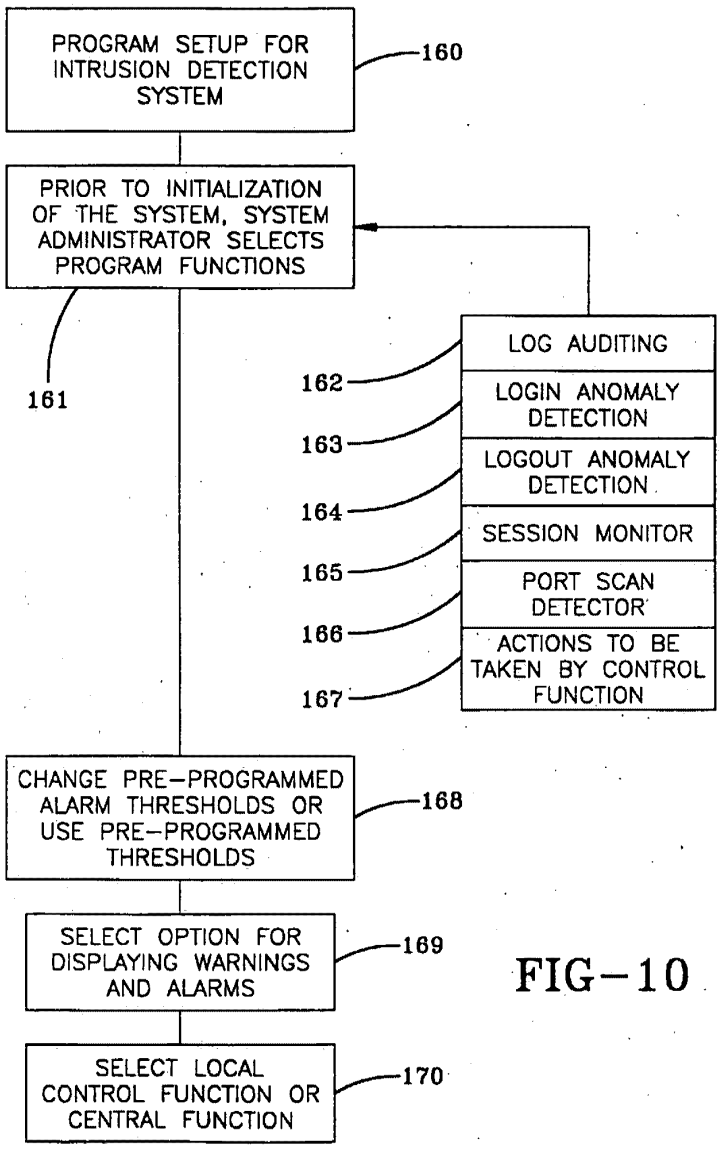

PROGRAM SETUP FOR INTRUSION DETECTION SYSTEM
160
PRIOR TO INITIALIZATION OF THE SYSTEM, SYSTEM ADMINISTRATOR SELECTS PROGRAM FUNCTIONS
161
LOG AUDITING
162
LOGIN ANOMALY DETECTION
163
LOGOUT ANOMALY DETECTION
164
SESSION MONITOR
165
PORT SCAN DETECTOR
166
ACTIONS TO BE TAKEN BY CONTROL FUNCTION
167
CHANGE PRE-PROGRAMMED ALARM THRESHOLDS OR USE PRE-PROGRAMMED THRESHOLDS
168
SELECT OPTION FOR DISPLAYING WARNINGS AND ALARMS
169
SELECT LOCAL CONTROL FUNCTION OR CENTRAL FUNCTION
170

FIG-10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/06313

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :H04L 9/32
US CL : 713/200, 164; 709/229
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)
U.S. : 713/200, 164; 709/229

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 practicable, 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	US 5,825,750 A (THOMPSON) 20 October 1998, abstract, column 2 lines 14-19, column 5 lines 14-26, figure 3, column 4 lines 54-67, column 1 lines 47-53, column 4 lines 23-67, column 6 lines 36-51	1-9, 11-19, 24-32
Y,P	US 5,991,881 A (CONKLIN et al) 23 November 1999, abstract, column 1 lines 50-65, figure 6, column 4 lines 9-43, column 5 lines 23-24, column 6 lines20 and 28-33	1-9, 11-19, 24-32
X,E	US 6,070,244 A (ORCHIER et al) 30 May 2000, abstract, column 12 lines 51-59	10, 20-23
A,E	US 6,073,240 A (KURTZBERG et al) 06 June 2000, disclosure	1-32

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"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
"A"	document defining the general state of the art which is not considered to be of particular relevance	"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
"B"	earlier document published on or after the international filing date	"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
"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)	"&"	document member of the same patent family
"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		

Date of the actual completion of the international search 14 JUNE 2000	Date of mailing of the international search report 03 AUG 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer GAIL HAYES Joni Hill Telephone No. (703) 308-3900

Form PCT/ISA/210 (second sheet) (July 1998)*

IPR2024-00027
CrowdStrike Exhibit 1002 Page 89

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/06313

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 5,495,521 A (RANGACHER) 27 February 1996, disclosure	1-32
A,E	US 6,044,465 A (DUTCHER et al) 28 March 2000, disclosure	1-32

Form PCT/ISA/210 (continuation of second sheet) (July 1998)*

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 January 2002 (24.01.2002)

PCT

(10) International Publication Number
WO 02/06928 A2

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

(21) International Application Number: PCT/US01/19142

(22) International Filing Date: 14 June 2001 (14.06.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/218,489 14 July 2000 (14.07.2000) US
09/642,625 18 August 2000 (18.08.2000) US

(71) Applicant: VCIS, INC. [US/US]; 522 Erskine Drive, Pacific Palisades, CA 90272 (US).

(72) Inventor: VAN DER MADE, Peter, A., J.; 17 Nooal Street, Newport Beach, NSW 2106 (AU).

(74) Agents: WRIGHT, William, H. et al.; Hogan & Hartson L.L.P., Biltmore Tower, Suite 1900, 500 South Grand Avenue, Los Angeles, CA 90071 (US).

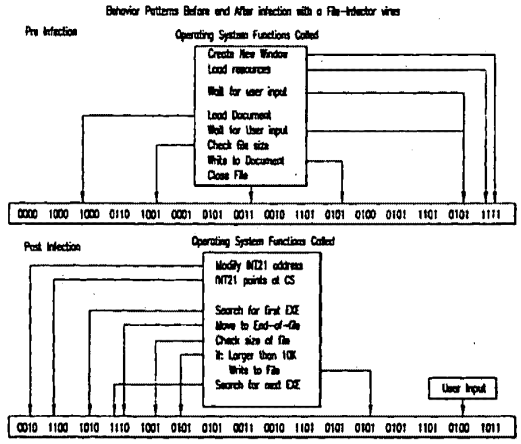
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU,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



(57) Abstract: An automated analysis system detects malicious code within a computer system by generating and subsequently analyzing a behavior pattern for each computer program introduced to the computer system. Generation of the behavior pattern is accomplished by a virtual machine invoked within the computer system. An initial analysis may be performed on the behaviour pattern to identify infected programs on initial presentation of the program to the computer system. The analysis system also stores behavior patterns and sequences with their corresponding analysis results in a database. Newly infected programs can be detected by analyzing a newly generated behaviour pattern for the program within reference to a stored behavior pattern to identify presence of an infection or payload pattern.

WO 02/06928 A2

IPR2024-00027
CrowdStrike Exhibit 1002 Page 91

Computer Immune System and Method for Detecting Unwanted Code in a Computer System

PRIORITY APPLICATION NOTICE

This application claims priority from United States provisional patent application Serial No. 60/218,489, filed July 17, 2000, which application is hereby incorporated by reference in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the field of computer security and specifically to the detection of computer programs that exhibit malicious or self-propagating behavior including, for example, computer viruses and trojans.

2. Discussion of the Related Art

Detection of viruses has been a concern throughout the era of the personal computer. With the growth of communication networks such as the Internet and increasing interchange of data, including the rapid growth in the use of e-mail for communications, the infection of computers through communications or file exchange is an increasingly significant consideration. Infections take various forms, but are typically related to computer viruses, trojan programs, or other forms of malicious code. Recent incidents of e-mail mediated virus attacks have been dramatic both for the speed of propagation and for the extent of damage, with Internet service providers (ISPs) and companies suffering service problems and a loss of e-mail capability. In many instances, attempts to adequately prevent file exchange or e-mail mediated

IPR2024-00027
CrowdStrike Exhibit 1002 Page 92

infections significantly inconvenience computer users. Improved strategies for detecting and dealing with virus attacks are desired.

One conventional technique for detecting viruses is signature scanning. Signature scanning systems use sample code patterns extracted from known malicious code and scan for the occurrence of these patterns in other program code. In some cases program code that is scanned is first decrypted through emulation, and the resulting code is scanned for signatures or function signatures. A primary limitation of this signature scanning method is that only known malicious code is detected, that is, only code that matches the stored sample signatures of known malicious code is identified as being infected. All viruses or malicious code not previously identified and all viruses or malicious code created after the last update to the signature database will not be detected. Thus, newly created viruses are not detected by this method; neither are viruses with code in which the signature, previously extracted and contained in the signature database, has been overwritten.

In addition, the signature analysis technique fails to identify the presence of a virus if the signature is not aligned in the code in the expected fashion. Alternately, the authors of a virus may obscure the identity of the virus by opcode substitution or by inserting dummy or random code into virus functions. Nonsense code can be inserted that alters the signature of the virus to a sufficient extent as to be undetectable by a signature scanning program, without diminishing the ability of the virus to propagate and deliver its payload.

Another virus detection strategy is integrity checking. Integrity checking systems extract a code sample from known, benign application program code. The code sample is stored, together with information from the program file such as the executable program header and the file length, as well as the date and time of the sample. The program file is checked at regular intervals against this database to ensure that the program file has not been modified. Integrity checking programs generate long lists of modified files when a user upgrades the operating system of the computer or installs or

IPR2024-00027
CrowdStrike Exhibit 1002 Page 93

upgrades application software. A main disadvantage of an integrity check based virus detection system is that a great many warnings of virus activity issue when any modification of an application program is performed. It is difficult for a user to determine when a warning represents a legitimate attack on the computer system.

Checksum monitoring systems detect viruses by generating a cyclic redundancy check (CRC) value for each program file. Modification of the program file is detected by a variation in the CRC value. Checksum monitors improve on integrity check systems in that it is more difficult for malicious code to defeat the monitoring. On the other hand, checksum monitors exhibit the same limitations as integrity checking systems in that many false warnings issue and it is difficult to identify which warnings represent actual viruses or infection.

Behavior interception systems detect virus activity by interacting with the operating system of the target computer and monitoring for potentially malicious behavior. When such malicious behavior is detected, the action is blocked and the user is informed that a potentially dangerous action is about to take place. The potentially malicious code can be allowed to perform this action by the user. This makes the behavior interception system somewhat unreliable, because the effectiveness of the system depends on user input. In addition, resident behavior interception systems are sometimes detected and disabled by malicious code.

Another conventional strategy for detecting infections is the use of bait files. This strategy is typically used in combination with other virus detection strategies to detect an existing and active infection. This means that the malicious code is presently running on the target computer and is modifying files. The virus is detected when the bait file is modified. Many viruses are aware of bait files and do not modify files that are either too small, obviously a bait file because of their structure or have a predetermined content in the file name.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 94

It is apparent that improved techniques for detecting viruses and other malicious types of code are desirable.

SUMMARY OF THE PREFERRED EMBODIMENTS

One aspect of the present invention provides a method for identifying presence of malicious code in program code within a computer system, including initializing a virtual machine within the computer system. The initialized virtual machine comprises software simulating functionality of a central processing unit and memory. The virtual machine virtually executes a target program so that the target program interacts with the computer system only through the virtual machine. The method includes analyzing behavior of the target program following virtual execution to identify occurrence of malicious code behavior and indicating in a behavior pattern the occurrence of malicious code behavior. The virtual machine is terminated at the end of the analysis process, thereby removing from the computer system a copy of the target program that was contained within the virtual machine.

Another aspect of the present invention provides a method for identifying the presence of malicious code in program code within a computer system. The method includes initializing a virtual machine within the computer system, the virtual machine comprising software simulating functionality of a central processing unit, memory and an operating system including interrupt calls to the virtual operating system. A target program is virtually executed within the virtual machine so that the target program interacts with the virtual operating system and the virtual central processing unit through the virtual machine. Behavior of the target program is monitored during virtual execution to identify presence of malicious code and the occurrence of malicious code behavior is indicated in a behavior pattern. The virtual machine is terminated, leaving behind a record of the behavior pattern characteristic of the analyzed target program.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 95

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a behavior pattern generated according to the analytical behavior method, showing the behavior pattern for code that is not infected and is infected with a computer virus. Each bit may be a flag indicating an action. The total stream of bits is a value indicative of the behavior of the program.

FIG. 2 shows a block diagram of components used in a preferred implementation of the analytical detection method.

FIG. 3 schematically illustrates the COM file format, used as an example of the function of the program structure extractor and program loader.

FIG. 4 illustrates an interface of the virtual PC to various program file formats. Before virtualization can take place, the program loader preferably extracts the correct entry point, code and initialized data from the program file. The file offset to the entry point code is given in the program header and varies depending on the type of file that contains the program.

FIG. 5 schematically illustrates the virtual PC memory map after loading a binary image (.COM) program and after loading a MZ-executable program. To virtualize the code in the desired manner, the structure of the virtual PC and its memory map contains the same information as it would if the code was executed on the physical PC which runs the virtual machine containing the Virtual PC.

FIG. 6 provides a detailed diagram showing components of a preferred implementation of the Virtual PC. The virtual PC contains the same components that are used in a physical computer, except that all Virtual PC components are simulated in software running as a virtual machine on a physical computer.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A particularly preferred embodiment of the present invention provides an automated analysis system that detects viruses and other types of

IPR2024-00027
CrowdStrike Exhibit 1002 Page 96

malicious code within a computer system by generating and subsequently analyzing a behavior pattern for each computer program introduced to the computer system. New or modified computer programs are analyzed before being executed by the computer system. Most preferably the computer system initiates a virtual machine representing a simulation of the computer system and the virtual machine executes the new or modified computer program to generate a behavior pattern prior to the new computer program being executed by the physical computer system. An initial analysis is performed on the behavior pattern to identify infected programs upon initial presentation of the program to the computer system. The analysis system also stores behavior patterns and corresponding analysis results in a database. Newly infected programs can be detected by subtracting the stored behavior pattern for that program from a newly generated behavior pattern, and analyzing the resulting pattern to identify an infection or payload pattern associated with malicious code.

A variety of different terms are used in programming to describe different functional programming subunits. At different times and for different programming languages subunits of various sorts have been called functions, routines, subprograms, subroutines and other names. Such designations and the context or differences they represent are not significant to the present discussion and so this discussion is made simply in terms of programs, intending the term program to refer to functional programming units of any size that are sufficient to perform a defined task within a computer system or computing environment. Such specialized functions as those performed by macros within certain word processing programs, including for example, in Visual Basic macros for Microsoft Word documents, are included within this general discussion. In this sense, individual documents may be considered to be programs within the context of this discussion.

For convenience and brevity, this discussion references viruses in the known sense of that term as being a self-propagating program generally

IPR2024-00027
CrowdStrike Exhibit 1002 Page 97

undesired in the infected computer system. As used here, the term Windows is intended to reference any of the personal desktop operating systems sold by the Microsoft Corporation under the Windows brand name. The term PC or personal computer is used, unless specifically modified to indicate otherwise, to indicate a computer system based on the well-known x86 architecture, including those machines that presently are based on the microprocessor sold by Intel Corporation under its Pentium brand name and successors to that microprocessor and architecture. This discussion is provided to illustrate implementation of aspects of the invention. Aspects of the present invention find application in a range of different computer systems in addition to the illustrated personal computer systems.

The present inventor has analyzed the behavior of a variety of different viruses and other malignant source code. Certain general characteristics of viruses have been identified. A virus needs to infect other programs and eventually other computers to propagate. Viruses consequently include infection loops that copy the virus into another executable program or sometimes into documents, in the exemplary case of Visual Basic macro viruses. Viruses and trojans generally contain payloads. The payload allows the virus to affect the infected system or communicate its presence. A payload might be, for example, a message that pops up to announce the virus or a malicious function that damages the infected computer, for example by corrupting or erasing the data on the hard disk or by altering or disabling the BIOS within the BIOS flash or EEPROM.

Another common characteristic of viruses is that the virus becomes resident in the memory. DOS viruses need to copy themselves into memory and stay resident. Most viruses do not use the obvious terminate and stay resident (TSR) call but instead use a procedure that copies the virus into high memory. The virus then can directly modify the data in the high memory blocks. In an additional aspect of this infection scheme, the interrupt vector is modified to point at memory blocks that have been modified by the memory resident virus or other malignant procedure. These modified memory blocks

IPR2024-00027
CrowdStrike Exhibit 1002 Page 98

store the infection procedure. Windows specific viruses bump themselves into ring0, for example using a callgate or DPMI call, and go resident in a system utility such as the system tray.

These behaviors are characteristic of a virus and are not, in the aggregate, characteristic of other, non-malignant programs. Consequently, a program can be identified as a virus or infected with a virus if it possesses certain ones of these behaviors, certain collections of these behaviors or all of these behaviors. In preferred embodiments of the present invention, the occurrence of these behaviors or combinations of the behaviors is indicated by collections of bits in a behavior pattern data set representing behavior characteristic of the infected program. An example of behavior patterns for a normal and an infected file are illustrated in FIG. 1.

In preferred embodiments of the present invention, the behavior of a newly loaded or called program is analyzed in a virtual machine that simulates a complete PC, or a sufficiently complete PC, in software and it is that virtual PC that generates the behavior pattern. The virtual PC simulates execution of the new or modified program, simulating a range of system functions, and the virtual PC monitors the behavior of the suspect program and makes a record of this behavior that can be analyzed to determine that the target program exhibits virus or malignant behaviors. The result of the virtual execution by the virtual machine is a behavior pattern representative of the new program. As discussed in greater detail below, the behavior pattern generated by the virtual PC identifies that a program is infected with a virus or is itself a virus. An advantage for the use of virtual execution and analysis of new programs for viruses is that the virtual machine is virtual and so, if the virtualized new program contains a virus, only the virtual machine is infected. The infected instance of the virtual machine is deleted after the simulation, so the infection is incomplete and the virus does not propagate. The behavior pattern survives the deletion of the virtual machine, allowing an analysis program to identify the existence of the virus and of the infection within the new program.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 99

Most preferably, each time a new program is analyzed a new instance of the virtual machine is generated, free of modification by any previously virtualized programs including any earlier analyzed viruses. The new program then is run on the new instance of the virtual machine preferably followed by initiation of a modified interrupt caller procedure, described in greater detail below. While the virtual machine is executing the new program in cooperation with the modified interrupt caller procedure, the virtual machine monitors all system calls, DPMI/DOS interrupts and I/O port read/write (r/w) operations, setting bits in the behavior pattern register according to the observed behaviors. It is these bits in the behavior pattern that are retained after the simulation is complete and the virtual PC has been terminated. The bits stored in the behavior pattern register are the behavior pattern and indicate whether the virtually-executed program includes behaviors indicative of the presence of a virus or other malignant code.

The modified interrupt caller procedure calls the interrupts that the program being analyzed has modified within the virtual PC and generates a behavior pattern for each of those interrupt service routines as well. This allows particularly preferred embodiments of the present invention to identify certain types of viruses that initially modify only the interrupt service routines and do not begin propagating until the modified interrupt or interrupts are called by another program. By allowing the various interrupt service routines in the virtual machine to be modified and then analyzing the modified interrupts, these embodiments of the invention can detect this delayed propagation mechanism.

In some presently preferred embodiments, only the static, final version of the behavior pattern is analyzed. It is possible, and in some circumstances desirable, to monitor the sequence in which the bits in the behavior pattern register are set. The order in which the behavior pattern bits are set provides additional information allowing identification of additional virus behaviors. Tracking of the order in which the behavior pattern bits are set is accomplished within the virtual machine.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 100

Preferred implementations of the analytical behavior method (ABM) proceed by extracting a behavior pattern and sequence from a modified, new, unknown or suspect program. The behavior pattern is preferably used to analyze the behavior of the unknown program to determine if the behavior of the unknown program is malicious. Identification of malicious behavior in this manner allows identification of virus carrying files prior to infection of the host computer system. The behavior pattern can also be stored in a database and the virtual machine can subsequently analyze the behavior of the program following modification to determine if its functionality has been modified in a suspect (malicious) manner. This provides post-infection analysis.

The described analytical behavior method differs from conventional virus detection methods in that it does not match program code to a set of stored patterns as do signature scanners and integrity checkers. Rather, a virtual machine is used to generate a behavior pattern and a sequence. The generated behavior pattern does not change significantly between version updates, but does change dramatically when a virus infects a program. For example, a word processor will still behave like a word processor when the program is replaced or updated with a new version of the program but the word processor changes significantly when the word processor is infected with a virus. The differences reflected in the behavior patterns are illustrated in FIG. 1. When a word processor is infected with a file infector computer virus, the word processor now opens executable files and inserts the viral code into them, thereby infecting additional files. This is clearly reflected in the illustrated behavior patterns.

In particularly preferred embodiments of the invention, the analysis procedure specifically targets infection methods such as, but not limited to, the insertion of code to other executables or documents, submitting code to other applications to be transmitted or stored, insertion of code into high memory blocks and the modification of memory control blocks. Preferred implementations of the analysis method further look for destructive content,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 101

such as, but not limited to, functions that overwrite disk areas or the BIOS ROM, or delete files or directories. Most preferably, the analysis makes an exception and does not identify as infected a program whose other behavior characteristics indicate that the program is a development tool or software debugging tool and where the modifying behavior is an integral part of the tool's normal function. A viral infection of a development tool can be detected where an expressed function is not part of the tool's normal function, that is, within the development process. Both active (1) and inactive (0) flags present in the behavior pattern are significant in this analysis, as well as the sequence in which actions take place.

In accordance with preferred embodiments of the present invention, the virtual machine or virtual PC represents a simulation of a complete computer system. A complete computer system preferably includes an emulated central processing unit (CPU), emulated memory, input/output (I/O) ports, BIOS firmware, operating system and the operating system data areas. This stands in contrast to simple emulation of a processor, in which only the processor processes are emulated. In emulation, program instructions are converted from their native form to a stream of instructions that perform the same function on a different hardware platform. Some signature scanning software employs emulation to decrypt the body of a suspect program before the suspect program is scanned for signatures. In virtualization, the entire computer is simulated including operating system calls, which are not actually executed but seem to the calling program to perform the desired functions and return the correct values as if they were executed.

As discussed above, the virtual PC includes a CPU, memory, I/O ports, a program loader, and the operating system application program interface (API's) entry points and interface. Using such a complete virtual PC is particularly preferred because it gives the analytical behavior method a high level of control over the virtualized program, including over the sophisticated direct calls to the operating system API. The virtualized program is not given access to any of the facilities of the physical machine, thereby avoiding the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 102

risk that the potential virus or other malicious code escapes from the controlled environment to infect the host computer system.

FIG. 2 provides an overview of the preferred analytic behavior method architecture including the relationship between the virtual machine and components of the host computer system. Program code is delivered to the ABM engine and analysis system by direct access to the hard disk through I/O port bit manipulation, hooking into the operating system file system or by sequentially scanning the hard disk. The program code is checked against the database for 'known' files. If the file is new or modified, it is processed. The resulting behavior signature is analyzed or compared and stored. A virus warning is returned when analysis shows that the file contains malicious code. The analytical behavior method preferably includes: (1) file structure extraction; (2) change detection; (3) virtualization; (4) analysis; and (5) decision.

Before the program can be virtualized, the file format containing the target program has to be evaluated. The entry point code is extracted and loaded into the virtual computer's memory at the correct simulated offset. In a physical computer this function would be performed by the program loader function, which is part of the operating system. The operating system can execute programs that are held in a collection of different file formats, such as:

DOS 1.0 and/or CP/M COM	Binary image file, loaded at 100h in memory, maximum size: 64K.
DOS 2.0 – DOS 7.1 EXE	MZ-type executable, header determines CS:IP of load address.
Windows 3.0 executables	NE-type executable which contains both the DOS MZ-header pointing at a DOS code area and a New Executable (NE) header containing the entry point of the Windows (protected mode) code. NE files are segmented.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 103

OS/2 executables	LE /LX type executable which contains both the DOS MZ-header and DOS code area and a protected mode section which is determined by the LE-header following the DOS code segment. Linear Executable (LE) files are used in Windows 3 for system utilities and device drivers. LE files are segmented. LX files incorporate some differences in the way the page table is stored and are intended for the OS/2 operating system. LE files are segmented and the segments are paged.
32-bit executables	PE-type executable which contains both the DOS MZ-header and DOS code area and the Portable Executable header containing the entry point and file offset of the protected mode code. PE files are segmented.
OLE Compound Files	OLE compound files (COM) are document files that can contain executable format streams, usually referred to as Macros. All office components incorporate Visual Basic for Applications, as does Internet Explorer versions 4 and 5. Windows98 systems can execute Visual Basic code directly from a script file. The Visual Basic code is compiled and stored in a stream, which is paged according to its file offset references stored in a linked list in the file header.
Binary Image	A binary image is used for the boot sector and Master Boot and Partition table. Both the boot-sector and the MBR contain executable

IPR2024-00027
CrowdStrike Exhibit 1002 Page 104

	code which is loaded into memory at 0:7C00 during the start-up process.
Driver files	System Drivers are stored as a binary image with a header. The header contains information about the drivers stored within the file. Multiple drivers can be stored within the same file.

The virtual computer loader function is capable of dealing with the file formats and binary image files shown above. The loader function is performed by virtualizing the operating system program loader and so varies depending on the operating system used in the host computer. The file structure analysis procedure looks in the file header and file structure to determine the file format, rather than using the file extension because file extensions are unreliable in general use. The .EXE formats described above therefore include DLL, AX, OCX and other executable file format extensions.

Compound document files can contain executable streams such as Visual Basic code or macros. The structure of a compound document file is illustrated in the diagram shown in FIG. 3. The header of a compound document file contains a linked list (or File Allocation Table) which is referenced in a directory structure that points to the entry point of the linked list. Each entry in the linked list refers to the next entry and a file offset. A value of –1 in the linked list indicates the end of a chain. Streams exist out of blocks, which may be scattered anywhere in the file in any order. In particularly preferred embodiments of the invention, code extracted from a compound document file is passed through a Visual Basic decompiler before it is presented to a Visual Basic emulator. Not all compound document files contain compiled Visual Basic code. Hypertext markup language (HTML) and Visual Basic Script (VBS) files can contain Visual Basic Script code as text. This code is preferably extracted and treated as a Visual Basic stream within the virtual machine.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 105

The NE /PE /LE executable file formats are similar in complexity, except that no linked list is used; rather these file formats use a segment or page table. The PE file format is based on the COFF file specification. FIG. 4 illustrates how these file formats interface with the preferred virtual PC in accordance with certain embodiments of the present invention. In evaluating how aspects of the preferred virtual PC interfaces to a particular file, the file loader preferably decides if the file presented is a document file or a binary file.

After the file format has been evaluated and the entry point-file offset has been calculated, the file is opened and the virtual machine reads the relevant code into memory as a data stream. The length of the code is calculated from fields in the header of the file. This information is passed to the virtual program loader. The virtual program loader uses information in the file header to load the extracted code at the correct simulated offset in a virtual memory array.

A memory mapping utility maps the virtual memory map to the offset for the file type that is virtualized:

DOS (CP/m) binary image files (.COM)	offset CS:100h
DOS (2.0 up) Executable format files (MZ-EXE)	offset CS:IP from header
Windows NE, PE, LE	offset C0000000+CS:IP from header
Binary Image MBR, Boot sector code	offset 0:7C00h
Document COM files, HTML and VBS files	no specific offset, VBA code

The Loader utility dynamically assigns physical memory to the virtual computer memory array each time a program is virtualized, and proceeds to build a new virtual machine. Each virtual machine contains a BIOS data area, a filled environment string area, DOS data area, memory control blocks, program segment prefix area, the interrupt vector table and descriptor tables.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 106

The final structure of the virtual machine depends on the type of program that is virtualized. Each virtualized program therefore runs in a fresh memory area, created when that program is loaded into the virtual PC. Previous instances, where infected programs may have been virtualized, therefore cannot affect the performance of subsequent programs. The virtual machine is shut down and its memory resources are released when the virtualized program terminates and the virtual machine completes assembly of the behavior pattern for the target, virtualized.

FIG. 5 illustrates how the virtual memory is configured for (COM) binary image files and DOS program (MZ-EXE) files. The memory map and mapper utility are adjusted depending on the file type.

The program loader simulates the loader functions of the operating system and creates system areas that represent similar system areas in the physical computer. This is particularly advantageous functionality because the code under evaluation most preferably runs in the same manner as if executed on a physical computer system. The virtualized program is executed by fetching instructions from the virtual memory array into a pre-fetch instruction queue. The instructions in the queue are decoded and their length is determined by their operational parameters.

The instruction pointer is incremented accordingly so that the instruction loader is ready to fetch the next instruction. The virtual machine determines from the r/m field of the instruction parameters where data on which the instruction operates is to be fetched. The data fetch mechanism fetches this data and presents the data to the logic unit, which then performs the operation indicated by the code. The destination of the processed data is determined from the parameters of the instruction code. The data write mechanism is used to write the processed data to emulated memory or the emulated processor register set. This process accurately reflects what takes place in a physical CPU (central processing unit).

All areas of this process are simulated, as generally illustrated in FIG. 6. The memory exists as an array of 400 Kbyte elements into which all

IPR2024-00027
CrowdStrike Exhibit 1002 Page 107

memory accesses are mapped by a memory mapping mechanism. The size of the memory array may be adjusted in future implementations to accommodate larger programs. The video display is simulated from a system viewpoint as 128 Kbyte of memory mapped between A000:0 and BFFF:F (inclusive) in the virtual computer's memory map. The standard IBM PC input/output area is simulated as an array of 1024 bytes representing I/O ports 0-3FFh. The CPU is simulated by performing the same low-level functions as the physical CPU, but in high-level software.

The operating system is implemented as an area in the memory array of 700h bytes containing the BIOS data fields, the DOS data area, Memory Control Blocks and DOS devices. The interrupt vector table takes up the first 1024 (400h) positions in the memory array as it would in a physical PC. The DOS interrupt structure is implemented as simulated functions that return the correct values and by filling the memory array with the correct values expected by simulating DOS functions.

The operating system is implemented as a virtual API (VAPI) that simulates the results returned by all operating system API's.

During the virtualization process, flags are set in the behavior pattern (Tstruct) field as the functions represented by those fields are virtualized. The sequence in which these functions are called is recorded in the sequencer. The behavior pattern therefore matches closely the behavior of the program under evaluation to the behavior of that program in a physical PC environment. Simulated interrupt vectors modified during the process of executing the virtualized program are called after program virtualization terminates, thus acting as applications that would call such interrupt vectors in a physical computer following modification of these vectors.

To illustrate this functionality, consider the following set of operations might be performed in operation of the analytical behavior method:

Search for the first EXE file in this directory	;set FindFirst Flag (Tstruct Structure)

IPR2024-00027
CrowdStrike Exhibit 1002 Page 108

```
Is this a PE executable (examine header)?        ;set EXEcheck Flag
If not, jump far
Else: Open the executable file                   ;set EXEaccess Flag
      Write to the section table                 ;set EXEwrite Flag
      Search for the end-of-file                 ;set EXEeof Flag
      Write to file                              ;set EXEwrite Flag
      Close file
Search next EXE file                             ;set EXEFindNext Flag
```

```
Bit+1  64--- ---------- ---------- ---------- ---------- ---------- -------- --------1
Returned: 0010 0100 1010 1010 1001 0101 0010 1111 0010 1010 0010 0100 0100 1001 0000
0101
Value:    2    4    A    A    9    5    2    F    2    A    2    4    4    9    0    5
Sequencer: 21,22, 23,24,26,29,3E,1,36,38,3B,3, 9,C,F,13,16,1A,1C,1E, 2B,2D,30,32,34,
```

The resulting behavior pattern is: 24AA952F2A244905

The behavior pattern contains flags that indicate that the user has not had the opportunity to interact with this process through user input (the userInput flag is not set). The sequencer contains the order in which the bits were set, identifying the infection sequence shown above. Therefore this observed behavior is most likely viral.

Many viruses are encrypted, polymorphic or use 'tricks' to avoid detection by signature scanners. Wherever such 'tricks' are used, the behavior pattern points more obviously towards a virus since such tricks are not normally used in normal applications. In any case, preferred implementations of the present invention require that an infection procedure be present to trigger a virus warning to avoid false positive warnings. Encrypted viruses are no problem, because the execution of the code within the virtual machine, which generates the behavior pattern, effectively decrypts any encrypted or polymorphic virus, as it would in a physical PC environment. Because all parts of the virtual computer are virtualized in preferred embodiments, and at

IPR2024-00027
CrowdStrike Exhibit 1002 Page 109

no time is the virtualized program allowed to interact with the physical computer, there is no chance that viral code could escape from the virtual machine and infect the physical computer.

The change detection module compares existing files at 6 levels to determine if the file was analyzed previously:

- The file is the same (entry point code, sample, file-name and file-size are the same).
- The file is not in the database (new file).
- The behavior pattern matches a stored pattern.
- The file's entry code is modified. The behavior pattern is binary subtracted from the previous stored pattern. The resulting bit pattern is analyzed.
- The file's entry code, CRC and header fields are the same, but the file is renamed. No other fields are modified.
- The file's behavior pattern is found in the database and matches a known viral behavior pattern.
- The file's behavior pattern is found in the database and matches a known benign behavior pattern.

The program is virtualized if the executable part of the file is modified. A file that does not contain modified executable code cannot contain a virus, unless the original file was infected. If this is the case, a previous analysis would have detected the virus. When an existing program is updated, its function remains the same, and therefore its behavior pattern closely matches its stored behavior pattern. If the altered bits indicate that an infection procedure has been added then the file is considered as infected.

Two detection mechanisms operate side-by-side, both using the behavior pattern:

Pre-infection detection

This is the most desirable case. In pre-infection detection, the behavior pattern is analyzed and is found to represent viral behavior for those new or modified programs introduced to the system. The program file under

IPR2024-00027
CrowdStrike Exhibit 1002 Page 110

evaluation can be repaired by removing the virus or erased if the virus infection proves too difficult to remove or if parts of the original code were overwritten. The infected program has not yet been executed on the physical PC at this time and so nothing need be done to repair the physical PC after discovery of the virus.

Post-infection detection

Post-infection detection takes place in cases when initial infection is missed by pre-infection detection. A virus could be missed by pre-infection detection when it does not perform any viral function on first execution and does not modify interrupt vectors that point to an infection routine. This is the case with so-called slow infectors and similarly behaving malignant code. In post-infection detection the virus is caught the moment it attempts to infect the first executable on the PC. The file hook mechanism detects this attempted change to an executable (including documents). The ABM engine then analyzes the first executable program and finds that its behavior pattern is altered in a manner indicating that a virus is active.

Database Structure:

File ID area:	Behavior pattern, program name, file size and path.
Repair Structures	Header fields, section table and relocation tables.
Segment tables	Size and Offset of each section in the section table (Windows programs only).

Macro viruses in documents are treated as if they were executables. The original Visual Basic code is recovered by decryption (where applicable) and reverse compiling the Visual Basic document (COM) stream. The resulting source code is neither saved nor shown to protect the rights of the original publishers of legitimate Visual Basic software. After virtualization the source code is discarded.

One drawback to the described virus detection system is that the initial analysis is slower than pattern scanning. This drawback is more than offset

IPR2024-00027
CrowdStrike Exhibit 1002 Page 111

by the advantages of the system. Using file system hooking means all new files are reported and analyzed 'on the fly' in background. This means that once a computer is virus-free, a complete scan is typically not required again, unless the protection system has been deactivated during a period in which new programs have been installed. In signature scanning based protection systems, the computer needs to be completely rescanned every time the virus signature database is updated. Unaltered files are not again virtualized when the user initiates subsequent disk scans, so that the process is at least as fast as pattern scanning, but with a higher degree of security. The stored information also helps to repair viral damage to files or system areas, securing complete or effectively complete recovery in most cases.

In tests of a prototype implementation ABM system, the combination of pre-infection (96%) and post-infection detection (4%) resulted in 100% detection of all known viral techniques, using a combination of new, modified and well-known viruses. Other methods detected only 100% of known viruses and scored as low as 0% for the detection of new, modified and unknown viruses. No exact figure can be quoted for tests involving signature scanner based products. The results for such products are a direct representation of the mix of known, modified and new, unknown viruses; e.g. if 30% of the virus test set is new, modified or unknown then the final score reflected close to 30% missed viruses. No such relationship exists for the implementations of preferred aspects of the present system, where the detection efficiency does not appreciably vary for alterations of the presented virus mix.

The present invention has been set forth with reference to certain particularly preferred embodiments thereof. Those of ordinary skill in the art will appreciate that the present invention need not be limited to these presently preferred embodiments and will understand that various modifications and extensions of these embodiments might be made within the general teachings of the present invention. Consequently, the present invention is not to be limited to any of the described embodiments but is instead to be defined by the claims, which follow.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 112

I claim:

1. A method for identifying presence of malicious code in program code within a computer system, the method comprising:

initializing a virtual machine within the computer system, the virtual machine comprising software simulating functionality of a central processing unit and memory;

virtually executing a target program within the virtual machine so that the target program interacts with the computer system only through the virtual machine;

analyzing behavior of the target program following virtual execution to identify occurrence of malicious code behavior and indicating in a behavior pattern the occurrence of malicious code behavior; and

terminating the virtual machine after the analyzing process, thereby removing from the computer system a copy of the target program that was contained within the virtual machine.

2. The method of claim 1, wherein the virtual machine simulates functionality of input/output ports, operating system data areas, and an operating system application program interface.

3. The method of claim 2, wherein the virtual machine further includes a virtual Visual Basic engine.

4. The method of claim 2, wherein virtual execution of the target program causes the target program to interact with the simulated operating system application program interface.

5. The method of claim 1, wherein the target program is newly introduced to the computer system and not executed prior to virtually executing the target program.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 113

6. The method of claim 1, wherein after a first instance of a first program is analyzed by the virtual machine and a first behavior pattern is generated and stored in a database within the computer system, the method further comprising:

determining that the first program is modified;

analyzing the modified first program by executing the modified first program in the virtual machine to provide a second behavior pattern; and

comparing the first behavior pattern to the second behavior pattern.

7. The method of claim 6, wherein a new behavior pattern is generated each time the first program is modified.

8. The method of claim 6, wherein introduction of malignant code during modification of the first program is detected by comparing the first behavior pattern to the second behavior pattern.

9. The method of claim 6, wherein the first behavior pattern is substantially similar to the second behavior pattern when the modified first program is a new version of the first program.

10. The method of claim 1, wherein the behavior pattern identifies functions executed in the virtual execution of the target program, the method further comprising tracking an order in which the functions are virtually executed by the target program within the virtual machine.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 114

11. A method for identifying presence of malicious code in program code within a computer system, the method comprising:

initializing a virtual machine within the computer system, the virtual machine comprising software simulating functionality of a central processing unit, memory and an operating system including interrupt calls to the virtual operating system;

virtually executing a target program within the virtual machine so that the target program interacts with the virtual operating system and the virtual central processing unit through the virtual machine;

monitoring behavior of the target program during virtual execution to identify presence of malicious code and indicating in a behavior pattern the occurrence of malicious code behavior; and

terminating the virtual machine, leaving behind a record of the behavior pattern characteristic of the analyzed target program.

12. The method of claim 11, wherein the record is in a behavior register in the computer system.

13. The method of claim 11, wherein after a first instance of a first program is analyzed by the virtual machine and a first behavior pattern is generated and stored in a database within the computer system, the method further comprising:

determining that the first program is modified;

analyzing the modified first program by executing the modified first program in the virtual machine to provide a second behavior pattern; and

comparing the first behavior pattern to the second behavior pattern.

14. The method of claim 13, wherein a new behavior pattern is generated each time the first program is modified.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 115

15. The method of claim 13, wherein introduction of malignant code during modification of the first program is detected by comparing the first behavior pattern to the second behavior pattern.

16. The method of claim 13, wherein the first behavior pattern is substantially similar to the second behavior pattern when the modified first program is a new version of the first program.

17. The method of claim 13, wherein the behavior pattern identifies functions executed in the virtual execution of the target program, the method further comprising tracking an order in which the functions are virtually executed by the target program within the virtual machine.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 116

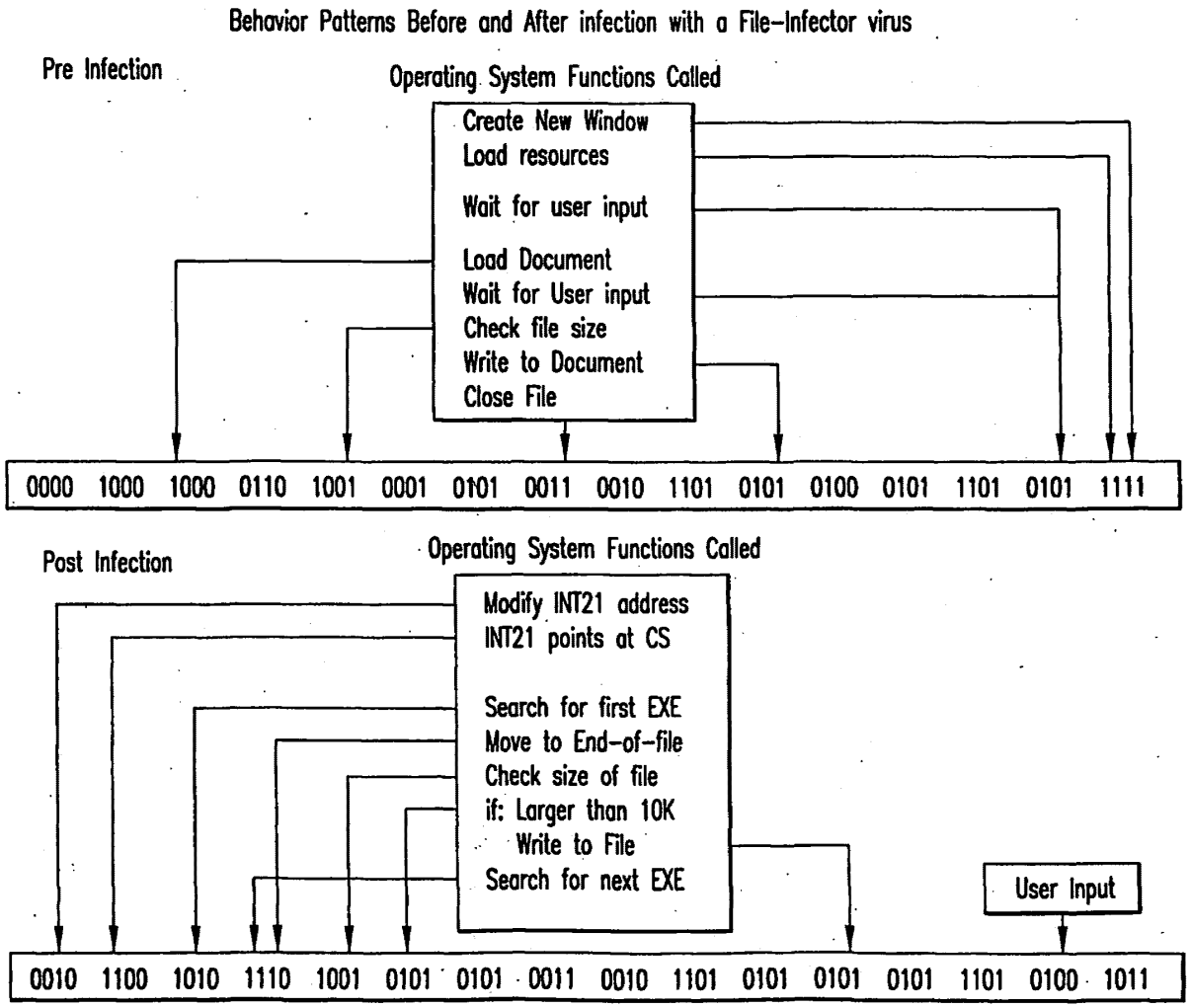


FIG.1

IPR2024-00027
CrowdStrike Exhibit 1002 Page 117

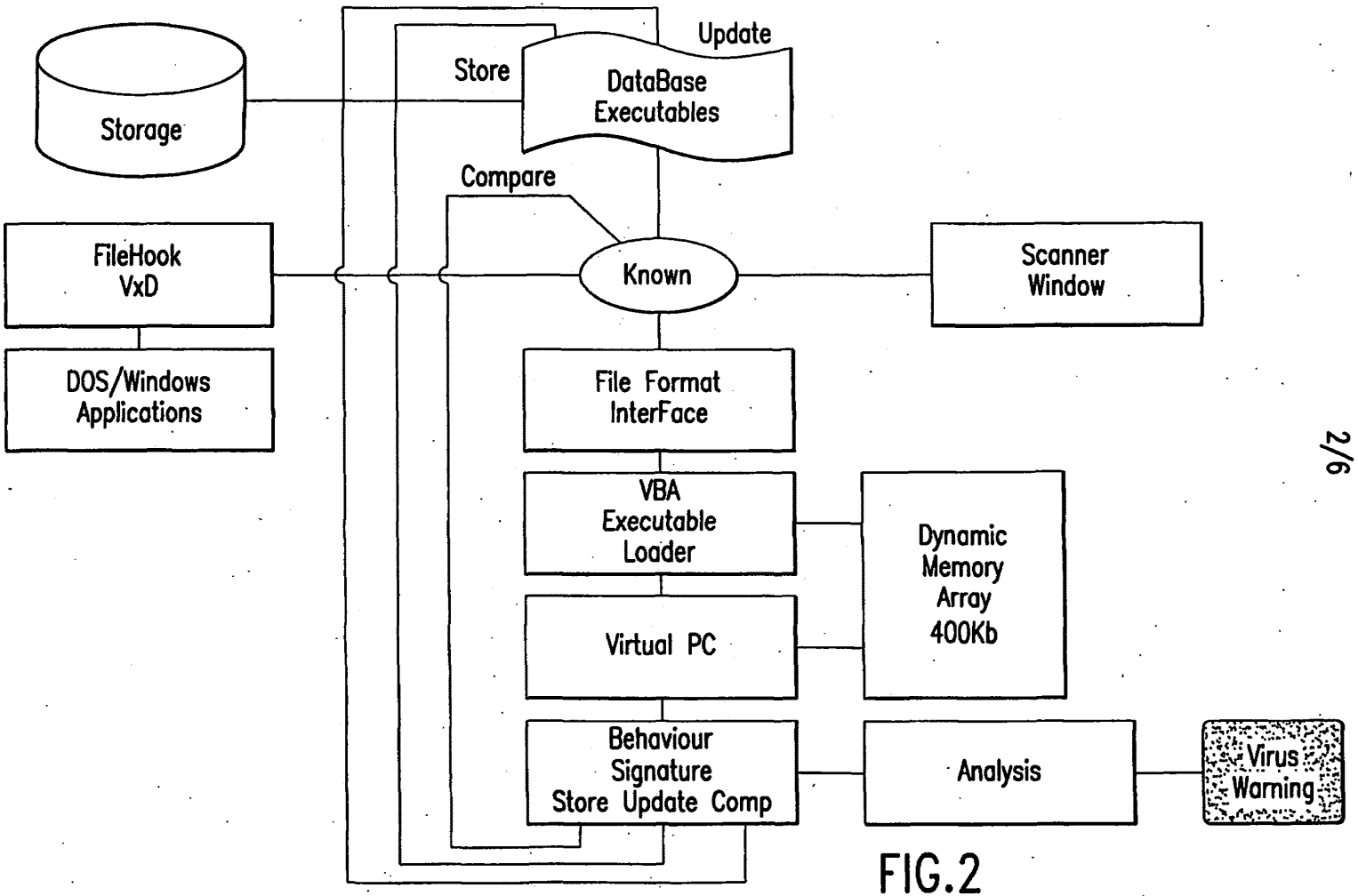


FIG.2

IPR2024-00027
CrowdStrike Exhibit 1002 Page 118

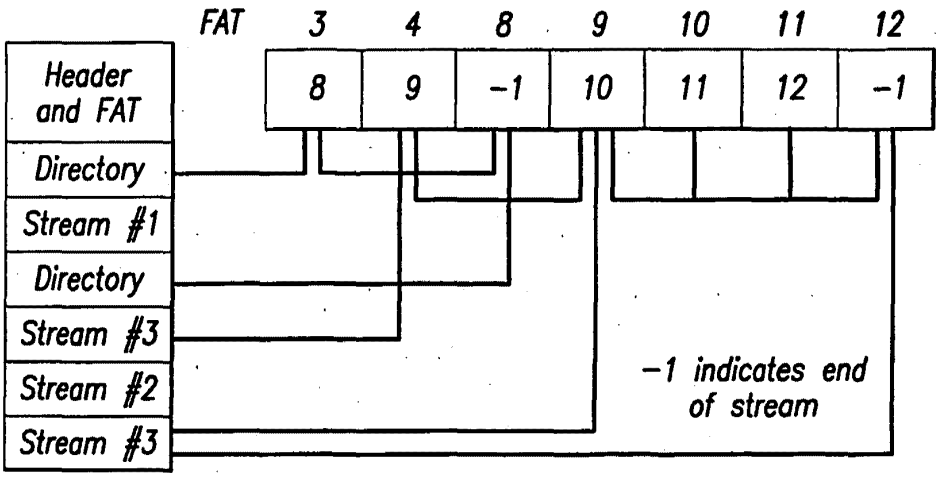

FAT
3
4
8
9
10
11
12
8
9
-1
10
11
12
-1
Header and FAT
Directory
Stream #1
Directory
Stream #3
Stream #2
Stream #3
-1 indicates end of stream

FIG.3

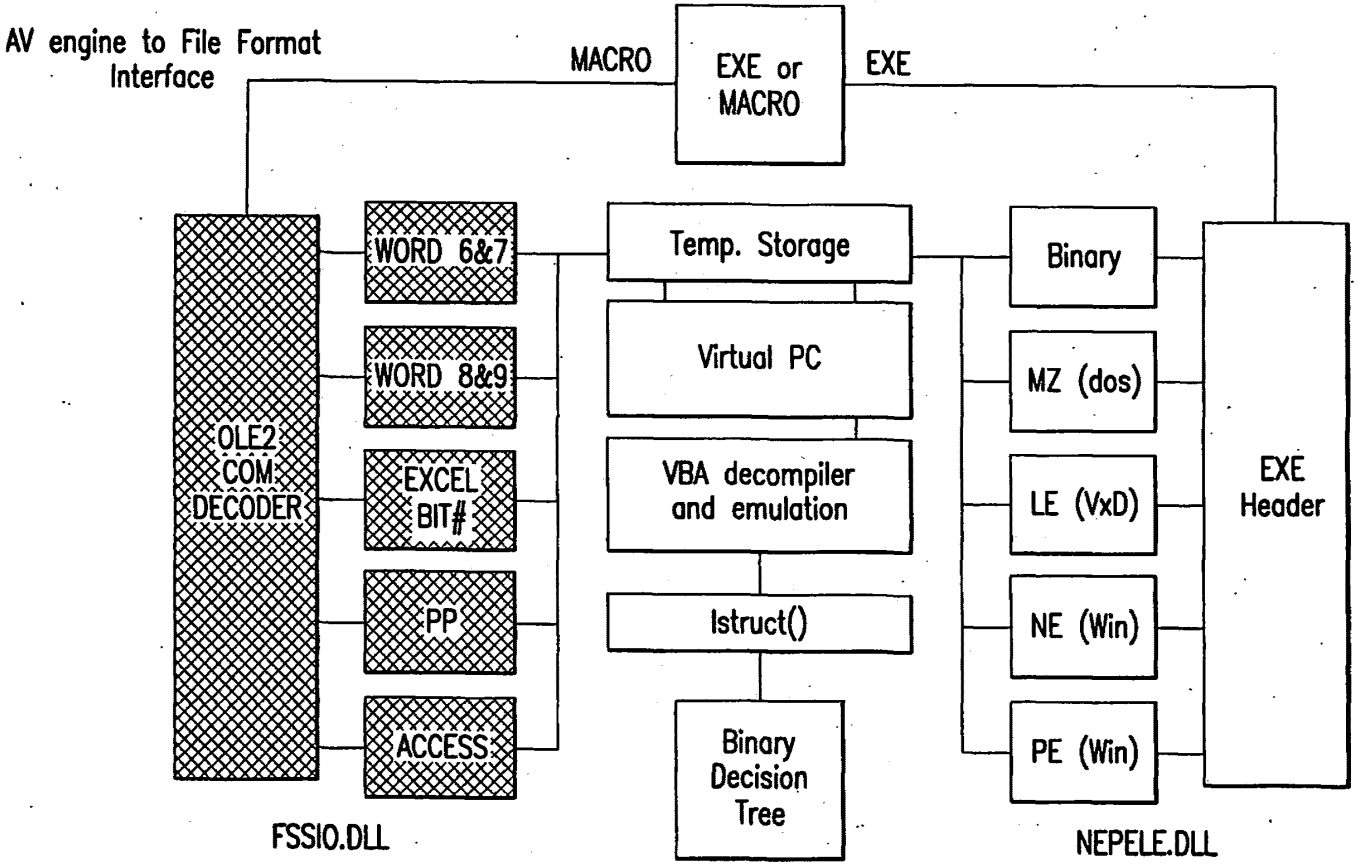


FIG.4

IPR2024-00027
CrowdStrike Exhibit 1002 Page 120

5/6

V80X86

MEMORY MAPS FOR BINARY COM AND EXE FILES

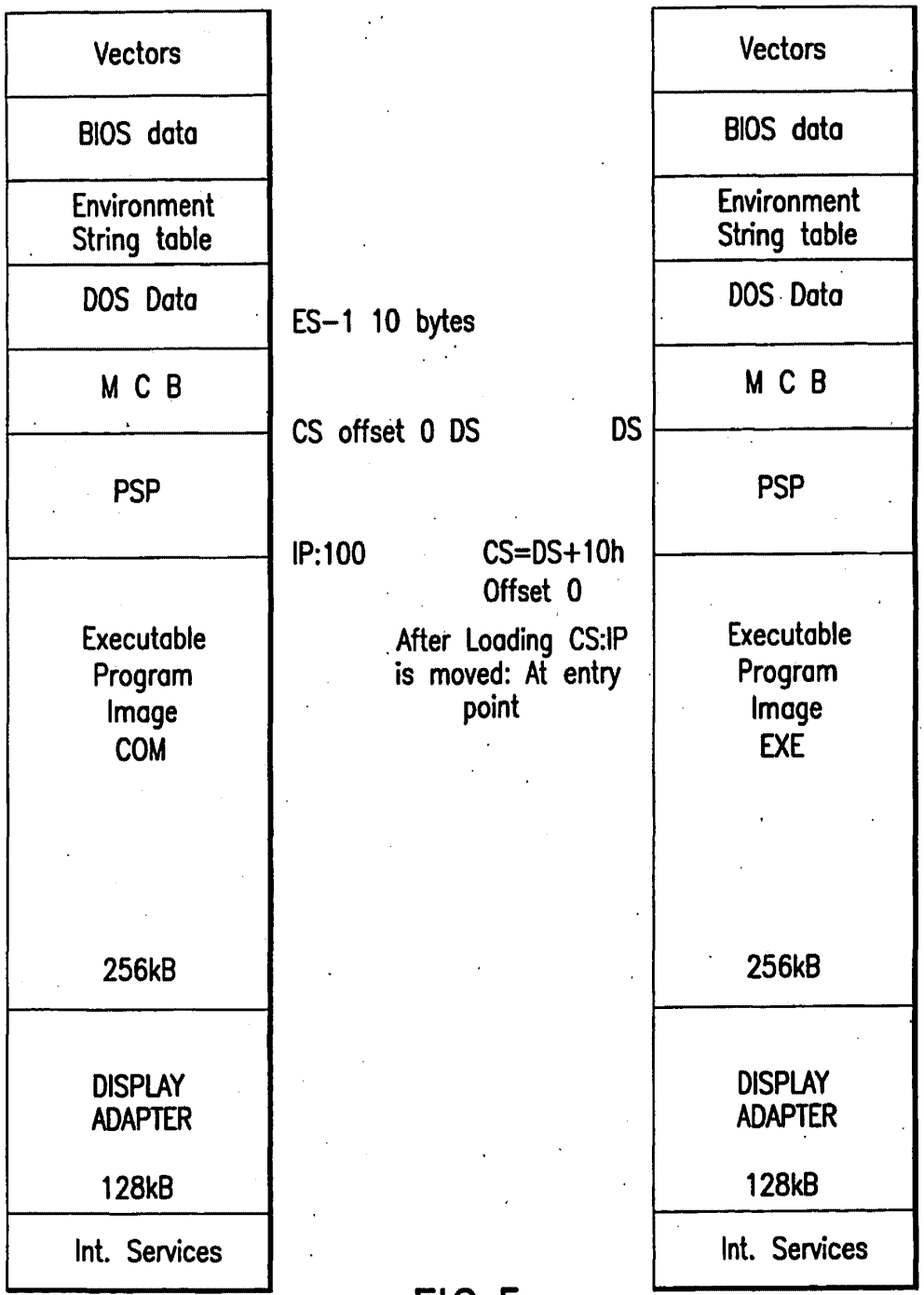


FIG.5

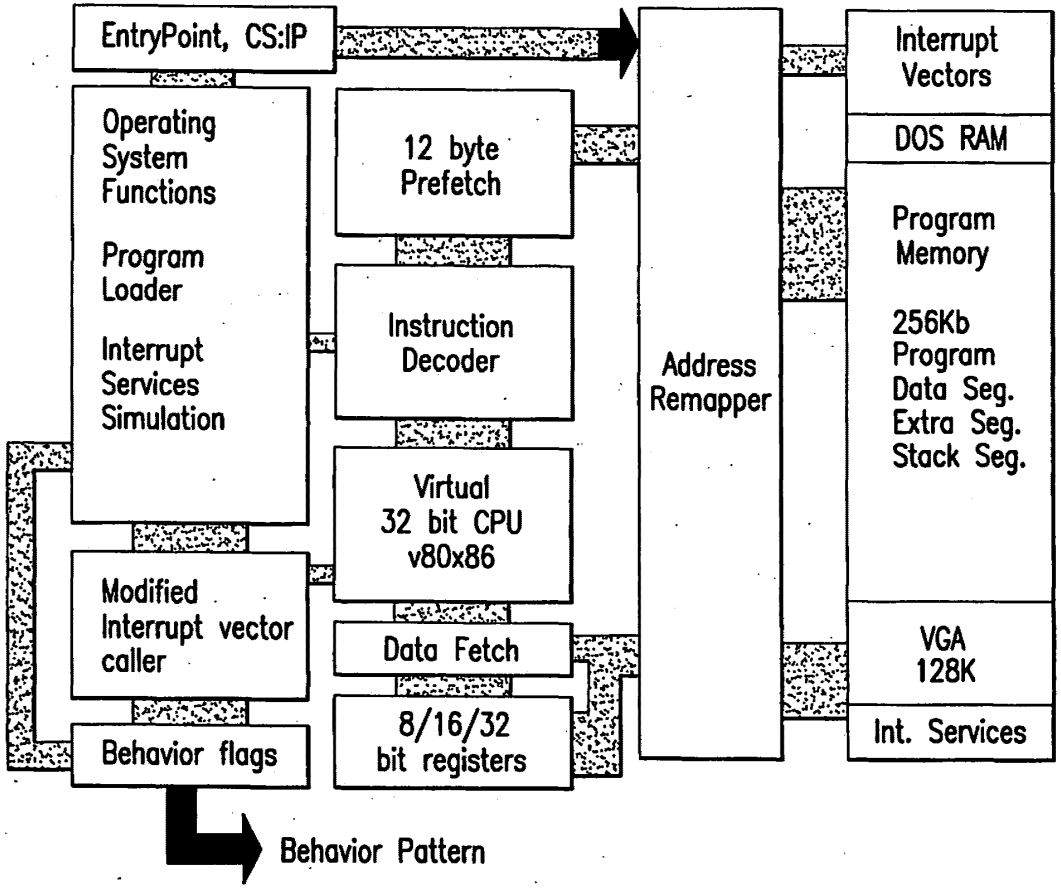


FIG.6

IPR2024-00027
CrowdStrike Exhibit 1002 Page 122

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 July 2002 (18.07.2002)

PCT

(10) International Publication Number
WO 02/056152 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US02/00900

(22) International Filing Date: 10 January 2002 (10.01.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/261,155 10 January 2001 (10.01.2001) US

(71) Applicant: **PSIONIC SOFTWARE, INC.** [US/US]; 7501 N Capital of Texas Hwy, Bldg B, Austin, TX 78731 (US).

(72) Inventors: **ROWLAND, Craig, H.**; 6908 Dogwood Hollow, Austin, TX 78750 (US). **PETTIT, Justin**; 70 Redding Road, Campbell, CA 95008 (US). **RHODES, Aaron**; 642 West Maple Street, Clyde, OH 43410 (US). **IRWIN, Vicki**; 1703 Persimmon Road, Cedar Park, TX 78613 (US).

(74) Agent: **RUSSELL, Douglas, D.**; Taylor Russell & Russell, P.C., 4807 Spicewood Springs Road, Building One, Suite 1200, Austin, TX 78759 (US).

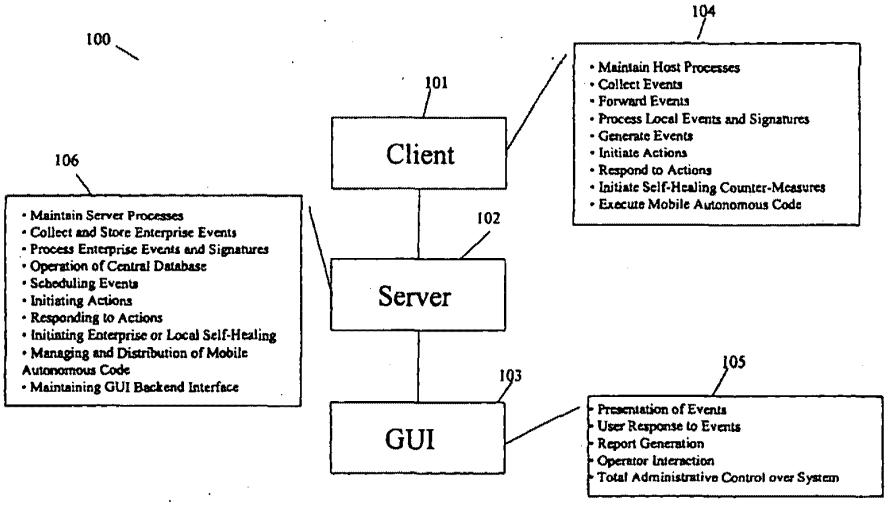
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, 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, 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*
— *entirely in electronic form (except for this front page) and available upon request from the International Bureau*

[Continued on next page]

(54) Title: COMPUTER SECURITY AND MANAGEMENT SYSTEM



(57) Abstract: The present invention provides a generic distributed command, control, and communication framework that allows computer systems, devices, and operational personnel to interact with the network as a unified entity. The present invention provides these services by building upon a core communication architecture that permits local or remote execution of mobile program code, static execution of program code, flexible communication formats, self-healing network techniques, and expansion by the addition of new system modules, software handlers, or mobile autonomous agents.

WO 02/056152 A2

IPR2024-00027
CrowdStrike Exhibit 1002 Page 123

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.

COMPUTER SECURITY AND MANAGEMENT SYSTEM

by Craig H. Rowland of Austin, Texas
Justin Pettit of Campbell, California
Aaron Rhodes of Clyde, Ohio
Vicki Irwin of Cedar Park, Texas

This application claims benefit of U. S. Provisional Application No. 60/261,155, filed on January 10, 2001.

Background

The invention relates generally to distributed intrusion detection, network management and host management systems. More particularly, it relates to autonomous self-healing computer network and computer management tools for computer security that operate in a distributed and localized manner.

The increased use of the Internet, intranets and extranets for gaining access to computer systems and networks has led to a commensurate increase in unauthorized access or attempted access into these systems and networks. This activity is unauthorized whether or not its purpose is of a malicious nature. As a result, intrusion prevention, detection and correction technologies have taken on a more significant role in computer system and network security.

Most of the systems in use today to prevent and detect intrusions are applicable to centralized client-server networks. These intrusion prevention and detection systems do not have the capability to operate effectively over widely distributed networks and systems in a unified manner. Nor do they have the capability to isolate and repair network and system elements that have been maliciously altered. They are also unable to re-allocate resources to compensate for defective network and system elements. Operation of many of these intrusion detection systems is limited to automatically collecting and reducing data, while the analysis of that data usually remains a manual process. Profiling and pattern recognition techniques also have been used to analyze the data collected and presented to an intrusion detection system. Some intrusion detection systems, based on anomaly detection techniques, look for statistically anomalous behavior, that is, behavior that appears unusual when compared to other user behavior. These systems are prone to both false positive and false negative alerts, resulting in a slow or inadequate response to the intrusion. Some intrusion detection systems use expert systems, which are driven from an encoded rule base to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 125

monitor policy compliance to ensure that all users are operating within their privileged rights. Other systems have passive monitor functions that continually analyze data presented to them. Another type of intrusion detection system is a scanner that actively attempts to find security holes (called vulnerabilities) and unauthorized hardware and software. Relying on systems with these limited capabilities can result in financial loss and system damage to an organization.

It is desirable to provide a computer security and management system that enables a distributed framework for command, control and communication that enables systems, devices and operational personnel to interact with a network as a unified entity. It is further desirable to provide this command, control and communication by using a core communication architecture that allows local and remote execution of mobile program code, and static execution of program code. Such a system should enable flexible communication formats, self-healing network techniques, and expansion by adding new program modules, software handlers, and mobile autonomous agents.

Summary

The present invention provides a generic distributed command, control, and communication framework that allows computer systems, devices, and operational personnel to interact with a network as a unified entity. The present invention provides these services by building upon a core communication architecture that permits local or remote execution of mobile program code, dynamic and static execution of program code, flexible communication formats, self-healing network techniques, and expansion by the addition of new system modules, software handlers, or mobile autonomous agents.

The system components may comprise at least one client, at least one server, and at least one graphical user interface. These components may be interconnected by encrypted and authenticated communication links. They may be customized for site-specific requirements. The system components interact to isolate and process system status messages, security alarms, administrative tasks, mobile autonomous agent functions, self-healing network functions, and user-implemented modules. The clients and servers each contain a Master Control Process and associated system handlers. The Master Control Process provides a message routing function for transferring messages between the various handlers.

A computer implemented method having features of the present invention for providing system security and resource management comprises managing event messages

IPR2024-00027
CrowdStrike Exhibit 1002 Page 126

by a master control processor between system handlers according to security system policies, processing network messages by a network handler between client and server computers, inserting native and third party event messages received by an insertion handler into the master control processor for processing by other system handlers, detecting and processing event message signatures by the signature handler from alarm, system, and insertion events for conversion into system alarm messages for action by the other system handlers, and performing actions by an action handler in response to action requests from the master control processor. The method may further comprise maintaining an execution environment by an agent handler for mobile autonomous code modules. The method may further comprise collecting and logging event messages by a logging handler. The method may further comprise managing system configuration parameters by a configuration handler.

In an alternate embodiment of the present invention, a system for providing system security and resource management comprises a master control processor for managing event messages between system handlers according to security system policies, a network handler for processing network messages between client and server computers, an insertion handler for inserting native and third party event messages into the master control processor for processing by other system handlers, a signature handler for detecting and processing event messages from alarm, system, and insertion events for conversion into system alarm messages for action by other system handlers, and an action handler for performing actions in response to action requests from the master control processor. The system may further comprise an agent handler for maintaining an execution environment for mobile autonomous code modules, a logging handler for collecting and logging event messages, and a configuration handler for managing system configuration parameters. The system may be installed on at least one server computer and at least one client computer. The system may further comprise at least one graphical user interface.

In another embodiment of the present invention, computer executable software code stored on a computer readable medium, the code being for a computer implemented method for providing system security and resource management comprises code for managing event messages by a master control processor between system handlers according to security system policies, code for processing network messages by a network handler between client and server computers, code for inserting native and third party event messages received by an insertion handler into the master control processor for processing by other system handlers, code for detecting and processing event message signatures by the signature

IPR2024-00027
CrowdStrike Exhibit 1002 Page 127

handler from alarm, system, and insertion events for conversion into system alarm messages for action by the other system handlers, and code for performing actions by an action handler in response to action requests from the master control processor. The computer executable software code method may further comprise code for maintaining an execution environment by an agent handler for mobile autonomous code modules, code for collecting and logging event messages by a logging handler, and code for managing system configuration parameters by a configuration handler.

Brief Description of the Drawings

These and other features, aspects and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings wherein:

Fig. 1 shows an overview of a computer security and management system according to an embodiment of the present invention;

Fig. 2 shows a block diagram of the master control process and system handlers according to an embodiment of the present invention;

Fig. 3 shows flowchart depicting handler registration according to an embodiment of the present invention;

Fig. 4 is a flowchart of action parameter processing for signature detection according to an embodiment of the present invention;

Fig. 5 is a functional block diagram of the Network Handler;

Fig. 6 is a functional block diagram of the Insertion Handler;

Fig. 7 is a flowchart of the processing of the Signature Handler;

Fig. 8 is a flowchart of the processing of the Signature Handler;

Fig. 9 shows the processing for decoding a "Port Scan" alarm;

Fig. 10 is a flowchart of signature registration process;

Fig. 11 is a diagram of centralized Mobile Autonomous Code (MAC) Agent distribution;

Fig. 12 is a diagram of a Peer-To-Peer Defensive Cluster with MAC Agents;

Fig. 13 is a diagram of MAC Agents for security-specific applications; and

Fig. 14 is a diagram of MAC Agents for network management-specific applications.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 128

Detailed Description of the Drawings

Turning now to Fig. 1, Fig. 1 shows an overview of a computer security and management system 100 according to an embodiment of the present invention. In the present embodiment, the system is comprised of a client computer 101, a server computer 102 and a graphical user interface 103. Although Fig. 1 depicts one embodiment of the present invention, other embodiments may include as few as one combined client/server computer or a multiple of client and server computers. These components may be linked together through encrypted and mutually authenticated communication links to prevent viewing of sensitive data by unauthorized personnel and to thwart any attempts at subverting the protective mechanisms. The components isolate and process system status messages, security alarms, basic and advanced administration tasks, mobile autonomous agent functions, self-healing network functions, and user-implemented modules to present a unified interface to administrative personnel. Many of the tasks are automated and self-configurable to reduce administrative workload and increase overall system reliability. All components are configurable by the end-user to allow for customization of the end product for site-specific needs. The client computer 101 is responsible for maintaining host processes, collecting and forwarding events, processing local client signatures, generating events, initiating and responding to action requests, initiating self-healing counter-measures, and hosting mobile autonomous agents 104. The server computer 102 is responsible for collecting and storing events, processing enterprise client signatures, generating events, operating the central system database, scheduling events, initiating and responding to action requests, initiating self-healing counter-measures, maintaining GUI backend infrastructure and managing the mobile autonomous agents 106. The GUI 103 is responsible for presentation of events, allowing users to respond to events, generation of reports, scheduling of events, and allowing administrative control over all system processes 105. The GUI 103 may be hosted on a separate computer or may communicate directly with either the client computer 101, the server computer 102 or both, or may communicate with the client computer 101 and the server computer 102 via a network such as an Internet, Intranet or the like.

The architecture of the system is designed to allow modularity and flexibility in the components. The modularity allows easy expansion, yet the design of the architecture allows effective monitoring and control with a minimum number of components operating. In Fig. 1, only one client computer 101, server 102 and GUI 103 are shown. However, multiple components of each type may be used in the present invention. The system reuses

IPR2024-00027
CrowdStrike Exhibit 1002 Page 129

software concepts and components wherever possible. The system is designed so that the server 102 and client 101 function as similarly as possible internally. In one embodiment, the difference between the two is the way in which data is sent and received. Most client and server components can be reversed depending upon the role they serve in the system. This design has an added benefit of creating clients that can "morph" or change into servers in case of central system failure or other problem. For example, the system is designed with "reversible" components so a client system can work as a server and the server software is reused wherever possible to avoid redundancy. The architecture has built-in mechanisms for detecting failures and circumventing problems. The architecture allows expansion through a simplified Application Programming Interface (API) and standardized messaging format.

Turning now to Fig. 2, Fig. 2 shows a block diagram of the master control process and system handlers 200 according to an embodiment of the present. Both the client 101 and server 102 computers of Fig. 1 contain a Master Control Process (MCP) 201 and system handlers 202. The MCP 201 functions as a message routing system that handlers 202 communicate with. Handlers 202 register themselves with the MCP 201 and then may use the MCP 201 to pass messages between themselves. In this way, the MCP 201 functions as a passive message traffic controller and is not involved in the actual operation of the handlers 202 themselves.

The MCP 201 is a component of both the client and server systems. The MCP 201 may be an identically functioning software component in both the client and server computers (or similarly functioning software component) that manages a series of system handlers 202-209 that have registered with the MCP 201. The MCP 201 is responsible for passing communication messages between the handlers 202-209 and for managing this message traffic according to system policies. The MCP 201 is designed to be an expandable lightweight multi-threaded application with expandable communication paths that manages a series event queues on the system. Attached to these queues are "handlers" 202-209 that are either static (always running) or dynamic (executed on demand) system processes.

A "system handler" 202-209 is software designed to perform one or more specific functions. These functions can be initiated by the handler 202 itself through external or internal mechanisms or be passed to the handler from the MCP 201 for processing. Handlers are written to process one or more type of messages depending on the application. Handlers register themselves as services on the MCP. These services are accessible through the MCP as Remote Procedure Calls (RPC). The RPC mechanism allows for local or remote callers equal access to system assets transparently. In one embodiment, the RPC services provided

IPR2024-00027
CrowdStrike Exhibit 1002 Page 130

are independent of the MCP's knowledge, that is, handlers may automatically register their capabilities with the MCP and the MCP requires no knowledge of their function. For example, the MCP 201 just needs to know how to accept registration of services and does not need to know what each specific service does. The MCP 201 can pass multiple message types between handlers depending on the auto-registration process. The MCP 201 uses a lightweight threaded programming design. The entire process is very fast with low overhead for rapid processing of messages and routing information. Using this generic mechanism allows the actual MCP 201 to contain a minimal amount of code for maximum reliability and speed.

Because the MCP 201 requires no knowledge of the handler functions, it can function in a "reversible" mode on both the client and server. That is, the same or similar software code may be used on both the client and the server with handlers designed for either server or client side operation. Additionally, a client on a network may switch into server operation in the event of a failure of the main central server. This provides for a fault-tolerant command and control mechanism. In another embodiment, the system with MCP reversible client/server software provides for client-to-client communications and "defense clusters" where groups of computers communicate with one another to defend themselves completely independent of central control. Clients can communicate in a peer-to-peer fashion to form these defense clusters where systems can defend themselves independent of a central control point.

As discussed above, "handlers" 202-209 are software code designed to perform one or more specific functions. These functions can be initiated by the handler itself through external or internal mechanism or be passed to the handler from the MCP on its input queue for processing. Handlers are written to process one or more types of messages depending on the application. Handlers register their services to the MCP and can provide these services to remote callers as an RPC mechanism.

Handlers are designed to be very focused pieces of code that perform a fixed set of very specific functions. Because in one embodiment, a handler focuses on one area of the system, the number of software errors ("bugs") that are contained in the code may be reduced which optimizes overall system debugging efforts. Additionally, by containing the function of handlers to very specific areas, overall system integrity may be increased because message types and data can be restricted to parts of the system that require its use. This can improve speed along with system security.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 131

Turning now to Fig. 3, Fig. 3 shows flowchart depicting handler registration 300 according to an embodiment of the present invention. Before a handler can begin operation the MCP must initialize it. Since the MCP is the ultimate controller of the system, all handlers must operate in a predictable and standardized manner. The MCP scans the local configuration file and handler directory for handlers that need to be initialized 301. Handlers are entered into the MCP by name and the Handler API is initialized 302. The Handler is then initialized to begin both generation and processing of system events it is designed to accommodate. If the Handler is to run as an RPC service 303, its capabilities are made available through the RPC interface for local or remote callers 305. Otherwise, if the Handler is not going to run as an RPC service, the handler input/output queues are initialized 304. Handler Registration 300 is performed on MCP system startup, operator request, handler "wakeup", or similar system operation. Handlers have a reversible software architecture that allows them to be used in either a client or server computer mode. This eliminates redundant code, eases debugging, and allows for optimization of handler operations.

Turning back to Fig. 2, handlers serve a variety of purposes depending on their type. Since the handlers perform the vast majority of system operations, they are often charged with performing a variety of tasks such as:

- Logging Handler 203 – Logs system events locally or remotely.
- Action Handler 204 – Responds to system requests to perform active actions locally.
- Network Handler 205 – Processes network traffic to and from the client and server.
- Configuration Handler 206 – Manages configuration of the client and server systems either locally or remotely
- Insertion Handler 207 – Allows native and third party applications the ability to insert events directly into the system architecture.
- Signature Handler 208 – Processes events and converts the events into system alarms.
- Agent Handler 209 – Provides remote execution and reporting environment for mobile autonomous code.

In addition to the above listed handlers, other handlers may provide for the processing of any other activity as specified by the operator of the system. Because the MCP is built upon a generic architecture and the handler contains an API and standardized messaging format, new handlers can be added. All handler modules are pluggable and can be changed by an operator or as the host computer system allows.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 132

Referring to Fig. 2, the Logging Handler 203 enables the system to audit events for the local system on which it resides. The Logging Handler 203 can accept messages to be logged from any part of the system for any reason. The Logging Handler 203 may log the event in any number of ways to the local system that includes:

- Text-based file
- Local system auditing facility (Unix® compatible syslog daemon, Windows NT® event log, etc.)
- Cryptographically signed secure log format
- Directly to the system console
- E-mail notification
- Direct user notification
- System wide notification

The Logging Handler 203 may forward log messages directly to a remote server for logging in the central database and for reporting of system generated events. The Logging Handler 203 can accept multiple file formats for input and output. The Logging Handler 203 can cryptographically secure log files for protection of forensic evidence. The Logging Handler can provide notification in a number of ways: via e-mail, direct messaging, system wide notification and user specified.

Referring to Fig. 2, the Action Handler 204 is responsible for receiving "action requests" from the system architecture and to react to those requests with appropriate responses. In this context an "action request" can consist of anything the Action Handler 204 is programmed to do. Typically, the Action Handler will be responsible for performing one of the following actions based upon its instructions:

- Block a host with a modified route command
- Block a host with a packet filter modification command
- Disable a user account
- Disable a network interface
- Run an external user-defined command
- Log the event
- Send E-Mail or pager alerts
- Send on-screen alerts to users or administrators
- Request server executed action
- Any other pluggable action as defined by the user

IPR2024-00027
CrowdStrike Exhibit 1002 Page 133

The Action Handler 204 is designed with pluggable action modules. These action modules allow the Action Handler 204 unlimited functionality. With pluggable action modules the Action Handler 204 can be presented with new action types to process with no additional configurations. Additionally, for an action the Action Handler 204 is not capable of executing, it can request those action types be sent to it from the server as new pluggable action modules. This allows the Action Handler 204 to use pluggable reaction modules to expand capabilities. The Action handler can respond with blocking hosts, users, networks, running commands, logging events, disabling interfaces, disabling computer, sending e-mail, paging personnel, providing on-screen alerts, etc.

The actions that the Action Handler 204 is capable of performing must describe the parameters required for proper execution. These parameters are defined in their configuration and vary depending on their function. Parameters for the action are defined in the configuration file for the action module. So if an action is designed to block hosts, the parameters required in the configuration file would be an address of a host to block. For example, for a host blocking action type, the Action Handler 204 requires a host IP address to block (usually the attacking host). In another example, for a user disabling action type, the Action Handler 204 requires a username to perform the disable action. The Action Handler 204 passes action parameter information to a pluggable action module where the actions can be executed. The Action Handler 204 does not need to know the details of the actions, but only that the correct parameters are being passed based on how the actions are registered with the system.

Turning now to Fig. 4, Fig. 4 is a flowchart of action parameter processing for signature detection 400 according to an embodiment of the present invention. Action type parameters are specified for the signature-processing portion of the present invention. Signatures entering the system are processed in real-time and the response action is retrieved from the local action policy. The action policy will tell the signature how to respond (block the host, disable the user, etc.). The signature will then pass the appropriate parameters to the action type depending on what the action type requires to operate correctly. A signature detects an attack (for example, a port scan) 401 by reading the signature module. The Signature Handler (208 in Fig. 2) consults action policy 402 to determine an action needs to take place (in this example, blocking the attacking host) 403. The Signature Handler looks up what parameters the block host action needs (attacker's IP address) 405. The Action Handler (204 in Fig. 2) processes the request and executes the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 134

actions 406 (in this example, passes the attacker's IP address to the block host action for execution). The Action Handler returns the status of the executed action to the caller 407 and the process is completed 404. The Action Handler finishes processing the request and sends the result of the operation to the MCP. The MCP routes the status message to the Signature Handler which then routes the message according to it policy (typically to a local log and to the central server) to report the status of the request.

Turning now to Fig. 5, Fig. 5 is a functional block diagram of the Network Handler 501 (205 in Fig. 2). The Network Handler 501 is responsible for all network-bound communications in the present invention. The Network Handler 501 is responsible for the following: incoming and outgoing Client/Server Common Network 502 communications from the Client 506 or Server 507, including Access Control 508 and Authentication 503, Compression 504 and Encryption 505 of all network communications.

The Network Handler 501 is designed to use any available network protocol as its communications medium. For example, the Network Handler may use a Transmission Control Protocol (TCP) based communications protocol to send data in the form of message units.

The Network Handler 501 only accepts connections from clients and servers defined in its internal Access Control List (ACL) 508. The ACL 508 is stored in a local file, database, or other suitable format and is consulted whenever an incoming connection request is made. Clients 506 wishing to connection to the TCP port of the Network Handler 501 are first looked up in the appropriate ACL 508 table. If the client is listed in the ACL 508 then it is allowed to proceed to full authentication. If the client is not listed, the connection is dropped and the Client 506 or Server 507 generates an appropriate connection refused alarm event for reporting purposes.

The Network Authentication function 503, which contains the protocols for authentication within the Network Handler 501 can be changed depending on new or updated security vulnerabilities, enhancements in technology, or for other reasons. The Network Handler 501 supports multiple authentication types depending on speed and security requirements.

The Network Handler 501 can be expanded to provide a large number of authentication mechanisms for client and server communication depending on the needs of the system. The Server 507 may negotiate the default protocol to be used upon accepting the Client 506 through the network ACL 508. The Server 507 default protocol may be set by the system administrator. Provided protocols that allow for mutual authentication are

IPR2024-00027
CrowdStrike Exhibit 1002 Page 135

preferred. During the negotiation phase, the Server 507 will list one or more protocols for authentication that it will accept. If the Client 506 does not support the protocol requested, it may choose from a list of other potential protocols the Server 507 will accept. If the Client 506 supports none of the protocols the Server 507 requests, the connection is dropped and both the Client 506 and Server 507 are notified. Notification may be by generating an appropriate alarm event on the Client 506 and Server 507.

The network data that the Network Handler 501 will be forwarding and receiving can be very large in size. To minimize bandwidth consumption, the Network Communication Compression function 504 compresses the data before it is sent using a compression algorithm. Many types of compression algorithms are possible including Huffman/LZ77 encoding or similar type of algorithm. Because encrypted data usually does not compress well, the data compression algorithm is normally implemented before processing by the Network Communication Encryption function 505 of the communication protocol. Any compression algorithm can be used for this purpose, or compression can be completely disabled if required by the administrator.

Because critical information will be passed over the client and server communications links, the Network Handler 501 includes a Network Communication Encryption function 505 which contains one or more data encryption algorithms for encrypting all network data. The Network Handler 501 is responsible for negotiating the encryption algorithm to be used on the data stream. Encryption algorithms may be added or removed from the Network Handler 501 depending on the configuration.

During authentication, the Server 507 presents a list of encryption algorithms it is willing to accept. The Client 506 may then select an algorithm. If the Client 506 supports none of the encryption protocols of the Server 507, then the connection is dropped and both the Client 506 and Server 507 are notified. Notification may be by generating an appropriate alarm event on the Client 506 and Server 507. The system administrator may add or remove encryption protocols.

The Network Handler 501 can switch to different authentication and encryption protocols and methods in the event of a security compromise of one of its methods. Authentication and encryption mechanisms are changeable on the Client 506 and Server 507, depending on system requirements or security needs.

Turning back to Fig. 2, the Configuration Handler 206 provides an interface to system configuration libraries and data stores for system parameters. The Configuration Handler 206 is responsible for interfacing these generic calls so other handlers, system

IPR2024-00027
CrowdStrike Exhibit 1002 Page 136

processes, and system operators can have quick, reliable, and uniform access to configuration data. The Configuration Handler 206 is responsible for providing the following access to system parameter data:

- Read – Allows system parameters to be read
- Write – Allows system parameters to be written or added
- Change – Allows system parameters to be changed
- Backout – Allows system parameters to be reverted back to the previous version
- Delete – Allows system parameters to be deleted
- Backup – Allows all system parameters to be backed up

The Configuration Handler 206 is responsible for all relevant file locking and multiple access protection mechanisms as well as providing sequential access to threads, processes, and system operators. The Configuration Handler 206 allows configuration of the system to occur locally or remotely from the central command and control system.

Turning now to Fig. 6, Fig. 6 is a functional block diagram of the Insertion Handler 600 (207 in Fig. 2). The Insertion Handler 600 is a mechanism whereby native or third party applications can insert events directly into the present system. The Insertion Handler 600 accepts messages, ensures they are in the correct format, and then forwards them to the MCP 601 for further processing. The Insertion Handler 600 has two modes of data insertion containing Insertion Libraries 602: a native insertion library and an external insertion library.

The Insertion Handler 600 provides the mechanism for inserting events from External Programs 606, 607 into the system by any of the following means: placing a File Descriptor 603 on the system for reading/writing, placing a Network Socket 604 on the system for reading/writing or placing a Named Pipe 605 on the system for reading/writing. The Insertion Handler 600 allows legacy, third party, or other applications the ability to send native messages directly into the system with few modifications. The Insertion Handler 600 allows communication with the system to occur with messages using a variety of input mechanisms (File Descriptor 603, Network Socket 604, Named Pipe 605 and the like). The Insertion Handler 600 can place file descriptors inside of protected environments to operate in a secure manner. In one embodiment, the Insertion Handler 600 is used to pass inserted alarm messages directly to the Signature Handler (208 in Fig. 2) for processing.

Referring to Fig. 6, in one embodiment, the Insertion Handler 600 can utilize a native Insertion Library 602 for injecting messages into the present system. The native

IPR2024-00027
CrowdStrike Exhibit 1002 Page 137

Insertion Library 602 is a fixed set of methods that the native system programs or add-ons can access and use to directly insert messages. The native Insertion Library 602 would typically be used by programs written in the same language as the core architecture or by those programs developed for the core system specifically (e.g., those from the manufacturer or by authorized third parties).

In another embodiment, the Insertion Handler 600 utilizes an external Insertion Library 602 that is essentially a "wrapped" version of the native Insertion Library 602 discussed above. The external Insertion Library is a C or C++ library designed to be linked to third-party sources (other languages such as Perl, Python, TCL, Unix® shell script, etc. can also be supported). This linked library provides a suite of system calls that allow the wrapped program to inject messages directly into the core system without special configurations. It only requires the addition of appropriate system calls throughout the code to audit relevant events (security, administrative, failure, error, etc.). Once the code has been modified in this way, it may then insert the messages directly into the core system as if it were a native application. The external Insertion Library 602 allows third-party applications to be modified and added to the present invention. The third-party applications then have the ability to detect whether the core system is installed and communicate directly with the system, minimizing user interaction and system overhead. The Insertion Handler 600 allows third parties to integrate system support using a variety of languages such as C, C++, Java, Perl, Python, and the like.

If File Descriptor 603 insertion mode is used, the Insertion Handler 600 uses a Unix®, Windows NT®, or similarly based mechanism for placing a "file descriptor" directly onto the system. This File Descriptor 603 is continuously polled by the Insertion Handler 600 for new data from the Insertion Library 602, either the native or external Insertion Library 602. Once data is received on the socket it is inserted into the MCP 601 after appropriate validation checks on the data have been performed. The File Descriptor 603 for the Insertion Handler 600 is placed in a secured directory and is only accessible to authorized programs and system services. This method may also allow the creation of multiple file descriptors that can be placed in user-defined directories. This allows the Insertion Handler 600 to work in UNIX® chroot() or similar secured operating environments for running un-trusted code.

If Named Pipe 605 support insertion mode is used, the Insertion Handler 600 uses either the Unix® and Microsoft® operating system environments. This mode functions almost identically to the File Descriptor 603 insertion method described above with the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 138

difference being that a "named pipe" instead of the local file descriptor is used. A pipe is a technique for passing information from one program process to another and is a one-way communication. A named pipe is a method for passing information from one computer process to other processes using a pipe or message holding place that is given a specific name. Unlike a regular pipe, a named pipe can be used by processes that do not have to share a common process origin. The message sent to the named pipe can be read by any authorized process that knows the name of the named pipe.

The Network Socket 604 insertion mode is the most generic method for accepting messages by using a local network socket listening on the system for new insertion messages. This mode of operation requires the opening of a locally bound network socket on the system. Like the other modes, the socket is continuously polled by the Insertion Handler 600 for new data from the Insertion Library 602, either the native or external Insertion Library 602. Once data is received on the Network Socket 604, it is inserted into the MCP 601 after appropriate validation checks on the data have been performed.

For security reasons, in one embodiment this socket is limited to connections from the local host system. The Network Socket 604 insertion handler is a TCP based socket to protect against maliciously forged packet attacks.

The Insertion Handler 600 (207 in Fig. 2) messages are generally passed to the Signature Handler (208 in Fig. 2) as described below. In one embodiment, the Insertion Message should at a minimum: be formatted in a mark up type language (such as Extensible Markup Language (XML)), define the originator of the message (for example a third party application or internal system application), define an alarm identification type to allow the Signature Handler (208 in Fig. 2) to properly pull out relevant alarm data and contain all relevant alarm data necessary to allow the corresponding signature to process the event.

All messages passed into the Insertion Handler 600 should have a corresponding signature in the Signature Handler (208 in Fig. 2). The signature determines how the message will be processed and what actions should be taken when it is received. If the message cannot be processed by the Client (506 in Fig. 5), the Signature Handler (208 in Fig. 2) passes the message onto the Server (507 in Fig. 5) for processing.

Returning now to Fig. 2, the Signature Handler 208 processes alarm events in real-time or near real-time and decides any actions and responses of the system to the alarms. In one embodiment, the Signature Handler 208 accepts the following types of messages:

1. Alarm events – Those events generated by native detection mechanisms to indicate a security, management, or similar event.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 139

2. System events – Errors, status, and other informative messages relating directly to the present invention system.
3. Insertion events – Those events generated by either native or external programs that were passed via XML format into the Insertion Handler 207 (600 in Fig. 6). These events are usually Alarm events relating to security, management, or similar situation.

The Signature Handler 208 operates similarly to the MCP 201 but it is only responsible for alarm events and not for individual handlers as is the MCP 201. The Signature Handler 208 is responsible for the conversion of events into system alarms with which the present invention system can report and respond to.

Turning now to Fig. 7, Fig. 7 is a flowchart of the processing of the Signature Handler (208 in Fig. 2). The Signature Handler accepts alarms events from the MCP 701. The Signature Handler decodes the alarm type and originator from the alarm event 702. The Signature Handler then consults its internal registry of signatures that wish to process alarms of the type received 703. The Signature Handler then hands the alarm message off to the signature for processing 704. The signature given the alarm message is responsible for pulling out the alarm data macro information it needs to function correctly (IP address, port numbers, usernames, and the like) 705. The signature determines if an alarm has occurred and then consults the action policy for that alarm type to determine what actions to take in response to the alarm 706. The Signature Handler extracts relevant information from the alarm, insertion, or system event to pass into the action message. This data can include such things as IP Addresses, Usernames, Process Identification and the like. When signature processing is complete, the Signature Handler may then pass the resulting message back to the MCP 201 in Fig. 2) for appropriate routing 707 (Action request, Network request, Logging request, and the like). The Signature Handler uses modular signatures that automatically pull necessary data from the alarm events received. The Signature Handler can operate independent of a central control system and can act autonomously based on pre-established policy in the system.

The Action Policy in the Signature Handler (208 in Fig. 2) is a list of actions that can be taken for each event that is processed by the Signature Handler itself. The Action Policy is configurable by the system administrator and maps the responses the system will perform relating to each detected event. In a simple embodiment the Action Policy contains a signature identification and a list of response actions to each signature. In addition, each action requires inherent data to operate correctly. With each response request coming in, the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 140

signature is responsible for supplying supporting data to ensure the action can operate correct. For example, if the Action Policy requires that a host be blocked as a result of a port scan, the Signature Handler must know the Internet Protocol (IP) address of the attacking host to pass into the blocking function otherwise the call will fail. Examples of required data include but are not limited to those listed in Table 1.

Action Type	Information Required from Signature
Block Host	Target IP Address
Disable User	Target Username
Disable Network Interface	Network Interface Name
Send e-mail	E-mail address and data
Move/copy/delete file	Filename

Table 1

The individual signatures require intelligence to properly pull out the required information from each signature (such as attacker IP addresses, usernames, and the like). In one embodiment, XML formatting is used to isolate the data types within each signature to make the processing easier. For many cases, the local detection modules that generate the event will have this data already separated. As an example, if a port scan is detected by a port scan local detection module, it generated and event and the processing is as follows. A port scan to a TCP port 143 (IMAP) is detected by the port scan detector . The port scan generates an alarm event that contains the alarm ID (which maps to the TCP port scan and the alarm data (for example, the IP Address of attacker, port number being scanned, and scan type). The alarm data is encapsulated into an alarm event format whereby the data is pre-processed with XML tags so items such as the attacker IP address, port number being attacked, and scan type are already separated for example:

a. <attacker_ip>192.168.2.30</attacker_ip>

b. <attacker_port _number>143</attacker_port_number>

c. <attacker_scan_type>FIN Scan</attacker_scan_type>

The entire alarm event is then sent to the Signature Handler (208 in Fig. 2).

Once inside the Signature Handler (208 in Fig. 2), the alarm event data is passed to the appropriate Signature Handler, in this case the TCP Port scan detector. The Signature

IPR2024-00027
CrowdStrike Exhibit 1002 Page 141

Handler then identifies that an alarm occurred and then looks up the action from the action event table:

1. TCP Scan detected
2. Lookup action in action event table which indicates:
    a. Block host
        i. Block host action requires <attacker_ip> data to run.
        ii. Data retrieved from <attacker_ip> tag and sent to action handler
    b. Log event locally
    c. Send event to server
        i. Entire event is packaged inside a network message and forwarded to server.
        ii. Server may use XML data tags within its signature handler for further actions and reporting.

Using the above processing, the Signature Handler (208 in Fig. 2) is isolated from having to know any direct information on how the signatures and actions operate. The Signature Handler just needs to route the event as it is received which is resolved during the signature registration process. The action policy defines what actions to take in response to a detected signature. This policy is configurable by the user or may be preset to defaults on system initialization. For each action request, the Signature Handler is responsible for pulling out the data necessary for the action to run correctly.

Turning now to Fig. 8, Fig. 8 is a flowchart of the processing of the Signature Handler (208 in Fig. 2). An Insertion, Alarm, or System Event enters Signature Handler 801. The Signature Handler passes message off to appropriate signature(s) who have registered to see signatures of that type 802. The Signature Handler determines if an alarm has occurred 803. If an alarm has occurred, the Signature Handler looks up the response from the action policy table 804. From the action policy table, the Signature Handler determines the action(s) to initiate 805. The action policy table also defines the data needed to run in their configuration and the signature handler is responsible for ensuring that the XML tagged data is sent correctly as parameters to the action 806. Each action type defines what data it needs to operate which may be in a separate configuration file. If the action being requested does not have the correct parameters defined, the action request fails (i.e. Blocking a host requires at a minimum that the IP address to be blocked be provided) 807.

The Signature Handler maps incoming events to appropriate actions to perform. The process comprises accepting an event from MCP, diverting the event to the signature

IPR2024-00027
CrowdStrike Exhibit 1002 Page 142

handler, determining if an alarm has occurred and if so, determining the appropriate response by looking up the local action policy and sending out appropriate response requests to the MCP.

In this case, all Alarm, System, and Insertion events must have corresponding signatures to process the alarm. However, not all signatures need to be directly related to each event. It is possible to have a generic "group" signature that only looks for system status events and simply re-packages them for passing across the network and logs them locally without processing them individually. This type of signature operates essentially as a "wildcard" parsing mechanism where any event type meeting a high-level mask is grabbed with minimal processing.

Alternatively, for specific signatures, the system may have only one particular signature type. An example may be a TCP port scan detection signature that looks for TCP port scans, pulls out the attacking host IP address, and then consults the action policy which directs that all alarms of this type blocks the host, logs the event locally, and passes it to the network controller. Then the signature will generate a block message for the action handler, a local log message, and finally a network message for the server to display on the GUI.

Turning now to Fig. 9, Fig. 9 shows the processing for decoding a "Port Scan" alarm. This is just one example of the types of alarms processed by the Signature Handler 900 (208 in Fig. 2). The steps of the processing are as follows. A "Port Scan" alarm event is received from the MCP 901. The Signature Handler decodes the alarm type 902. The Signature Handler 900 sees which pluggable signatures want to see the alarm 903. In this case the port scan alarm is processed by a scan detect module. The Signature Handler 900 passes the alarm off to those signatures interested in seeing it and returns appropriate message if the signature module should activate (scanDetect) 904. The Signature Handler 900 is responsible for looking up the local action policy for the event type received and sending that back to the MCP. The Signature Handler 900 repeats for next alarm 905.

The Signature Handler 900 functions similarly to the MCP. Signatures can be added or removed without modifying the core code base. Signatures can process what is of interest to them, sparing system resources. Signatures can maintain their own data between executions and can hold this data even between system re-starts or failures. Signatures can be customized depending on operator requirements, system optimizations, or other attributes. The Signature Handler 900 acts as a central point of control and can focus advanced control concepts into the system operation such as dynamic throttling of signature throughput under heavy load.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 143

The Signature Handler 900 includes "pluggable" signature modules. A pluggable signature module is code that is designed to perform a specific detection task. Each of the pluggable signature modules is focused on a minimal number of alarm events to process and is capable of being written independent of the Signature Handler 900 itself. Signature modules may be inserted into and out of the Signature Handler 900 at will. During initialization of the Signature Handler 900, the handler determines what signatures are available for it to use. The signatures automatically register their capabilities with the Signature Handler 900. During this phase the Signature Handler 900 "registers" the signature modules as depicted in Fig. 10.

Turning now to Fig. 10, Fig. 10 shows a flowchart of a signature registration process 1000. The Signature Handler (900 in Fig. 9) reads the signature module 1001, determines the signature name 1002 and determines the signature version 1003. If the signature is registered 1004, the required alarm types are determined 1005. How the signature is to be run is determined (for example, with threads, without threads, or as a process) 1006. The signature priority is determined 1007 and then registration is complete 1008. After the registration process 1000 is complete, the Signature Handler may then receive messages from the MCP (901 in Fig. 9) for processing. The signatures themselves are of two major types: Singular Signature Modules and Composite Signature Modules.

The Singular Signature Module requires the signature to take an input and return a true or false output depending on the alarm event data. This signature contains basic fall-through logic to determine the result. As part of its operation this signature type can store local state information in the database (for example a structured query language (SQL) database) for use in processing of future alarms and for tracking trends in data.

The Composite Signature Modules can use data stored by them in combination with data stored by other signatures or in globally available resources (such as SQL tables and the like). These signatures require that several items be true before they activate. These signatures can store local state information in the SQL database for use in processing future alarms and for tracking trends in data.

Pluggable signatures can use stored data from other signatures to detect alarms. Signatures can maintain long-running state information between executions. Signatures can maintain long-running state information between system startup and shutdown sequences.

The Signature Handler can operate in several modes of operation such as sequential processing or multi-threaded processing depending on where the handler is executing. In the Sequential Processing Mode, the Signature Handler pulls messages off of the MCP queue

IPR2024-00027
CrowdStrike Exhibit 1002 Page 144

one at a time and processes them in sequential order. This mode is the simplest implementation of the system architecture and has several advantages such as ease of implementation, no re-entrant code issues in a multi-threaded environment, composite signatures can be executed without fear of corrupting internal signature state data and debugging of the system and following signature data flow is simplified. In the Multi-threaded Mode, the Signature Handler extracts messages from the MCP queue and directs them to a thread pool where signatures can be processed concurrently. This mode has several advantages such as faster processing of signatures, processing of other queued alarms is not delayed, threads can be assigned priorities so more important alarms can be serviced faster. The signature handler can process signatures one at a time or concurrently using threads or event scheduling.

Returning to Fig. 2, the Agent Handler 209 is responsible for the execution environment of Mobile Autonomous Code (MAC) modules that circulate throughout the network. Mobile Autonomous Code is described in more detail below. The primary responsibilities of the Agent Handler 209 are:

1. To ensure the MAC modules carry appropriate credentials and are authenticated and cryptographically signed by a trusted introducer (network administrator, operator, client system, etc.).
2. To ensure the MAC modules are able to execute on the host operating system by checking version numbers and other appropriate identification information.
3. To execute the MAC modules and not interfere with operation until the MAC module is complete or is ordered to stop by an authorized party.
4. To allow the MAC to complete and report its findings and/or execute any actions on the local system as requested by the MAC.

The present invention has several core security and intrusion detection mechanisms such as log security in the form of log audit functions, login and logout anomaly detection functions; session monitors and a port scan detector functions. These and other features are described in United States Patent Application No. 09/268,084 filed on March 12, 1999, which is incorporated by reference herein.

The present invention comprises autonomous software programs that can patrol a communications network independent of the clients they are monitoring. Mobile Autonomous Code (MAC Agents) are independently distributed pieces of code that operate within a special execution environment within the client and server system. The MAC Agents are capable of performing any local system operation and reporting their findings to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 145

either a local, central, or distributed environment. While the initial goal of the present invention is to use MAC Agents to perform security operations, the overall goal is to use MAC Agents to administer and control all aspects of the networking environment and ultimately lead to a self-healing network architecture that can sustain itself with minimal human interaction. The MAC Agents are capable of moving between systems independent of the actual client. The MAC Agents are capable of performing any type of system operation permitted on the client. The MAC operate within a separate execution environment from the rest of the system that can be open or restricted depending on configuration. The MAC Agents can perform any security or system administration tasks and are targeted for use in self-healing network environment deployments. MAC Agents allow self-healing components of the system to move between clients and operate independently and where needed. MAC Agents can be written in either native object code or in a platform-independent interpreted language such as Python, Perl, or Java®.

Turning now to Fig. 11, Fig. 11 shows a diagram of this centralized MAC Agent distribution. In this embodiment, MAC Agents 1103 are designed to be dispatched from the central Server 1102 to the Clients 1104-1106. In this operational mode, the Server 1102 handles all configuration and deployment issues by pushing MAC Agents 1103 to one or more Clients 1101. This configuration allows for the MAC Agents 1103 to be centrally managed and deployed, easily updated from a central console, not subject to alteration or tampering by hostile adversaries and can be dispatched through a predictable or random schedule from a central control point. The central Server 1102 can launch agents to perform Client 1104-1106 operations or operations on the Server 1102 itself, such as monitoring and administrative tasks.

Turning now to Fig. 12, Fig. 12 shows a diagram of this Peer-To-Peer Defensive Cluster 1201 with MAC Agents 1206. In this embodiment, MAC Agents may travel from client-to-client/peer-to-peer independent of central control. Mobile MAC Agents 1206 can begin patrolling the network immune from centralized attack and can propagate to trouble spots automatically through neighboring hosts. MAC Agents 1206 can be used to form "defense clusters" to form a Defensive Barrier 1202 where neighboring computers work together to defend themselves from concerted hostile attack 1203-1205. In this form of defense, a security policy is enforced using the MAC Agents 1206 along with the extensive reactive capability built into the Client Systems 1207-1209. The MAC Agents 1206 can be used within "defense clusters" to allow groups of computer systems to rapidly defend themselves from attack or failure. MAC Agents 1206 can move from Client to Client 1207-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 146

1209, independent of a central controller. MAC Agents 1206 can run on the Client 1207-1209 or on the Server itself for monitoring or administration tasks.

Allowing the use of either centralized or distributed MAC Agents 1103, 1206 on the network allows for modifications of MAC centrally without interrupting system-wide operations. MAC Agents 1103, 1206 cannot be attacked on the host-level in the event of client compromise. MAC Agents 1103, 1206 can be customized to the network environment and only sent to clients that need their services. They can be assigned to patrol the network continuously to look for problems or be dispatched in response to certain events (security, administrative, or other events). They can be utilized to retrieve and process data independent of the client configuration on which they run. MAC Agents 1103, 1206 can move freely between clients with little communication overhead or coordination. They allow client-to-client communication and protection perimeters to be established. This allows the creation of "defense clusters" of systems that work together to protect themselves from attack.

MAC Agents 1103, 1206 can be used in a variety of ways on the network. In one embodiment, they are used to patrol the network looking for security problems. In another embodiment, they can control all aspects of network administration and control alleviating human operators from most aspects of network maintenance.

Besides the used discussed above, MAC Agents 1103, 1206, when used for security, provide an administrator several tactical advantages over an attacker. They are unpredictable and can arrive to remove or detect attackers at random without warning, and may store their code centrally and cannot be tampered with on the client system to bypass security mechanisms. They can be programmed to react to any system security event in real-time that is often significantly faster than a human response. MAC Agents 1103, 1206 can be updated with the latest security-relevant detection strategies without having to alter all clients. MAC Agents 1103, 1206, once alerted to an intruder's presence, can begin active searching for attackers and perform automated corrective measures to remove the intruder and save forensic evidence. MAC Agents 1103, 1206 in a peer-to-peer networking environment can propagate through the network absent of centralized control to thwart attack and enable fluid response to network problems. In a peer-to-peer networking environment, they can be introduced by any trusted third party within the network and be made immediately available to all clients.

Turning now to Fig. 13, Fig. 13 is a diagram of MAC Agents for security-specific applications 1300. Description of these applications are given below.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 147

Forensic Evidence Agent 1301 – This MAC agent is dispatched to a client system that has been compromised. This agent collects log files, system accounting records, auditing information and tampered files. It also collects other relevant pieces of computer forensic evidence. It protects all collected information and evidence with tamper-resistant cryptographic signatures. It centrally stores all collected evidence for later review by authorized personnel. The Forensic Evidence Agent allows an administrator or law enforcement personnel to re-build how an attack happened and to gather evidence to prosecute attackers.

Intrusion Control Agent 1302 – This MAC agent is dispatched in the event of an immediate compromise situation on a host computer system. This agent performs the following at the host system: disabling of the network interfaces to disallow unauthorized traffic except back to the central control platform; shutdown of active user accounts; locating and logging all suspicious activities, files, and processes on the system; notifying the central controller of its actions and request forensic evidence collection; moving between other affected client systems and attempting to contain the intrusion situation.

File Integrity Agent 1303 – This agent is designed to use cryptographic secure one-way hashes to detect tampering of system files. This agent can be dispatched from the central server with a list of known good file hashes to compare against on the target system. This list of known good hashes is carried with the MAC itself as part of its package. This mode of operation may allow an attacker to bypass detection if they can access the file hashes stored within the MAC code. The File Integrity Agent 1303 can be dispatched from the central server with only the ability to derive hashes from files with the known-good file hashes residing on the secure server. This mode of operation forces the MAC to communicate back the file hash results, which are then compared by the server to the known good values. This mode of operation prevents certain attacks to the MAC because the known-good hash values are not present on the client. The File Integrity Agent 1303 can be dispatched from the server to audit all system files, randomly select files, or pick from a known set of critical or commonly tampered with files. It can be dispatched from the server at random during the day to thwart attempts at intruders to time scheduled runs to bypass detection. The File Integrity Agent 1303 can look for "suspicious" files or directories on the host that indicate intrusion. Such files or directories typically include those with spaces, hidden characters and control characters in the name and those that are out of place for the target operating system (a plain text file in a Unix device directory, for example) or those with known security exploit scripts (exploit residual information).

IPR2024-00027
CrowdStrike Exhibit 1002 Page 148

Host Scanning Agent 1304 - This agent is designed to perform host vulnerability assessment and vulnerability detection from within the host looking outward ("inside out scanning"). This mode of operation is significantly different from other scanners that probe hosts from the network. This feature is accurate for detecting problems because the scanner has privileged access to the host internally and can automatically look for misconfigurations with certainty instead of having to make assumptions the way a network based scanner does. The scanner can spot local host vulnerabilities that aren't visible from the network that may allow for exploitation internally. The scanner can be updated from a central server location with optimized scanning data designed for the host-platform it is running on. This allows for faster operation. The scanner can search for known signs of intrusion that are often hidden from network scanners. The scanner does not consume network bandwidth when inspecting large number of hosts. This allows it to operate efficiently even in limited-bandwidth or remote deployment environments. The scanner can condense its findings at the host before sending to the server to conserve CPU time when generating reports and optimizing gathering of information across the network. The host scanning agent has several built in detection features which include a database of known vulnerable files for the platform it is scanning; an expert system for detecting common erroneous or corrupted configurations on the host; generic detection mechanisms for common problems that may cause vulnerabilities within the host; and the ability to detect suspicious files and activity on a host that may indicate a compromise has taken place.

Known Intrusion Agent 1305 – This agent is designed with signatures specifically to look for and alarm on signs of known intrusion. This agent is designed to randomly roam the network and detect this activity before it becomes widespread. Upon detection it notifies the central server that can then dispatch Intrusion Control Agents, Forensic Evidence Agents, or other MAC Agents to assess and respond to the incursion. The signatures that this agent rely upon are varied depending on the type of attack, but they can include: known Trojan horse detection; common backdoors; common binary alterations; suspicious directories which are known hiding places for attackers; suspicious files, for example files with known suspicious filenames indicating a break-in attempt or success; tampered system critical files; suspicious loadable kernel modules; suspicious running operating system modifications; suspicious usernames; known suspicious usernames; usernames that were not present on the system last time it was scanned; usernames that are not authorized to be present on the system regardless of last time scanned; altered or missing log files; altered or

IPR2024-00027
CrowdStrike Exhibit 1002 Page 149

missing accounting records; and other suspicious activity as specified in the accompanying database with the agent.

Loadable Kernel Module Agent 1306- This agent looks for known or unknown loadable kernel modules (LKMs) for UNIX® compatible systems. Modified LKMs are a common method used by attackers to conceal activity on compromised systems. This agent looks for the following: unauthorized LKMs loaded; known suspicious LKMs loaded; unknown suspicious LKMs loaded; intercepting system calls; employing protective anti-probe or stealth techniques; accessing or hooking normally restricted data areas in memory or on disk; and other suspicious activity.

Password Cracking Agent 1307 - This agent attempts to break user passwords using a list of common, uncommon or customized words. This mechanism helps spot weak passwords before intruders can abuse them. This agent can travel from host to host attempting to break and report on bad passwords to the central controller. It can also travel from host to host attempting to break passwords given to it by the central controller to distribute cracking efforts over all the computers on the network.

Log Archive Agent 1308 - This agent is responsible for archiving, cryptographically signing, and saving all relevant system logs, accounting records, and other audit data from the target host on which it is executing. This information is pulled back to the main server for archival purposes. This agent exists to help track down security or maintenance issues across the enterprise. This agent ensures that critical audit data is stored in a centrally managed repository that is not susceptible to tampering the way it would be if it were only present on the target host. Audit data collected by this agent can be historically categorized on the back-end server where data-mining and other advanced data correlation techniques can be used to detect and monitor trends for security or other problems.

"Rootkit" Agent 1309 – This is a complement to the Known Intrusion Agent and is designed to specifically look for all variants of "rootkits" that may be on the target host. "Rootkits" are used by intruders to automatically conceal activity, install backdoors, and create Trojan horse binaries to fully compromise a system. This agent is capable of detecting all major categories and derivatives of "rootkits" in existence.

Suspicious File Agent 1310 - This agent is designed to look for known or user-specified banned files on client systems. Upon detection of these file this agent can erase, archive, send an alarm, or perform other action as specified. Examples of suspicious files include pirated MP3 music files; pirated software; pornography or other inappropriate data; confidential corporate data outside of controlled spaces; user-specified files or data; and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 150

core files from prominent running system daemons that could be the result of a buffer overflow attack

Promiscuous Mode Agent 1311 – This agent detects network cards in promiscuous modes on remote clients. This is a random audit capability.

Hidden Process Detection Agent 1312 - This agent can detect hidden processes on the remote host.

Unauthorized Network Daemon Agent 1313– Looks for network daemons being run by unauthorized users.

Self-Test Agent 1314– Agents that can simulate attacks on the host to test the IDS functions independently.

Spy Agent 1315– This agent follows around users to observe their activity if it is suspicious or if they have been tagged for surveillance.

Zombie Shells Agent 1316– This agent will attempt to detect zombie shells left over from buffer overflow attacks.

Insider Attack Agent 1317 – This agent will be dispatched to the attacking host if both the host being attacked and the attacker are under the system control. This agent is used to stop the insider attack at the host in case it is being used as a hop point to move around inside.

Turning now to Fig. 14, Fig. 14 is a diagram of MAC Agents for network management-specific applications 1400. Descriptions of these application are given below.

Backup Agent 1401 – This agent is designed to perform a full or partial backup of the client system depending on its configuration. Applications for this include: automated scheduled backup of systems to a central storage facility and automated response backup upon detection of a system fault to preserve critical data (such as a bad hard drive being detected)

Host inventory agent 1402 - This agent performs a detailed hardware inventory of the target system noting such data as: system type; CPU Type; memory capacity; operating system installed; software versions installed; storage capacity; serial numbers; and user-specified. This data is retrieved and stored at the central server repository for cataloging systems on the network for use by other agents, data-mining needs, central reporting, tracking of installed patches, or other purposes.

System monitor/status agent 1403- This agent reports on the system status of the client on which it is executing. This data may include amount of free storage available;

IPR2024-00027
CrowdStrike Exhibit 1002 Page 151

number of active users; network activity; CPU activity; and other relevant system data as defined by the user.

System task agent 1404 – This agent can perform a variety of system maintenance related tasks as configured by the administrator. Such tasks may include: removing files from temp directories; freeing up drive space by removing unused files and directories; and running any other user-specified action on the target systems

PatchWatch™ agent 1405 – This agent is designed to detect what operating system a remote host is running. Once this data is collected it is deposited into the central repository. The central server then will utilize a subscription-based monitoring service that reports on a daily basis what new security patches have become available from all known vendors. The PatchWatch service then downloads the patch and uses the PatchWatch agents to distribute the security or administrative patch information to affected hosts automatically. This service allows administrators to keep systems up-to-date with minimal downtime and involvement. PatchWatch inventories host systems automatically. The central PatchWatch server monitors vendor sites for new product updates, patches, or hot fixes. The central PatchWatch server downloads the newest updates. The updates are categorized and pushed down to affected clients automatically to repair the problem.

Using the foregoing, the invention may be implemented using standard programming or engineering techniques including computer programming software, firmware, hardware or any combination or subset thereof. Any such resulting program, having a computer readable program code means, may be embodied or provided within one or more computer readable or usable media, thereby making a computer program product, i. e. an article of manufacture, according to the invention. The computer readable media may be, for instance a fixed (hard) drive, disk, diskette, optical disk, magnetic tape, semiconductor memory such as read-only memory (ROM), or any transmitting/receiving medium such as the Internet or other communication network or link. The article of manufacture containing the computer programming code may be made and/or used by executing the code directly from one medium, by copying the code from one medium to another medium, or by transmitting the code over a network.

An apparatus for making, using or selling the invention may be one or more processing systems including, but not limited to, a central processing unit (CPU), memory, storage devices, communication links, communication devices, server, I/O devices, or any sub-components or individual parts of one or more processing systems, including software,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 152

firmware, hardware or any combination or subset thereof, which embody the invention as set forth in the claims.

User input may be received from the keyboard, mouse, pen, voice, touch screen, or any other means by which a human can input data to a computer, including through other programs such as application programs.

Although the present invention has been described in detail with reference to certain preferred embodiments, it should be apparent that modifications and adaptations to those embodiments may occur to persons skilled in the art without departing from the spirit and scope of the present invention as set forth in the following claims.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 153

What is claimed is:

1. A computer implemented method for providing system security and resource management, comprising:

managing event messages by a master control processor between system handlers according to security system policies;

processing network messages by a network handler between client and server computers;

inserting native and third party event messages received by an insertion handler into the master control processor for processing by other system handlers;

detecting and processing event message signatures by the signature handler from alarm, system, and insertion events for conversion into system alarm messages for action by the other system handlers; and

performing actions by an action handler in response to action requests from the master control processor.

2. The method according to claim 1, further comprising maintaining an execution environment by an agent handler for mobile autonomous code modules.

3. The method according to claim 1, further comprising collecting and logging event messages by a logging handler.

4. The method according to claim 1, further comprising managing system configuration parameters by a configuration handler.

5. The method according to claim 1, wherein the step of managing event messages comprises:

registering the system handlers;

passing event messages between system handlers; and

managing a event queues attached to the system handlers.

6. The method according to claim 5, wherein the step of registering each system handler comprises:

reading the handler module to determine initialization requirement;

IPR2024-00027
CrowdStrike Exhibit 1002 Page 154

initializing the handler application programming interface;
determining if the handler is to run as a remote procedure call;
making the handler available through the remote procedure call interface if run as a remote procedure call; and
initializing the handler input/output queues if not run as a remote procedure call.

7. The method according to claim 1, wherein the system handlers comprise static and dynamic system processes.

8. The method according to claim 1, further comprising;
initiating the system handlers by internal mechanisms;
initiating the system handlers by external mechanisms; and
initiating the system handlers from the master control processor.

9. The method according to claim 1, wherein the system handlers have a reversible architecture to enable the system handlers to be used in either a client or server computer mode.

10. The method according to claim 1, wherein the step of processing network messages by the network handler comprises:
allowing connection only from clients and servers defined in a access control list;
authenticating protocols with clients and servers;
compressing data to minimize bandwidth requirements; and
encrypting data to provide secure communication.

11. The method according to claim 1, wherein the step of inserting native and third party event messages received by the insertion handler from external programs comprises:
reading and writing messages using an insertion method selected from the group consisting of file descriptor, network sockets and named pipe;
using a native insertion mode library to directly insert messages into the master control processor; and
using an external insertion mode library linked to a third party sources to directly insert messages into the master control processor.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 155

12. The method according to claim 1, wherein the step of detecting and processing event messages received by the signature handler comprises:

accepting alarm events from the master control processor;

decoding the alarm type and originator from the alarm event;

consulting internal signature registry for alarms of the type accepted;

handing the alarm message off to the signature module for processing;

extracting alarm data macro information;

determining if an alarm has occurred;

consulting the action policy if an alarm has occurred to determine response; and

passing the resulting response message to the master control processor for action by the other system handlers.

13. The method according to claim 1, wherein the action performed by the action handler is selected from the group consisting of blocking a host with a modified route command, blocking a host with a packet filter modification command, disabling a user account, disabling a network interface, running an external user-defined command, logging an event, sending email or pager alerts, sending on-screen alerts to users or administrators, requesting server executed action, and a pluggable action defined by a user.

14. The method according to claim 1, wherein the steps of detecting and processing event messages received by the signature handler and performing actions by the action handler comprises the steps of:

receiving an event message containing a signature by the signature handler from the master control processor;

detecting an attack by the signature handler from the event message signature;

consulting action policy by the signature handler;

determining action to be taken by the signature handler;

ending the process if no action is required;

determining required action parameters by the signature handler if action is required;

sending an action request to the action handler by the signature handler via the master control processor;

processing and executing the action request by the action handler; and

returning status of the executed action to the signature handler.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 156

15. The method according to claim 2, wherein the step of maintaining an execution environment by the agent handler for mobile autonomous code modules comprises:

ensuring that the mobile autonomous code modules carry appropriate credentials, are authenticated and cryptographically signed by a trusted introducer, and able to execute on the host operating system;

distributing mobile autonomous code modules to one or more client computers;

executing the mobile autonomous code modules without interference;

allowing the mobile autonomous code modules to collect and report its results; and

shutting down the mobile autonomous code modules.

16. The method according to claim 15, wherein the step of executing the mobile autonomous code modules comprises:

verifying detected alarms;

reducing false alarm rates; and

providing immediate response.

17. The method according to claim 15, wherein the step of executing the mobile autonomous code modules comprises the steps of actively looking for problems and identifying attackers when problems are detected.

18. The method according to claim 15, wherein the step of executing the mobile autonomous code modules comprises performing security and system administration tasks in self-healing network environments.

19. The method according to claim 15, wherein the step of executing the mobile autonomous code modules comprises allowing self-healing components of the system to move between clients and operate independently where required.

20. The method according to claim 2, wherein the step of maintaining an execution environment by the agent handler for mobile autonomous code modules comprises:

enabling self-healing and adaptive networks;

facilitating distribution of updates for the mobile autonomous code modules; and

centralizing command and control functions for increased reliability.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 157

21. The method according to claim 2, wherein the step of maintaining an execution environment by the agent handler further comprises forming a peer-to-peer defensive cluster with mobile autonomous code modules.

22. The method according to claim 2, wherein the step of maintaining an execution environment by the agent handler further comprises protecting the mobile autonomous code modules from alteration or tampering by hostile adversaries, and dispatching the mobile autonomous code modules through a predictable schedule from a central control point.

23. The method of claim 22, wherein the mobile autonomous code modules are dispatched through a random schedule.

24. The method according to claim 2, wherein the step of maintaining an execution environment by the agent handler further comprises:

programming the mobile autonomous code modules to detect and remove attackers at random;

storing code for the mobile autonomous code modules at a central location;

preventing alteration of the mobile autonomous code modules on client computers;

updating the mobile autonomous code modules with updated security detection strategies without modifying client computers;

beginning an active search for attackers when alerted to an intruder's presence;

performing automated corrective measures to remove the intruder; and

saving forensic evidence.

25. The method according to claim 2, wherein security-specific mobile autonomous code modules are selected from the group consisting of forensic evidence agent, intrusion control agent, file integrity agent, host scanning agent, known intrusion agent, loadable kernel module agent, password cracking agent, log archive agent, rootkit agent, suspicious file agent, promiscuous mode agent, hidden process detection agent, unauthorized network daemon agent, self-test agent, spy agent, zombie shells agent, and insider attack agent.

26. The method according to claim 25, wherein forensic evidence gathered by the forensic evidence agent from protected systems is cryptographically signed to prevent tampering.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 158

27. The method according to claim 2, wherein network management-specific mobile autonomous code modules are selected from the group consisting of backup agent, host inventory agent, system monitor and status agent, system task agent, and PatchWatch™ agent.

28. The method according to claim 3, wherein the method of logging event messages is selected from the group consisting of text-based files, a local system auditing facility, a cryptographically signed secure log format, directly to a system console, e-mail notification, direct used interface, and system wide notification.

29. The method according to claim 4, wherein the step of managing system configuration parameters comprises;

interfacing generic calls to other system handlers;
reading system configuration parameters;
writing system configuration parameters;
changing system configuration parameters;
reverting system configuration parameters back to a previous version;
deleting system configuration parameters;
backing up system configuration parameters; and
providing multiple access protection mechanisms.

30. A system for providing system security and resource management, comprising:

a master control processor for managing event messages between system handlers according to security system policies;
a network handler for processing network messages between client and server computers;
an insertion handler for inserting native and third party event messages into the master control processor for processing by other system handlers;
a signature handler for detecting and processing event messages from alarm, system, and insertion events for conversion into system alarm messages for action by other system handlers; and
an action handler for performing actions in response to action requests from the master control processor.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 159

31. The system according to claim 30, further comprising:

an agent handler for maintaining an execution environment for mobile autonomous code modules;

a logging handler for collecting and logging event messages; and

a configuration handler for managing system configuration parameters.

32. The system according to claim 31, wherein the system handlers auto-register themselves and their capabilities with the master control processor.

33. The system according to claim 30, wherein the action handler utilizes pluggable action modules.

34. The system according to claim 30, wherein the signature handler utilizes pluggable signature modules.

35. The system according to claim 34, wherein the pluggable signature modules auto-register with the signature handler.

36. The system according to claim 34, wherein a pluggable signature modules uses stored data from other signature modules, stores data between execution, stores data between system startup and shutdown sequences, and only processes signatures relating to the signature module.

37. The system according to claim 34, wherein the pluggable signature modules are added and removed from the system without modifying the core system code.

38. The system according to claim 31, wherein the system is installed on at least one server computer and at least one client computer.

39. The system according to claim 38, wherein the system installed on at least one client operates independently of the system installed on at least one server for reduced processing and reaction time and when network communications are disrupted.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 160

40. The system according to claim 38, further comprising at least one graphical user interface.

41. The system according to claim 40, further comprising multiple levels of independent alarm filters on the client and server computers for reduced false alarm reporting, configuration flexibility, and system granularity.

42. The system according to claim 40, further comprising encrypted and mutually authenticated communication links between the server computer, client computer and graphical user interface.

43. The system according to claim 40, further comprising a means for re-allocating system resources to circumvent system problems or failures.

44. The system according to claim 31, wherein the system is installed on a client computer to maintain host processes, collect and forward events, process local client signatures, generate events, initiate and respond to action requests, initiate self-healing counter-measures, and host mobile autonomous code agents.

45. The system according to claim 31, wherein the system is installed on a server computer to collect and store events, process enterprise client signatures, generate events, operate a central system database, schedule events, initiate and respond to action requests, initiate self-healing counter-measures, maintain a graphical user interface backend structure, and manage mobile autonomous code agents.

46. The system according to claim 40, wherein the client computer has a capability to perform server computer functions in the event of failure of a server computer.

47. The system according to claim 40, wherein the server computer has a capability to perform client computer functions in the event of failure of a client computer.

48. Computer executable software code stored on a computer readable medium, the code for a computer implemented method for providing system security and resource management, comprising:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 161

code for managing event messages by a master control processor between system handlers according to security system policies;

code for processing network messages by a network handler between client and server computers;

code for inserting native and third party event messages received by an insertion handler into the master control processor for processing by other system handlers;

code for detecting and processing event message signatures by the signature handler from alarm, system, and insertion events for conversion into system alarm messages for action by the other system handlers; and

code for performing actions by an action handler in response to action requests from the master control processor.

49. The computer executable software code method of claim 48, further comprising:

code for maintaining an execution environment by an agent handler for mobile autonomous code modules;

code for collecting and logging event messages by a logging handler; and

code for managing system configuration parameters by a configuration handler.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 162

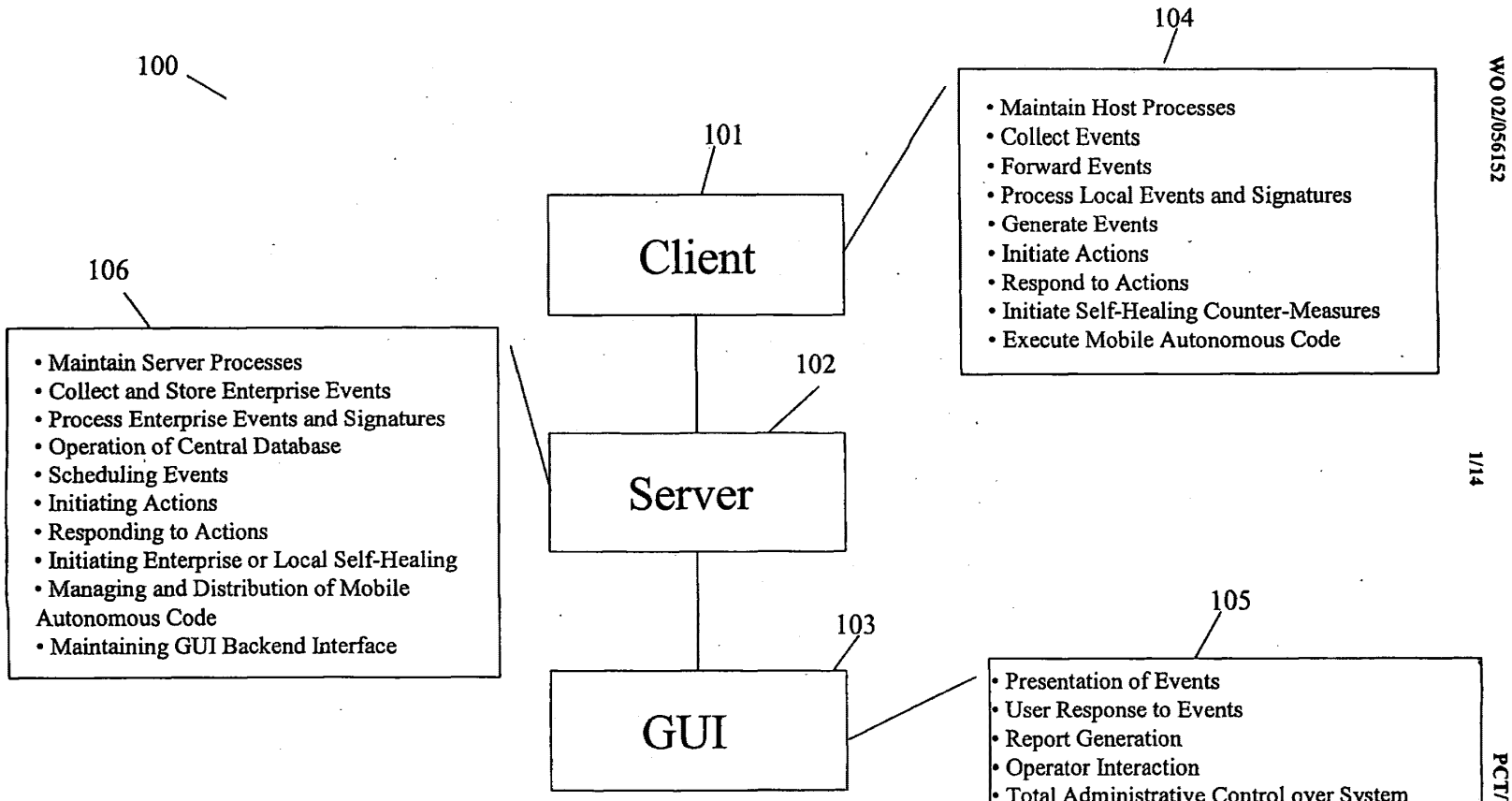

100
101
Client
104
• Maintain Host Processes
• Collect Events
• Forward Events
• Process Local Events and Signatures
• Generate Events
• Initiate Actions
• Respond to Actions
• Initiate Self-Healing Counter-Measures
• Execute Mobile Autonomous Code
106
• Maintain Server Processes
• Collect and Store Enterprise Events
• Process Enterprise Events and Signatures
• Operation of Central Database
• Scheduling Events
• Initiating Actions
• Responding to Actions
• Initiating Enterprise or Local Self-Healing
• Managing and Distribution of Mobile Autonomous Code
• Maintaining GUI Backend Interface
102
Server
103
GUI
105
• Presentation of Events
• User Response to Events
• Report Generation
• Operator Interaction
• Total Administrative Control over System

Fig. 1

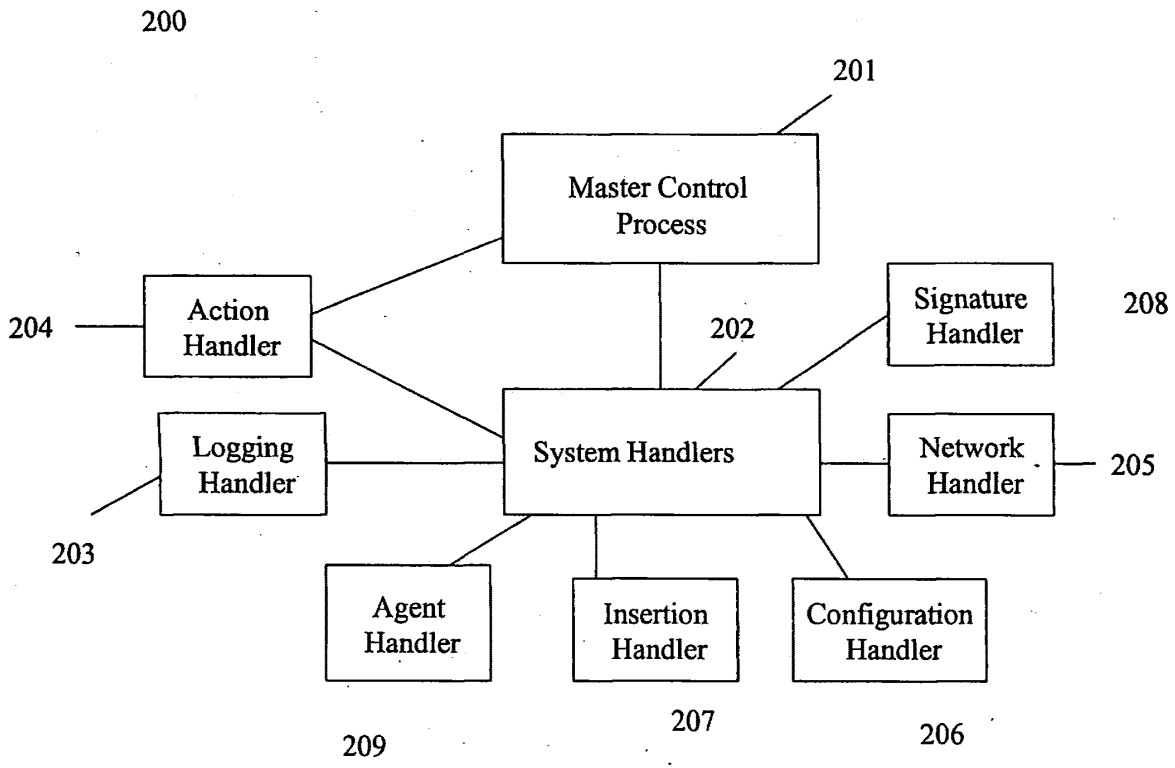

200
201
Master Control Process
204
Action Handler
202
Signature Handler
208
Logging Handler
System Handlers
Network Handler
205
203
Agent Handler
Insertion Handler
Configuration Handler
209
207
206

Fig. 2

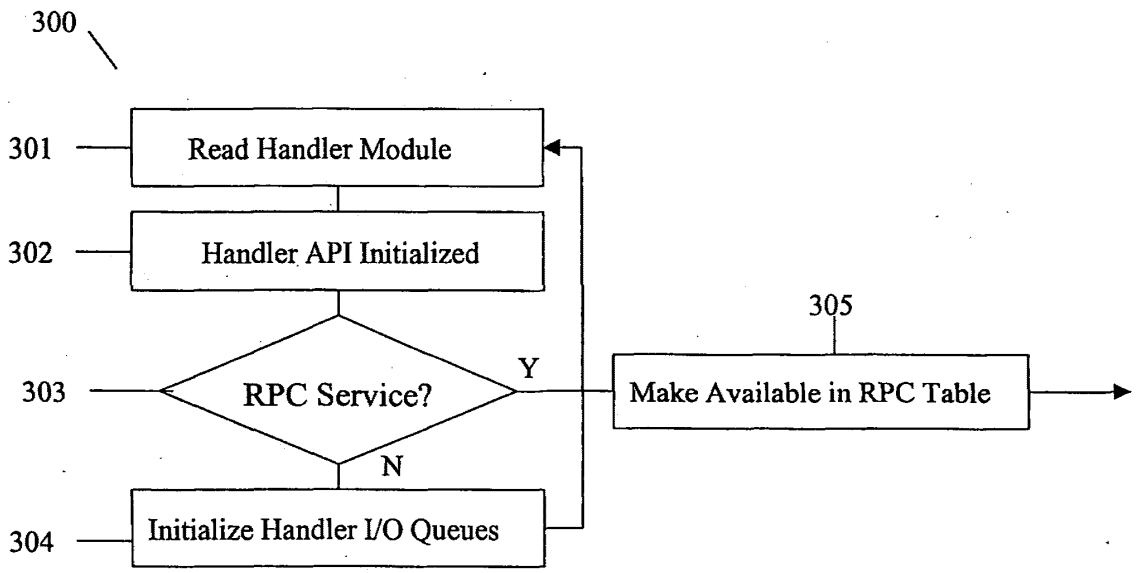

300
301
Read Handler Module
302
Handler API Initialized
303
RPC Service?
Y
N
305
Make Available in RPC Table
304
Initialize Handler I/O Queues

Fig. 3

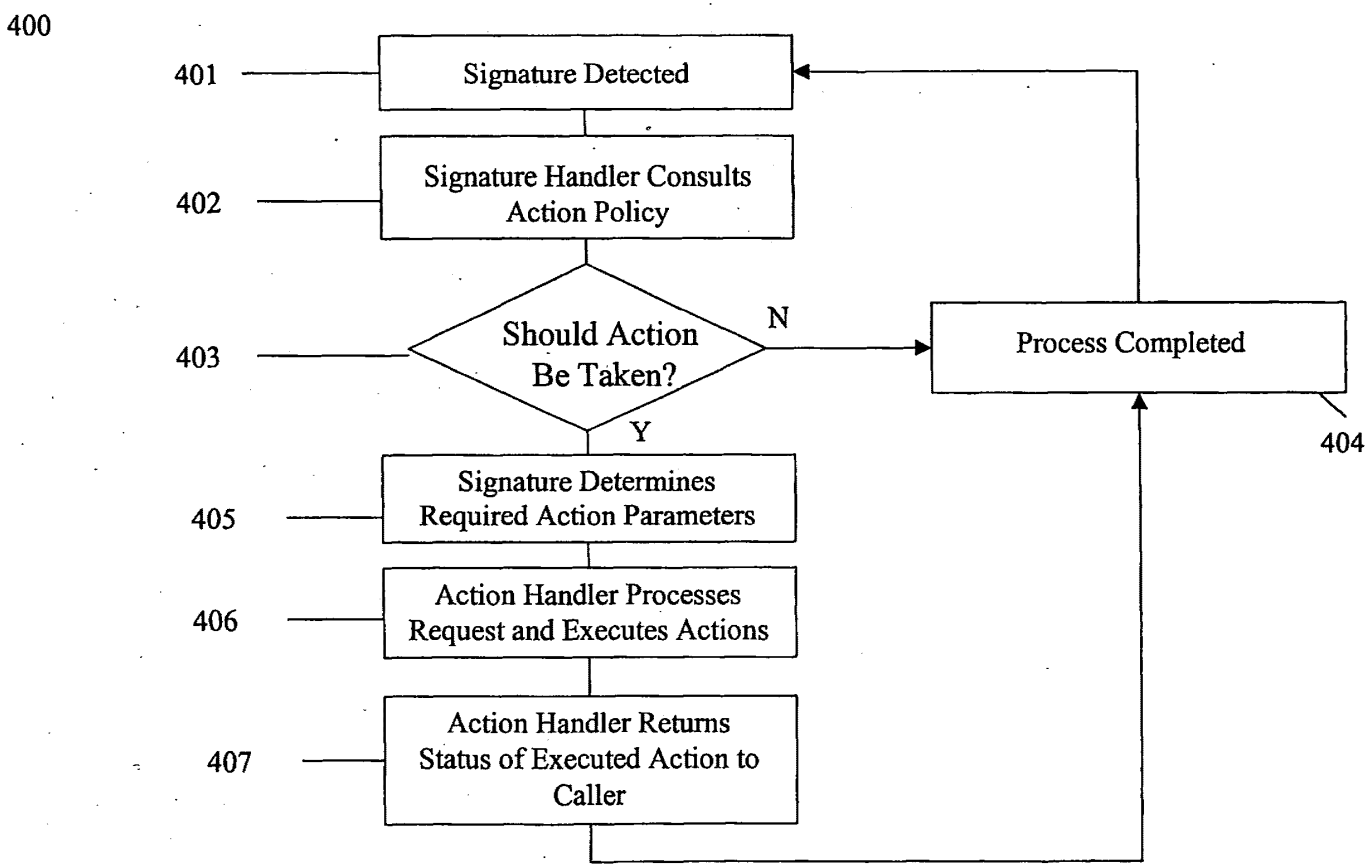

400
401
Signature Detected
402
Signature Handler Consults Action Policy
403
Should Action Be Taken?
N
Y
Process Completed
404
405
Signature Determines Required Action Parameters
406
Action Handler Processes Request and Executes Actions
407
Action Handler Returns Status of Executed Action to Caller

Fig. 4

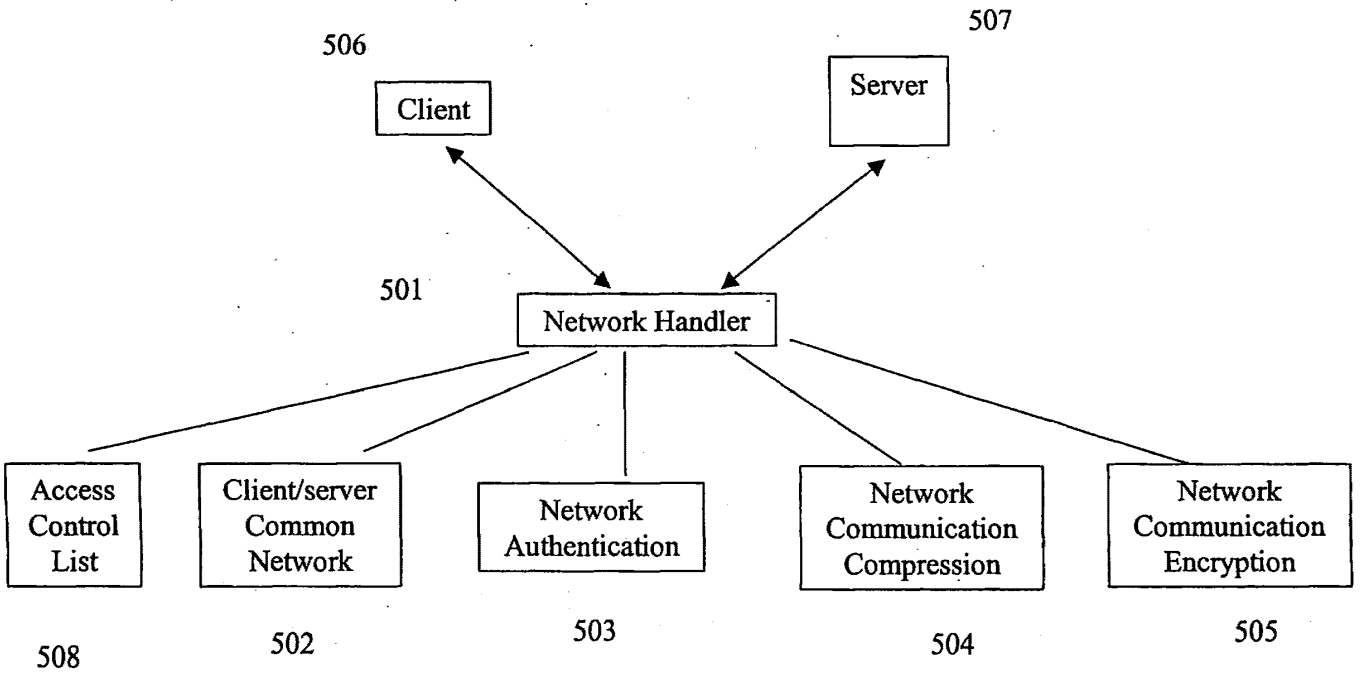

506
Client
507
Server
501
Network Handler
Access Control List
Client/server Common Network
Network Authentication
Network Communication Compression
Network Communication Encryption
508
502
503
504
505

Fig. 5

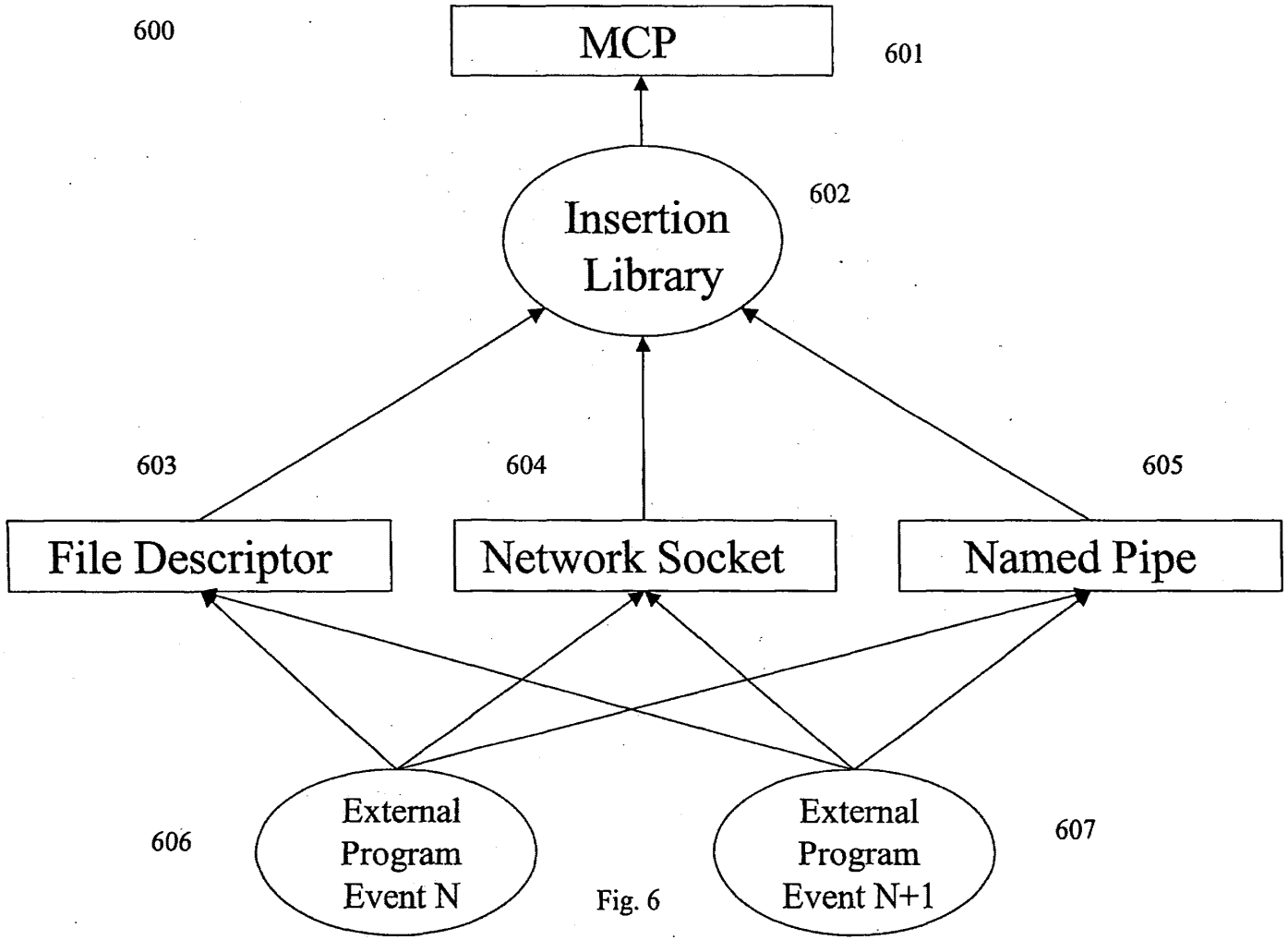

600
MCP
601
Insertion Library
602
603
604
605
File Descriptor
Network Socket
Named Pipe
606
External Program Event N
External Program Event N+1
607

Fig. 6

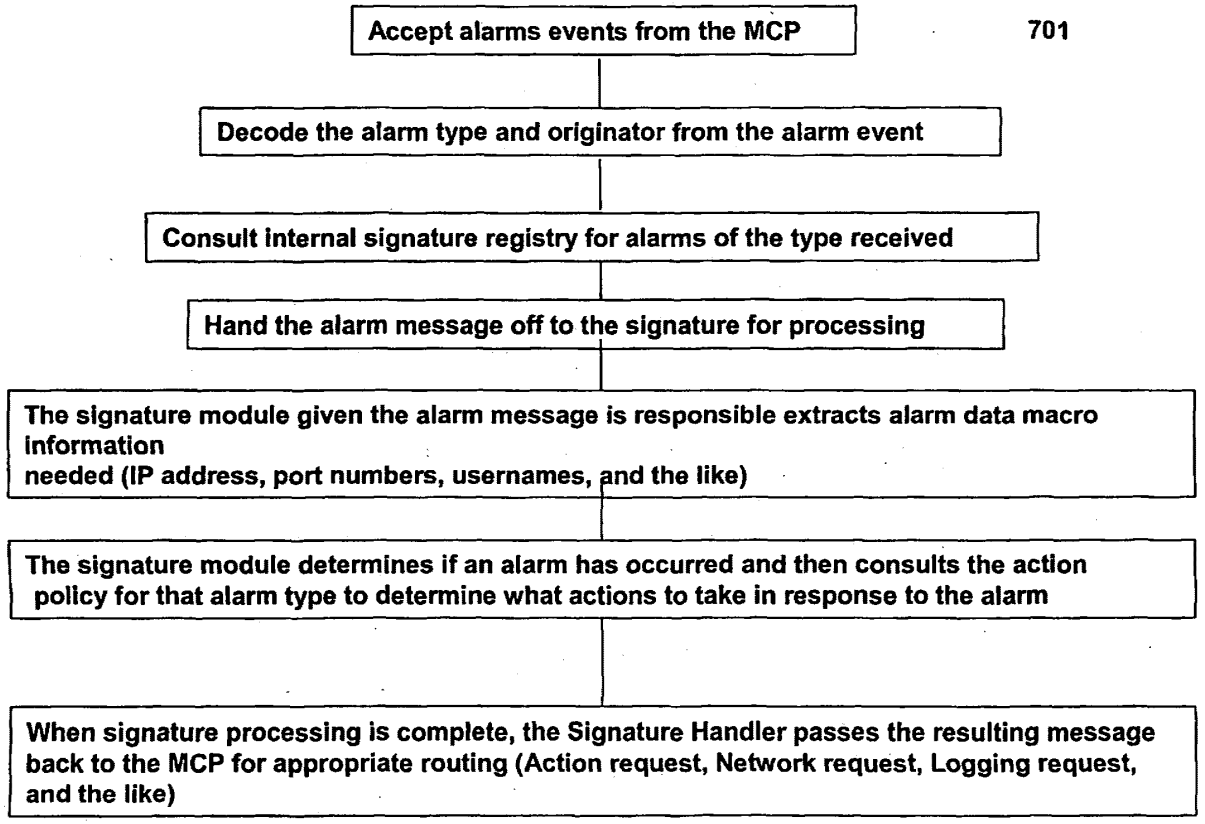


Fig. 7

IPR2024-00027
CrowdStrike Exhibit 1002 Page 169

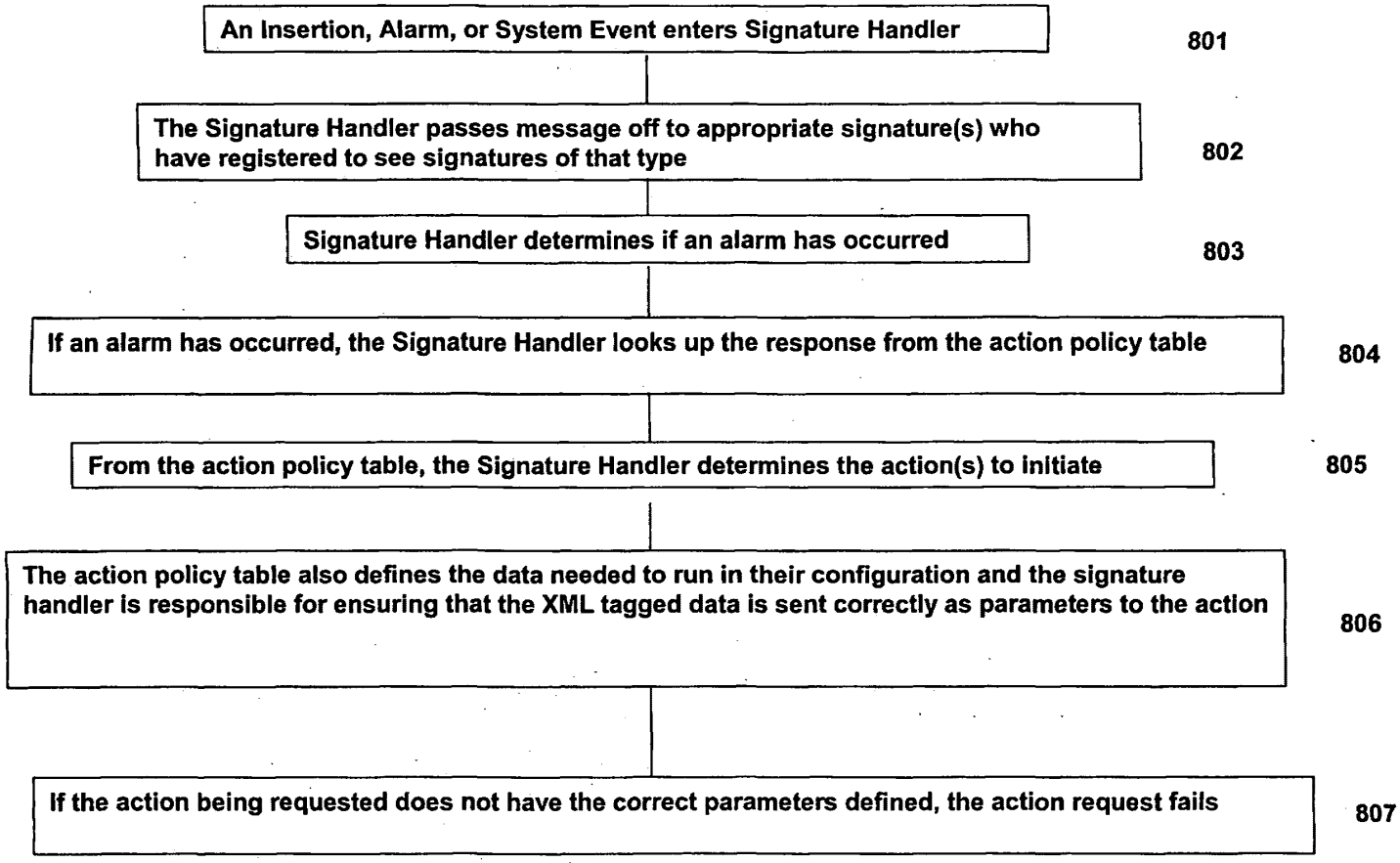


Fig. 8

IPR2024-00027
CrowdStrike Exhibit 1002 Page 170

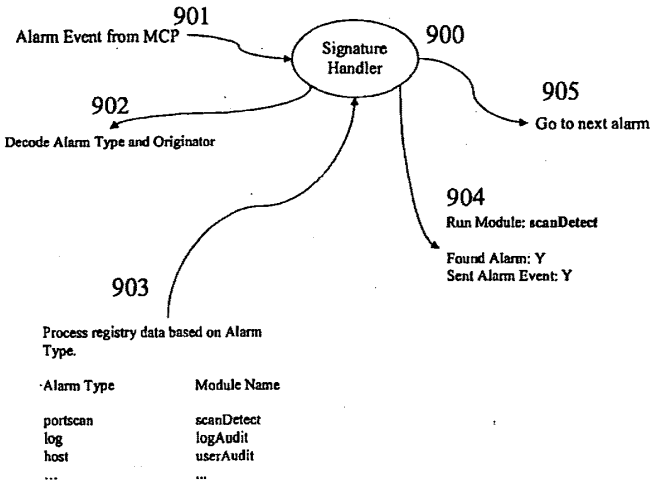

901
Alarm Event from MCP
Signature Handler
900
905
Go to next alarm
902
Decode Alarm Type and Originator
904
Run Module: scanDetect
Found Alarm: Y
Sent Alarm Event: Y
903
Process registry data based on Alarm Type.
Alarm Type
Module Name
portscan
scanDetect
log
logAudit
host
userAudit
...
...

Fig. 9

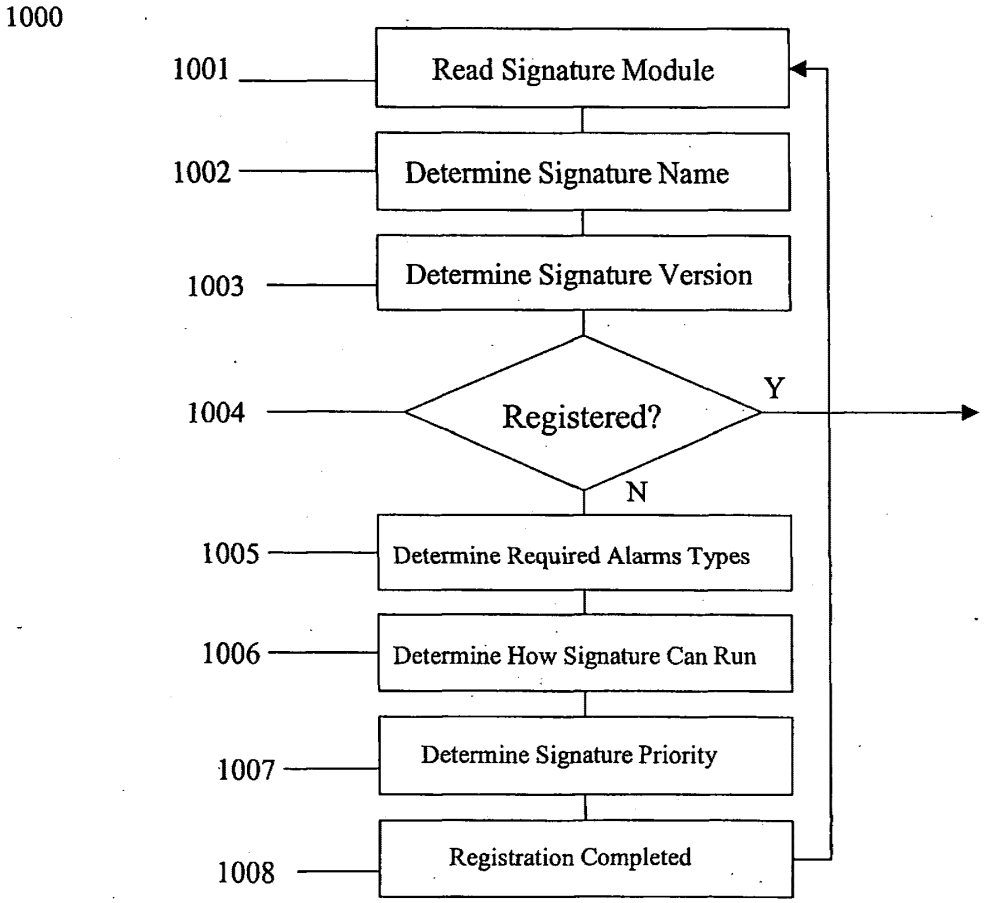

1000
1001
Read Signature Module
1002
Determine Signature Name
1003
Determine Signature Version
1004
Registered?
Y
N
1005
Determine Required Alarms Types
1006
Determine How Signature Can Run
1007
Determine Signature Priority
1008
Registration Completed

Fig. 10

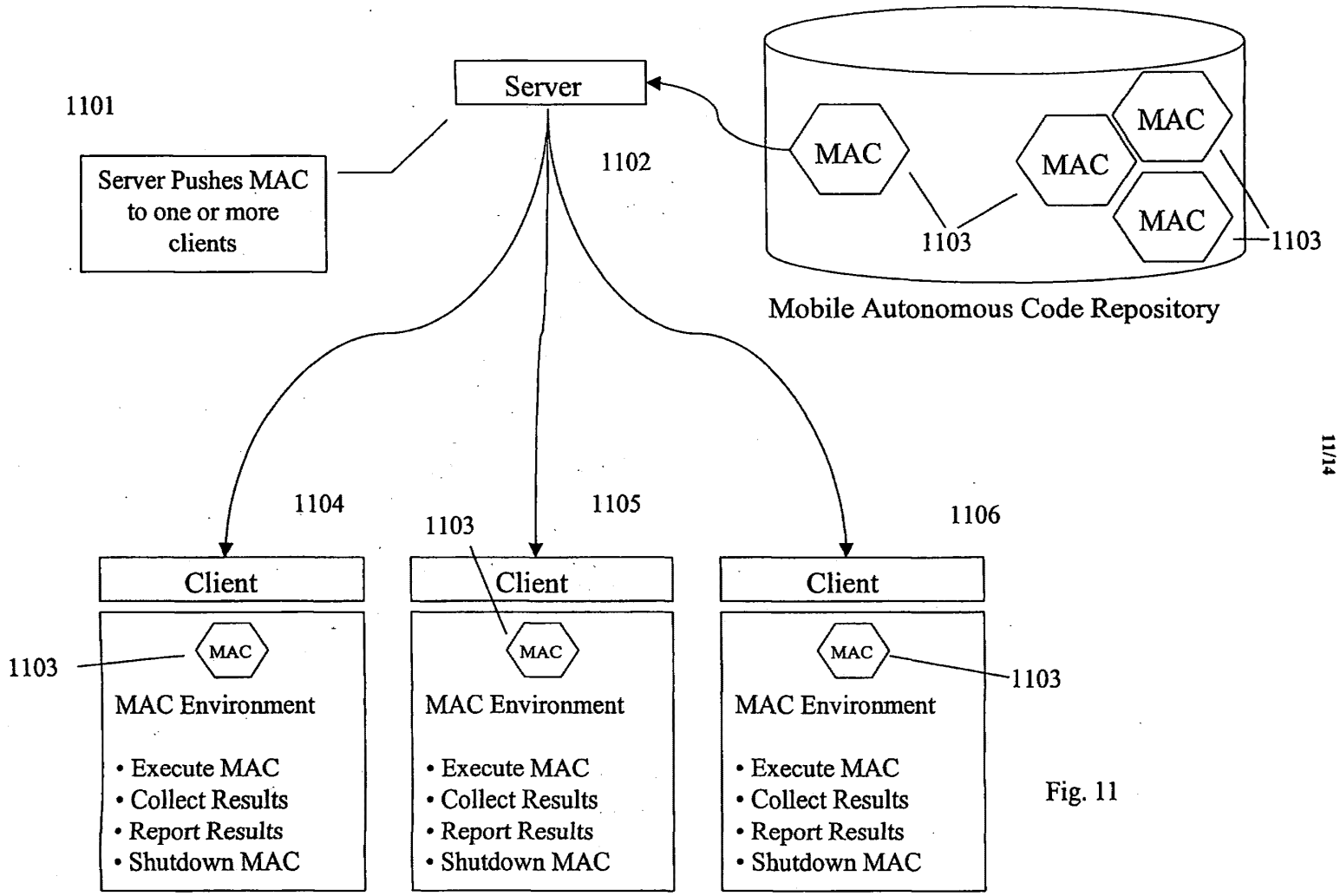
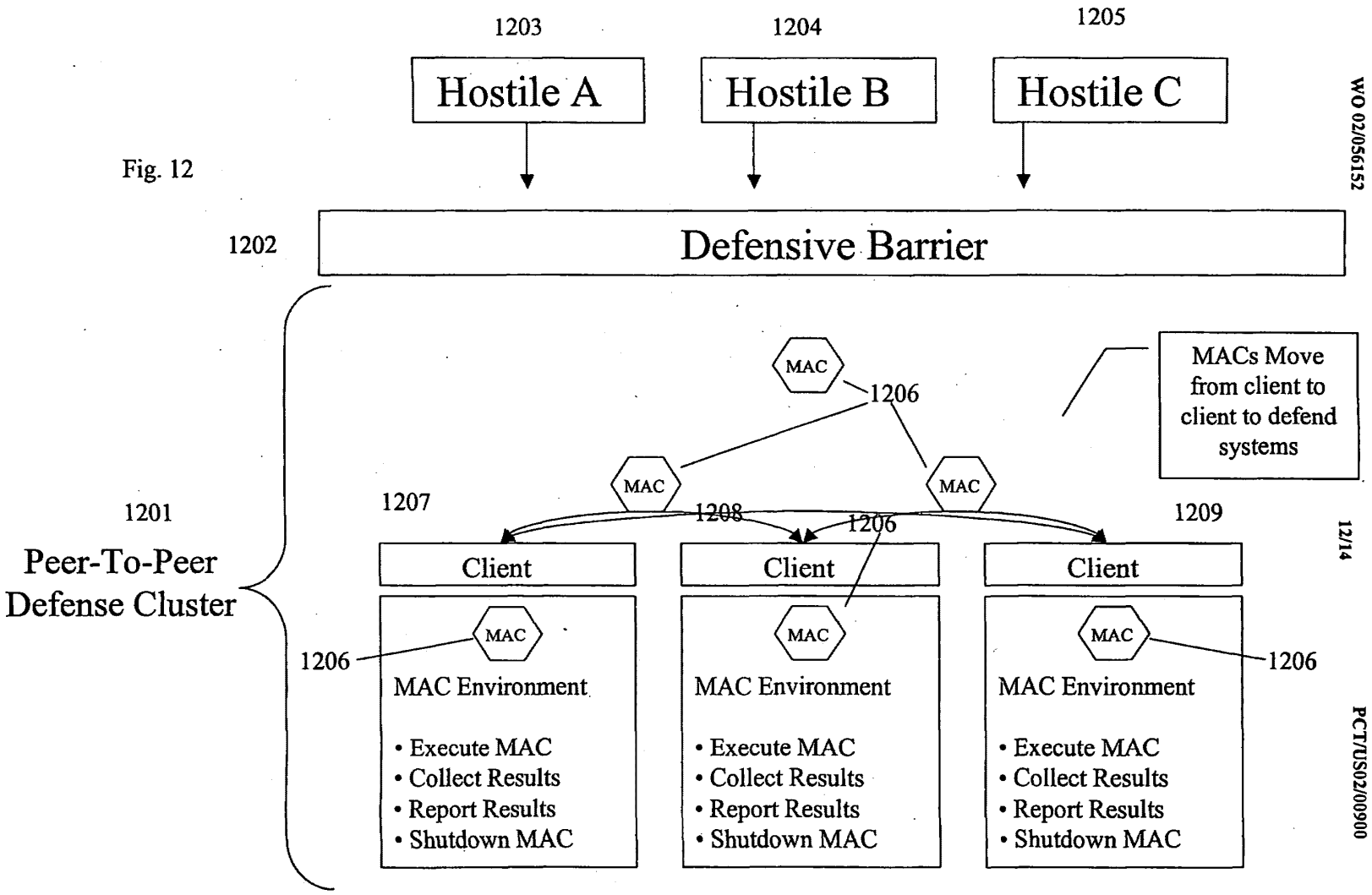


Fig. 11

IPR2024-00027
CrowdStrike Exhibit 1002 Page 173

Fig. 12


1203
1204
1205
Hostile A
Hostile B
Hostile C
1202
Defensive Barrier
MAC
1206
MACs Move from client to client to defend systems
MAC
MAC
1207
1208
1206
1209
1201
Peer-To-Peer Defense Cluster
Client
Client
Client
MAC
MAC
MAC
1206
1206
MAC Environment
MAC Environment
MAC Environment
• Execute MAC
• Collect Results
• Report Results
• Shutdown MAC
• Execute MAC
• Collect Results
• Report Results
• Shutdown MAC
• Execute MAC
• Collect Results
• Report Results
• Shutdown MAC

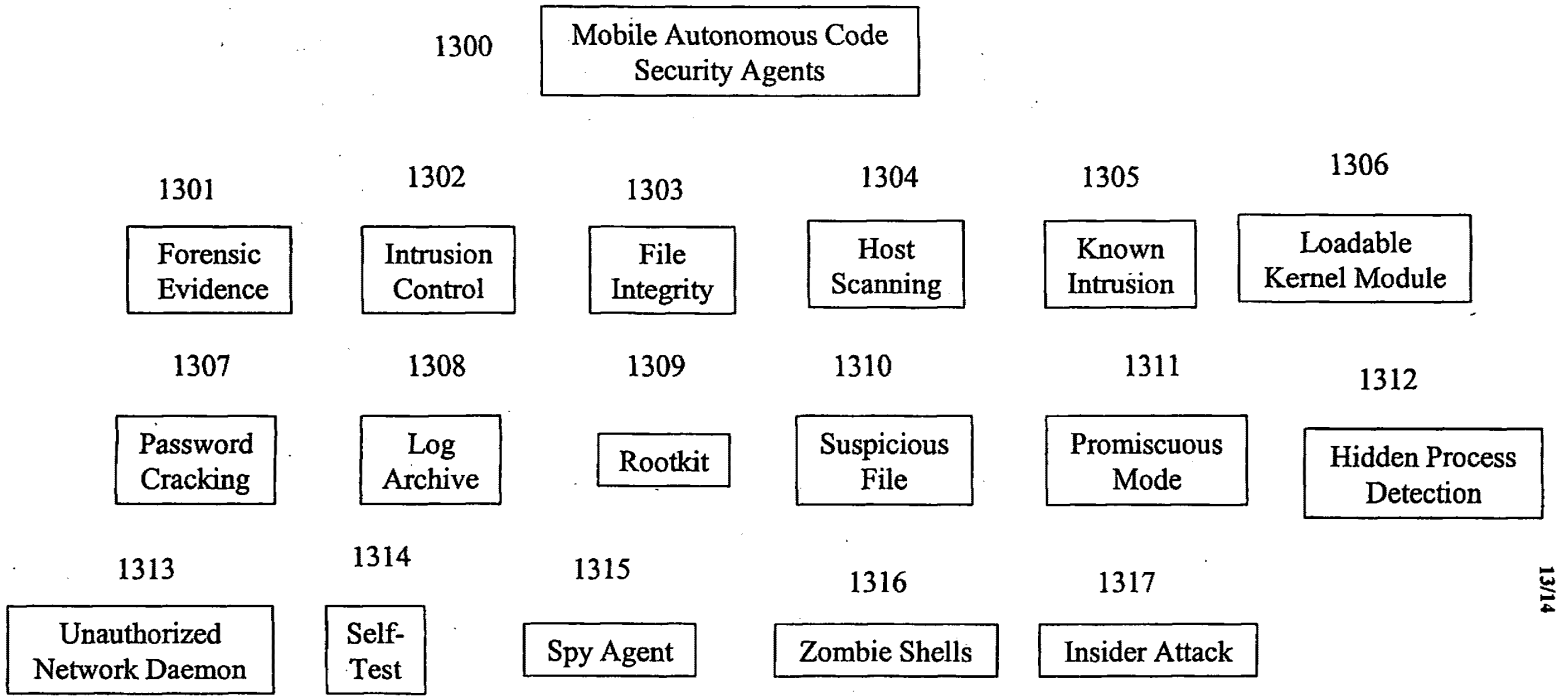

1300
Mobile Autonomous Code Security Agents
1301
Forensic Evidence
1302
Intrusion Control
1303
File Integrity
1304
Host Scanning
1305
Known Intrusion
1306
Loadable Kernel Module
1307
Password Cracking
1308
Log Archive
1309
Rootkit
1310
Suspicious File
1311
Promiscuous Mode
1312
Hidden Process Detection
1313
Unauthorized Network Daemon
1314
Self-Test
1315
Spy Agent
1316
Zombie Shells
1317
Insider Attack

Fig. 13

1400
Network Management
Mobile Autonomous Agents
1401
1402
1403
1404
1405
Backup
Host
Inventory
System
Monitor/status
System
Task
PatchWatch

Fig. 14

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 November 2001 (08.11.2001)

PCT

(10) International Publication Number
WO 01/84285 A3

(51) International Patent Classification[7]: H04L 29/06, 12/24

(21) International Application Number: PCT/US01/13799

(22) International Filing Date: 27 April 2001 (27.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/200,316 28 April 2000 (28.04.2000) US

(71) Applicant: INTERNET SECURITY SYSTEMS, INC. [US/US]; 6303 Barfield Road, Atlanta, GA 30328 (US).

(72) Inventors: FARLEY, Timothy, P.; 128 Old Holcomb Bridge Way, Roswell, GA 30076 (US). HAMMER, John, M.; 5584 Wilmer Drive, Norcross, GA 30092 (US). WILLIAMS, Bryan, Douglas; 430 Thorntree Pass, Lawrenceville, GA 30043 (US). BRASS, Philip, Charles; 1140 Pine Grove Pointe Drive, Roswell, GA 30075 (US). YOUNG, George, C.; 3355 Commons Gate Bend, Norcross, GA 30092 (US). MEZACK, Derek, John; 3615 Blackwell Run, Marietta, GA 30066 (US).

(74) Agent: WIGMORE, Steven, P.; King & Spalding, 191 Peachtree Street, Atlanta, GA 30303-1763 (US).

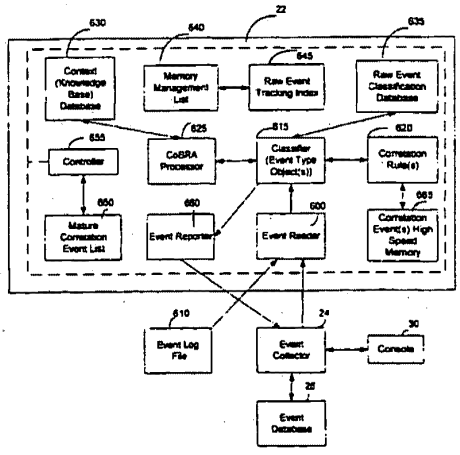
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD,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

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR MANAGING COMPUTER SECURITY INFORMATION



(57) Abstract: A security management system includes a fusion engine which "fuses" or assembles information from multiple data sources and analyzes this information in order to detect relationships between raw events that may indicate malicious behavior and to provide an organized presentation of information to consoles without slowing down the processing performed by the data sources. The multiple data sources can comprise sensors or detectors that monitor network traffic or individual computers or both. The sensors can comprise devices that may be used in intrusion detection systems (IDS). The data sources can also comprise firewalls, audit systems, and other like security or IDS devices that monitor data traffic in real-time. The present invention can identify relationships between one or more real-time, raw computer events as they are received in real-time. The fusion engine can also assess and rank the risk of real-time raw events as well as mature correlation events.

WO 01/84285 A3

IPR2024-00027
CrowdStrike Exhibit 1002 Page 177

(88) Date of publication of the international search report:
13 June 2002

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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 178

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/13799

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06 H04L12/24

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)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 985 995 A (IBM) 15 March 2000 (2000-03-15) column 4, line 2 -column 5, line 50	1-4,14, 18,22
X	JAGANNATHAN R ET AL: "SYSTEM DESIGN DOCUMENT: NEXT-GENERATION INTRUSION DETECTION EXPERT SYSTEM (NIDES)" INTERNET CITATION, 9 March 1993 (1993-03-09), XP002136082 page 1 -page 8	1-4,14, 18,22

☐ 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
12 April 2002	24/04/2002

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	Veen, G

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

In tional Application No
PCT/US 01/13799

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0985995 A	15-03-2000	EP 0985995 A1	15-03-2000

IPR2024-00027
CrowdStrike Exhibit 1002 Page 180

METHOD AND SYSTEM FOR MANAGING COMPUTER SECURITY INFORMATION

PRIORITY AND RELATED APPLICATIONS

The present application claims priority to provisional patent application entitled, "Intrusion Detection Fusion System of a Network Security System," filed on April 28, 2000 and assigned U.S. Application Serial Number 60/200,316. The present application is also related to non-provisional application entitled, "System and Method for Managing Security Events on a Network," (Attorney Docket No. 05456-105005) filed on April 27, 2001 and assigned U.S. Application Serial Number _____.

TECHNICAL FIELD

The present invention relates to computer systems and the security of such systems. More particularly, the present invention relates to a method and system for ranking individual security events according to risk and fusing or identifying relationships between two or more security events that may occur on or within a computer system. The invention can also identify relationships in other security related information.

BACKGROUND OF THE INVENTION

The nature of a distributed network, such as the internet, makes it vulnerable to attack. The internet was designed to allow for the freest possible exchange of information, data, and files. However, this free exchange of information carries a price: many users will try to attack the networks and computers connected to the internet; many users will also try to invade other users' privacy and attempt to crack databases of sensitive information or intercept information as it travels across internet routes.

To detect or prevent such computer attacks, intrusion detection systems (IDS) and software programs that gather information and make changes to security configurations of network computers have been developed. However,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 181

these conventional intrusion detection systems can typically have many problems and drawbacks. Conventional intrusion detection systems typically comprise hardware that is dedicated to intrusion detection on networks. Other intrusion detection systems can simply comprise programs running on a host computer.

The problems and drawbacks of many conventional intrusion detection systems can be attributed to at least two parameters that are part of any detection design: The first parameter is the speed in which a detector of an intrusion detection system must run in order to be transparent to the data or communication that flows through the detector. Detectors that typically run on dedicated personal computers must be able to handle constantly increasing loads of information traffic, as network speeds increase from 100 megabits per second to gigabit per second speed and beyond. Because of these high speeds, a detector of an intrusion detection system cannot perform complex analysis of the information that flows through the detector for obvious reasons. That is, if a detector were to perform complex analysis of the information flowing through it, then such analysis would fail to keep up with the flow of information that passes through the detector.

A second key parameter that is part of any detection design is typically the volume of information that may pass through a detector. Because of the high speed at which information passes through a detector, a detector must be able to analyze high volumes of data packets.

In light of current network speeds and the corresponding volume of information that is generated as a result of the network speeds, many detectors of conventional intrusion detection systems can provide very limited protection against complex and more sophisticated computer attacks. This limited protection can manifest itself when many false positives are generated by an intrusion detection system. In other words, many conventional intrusion detection systems may generate false alarms based on communications between computers that do not comprise any threat or attacks.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 182

In addition to false alarms, conventional intrusion detection systems are typically not equipped to handle complex analysis because of the limitations on current processing speeds. For example, many conventional intrusion detection systems cannot execute central processing unit-intensive checks such as the well-known L0pht Crack. The L0pht Crack decode can use cryptographic challenge-response data from Windows (SMB) connections to crack passwords in use on a network. The conventional method for executing L0pht Crack is to obtain packets using a packet-capturing tool and then crack the passwords offline. Conventional intrusion detection system typically cannot employ the L0pht Crack method in any real-time analysis.

Another obstacle of conventional intrusion detection systems is that most intrusion detection systems have very limited or short term memory capacity. In other words, long histories of data streams are seldom kept by the detectors in conventional intrusion detection systems.

Another problem of conventional intrusion detection systems is that the detectors of such systems typically only watch or observe a single environment. For example, detectors usually observe only parts of networks. Conventional detectors typically have a limited scope of awareness since they are designed to observe only portions of a network instead of the entire network as a whole. Because conventional detectors typically monitor only portions of a network, they are unable to track more sophisticated computer attacks such as distributed attacks.

In addition to the inability to track more sophisticated computer attacks, many conventional intrusion detection systems do not permit active probing of an attacker or the target of a computer attack. Active probing typically involves making a determination to see whether a computer attack has had an effect on its target. Further, probing can also comprise methods for discovering additional information about an attacker. However, as mentioned above, most intrusion detection systems do not permit active probing since such probing could reveal

IPR2024-00027
CrowdStrike Exhibit 1002 Page 183

the location of the detector. And if the location of a detector is revealed, it sometimes may also become a target for a computer attack.

Accordingly, there is a need in the art for a method and system for managing security information for an entire network. That is, there is a need in the art to log, investigate, respond to, and track computer security incidents that may occur in a network computer system. There is also a need in the art to determine whether security within a network or over a network has been compromised or if an incident is just some odd behavior that should be disregarded by an intrusion detection system. Another need exists in the art for a method and system that can monitor and analyze security information from multiple data sources so that rather complex and sophisticated computer attacks can be identified, stopped, or prevented. A further need exists in the art for a method and system for managing security information in real-time.

Another need exists in the art for a method and system for managing security information such that it can be determined if one or more real-time computer events are related to each other and if they are a part of a larger scheme or sophisticated attack. An additional need exists in the art for a method and system for managing security information where multiple computer events can be correlated together if the computer events are part of a larger scheme or attack. Another need exists in the art for a method and system for managing security information where computer events that are detected can be prioritized so that attention can be focused on those computer events which could cause the most damage to a network or individual computers. Similarly, another need exists in the art for a method and system for managing security information that enables rapid response to existing computer attacks in addition to prevention of the additional computer attacks which may spin off from or be generated from a single computer attack. A further need exists in the art for a method and system for managing security information such that real-time computer events can be classified and ranked according to their respective priorities in the context of the environment in which the event occurred.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 184

SUMMARY OF THE INVENTION

The present invention can solve the aforementioned problems by providing a computer security management system that can log, investigate, respond to, and track computer security incidents that can occur in a networked computer system. The invention can track suspicious computer activity or actual computer security threats. Actual security threats can include, but are not limited to, integrity attacks, confidentiality attacks, denial of service attacks, multi-stage attacks, or other similar attacks on computers or computer networks. The invention typically refers to suspicious computer activity descriptions obtained from data sources as real-time raw events and actual computer security threats as mature correlation events. The invention can comprise a method and system for managing security information collected from one or more data sources. More specifically, the present invention can comprise a fusion engine which "fuses" or assembles information from multiple data sources and analyzes this information in order to detect relationships between raw events that may indicate malicious behavior and to provide an organized presentation of information to one or more consoles without slowing down the processing performed by the data sources.

The multiple data sources can comprise sensors or detectors that monitor network traffic or individual computers or both. The sensors can comprise devices that may be referred to as intrusion detection systems (IDS). Because the present invention can be separate from IDS devices, it permits the IDS devices to operate efficiently and at high speeds when real-time processing of high volumes of data traffic is essential.

The data sources can also comprise firewalls and other like security or IDS devices. Further, the data sources can comprise any devices that may or may not provide real-time information, such as audit systems, that provide additional environmental information about a network or computer of interest. For example, one data source could comprise a database. The database may include a raw event classification database that contains categories of different types of raw events. Another database can comprise a context or knowledge database that includes

IPR2024-00027
CrowdStrike Exhibit 1002 Page 185

network context information, such as host vulnerability statuses, historical computer event frequency values, and network zone definitions.

From the multiple data sources, the fusion engine of the present invention can correlate and classify real-time, raw computer events. That is, unlike the conventional art which usually processes computer events after some period of time, the present invention can identify relationships between one or more real-time, raw computer events as they are received in real-time. Real-time raw computer events or raw events may comprise any computer activity that may be tracked by an intrusion detection system as a possible attack on a computer or a plurality of computers. Raw events can be generated by detectors of intrusion detection systems. Each raw event may comprise various parameters that may include, but are not limited to the following: source internet protocol address of the computer activity, destination internet protocol address of the computer activity, priority status assigned by the detector, a vulnerability status assigned by the detector, a time stamp, and an event type parameter.

The fusion engine can determine if one or more real-time raw events are related to each other and if they are part of a larger scheme or computer attack. Real-time raw events that are related to each other and that may indicate that a computer attack may be occurring are referred to by the fusion engine as a mature correlation event. A correlation event can comprise one or more raw events. However, a correlation event does not mean an actual security threat or attack has been detected. Correlation events typically store related raw events and usually indicate that a security event or computer attack has occurred when the correlation event is deemed to be mature. In order to be deemed mature, a correlation event must satisfy the criteria or algorithm of a corresponding correlation rule. Therefore, it is possible to track numerous correlation events that may comprise one or more raw events that have not yet been identified as being a mature correlation event or actual computer security threat or computer attack.

The fusion engine can also assess and rank the risk of real-time raw events as well as mature correlation events base on information about the environment or context in which the event occurred. The fusion engine can display this risk and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 186

rank information as messages on a console. The fusion engine can generate and send updates related to mature correlation events to a console. Further, the fusion engine can determine and indicate when a mature correlation event has stopped occurring.

In order to assess risks and determine ranks of real-time raw events, the fusion engine can utilize the aforementioned raw event classification database and the knowledge database. The raw event classification database can permit the fusion engine to classify raw computer events while the knowledge database can permit the fusion engine to rank and evaluate the risk of a raw computer event based upon the context of the raw computer event. The raw event classification database can comprise one or more tables of security information. That is, the raw event classification database can comprise tables that include information that can categorize raw events based on their impact on the target host (confidentiality, integrity, or availability), their scope (network, host, or service), and the method they employ (backdooring, IDS evasion or detection evasion, etc.). The context of the raw computer event can be determined by comparing parameters of the raw event with context parameters in a context or knowledge database, such as the aforementioned event vulnerability statuses, historical computer event frequency values, and zone definitions.

To determine if one or more raw computer events are part of or form a mature correlation event, the fusion engine can apply one or more rules that can be triggered based upon how the fusion engine classifies a raw computer event. In other words, the rules applied by the fusion engine can be activated and applied to raw computer events according to the classification (identification of the type or kind) of the raw events.

In addition to determining whether raw computer events are part of or form a mature correlation event or actual security threat, the fusion engine can also manage its high speed memory resources very efficiently. For example, the fusion engine can employ memory management techniques that erase raw events, immature, and mature correlation events that have either exceeded a predetermined time period or that have met predetermined conditions or both. The

IPR2024-00027
CrowdStrike Exhibit 1002 Page 187

high speed memory resources can comprise RAM containing data that is categorized according to the classifications of the raw events and mature correlation events.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a network personal computer that provides the exemplary operating environment for the present invention.

Fig. 2 is a functional block diagram illustrating exemplary network architecture for the present invention.

Fig. 3 is a functional block diagram illustrating an exemplary software architecture for the present invention.

Fig. 4 is a functional block diagram illustrating exemplary software and hardware architecture for the present invention.

Fig. 5A is a functional block diagram illustrating security information data sources feeding information about a computer incident source to an event collector that is connected to a fusion engine.

Fig. 5B is a diagram illustrating the type of data that may be present in a raw event generated by a detector in an intrusion detection system.

Fig. 5C is a diagram illustrating an exemplary raw event that has been processed by the CoBRA processor of the fusion engine.

Fig. 5D is a functional block diagram illustrating an exemplary attack from attacked host computer security threat.

Fig. 5E is a diagram illustrating the possible data of an exemplary correlation event that is based on Fig. 5D.

Fig. 5F is a diagram illustrating the possible data of another exemplary correlation event that is based on Fig. 5D.

Fig. 6 is a functional block diagram illustrating some components of the fusion engine illustrated in Fig. 2.

Fig. 7 is a logic flow diagram illustrating an exemplary embodiment of a method for managing security information collected from one or more data sources.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 188

Fig. 8 is a data flow diagram illustrating the exchange of information between various software components that are illustrated in Fig. 6 and discussed with reference to Figs. 7, and 9-15.

Fig. 9 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for assigning real-time raw events to one or more categories in an event type list.

Fig. 10 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for assigning context parameters to each real-time raw event.

Fig. 11 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for adjusting the priority status of each real-time raw event.

Fig. 12 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for adjusting the priority status of each real-time raw event.

Fig. 13 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for forwarding real-time raw event data to corresponding rules.

Fig. 14 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for determining whether a correlation event is mature.

Fig. 15 is a logic flow diagram illustrating an exemplary subprocess or routine of Fig. 7 for determining whether a mature correlation event has stopped occurring.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

The present invention may be embodied in program modules that run in a distributed computing environment. The present invention can comprise a computer security management system that can log, investigate, respond, and track computer security incidents that can occur in a network computer system. The present invention can comprise a fusion engine which "fuses" or assembles

IPR2024-00027
CrowdStrike Exhibit 1002 Page 189

information from multiple data sources and analyzes this information in order to provide an organized, and sometimes ranked, presentation of information to one or more consoles. The fusion engine can classify raw real-time computer events while also ranking the real-time computer events based upon comparisons with one or more databases.

Illustrative Operating Environment

Although the illustrative embodiment will be generally described in the context of an program modules running on a personal computer and a server, those skilled in the art will recognize that the present invention may be implemented in conjunction with operating system programs or with other types of program modules for other types of computers. Furthermore, those skilled in the art will recognize that the present invention may be implemented in either a stand-alone or in a distributed computing environment or both. In a distributed computing environment, program modules may be physically located in different local and remote memory storage devices. Execution of the program modules may occur locally in a stand-alone manner or remotely in a client server manner. Examples of such distributed computing environments include local area networks and the Internet.

The detailed description that follows is represented largely in terms of processes and symbolic representations of operations by conventional computer components, including a processing unit (a processor), memory storage devices, connected display devices, and input devices. Furthermore, these processes and operations may utilize conventional computer components in a heterogeneous distributed computing environment, including remote file servers, computer servers, and memory storage devices. Each of these conventional distributed computing components is accessible by the processor via a communication network.

The processes and operations performed by the computer include the manipulation of signals by a processor and the maintenance of these signals within data structures resident in one or more memory storage devices. For the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 190

purposes of this discussion, a process is generally conceived to be a sequence of computer-executed steps leading to a desired result. These steps usually require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, or otherwise manipulated. It is convention for those skilled in the art to refer to representations of these signals as bits, bytes, words, information, elements, symbols, characters, numbers, points, data, entries, objects, images, files, or the like. It should be kept in mind, however, that these and similar terms are associated with appropriate physical quantities for computer operations, and that these terms are merely conventional labels applied to physical quantities that exist within and during operation of the computer.

It should also be understood that manipulations within the computer are often referred to in terms such as creating, adding, calculating, comparing, moving, receiving, determining, identifying, populating, loading, executing, etc. that are often associated with manual operations performed by a human operator. The operations described herein can be machine operations performed in conjunction with various input provided by a human operator or user that interacts with the computer.

In addition, it should be understood that the programs, processes, methods, etc. described herein are not related or limited to any particular computer or apparatus. Rather, various types of general purpose machines may be used with the program modules constructed in accordance with the teachings described herein. Similarly, it may prove advantageous to construct a specialized apparatus to perform the method steps described herein by way of dedicated computer systems in a specific network architecture with hard-wired logic or programs stored in nonvolatile memory, such as read-only memory.

Referring now to the drawings, in which like numerals represent like elements throughout the several Figures, aspects of the present invention and the illustrative operating environment will be described.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 191

Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Referring now to Fig. 1, an illustrative environment for implementing the invention includes a conventional personal computer **100**, including a processing unit **102**, a system memory, including read only memory (ROM) **104** and random access memory (RAM) **108**, and a system bus **105** that couples the system memory to the processing unit **102**. The read only memory (ROM) **104** includes a basic input/output system **106** (BIOS), containing the basic routines that help to transfer information between elements within the personal computer **100**, such as during start-up. The personal computer **100** further includes a hard disk drive **118** and an optical disk drive **122**, e.g., for reading a CD-ROM disk or DVD disk, or to read from or write to other optical media. The drives and their associated computer-readable media provide nonvolatile storage for the personal computer **100**. Although the description of computer-readable media above refers to a hard disk, a removable magnetic disk and a CD-ROM or DVD-ROM disk, it should be appreciated by those skilled in the art that other types of media are readable by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, and the like, may also be used in the illustrative operating environment.

A number of program modules may be stored in the drives and RAM **108**, including an operating system **114** and one or more application programs **110**, such as a program for browsing the world-wide-web, such as WWW browser **112**. Such program modules may be stored on hard disk drive **118** and loaded into RAM **108** either partially or fully for execution.

A user may enter commands and information into the personal computer **100** through a keyboard **128** and pointing device, such as a mouse **130**. Other control 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 **102** through an input/output interface **120** that is coupled to the system bus, but may be connected by other interfaces, such as a game port, universal serial bus, or firewire port. A display monitor **126** or other

IPR2024-00027
CrowdStrike Exhibit 1002 Page 192

type of display device is also connected to the system bus **105** via an interface, such as a video display adapter **116**. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers or printers. The personal computer **100** may be capable of displaying a graphical user interface on monitor **126**.

The personal computer **100** may operate in a networked environment using logical connections to one or more remote computers, such as a host computer **140**. The host computer **140** may be a server, a router, a peer device or other common network node, and typically includes many or all of the elements described relative to the personal computer **100**. The LAN **136** may be further connected to an internet service provider **134** ("ISP") for access to the Internet **138**. In this manner, WWW browser **112** may connect to host computer **140** through LAN **136**, ISP **134**, and the Internet **138**. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer **100** is connected to the LAN **136** through a network interface unit **124**. When used in a WAN networking environment, the personal computer **100** typically includes a modem **132** or other means for establishing communications through the internet service provider **134** to the Internet. The modem **132**, which may be internal or external, is connected to the system bus **105** via the input/output interface **120**. It will be appreciated that the network connections shown are illustrative and other means of establishing a communications link between the computers may be used.

The operating system **114** generally controls the operation of the previously discussed personal computer **100**, including input/output operations. In the illustrative operating environment, the invention is used in conjunction with Microsoft Corporation's "Windows NT" operating system and a WWW browser **112**. However, it should be understood that the invention can be implemented for use in other operating systems, such as Microsoft Corporation's "WINDOWS 3.1," "WINDOWS 95", "WINDOWS 98" and "WINDOWS 2000" operating systems, IBM Corporation's "OS/2" and "AIX "operating system, SunSoft's

IPR2024-00027
CrowdStrike Exhibit 1002 Page 193

"SOLARIS" operating system used in workstations manufactured by Sun Microsystems, and the operating systems used in "MACINTOSH" computers manufactured by Apple Computer, Inc. Likewise, the invention may be implemented for use with other WWW browsers known to those skilled in the art.

Host computer **140** is also connected to the Internet **138**, and may contain components similar to those contained in personal computer **100** described above. Additionally, host computer **140** may execute an application program for receiving requests for WWW pages, and for serving such pages to the requestor, such as WWW server **142**. WWW server **142** may receive requests for WWW pages **150** or other documents from WWW browser **112**. In response to these requests, WWW server **142** may transmit WWW pages **150** comprising hyper-text markup language ("HTML") or other markup language files, such as eXetnsible Markup Language (XML), to WWW browser **112**. Likewise, WWW server **142** may also transmit requested data files **148**, such as graphical images or text information, to WWW browser **112**. WWW server **142** may also execute scripts **144**, such as CGI, PERL, ASP, or JSP (Java Server Pages) scripts, to dynamically produce WWW pages **150** for transmission to WWW browser **112**. WWW server **142** may also transmit scripts **144**, such as a script written in JavaScript, to WWW browser **112** for execution.

Similarly, WWW server **142** may transmit programs written in the Java programming language, developed by Sun Microsystems, Inc., to WWW browser **112** for execution. The WWW server **142** could comprise a UNIX platform running Apache or Netscape webserver. Alternatively, the WWW server **142** could comprise an Internet Information Server (IIS). The present invention is not limited to these enumerated examples. Other web server environments are not beyond the scope of the present invention.

As will be described in more detail below, aspects of the present invention may be embodied in application programs executed by host computer **142**, such as scripts **144**, or may be embodied in application programs executed by computer **100**, such as Java applications **146**. Those skilled in the art will also appreciate

IPR2024-00027
CrowdStrike Exhibit 1002 Page 194

that aspects of the invention may also be embodied in a stand-alone application program.

Exemplary Computer Architecture

Referring now to Figure **2**, the computer architecture for one exemplary embodiment of the present invention will be described. Figure **2** illustrates the System **20** for managing security information collected from one or more data sources. The security system **20** can comprise a fusion engine **22** that is linked to an event collector **24**. The event collector **24** can comprise an event sink or device that can organize events received from multiple data sources in a logical manner. Further details of the event collector **24** are described in a related application entitled, "System and Method for Managing Security Events on a Network," (Attorney Docket No. 05456-105005) filed on April 27, 2001 and assigned U.S. Application Serial Number _____, the contents of which is hereby incorporated by reference.

The security management system **20** can further comprise an event database **26** that is also linked to the event collector **24**. The security management system can also comprise data sources **28** that are linked to the event collector **24** and a console **30** which is also linked to event collector **24**. Information from the databases are typically loaded into fusion engine **22** that comprises high-speed memory devices such as random access memory (RAM) since comparisons between raw events and the databases must be performed in a very rapid and in a very efficient manner. Most memory resources used in the fusion engine **22** comprise high-speed memory devices such as RAM (sometimes referred to as "caches" hereinbelow). However, other memory resources are not beyond the scope of the present invention. The memory resources of the fusion engine **22** should be designed to handle high volumes of information with increased speed.

The one or more data sources **28** can comprise many different hardware and software devices. For example, a data source **28** can comprise a network detector or a host detector. Similarly, a data source **28** could also comprise a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 195

firewall or an audit system. The present invention is not limited to the types of data sources illustrated. The function of a data source **28** is to provide the event collector **24** with various types of information as it may relate to the network, host, or single computer being monitored by the security management system **20**. Other like data sources **28** are not beyond the scope of the present invention. One data source **28** can comprise a host detector which monitors network traffic in the form of data packets. Another data source **28** could comprise observations made by users who are monitoring any network or computer activity.

The one or more data sources **28** forward their information to the event collector **24**. The event collector **24** may comprise one or more program modules designed to store and collect the data received from the one or more data sources **28**. The event collector **24** can arrange the data and store it in the event database **26**. The event collector **24** also forwards any information received from the data sources **28** to the fusion engine **22**. The detectors **28** of intrusion detection systems scan raw network traffic or local system events for predefined patterns. Once the detectors identify these predefined patterns of information, the detectors generate a raw event which is then sent to the event collector and later to the fusion engine **22**. The fusion engine assembles or "fuses" the raw events or information received from the event collector **24**. In other words, the fusion engine **22** organizes and analyzes the information received from the one or more data sources **28** in order to provide an organized presentation of information by correlating (identifying relationships between) raw computer events that are related to each other.

Once the fusion engine **22** determines that two or more events are related to each other (to form a "correlation" event), the fusion engine **22** generates messages and forwards these messages to the event collector **24**. The event collector **24**, in turn, forwards the messages generated by the fusion engine **22** to a console **30**.

Console **30** may comprise a program module that runs on a separate personal computer. The fusion engine **22** may comprise one or more program modules running on a personal computer. The fusion engine **22**, the event

IPR2024-00027
CrowdStrike Exhibit 1002 Page 196

collector **24**, and the event database **26** have been circumscribed by a box **32** to demonstrate that each of these software components can reside on a single computer. However, the present invention is not limited to this configuration. And therefore, the fusion engine **22**, the event collector **24**, and the event database **26** could also reside on separate computer devices. Other combinations of the software components illustrated could be implemented. That is, the fusion engine **22** and event collector **24** could reside on one hardware device while the event database **26** resides on another hardware device. Conversely, the event collector **24** and event database **26** could reside on one hardware device while the fusion engine **22** resides on another hardware device. Those skilled in the art will appreciate that disclosed software architecture is not limited to the architecture illustrated in the drawings.

Referring now to Figure **3**, a functional block diagram illustrating another exemplary software architecture for the present invention is illustrated. In Figure **3**, the fusion engine program module **22** and a data source such as a detector module **28** could reside in a single machine. That is, the high speed IDS functions of the detector **28** could reside near the kernel of a computer while the fusion engine **22** could reside in the user mode part of the computer. In this way, the additional processing of the fusion engine **22** would not slow down the high speed intrusion detection system functions performed by the detector **24**.

Referring now to Figure **4**, this Figure illustrates another functional block diagram of exemplary software and hardware architectures for the present invention. In this one exemplary embodiment, the data source **28** comprising a detector could be implemented in a hardware device such as a detector board or a detector chip so that the high speed intrusion detection system functions could be performed. In this exemplary embodiment, the fusion engine **22** could simply reside as a program module in software. Figure **4** demonstrates that the data sources **28** that require access to high speed data streams can be separated from the fusion engine **22** such that network processing speeds can be achieved without significant interpretation or delay or both.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 197

Referring now to Figure 5A, this Figure illustrates a functional block diagram of the security information data sources 28 feeding information about a computer incident source 500 to the event collector 24 which is also connected to the fusion engine 22. Figure 5A further illustrates a network 510 that may comprise numerous data sources 28, user work stations 520, a server 530 that is a target for a computer incident source 500, an internal router 540, and the server 550. The network 510 is connected to the internet 590 by an external router 580 and by a firewall 28. The firewall 28 can comprise a bastion host or similar device. The firewall 28 can also be connected to an internal screening router 540 that may examine all packets of data travelling to and from the internal screening router 540. The user work stations 520 can be stand-alone personal computers that access the servers 530, 550.

The computer incident source 500 can be a computer or a network of computers that originate an attack against the network 510 and more specifically, the server 530 (attacked host). The computer incident source 500 can be connected to the server 560 of a local area network. Alternatively, instead of a server 560, the computer incident source 500 can be connected to a dial-in internet service provider (ISP) or any computer connected to the Internet. The server 560 or ISP (or other computer connected to the internet) can then be connected to a router 570. A router 570 provides access to a distributed computer network such as the Internet 590.

While the computer incident source 500 can be located outside of the network 510, it is possible for the computer incident source 500 to be located within the network 510. That is, a computer incident source 500 could be a user workstation 520 located within the network 510. For example, in case of a disgruntled employee within a company, a user workstation 520 could be used as the computer incident source 500 when the employee decides to interfere or hamper the operation of a network 510 or one or more other workstations 520 within the network 510.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 198

Each of the data sources **28** has a data line illustrated by dashed lines that feed into the event collector **24**. The dashed data lines could comprise actual physical data lines, however, these data lines are more for illustrative purposes to demonstrate that each of the data sources is operably linked to the event collector **24**. Also, the event collector **24** could reside within the network **510** so that it would not be vulnerable to a direct attack from a computer incident source **500**. The placement of event collector **24** within Figure **5** illustrates the collection function of the event collector **24**. Figure **5** illustrates fundamental concepts of a system for managing security information rather than the actual physical architecture that would be implemented to support such a system.

Exemplary Data Processed by Fusion Engine

Referring now to Figure **5B**, this diagram illustrates an exemplary raw event **505** that is generated by a detector of an intrusion detection system. The raw event **505** may comprise a source Internet protocol address **515**; a destination Internet protocol address **525**; a priority status **535**; a detector assigned vulnerability status **545**, an event type parameter **555**; and a time stamp **565**. As will be discussed in further detail below, the priority status **535** assigned by detectors of an intrusion detection system are typically very conservative in nature. That is, since detectors must process information very quickly, they are unable to run complex algorithms or tests to ascertain the risk of certain computer raw events. Therefore, the priority status **535** of many raw events generated by detectors will be very high relative to an actual priority of a raw event.

Referring now to Figure **5C**, this Figure is a diagram illustrating a CoBRA-(Context Based Risk Adjustment) processed raw event. The CoBRA-processed raw event **502** typically contains all of the previously detector assigned parameters of the raw event and in addition the CoBRA-processed parameters that may comprise any one of the following: a CoBRA-assigned vulnerability value **504**; a CoBRA-assigned historical frequency value **506**; a CoBRA-assigned source zone value **508**; a CoBRA-assigned destination zone value **510**; a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 199

CoBRA-assigned sensor zone value **512**; a CoBRA-assigned original priority status **514**; and a priority change reason **516** text string comprising a reason why the priority of the raw event was adjusted (if adjusted). These CoBRA-assigned values will be discussed below in further detail with respect to Figures **11** and Figure **12**.

Exemplary Raw and Correlation Events Processed by Fusion Engine

Referring now to Figure **5D**, this Figure is a functional block diagram illustrating an exemplary Attack From Attacked Host (AFAH) computer security threat. Figure **5D** illustrates a computer incident source **503** with an Internet protocol address of 1.1.1.1 sending an attack to host (attacked host) **505** that has an Internet protocol address of 2.2.2.2. The attack between the computer incident source **503** and the attacked host **505** may be characterized as a raw computer event **I**. After being attacked, the attacked host **505** then sends another attack to a second host **507**, having an Internet protocol address of 3.3.3.3. The attack between the attacked host **505** and the second host **507** may be characterized as a second raw event **II**. The second host **507** generates an attack on a third host **509**, having an Internet protocol address of 4.4.4.4. The attack between the second host **507** and third host **509** may be characterized as a third raw event **III**.

After processing the raw events **I**, **II** and **III**, the fusion engine **22** may identify the relationships between respective raw events. Therefore, after processing the raw events illustrated in Figure **5B**, the fusion engine may generate a mature correlation event **511** that corresponds to the first and second raw events **I** and **II**. Further, the fusion engine **22** may further generate a second mature correlation event **513** that identifies a relationship between the second and third raw events **II** and **III**. Further details of the processing performed by the fusion engine **22** to generate the first and second mature correlation events **511** and **513** will be discussed below in further detail with respect to Figure **7** and Figure **14**.

Referring now to Figure **5E**, this Figure is a diagram illustrating the possible data of an exemplary correlation event that is based on Figure **15**. The

IPR2024-00027
CrowdStrike Exhibit 1002 Page 200

correlation event **511** illustrated in Figure **5E** may comprise two sets of lists. The first list may identify inbound attacks relative to the attacked host **505** and outbound attacks relative to the attacked host **505**. Further details of the first exemplary correlation event **511** will be discussed in further detail below with respect to Figure **7** and Figure **14**.

Referring now to Figure **5F**, this Figure is a diagram illustrating the possible data of the second correlation event **513** illustrated in Figure **15**. The second correlation event **513** may also comprise two lists: one list identifying inbound attacks relative to the second host **507** and a second list identifying outbound attacks relative to the second host **507**. Further details of the second mature correlation event **513** will be discussed below with respect to Figure **7** and Figure **14**.

The exemplary attack from attacked host computer security threat illustrated by Figures **5D** through **5F** is just but one example of the possible computer security threats that can be analyzed with the fusion engine **22**. As discussed above and below, other types of computer security threats are not beyond the scope of the present invention. In one exemplary embodiment, the fusion engine may track at least twenty different types of possible correlation events. Those skilled in the art will appreciate the present invention is not limited to the exemplary correlation events illustrated in Figure 5D and that fewer or more correlation events can be utilized by the present invention without departing from the scope and spirit thereof.

Exemplary Software Components of Fusion Engine

Figure **6** is a function block diagram illustrating some components of the fusion engine **22** that is illustrated in Figure **2**. Basically, Figure **6** illustrates some of the numerous software components that make up the software architecture for the fusion engine **22**.

The present invention includes a computer program which embodies the functions described herein and illustrated in the appended flow charts. However, it should be apparent that there could be many different ways of implementing the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 201

invention in computer programming, and the invention should not be construed as limited to any one set of computer program instructions. Further, a skilled programmer would be able to write such a computer program to implement the disclosed invention without difficulty based on the flow charts and associated description in the application text, for example. Therefore, disclosure of a particular set of program code instructions is not considered necessary for an adequate understanding how to make and use the invention. The inventive functionality of the claimed computer program will be explained in more detail in the following description in conjunction with the remaining Figures illustrating the program flow.

In one exemplary embodiment, the fusion engine **22** can be implemented with object-oriented programming. Therefore, some of the software components illustrated in Figure **6** can have both data and code associated with a respective software object. However, the general functionality of each software object will be generally described such that a skilled programmer will be able to write a computer program to implement the disclosed functionality of the software object.

The fusion engine **22** may comprise several software components. In the exemplary embodiment illustrated in Figure **6**, the fusion engine **22** may comprise an event reader **600** that receives raw computer event information from the event collector **24**. The event reader **600** is operably linked to the classifier **615**. The classifier **615** organizes the raw event information received from the event reader **600**. In other words, the classifier **615** categorizes the raw event information by separating the raw event information according to an event type property that is included in each raw event. The event type property of each raw event is typically generated by a detector in an intrusion detection system.

The classifier **615** can be responsible for forwarding raw event information to the CoBRA processor **625** and one or more correlation rules **620**. The one or more correlation rules **620** may comprise algorithms for testing and determining whether a security incident may be occurring. The correlation rules track raw event information that is received from the classifier and stores the raw event

IPR2024-00027
CrowdStrike Exhibit 1002 Page 202

information in correlation event high speed memory **665**. The correlation event high speed memory **665** may comprise random access memory (RAM) for storing information. However, the present invention is not limited to RAM type memory. Other high speed memory devices are not beyond the scope of the present invention. The classifier **615** can be established based upon a raw event classification database **635**. The classifier **615** can be generated upon initialization of the fusion engine **22** when event classification data is read from the raw event classification database **635** into the classifier **615**.

The CoBRA processor **625** may comprise algorithms or software components for the context based risk adjustment of raw computer events. The CoBRA processor **625** can adjust the priority values of raw computer events by comparing a raw computer event against data contained within a context or knowledge base database **630**. The priority status of raw events is typically established by detectors of intrusion detection systems before forwarding the raw event data to the fusion engine **22**. After processing raw computer events, the fusion engine **22** can inform the event collector **24** whether a security event is occurring. The fusion engine **22** typically formats and sends one or more correlation events to the event collector via the event reporter **660**. As noted above, a correlation event may comprise one or more computer events that are related to each other as determined by the fusion engine **22**.

The fusion engine **22** may further comprise memory management devices that can conserve memory resources for the fusion engine **22**. For example, in one exemplary embodiment, the fusion engine **22** may comprise a memory management list **640**, a raw event tracking index **645** and a mature event list **650**. The memory management list **640** is typically linked to the raw event tracking index **645**. Further details of the functionality with respect to the memory management list **640**, raw event tracking index **645**, and the mature event list **650** will be discussed below in the brief process description of the software components illustrated in Figure **6**.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 203

Exemplary Object-Oriented Architecture for Figure 6

One of the software components of the fusion engine **22** that can be implemented as a software object, in one exemplary embodiment, is the event reader **600**. The event reader **600** can receive raw computer events from either the event collector **24** or an event log file **610**. The event log file **610** can comprise files having comma separated values (CSV) formats that store computer event data from an intrusion detection system. The event reader **600** typically reads in raw computer events or raw events which can be any computer activity that may be tracked by an intrusion detection system as a possible attack on a computer or a plurality of computers. The event reader typically creates raw event data objects (not shown) that are processed by other software components on a fusion engine **22**.

In one exemplary embodiment, the event reader **600** can be linked to a classifier **615** which may comprise one or more event type objects. The classifier **615** receives the raw event objects that are generated by the event reader **600**. A classifier **615** associates each raw event object with a corresponding event type object that has been established for a specific event type parameter 555. In other words, the classifier assigns raw event objects to event type objects according to the type of raw event. It is noted that each raw event received by the event reader **600** has been assigned a type or categorization based upon the intrusion detection system that generated the raw event.

One function of the classifier **615** is to categorize or classify each of the raw events and then forward the raw event objects to specific correlation rules **620** based upon their type. The correlation rules **620** can also take form of software objects that receive the raw event objects from the classifier **615**.

The classifier **615** can also forward the raw event object to the Context Based Risk Adjustment (CoBRA) processor **625**. The CoBRA processor is a risk assessment mechanism that can adjust priority parameters of raw event objects. The CoBRA processor **625** accesses a context or knowledge base database **630** in order to perform its context based risk adjustment for each received raw event

IPR2024-00027
CrowdStrike Exhibit 1002 Page 204

object. Basically the CoBRA processor determines the risk of a raw computer event by assessing the event type parameter 555 in combination with environmental factors such as the destination internet protocol address of an attack in addition to the source of the attack.

The context or knowledge base database **630** can include vulnerability statuses assigned to machines within a network, historical event frequency values, and network zone definitions. The vulnerability statuses can be results of vulnerability scans performed by devices outside of the fusion engine **22** that determine the strength or resistance of a network or single computer to an attack. The historical event frequency value can comprise signatures or data relating to computer attacks that occur over very long periods of time. The network zone definitions can comprise values assigned to parts of a network based upon the amount and type of information that may be assessable in certain parts of a network. For example, it is useful to distinguish internal, external, and demilitarized zones as will be discussed below.

The fusion engine **22** can further comprise a raw event classification database **635** that can be responsible for establishing the different event type objects which form the classifier **615**. The raw event classification database **635** can comprise one or more tables of security information. The tables can include information relating to the type parameter **555** of a raw event assigned by detectors. The raw event classification database **635** can categorize raw events based on their impact on the target host (confidentiality, integrity, or availability), their scope (network, host, or service), and the method they employ (backdooring, cloaking, etc.). Confidentiality events can be those events that indicate an attacker is attempting to obtain information from or about a host. Integrity events can be those events that indicate an attacker is attempting to alter data on a host, possibly to gain unauthorized access.

Availability events can be those events that indicate an attacker is attempting to cause a denial of service, such as by causing a host to crash. In addition to the above general criteria, specialized criteria useful in recognizing

IPR2024-00027
CrowdStrike Exhibit 1002 Page 205

particular correlation events can serve as a basis for classifying events. For example, events that confirm the success of a denial of service attempt can be grouped into a category used by a correlation rule **620** that identifies denial of service attacks that are believed to have succeeded. However, the raw event classification database **635** is not limited to these categories or parameters. Other categories and parameters which further define raw events are not beyond the scope of the present invention.

The fusion engine **22** can further comprise a memory management list **640**, a raw event tracking index **645**, and a mature event list **650**. The memory management list **640** enables the fusion engine **22** to manage its memory resources by eliminating or deleting the oldest raw events when memory resources exceed a predetermined threshold (i.e. - when memory resources are running low). The memory management list **640** can be implemented as a software object that deletes raw events that are considered oldest when memory resources run low. Related to the memory management list **640** is the raw event tracking index **645** which can also be implemented as another software object. A raw event tracking index **645** can identify which software objects may contain a particular raw event object. That is, the raw event tracking index identifies those software objects that may be storing a raw event that has now become old and should be deleted from the fusion engine **22**.

Related to the memory management list **640** and raw event tracking index **645** is the mature correlation event list **650** which tracks those raw events that have been identified as a pattern of activity or an actual computer threat that should not be removed from the memory management list **640**. In other words, the mature correlation event list identifies the raw events which should not be deleted from the fusion engine **22** since these events are deemed to be part of mature correlation events or actual computer security threats.

The fusion engine **22** may further comprise a controller **655** that may be responsible for the data flow between respective software objects. In other words,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 206

the controller **655** can be implemented as a high level software object which controls the data flow between lower level software objects.

The fusion engine **22** may further include the event reporter **660** that can also be implemented as a software object in the exemplary and preferred object-oriented programming environment. The event reporter **660** can be a software object that receives mature correlation events which are forwarded to the event collector **24**. Mature correlation events can comprise one or more raw events that are associated together because the one or more raw events may pose an actual computer security threat.

Computer-Implemented Process for Managing Security Information

Referring now to Figure **7**, this Figure illustrates an exemplary logic flow diagram of a computer-implemented process for managing security information collected from one or more data sources. More specifically, the logic flow diagram illustrated in Figure 7 illustrates a computer-implemented process for fusing or assembling security information received from multiple data sources and analyzing the security information in order to provide an organized presentation of information to one or more consoles. The logic flow described in Figure 7 is the core logic of the top-level processing loop of the fusion engine **22**, and as such is executed repeatedly as long as the fusion engine **22** is operating.

It is noted that the logic flow diagram illustrated in Figure **7** illustrates a process that occurs after initialization of several of the software components illustrated in Figure **6**. That is, in the exemplary object-oriented programming architecture of the present invention, several of the software components or software objects that are required to perform the steps illustrated in Figure **7** are initialized or created prior to the process described by Figure **7**. Therefore, one of ordinary skill in the art recognizes that several steps pertaining to initialization of the software objects illustrated in Figure **6** are not illustrated. For example, as noted above, the software component or software object comprising the classifier **615** is established after initialization of the fusion engine **22**.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 207

During initialization of the fusion engine **22**, the classifier **615** is built by reading information from the raw event classification database **635**. The classifier **615** may comprise a comprehensive list of event type objects corresponding to the types of raw events that can be processed by the fusion engine **22**, and a distinct event type object list for each event category defined in the raw event classification database **635**. Each distinct event type list can contain the subset of the comprehensive event type list that constitutes the set of raw event types defined by the raw event classification database **635** as belonging to one category. While the initialization of the various software components illustrated in Figure **6** are not described with specificity, a skilled programmer would be able to write such computer programs to implement the disclosed invention without difficulty based upon the following flow charts and associated description of the software architecture in the current application.

Certain steps in the processes described below must naturally precede others for the present invention to function as described. However, the present invention is not limited to the order of the steps described if such order or sequence does not alter the functionality of the present invention. That is, it is recognized that some steps may be performed before or after other steps without departing from the scope and spirit of the present invention.

Referring back to Figure **7**, this Figure provides an overview of the core logic of the top-level processing loop of the entire computer security management process where step **705** is the first step of the process **700**. In decision step **705**, it is determined whether there are any raw events to be processed by the fusion engine **22**. As described above, raw events may comprise computer events reported from detectors of intrusion detection systems. Raw computer events identified by intrusion detection systems may include various parameters. For example, in one exemplary embodiment, each raw event may comprise a source internet protocol address, a destination internet protocol address, the type of computer event being reported, a priority status, a vulnerability status, and a time stamp.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 208

If the inquiry to decision step **705** is negative, then the "no" branch" is followed in which the process proceeds to step **785**. If the inquiry to decision step **705** is positive, then the "yes" branch is followed to step **710** in which the raw computer events or event information is retrieved from a data source. The data source may comprise at least one of the event database **26**, an event log file **610**, or the event collector **24** as illustrated in Figure **8**.

Referring briefly to Figure **8**, this Figure is a data flow diagram illustrating the exchange of information between various software components that are illustrated in Figure **6**. This data flow diagram of Figure **8** parallels the steps described in Figure **7**. For example, step **710** for retrieving event information from data sources is illustrated in Figure **8** where the event reader object **600** in the exemplary object-oriented software architecture reads in event information. References to Figure **8** will be made throughout the detail description of Figure **7**.

Referring back to Figure **7**, after step **710** and in step **715**, the event information or raw events are arranged and assigned a predefined format referred to as raw events. In other words, in the exemplary object-oriented programming environment, the event reader object **600** can create software objects for each raw event as it is received from one of the data sources such as the event database **26**, the event collector **24**, and the event log file **610**. The event reader **600** generates the raw event objects in response to commands received from the controller **655**. In other words, the controller **655** requests the event reader **600** to retrieve raw events from each of the data sources.

After step **715**, in routine **720**, the event type from each raw event is ascertained and each raw event is then assigned to a corresponding event type object in an event type list. In other words, in the exemplary object-oriented software architecture, each raw event object that is created by the event reader **600** is sent to a corresponding event type object that is present within the classifier **615**. Further details of routine **720** will be discussed with reference to Figure **9**.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 209

Next, in decision step **725**, it is determined whether the context based risk adjustment processor (CoBRA) **625** is activated. In other words, a user may elect to not adjust any of the priority status information that is present in each raw event. As noted above, each raw event generated by a detector in an intrusion detection system typically contains parameters drawn to the priority of the event. That is, the detectors of intrusion detection systems assign relative values to computer events to measure the risk or potential damage that could be associated with a raw event. For example, a distributed attack against a network could be assigned a higher priority status relative to a computer attack against a single machine or computer.

If the inquiry decision step **725** is negative, then the "no" branch is followed to routine **740**. If the inquiry decision step **725** is positive, then the "yes" branch is followed to routine **730** in which parameters of a raw event are compared with information in the context or knowledge base database **630**. Also in this routine, context parameters are assigned for each raw event based upon the context information present in the context database **630**. Referring briefly to Figure **8**, the classifier **615** containing the event type objects forwards each raw event to the CoBRA processing object or CoBRA processor **625**. In routine **730**, the CoBRA processor **625** can assign context parameters that relate to the environment or surrounding conditions of a raw event.

Following routine **730**, in routine **735**, the priority status of each raw event can be adjusted or the original status can be left intact based upon the CoBRA assigned context parameters or detector assigned type parameters or both of the raw event. Basically routines **730** and **735** can comprise the exemplary algorithms and methods of the CoBRA processor **625**. Further details of routines **730** and **735** will be discussed below with respect to Figures **10, 11,** and **12.**

Next, in step **737**, the CoBRA processed raw event or unprocessed raw event can be sent to an output device, such as the event collector **24**. The event collector **24** then typically stores the CoBRA processed raw event or unprocessed raw event in the event database **26** and then forwards the event to the console **30**.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 210

As will become apparent from the discussion below, the console **30** can be provided with unprocessed raw events, CoBRA processed raw events, and correlation events. All such events can be handled by the fusion engine **22** and forwarded by the event collector **24** so that they can be displayed to user. It is noted that when a raw event is received by the event collector **24** from a data source **28**, the event collector first sends the raw event to the fusion engine **22**. However, if after a predetermined amount of time, the fusion engine **22** does not respond, then the event collector **24** will store the event in the event database **26** and then forward the unprocessed (not handled by the fusion engine **22**) raw event to the console **30**.

In routine **740**, the raw event is associated with correlation rules **620** based upon the event type assigned by a detector **28**. In this routine, the classifier **615** containing the event type objects determines which correlation rule(s) **620** should process the raw event based upon the event type parameter **555**. Further details of routine **740** will be discussed below with respect to Figure **13**.

In decision step **745**, if a rule corresponding with a raw event exists, then it is determined whether a correlation event exists that is related to the correlation rule. Note that although depicted as a single process flow in Figure 7, steps **745** through **780** are actually performed independently for each Correlation Rule **620** associated with the raw event. Basically, in decision step **745**, the correlation rule object or correlation rule **620** determines if a correlation event object has been created for the current raw event being processed. As illustrated in Figure **8**, the correlation rule objects or correlation rule **620** check the correlation event cache or correlation event high speed memory **665** to determine whether the correlation event for the current raw event being processed has been created. As noted above, the correlation event (or correlation event object in an object-oriented software architecture) can comprise a number of raw events that are grouped together to form a single higher level event.

For step **745**, each Correlation Event has an anchor Internet protocol (IP) address that is used to index the Correlation Event in the Correlation Event type's

IPR2024-00027
CrowdStrike Exhibit 1002 Page 211

area within the correlation event cache **665**. The anchor IP address will be the source IP address or destination IP address of one or more of the Raw Events within the Correlation Event, depending on the Correlation Event type. For example, the anchor IP address of the Attack From Attacked Host event is the IP address of the attacked host. This is the destination IP address of inbound attacks, and the source IP address of outbound attacks. The Correlation Rule for the Attack From Attacked Host event uses the destination IP address of an inbound Raw Event as the Correlation Event lookup key when attempting to retrieve the Correlation Event for which the Raw Event would be an inbound attack. The AFAH Correlation Rule uses the source IP address of the Raw Event as the Correlation Event lookup key when attempting to retrieve the Correlation Event for which the Raw Event would be an outbound attack.

If the inquiry to decision step **745** is positive, then the "yes" branch is followed to step **760**. If the inquiry to decision step **745** is negative, then the "no" branch is followed to step **750** in which a correlation event of the predetermined type associated with the current correlation rule is created. That is, in the exemplary object-oriented software architecture, at this point in processing of a raw event's associated correlation rules **620**, one or more correlation event objects can be created.

Next, in step **755**, the correlation events are stored in the high speed memory devices **665**. The high speed memory devices in one exemplary embodiment can comprise random access memory (RAM). However, other high speed memory devices are not beyond the scope of the present invention. Because of current network processing speeds and the corresponding volumes of information, it is necessary to use high speed memory devices like RAM so that rapid processing of raw information can be obtained.

In step **760**, the raw event is associated with the corresponding correlation event (which was either just created in step **750**, or retrieved from the correlation event cache **665** in step **745**) based upon the type of raw event. In other words, in this step in the exemplary object-oriented software architecture, each correlation

IPR2024-00027
CrowdStrike Exhibit 1002 Page 212

event object stores the raw event based upon its type. In addition to associating the raw event with the correlation event, the raw event tracking index 645 is updated to indicate that the raw event is associated with the correlation event.

Next in decision step **765**, it is determined whether the current correlation event being processed is already mature. Typically, to be mature, a correlation event can contain two or more raw events that meet maturity criteria defined for that specific type of correlation event. The maturity criteria for each correlation event type are defined to identify the conditions under which the occurrence of two or more raw events indicates that a likely security incident is occurring. In step **765**, the correlation event is being examined to determine if it had already been deemed mature as a result of processing of an earlier raw event.

If the inquiry to decision step **765** is positive, then the "yes" branch is followed to step **780**. If the inquiry to decision step **765** is negative, then the "no" branch is followed to routine **770**. In routine **770**, it is determined whether the current correlation event with the newly associated raw event being processed meets or fulfills the maturity criteria set forth in one or more of the correlation rules **620**. In routine **770**, each of the rules that correspond to the type of raw event being processed determines whether the current raw event and any other raw events listed in the current correlation event together satisfy the maturity criteria as set forth in the rule. The present invention can include any number of rules that correspond to the type parameter of a given raw event.

In one exemplary embodiment, the fusion engine **22** can employ numerous correlation rules **620**. The correlation rules can use the event categories defined in the Raw Event Classification Database **635** as a basis for identifying event patterns that indicate either malicious or nonmalicious activity. Many of the correlation events and corresponding correlation rules can reveal the intent of an attacker. The set of correlation events detected by the present invention and corresponding correlation rules includes, but is not limited to, the following:

IPR2024-00027
CrowdStrike Exhibit 1002 Page 213

1) Attack From Attacked Host. This event can be generated when an Integrity attack is seen against a host followed by a Confidentiality, Integrity, or Availability attack originating from that host.

2) Availability Attack Sweep (Multihost DoS Attack). This event can be generated when two or more different types of Availability attacks originating from the same source IP address are seen against multiple target IP addresses.

3) Confidentiality Attack Sweep (Multihost Information Gathering). This event can be generated when two or more types of Confidentiality attacks are seen originating from a single source IP address against multiple target IP addresses.

4) DoS Followed By Confirming Event. This event can be generated when an Availability attack is seen against a target IP address followed by another event indicating that the target is no longer behaving normally. Confirming events include events detected by a network-based sensor indicating the host is not reachable (for example, detection of ARP requests from other hosts for the target), and events detected on the target system itself by a host-based sensor indicating that system resources (such as memory) have become exhausted.

5) External Source Using Internal IP Address. This event can be generated when a network-based sensor that monitors an external network detects a duplicate internal IP address. The occurrence of this condition indicates that an external host is attempting to use the IP address of an internal host, a practice known as spoofing.

6) Integrity Attack Followed By Remote Login. This event can be generated when an Integrity attack is seen against a host followed by a remote login originating from that host.

7) Integrity Attack Followed By Start Of Service. This event can be generated when an Integrity attack is seen against a host followed by a report from a host-based sensor that a new service has been started on the host.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 214

8) Internet Scanner Scan. This event can be generated when an ISS Internet Scanner scan is detected from a host. For a period following detection of the start of the scan, all other events originating from the same host are subsumed into the Internet Scanner Scan event. If the source IP address is configured as an approved scan source, the event can be treated as a nonmalicious event; otherwise it can be treated as a malicious event.

9) Probe Followed By Integrity Attack. This event can be generated when a Probe event is seen against a host followed by an Integrity attack against the host.

10) Integrity Attack Sweep (Trolling For Victims). This event can be generated when two or more types of Integrity attacks are seen originating from a single source IP address against multiple target IP addresses.

11) Login From DoS-attacked Host. This event can be generated when a remote login is seen from a source IP address that is currently the target of an ongoing Availability attack. This combination of events can indicate that an attacker is masquerading as a particular host (the target of the Availability attack) in order to exploit network trust relationships to access other machines on the network.

12) Login Failure Of One User On Multiple Hosts. This event can be generated when login failures of the same user are reported by multiple network- or host-based sensors.

13) Suspicious Activity Followed By Availability Attack. This event can be generated when an event that involves a Cloaking method is reported, followed by an Availability attack. The term "cloaking" applies to any technique that attempts to conceal an attack from intrusion detection systems.

14) Suspicious Activity Followed By Integrity Attack. This event can be generated when an event that involves a Cloaking method is reported, followed by an Integrity attack. The term "cloaking" applies to any technique that attempts to conceal an attack from intrusion detection systems.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 215

15) Suspicious Activity Followed By Integrity Attack. This event can be generated when an event that involves a Cloaking method is reported, followed by an Integrity attack. The term "cloaking" applies to any technique that attempts to conceal an attack from intrusion detection systems.

16) Sustained Availability Attack (Focused DoS Attack). This event can be generated when two or more types of Availability attacks are seen from a single source IP address targeted at a single destination IP address.

17) Sustained Confidentiality Attack (Focused Information Gathering Attack). This event can be generated when two or more types of Confidentiality attacks are seen from a single source IP address targeted at a single destination IP address.

18) Sustained Integrity Attack (Focused Break-in Attempt). This event can be generated when two or more types of Integrity attacks are seen from a single source IP address targeted at a single destination IP address.

19) Web Scan. This event can be generated when multiple Web-related attacks targeted against a Web server are detected within a certain interval. By examining features of the Web-related attacks such as the sequence of URLs being probed, it can be possible to identify the use of specific Web scanning tools such as Whisker.

Additional rules may be employed without departing from the scope and spirit of the present invention. Further details of routine **770** will be discussed in further detail below with respect to Figure **14**. However, it is noted that routine **770** as illustrated in Figure **14** only covers the application of one rule. The exemplary rule illustrated in Figure **14** is the rule corresponding to the "Attack From Attacked Host" (AFAH) correlation event listed above. The attack from attacked host scenario will also be described in further detail below with respect to Figures **5D** through **5F**.

If the inquiry to decision routine **770** is negative, then the "no" branch is followed to decision routine **785**. If the inquiry to decision routine **770** is positive,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 216

then the "yes" branch is followed to step **775** in which a mature event message is generated and forwarded to an output device such as the event collector **24**. In step **775**, the event reporter **660** receives an indication that the correlation event is mature and then the event reporter **660** forwards this message to the event collector **24**.

In step **780**, a correlation event update notification is sent to the output device when a raw event is added to a correlation event that is already mature. In this step, the event reporter forwards the correlation event update to event collector **24** which, in turn, updates the representation of the correlation event in the Event Database **26** and forwards this information to the console **30** where it may be viewed by a user. This allows the user to be notified when additional raw events occur that are part of an ongoing security incident (i.e., a mature correlation event).

Next, in decision routine **785**, it is determined whether any mature correlation events have stopped occurring. Further details of decision routine **785** will be discussed below with respect to Figure **15**.

If the inquiry to decision routine **785** is negative, then the "no" branch is followed to step **795**. If the inquiry to decision step **785** is positive, then the "yes" branch is followed to step **790** in which a message is sent indicating that a correlation event has stopped occurring. This message can be forwarded from the event reporter **660** to the event collector **24**. In turn, the event collector **24** would update the representation of the now-concluded correlation event in the event database **26** and then forward this message to the console **30**.

In step **795**, the oldest raw events and immature correlation events in the memory management list may be erased. Because the fusion engine **22** has a limited amount of memory, it is necessary to keep the memory filled with raw events that are the most likely to become mature. The fusion engine has several memory usage monitoring devices such as the memory management list **640**, the raw event tracking index **645**, and the mature event list **650**. In one exemplary embodiment, the memory usage monitoring devices of the fusion engine

IPR2024-00027
CrowdStrike Exhibit 1002 Page 217

determine how much memory is available and when the memory is close to being filled to capacity, the memory usage monitoring devices will erase the oldest existing stored raw events and immature correlation events in order to increase available memory. Raw events that are included within mature correlation events are removed from the memory management list **640** but are not erased. Whenever a raw event is deleted, the raw event tracking index **645** is used to locate any immature correlation events that contain the raw event, and the raw event is removed from those immature correlation events. When a raw event is removed from an immature correlation event and the immature correlation event then contains no other raw events, the immature correlation event is also erased.

Referring now to Figure **9**, this Figure illustrates the computer-implemented process for routine **720** of Figure **7** which identifies the type of raw event and assigns each raw event to a corresponding event type object of the classifier **615**. Routine **720** begins with step **910** where each raw event is matched with a corresponding event type in the classifier **615**. Next, in step **915**, the time stamp of each raw event is identified. In step **920**, each raw event is added to the memory management list **640** based upon the time stamp identified in step **915**. The entries in this list are typically maintained in order by timestamp to facilitate locating the oldest events during the memory cleanup processing described above In step **925**, each raw event is stored in the high speed memory device associated with its event type object as contained in the classifier **615**. Next, in step **930**, each event type object receiving a raw event is added to the raw event tracking index **645**. That is, typically, each software component of the fusion engine registers itself with the raw event tracking index **645** upon receiving a raw event. In this way, when a raw event is determined to be deleted from the system, the raw event tracking list **645** can be used to identify the location of the raw event references that need to be erased. After step **930**, the process returns back to decision step **725** of Figure **7**.

Figure **10** illustrates the computer-implemented process for routine **730** of Figure **7** in which parameters of each raw event are compared with the context or knowledge base database **630**. Also in this routine, additional parameters are

IPR2024-00027
CrowdStrike Exhibit 1002 Page 218

assigned to each raw event based upon this comparison with the context database **630**. As noted above, the context database **630** can comprise environmental information that may be helpful in evaluating the importance of a raw event. For example, the context database **630** can comprise vulnerability information about machines or computers within a network, the relative location of a computer or detector based upon predetermined zones, and information relating to historical event frequency.

The vulnerability information of the context database **630** is usually derived from scans made across a network to determine the relative security risk that may be present at one or more machines that make up a network. A tool that analyzes historical raw event logs for the network being monitored by the fusion engine **22** typically derives the historical event frequency information of the context database **630**. This tool typically calculates average event frequencies for groups of events sharing the same raw event type, source internet protocol address, and destination internet protocol address, though other approaches to grouping raw events for the purpose of calculating average event frequencies could be used are within the scope of the present invention. The zone definitions of the context database **630** are usually derived by categorizing parts of a network as they relate to the entire network. For example, an internal zone and demilitarized zone (DMZ) may be defined such that the internal zone includes the internet protocol network addresses of the networks that should not be accessible from the Internet, and the DMZ zone includes the internet protocol network addresses of the networks that are accessible from the Internet. These zones would be defined as appropriate for the specific network being monitored by the fusion engine **22**.

Routine **730** is typically performed by the CoBRA processor **625**. The CoBRA processor **625** typically examines each raw event and compares it to the context database **630**. More specifically, in step **1010** (the first step of Routine **730**) a CoBRA vulnerability status **504** is assigned for each raw event based upon destination internet protocol address information and a comparison with the context database **630**. In one exemplary embodiment, the vulnerability value

IPR2024-00027
CrowdStrike Exhibit 1002 Page 219

assigned can be any one of the following: believed vulnerable; believed not vulnerable; and unknown.

Next, in step **1015**, a historical frequency value **506** is assigned for each raw event based upon another comparison with the context database **630**. This value can be a number of events per unit time, such as events per day, or a mathematically related value, such as an average time between events. The historical event frequency value typically indicates how frequently raw events of a particular type from a particular source machine to a particular destination machine are seen on the network being monitored by the fusion engine **22**. Historical frequency data is used by the fusion engine to aid in distinguishing events caused by normal non-malicious network activity from those caused by unusual and malicious activity.

In step **1020**, a source zone **508** value is assigned to each raw event based upon the source internet protocol address of the raw event and a comparison with the context database **630**. In step **1025**, a destination zone **510** value is assigned to each raw event based upon the destination internet protocol address of each raw event and a comparison with the context database **630**.

In step **1030**, a sensor zone **512** value is assigned to each raw event based upon the sensor internet protocol address and a comparison with the context database **630**. The sensor zone value can comprise the internet protocol address of the sensor or detector of an intrusion detection system that detected the suspicious computer activity and generated the raw event. After step **1030**, the process returns to routine **735** of Figure **7**.

Referring now to Figure **11**, this Figure illustrates the computer-implemented process for routine **735** of Figure **7**, which can adjust the priority status or leave an original priority status of a raw event intact based upon the CoBRA-assigned context parameters or detector-assigned type parameters or both. Routine **735** is another core function of the CoBRA processor **625**. This routine enables the fusion engine to rank raw events based upon their importance to a network or a computer being protected. In this way, the security

IPR2024-00027
CrowdStrike Exhibit 1002 Page 220

administrator can monitor computer security events more efficiently and effectively, since the important computer security events will have a higher ranking and priority relative to other lower-level security events.

The present invention can employ user-defined attributes for those events and parts of a network that are most important to a user. For example, the zone definitions that form part of the context database **630** can be supplied by the user. In one exemplary embodiment, an internal zone and a so-called demilitarized zone (DMZ) of the monitored network can be established by the user. Rather than being explicitly defined by the user, an external zone can be any IP address that doesn't fall within an explicitly defined zone or zones supplied by the user. The present invention is not limited to these types of zones and can include other types of zones such as a business partner zone, for example. Those skilled in the art will appreciate that the present invention can be designed to associate Internet protocol addresses with any number of zones defined by a user.

As noted above, each raw event comprises a priority status parameter **535** that was assigned to it by a detector within an intrusion detection system. In one exemplary embodiment, the priority status parameter can comprise any one of the following three values: 1, 2, or 3. The highest priority status value in the exemplary embodiment is typically the number 1. Meanwhile, the lowest priority status in the exemplary embodiment is typically the value of 3. The mid-range priority value is typically the number 2. Adjustments of the priority status values for each raw event are necessary since the priority status values assigned by the detectors typically are very conservative in nature. That is, raw events are typically the result of simple processing techniques that must occur at the detector level in order to maintain high network traffic speeds.

Therefore, the priority status values coming from the detector level of conventional intrusion detection systems typically are defined as appropriate for the worst-case scenario that could apply for each event type. For example, in the exemplary embodiment, if a given type of raw event could have an actual priority of 1, 2, or 3 depending on the circumstances that apply on a given network, the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 221

detector would typically assign the worst-case priority (denoted as one (1) in the exemplary embodiment) to all events of this type. Whenever the CoBRA processor **625** modifies the value of the priority status **535**, it does so only after storing the both the original detector-assigned priority status and the updated or CoBRA-adjusted priority parameter in priority status **535**. That is, in the exemplary embodiment, after CoBRA processing and if priority is adjusted, the adjusted priority status **535** will contain two values: the original detector-assigned priority status and the CoBRA-adjusted priority status.

The fusion engine **22** permits the ranking of raw events based upon the environmental conditions or surroundings in which a raw event is generated. In this way, the security network administrator will only be presented with the computer security events that are most important to the network or computer being monitored. The present invention is not limited to the priority status scale illustrated. That is, the present invention is not limited to a priority scale that spans between 1 and 3 where one is designated as the highest priority. Other ranges or values and increments between values are not beyond the scope of the present invention. Those skilled in the art will appreciate that more complex scales can be generated to further reduce the possibility of ranking an unimportant raw event over an important raw event.

Routine **735** begins with step **1110**, in which it is determined whether a target of a raw event is resistant to the computer attack. This determination is made based on the CoBRA vulnerability status **504** value of the raw event previously established by step **1010** of procedure **730** described in Figure 10. If the inquiry to decision step **1110** is negative, then the "no" branch is followed where the process continues to step **1210** of Figure **12**. If the inquiry to decision step **1110** is positive, then the "yes" branch is followed to step **1115**. In step **1115**, the raw event is compared to vulnerability-adjustable event types stored in a list in the context database **630**. These vulnerability-adjustable event types stored in the context database **630** are events identified by either a user or a system for which the assessment of a machine's vulnerability status is believed trustworthy

IPR2024-00027
CrowdStrike Exhibit 1002 Page 222

and for which therefore it is allowed to adjust priority based on vulnerability status information.

Alternatively, in another embodiment (not illustrated) the context database **630** can identify those raw event types for which a user or system does not believe the assessment of vulnerability status to be trustworthy, and the assessment of vulnerability status can be deemed trustworthy for all other event types. In this way, raw events that are not desired to be adjusted with respect to their priority status can be identified so that the CoBRA processor **625** will not reduce the priority of such raw events. In another alternative exemplary embodiment (not shown), the context database **630** can also contain both types of lists. That is, the context database **630** can comprise a list of raw event types that are permitted to have the priority status to be adjusted and a list containing raw event types that are not permitted to have the priority status adjusted. In this case a conflict resolution rule must also be established, so that if a particular event type appears in both lists, it is well-defined which entry will take precedence. Those skilled in the art will appreciate that other configurations of lists are not beyond the scope of the present invention.

Next, in decision step **1120**, it is determined whether a match exists with the stored vulnerability-adjustable events of the context database **630**. If the inquiry to decision step **1120** is negative, then the "no" branch is followed to step **1135**. If the inquiry to decision step **1120** is positive, then the "yes" branch is followed to decision step **1125**.

In decision step **1125**, it is determined whether the current raw event being processed is at its lowest priority status. In other words, if the current raw event being processed has an exemplary priority status value of 3, then it is recognized that its priority cannot further be adjusted. Therefore, if the inquiry to decision step **1125** is positive, then the "yes" branch is followed to step **1135**. If the inquiry to decision step **1125** is negative, then the "no" branch is followed to step **1130**, in which the priority status **535** of the current raw event is reduced and the reason for the change in priority status **535** is recorded in the raw event. For

IPR2024-00027
CrowdStrike Exhibit 1002 Page 223

example, in the exemplary embodiment, if a raw event has an original priority status value of a 1, and if the CoBRA processor **625** determines that the raw event is not believed to be vulnerable, then it will adjust the original priority status value of 1 to a lower value such as the value of 2 (the mid-range priority status value). The reason for changing the priority status value is recorded in the priority change reason **516** parameter of the raw event so that it can be determined at the console **30** why a particular raw event was assigned a reduced priority. In one exemplary embodiment, the reason for changing priority status of a raw event can comprise a text string.

In step **1135**, each raw event is compared to frequency-adjustable event types stored in a list in the context database **630**. Similar to the vulnerability-adjustable event types discussed above, the frequency-adjustable event types can comprise those raw event types for which a high historical event frequency between a given pair of machines is seen as a reliable indicator of non-maliciousness for the network or computer being monitored by the fusion engine **22**. Alternatively, also similar to the vulnerability-adjustable event types discussed above, in another exemplary embodiment (not shown) the context database **630** could instead comprise a list that identifies those raw event types for which a high historical event frequency between a given pair of machines for a network or computer being monitored by the fusion engine **22** is not seen as a reliable indicator of non-maliciousness, and historical event frequency can then be considered a trustworthy indicator of non-maliciousness for all other event types. In such a scenario, the list would identify those raw events where it is undesirable to adjust the priority status thereof based on historical event frequency. Alternatively, in yet another exemplary embodiment (not shown), the context database **630** could also comprise both types of lists where one list would identify those raw event types for which frequency-based priority adjustment is allowed and the other would identify those raw event types for which frequency-based priority adjustment is not allowed. In this case a conflict resolution rule must also be established, so that if a particular event type appears in both lists, it is well-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 224

defined which entry will take precedence. Those skilled in the art will appreciate that other configurations of lists are not beyond the scope of the present invention.

Following step **1135**, in decision step **1145**, it is determined whether a match exists with the stored frequency-adjustable event types. If the inquiry to decision step **1145** is negative, then the "no" branch is followed to step **1210** of Figure **12**. If the inquiry to decision step **1145** is positive, then the "yes" branch is followed to decision step **1150**.

In decision step **1150**, it is determined whether historical frequency information exists for the current raw event being evaluated. This determination is made based on the historical frequency value **506** of the raw event previously established by step **1015** of procedure **730** described in Figure 10. In other words, some raw events may be of a type, source, and destination that was not seen in the historical data analyzed to produce the historical frequency information. If the inquiry to decision step **1150** is negative, then the "no" branch is followed to step **1210** of Figure **12**. If the inquiry to decision step **1150** is positive, then the "yes" branch is followed to decision step **1155**.

In decision step **1155**, it is determined whether the historical frequency for the current raw event being evaluated is greater than a frequent event threshold. In other words, in this decision step, it is determined whether a raw event is of a type that occurs frequently enough between a specific source and destination that it can be considered to be likely to be a non-malicious event. The frequent event threshold may be a value that corresponds to an average number of events per unit time, such as per day. However, other mathematically related forms of this value, such as the average time between events, could also be used and are not beyond the scope of the present invention. If the current raw event being processed has an historical event frequency that is greater than the threshold, then it is considered to be a frequent event and likely to be non-malicious.

If it is determined to be a frequent raw event, then its priority status can be lowered. However, if the current raw event being processed has been seen less frequently on the network, then it is not considered to be a frequent raw event and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 225

an adjustment to its priority status based on historical event frequency is considered undesirable. Therefore, if the inquiry to decision step **1155** is negative (meaning that events like the current raw event being processed have not been seen frequently on the network monitored by the fusion engine **22**), then the "no" branch is followed to step **1210** of Figure **12**. If the inquiry to decision step **1155** is positive (meaning that events like the current raw event being processed have been seen frequently on the network monitored by the fusion engine **22**), then the "yes" branch is followed to decision step **1160**.

In decision step **1160**, it is determined whether the raw event being processed is at its lowest priority status. If the inquiry to decision step **1160** is positive, then the "yes" branch is followed to step **1210**, Figure **12**. If the inquiry to decision step **1160** is negative, then the "no" branch is followed to step **1165**, in which the priority of the current raw event is reduced and the reason for changing the status of the priority of the current raw event is recorded. The reason is typically recorded as being that the raw event being evaluated occurs frequently.

The process then continues to Figure **12**. Figure **12** illustrates a second portion of the computer-implemented process for routine **735** of Figure **7**, in which the CoBRA processor **625** adjusts the priority status or leaves an original priority status intact based upon the CoBRA assigned context parameters or detector-assigned type parameters of the raw event.

In step **1210**, the raw event is compared to zone-adjustable event types stored in a list of the context database **630**. Similar the vulnerability-adjustable event types and frequency-adjustable event types discussed above, the zone-adjustable event types are raw event types that may be defined by a network security administrator that are deemed to present low risk to the network or computer being monitored by the fusion engine **22** if they occur internally (that is, if both the source Internet protocol address and destination Internet protocol address in the raw event are located in networks defined in the context database **630** as belonging to the internal zone). However, in an alternative embodiment (not shown), the context database **630** may instead comprise a list that identifies

IPR2024-00027
CrowdStrike Exhibit 1002 Page 226

raw event types that cannot be deemed to present low risk to the network or computer being monitored by the fusion engine **22** based solely on the zone(s) in which the source and destination are located.

In such an embodiment, event types other than those listed are deemed to present low risk to the monitored computer or network if they occur internally. In a further alternative embodiment (not shown), the context database **630** may also comprise both types of list: one list identifying raw event types that cannot be deemed to present low risk based solely on the source and destination zones, where the priority status thereof should not be adjusted, and a second list of raw events that are deemed to present low risk when seen internally, and where priority status should be adjusted such that these events will have a lower priority status. In this case a conflict resolution rule must also be established, so that if a particular event type appears in both lists, it is well-defined which entry will take precedence.

In decision step **1215**, it is determined whether a match exists with the stored zone-adjustable event types in the context database **630**. If the inquiry to decision step **1215** is negative, then the "no" branch is followed back to routine **740** of Figure **7**. If the inquiry to decision step **1215** is positive, then the "yes" branch is followed to decision step **1220**.

In decision step **1220**, it is determined whether the source zone and destination zone of the current raw event being processed are both internal relative to the network or computer being monitored by the fusion engine **22**. This determination is made by examining the values of the source zone parameter **508** and destination zone parameter **510** of the raw event assigned by steps **1020** and **1025**, respectively, of routine **730** shown in Figure 10. For many event types, raw events classified as being internal are less of a threat to a network of computers being monitored compared to an event that may be external to a network or computer being monitored by the fusion engine **22**.

Therefore, for internal events, it may be desirable to lower the priority status of such raw events. Conversely for raw events for which either the source

IPR2024-00027
CrowdStrike Exhibit 1002 Page 227

or destination Internet protocol address is either in the DMZ zone or not in any defined zone (and therefore considered external), it may be desirable to keep the priority status of a raw event that was assigned to it by the detectors in the intrusion detection system. If the inquiry to decision step **1220** is negative, then the "no" branch is followed back to routine **740** if Figure **7**. If the inquiry to decision step **1220** is positive, then the "yes" branch is followed to decision step **1225**.

In decision step **1225**, it is determined whether the current raw event is at its lowest priority status. If the inquiry to decision step **1225** is positive, then the "yes' branch is followed back to routine **740**, Figure **7**. If the inquiry to decision step **1225** is negative, then the "no" branch is followed to step **1230**, in which the priority status of the current raw event is reduced and the reason for change in the priority status of the raw event is recorded. Typically, the reason in step **1230** will indicate that the priority status of the current raw event was lowered because it comprises an internal attack. The process then returns to routine **740** of Figure **7**.

The present invention is also not limited to the technique of reducing priority status values. In other words, the present invention can also comprise a scale where values are increased in order to reflect either reduced priority or increased priority. Those skilled in the art will appreciate that any number of risk adjustment schemes can be utilized and not depart from the scope and spirit of the present invention.

Referring now to Figure **13**, this Figure illustrates the computer-implemented process for routine **740** of Figure **7** in which raw events are associated with predetermined correlation rules based upon the event type parameter **555**. In this routine, the classifier **615** may identify one or more correlation rules **620** that should process each given raw event. Step **1310** is the first step of routine **740**, in which all lists containing the raw events are updated to reflect any CoBRA processing changes. In other words, all objects in the exemplary object-oriented architecture containing the raw events that were

IPR2024-00027
CrowdStrike Exhibit 1002 Page 228

adjusted by the CoBRA processor **625** are updated to reflect any adjustments in priority status.

Next, in step **1315**, the raw events are forwarded to the correlation rules that apply to the raw event type parameter **555**. More specifically, in step **1315**, the definition of each correlation rule **620** includes a list of the raw event categories that are of interest to it. The raw event types included in each raw event category are defined in the raw event classification database **635**. Therefore, the list of raw event types of interest to a correlation rule **620** is the union of the category-specific lists of raw event types for the categories of interest to the rule, where each category-specific list of raw event types is defined by the raw event classification database **635**. The category-specific lists of raw event types are stored in the classifier **615**, which is initialized based on the contents of the raw event classification database **635**.

When the controller **655** loads a correlation rule **620** during system initialization, it associates the rule with all the event types included in the event categories of interest to the rule (determined as described in the previous paragraph) by adding the rule to a list of interested rules maintained within each such event type. Thus, after initialization, each event type includes a list of all of the correlation rules **620** that are interested in events of its type. As each raw event is received, the event reader **600** determines which correlation rules **620** should process it by retrieving the raw event's event type and then retrieving the event type's list of interested rules. Having determined the set of correlation rules **620** that should process the raw event, the process then returns to step **745** of Figure **7**.

Referring now to Figure **14**, this Figure illustrates the computer-implemented process for routine **770** of Figure **7**, which determines if a currently immature correlation event to which the current processed raw event has been added meets or satisfies the maturity criteria of a corresponding rule **620**. The process described here is for an exemplary event type, Attack From Attacked Host (AFAH), rather than being generic. However, given this processing description and the descriptions of exemplary correlation event types presented

IPR2024-00027
CrowdStrike Exhibit 1002 Page 229

earlier, it should be apparent to those skilled in the art how similar processing could be used to recognize the occurrence of each of the described exemplary event types. As noted above, each rule **620** may be implemented as a rule object in an object-oriented computer architecture. As should be clear to those skilled in the art from the previously described processing of step 1315 of Figure 13, a single raw event may be processed by multiple correlation rule objects.

Though not depicted in Figure 7 or Figure 14, the processing of steps **745** through **780** of Figure 7 (including the routine **770** processing described in Figure 14) can be performed twice for the current processed raw event in the case of the exemplary AFAH correlation event. In one exemplary embodiment, the raw event is processed once to consider it as an inbound attack only if the raw event is an integrity attack, and is always processed to consider it as an outbound attack. When the raw event is considered as an inbound attack by steps **745** through **780**, step **745** uses the raw event's destination Internet protocol address as the lookup key when attempting to retrieve a corresponding AFAH correlation event from the correlation event cache **665** (as described earlier in the discussion of step **745** processing).

When the raw event is considered as an outbound attack by steps **745** through **780**, step **745** uses the raw event's source Internet protocol address as the lookup key when attempting to retrieve a corresponding AFAH correlation event from the correlation event cache **665** (as also described earlier in the discussion of step **745** processing). This "double processing" of the raw event is a unique aspect of the exemplary AFAH correlation event relative to other correlation events that can be processed by the fusion engine **22**. For all of the other exemplary correlation event types described earlier, the processing of steps **745** through **780** is performed once, as should be apparent to those skilled in the art based on the descriptions of the exemplary correlation event types.

Referring again to Figure 14, step **1410** is the first step of routine **770**, in which the event type object of the classifier **615** for the current raw event being processed is added to the raw event tracking index **645**. Also, the correlation

IPR2024-00027
CrowdStrike Exhibit 1002 Page 230

event object corresponding to the current raw event object being processed is either added to the memory management list **640** (if it is a new correlation event that was just created in step **750** of Figure 7), or is moved to a new position in the memory management list if the timestamp of the current raw event has the most recent timestamp of all the raw events associated with the current correlation event.

In addition, the current correlation event object is added to the raw event tracking index **645** in association with the raw event. The event type object and correlation event object are added to the raw event tracking index **645** so that they can later be informed by the memory management processing if the raw event is erased from memory, so they can erase their own references to the raw event. The current correlation event is also added to the memory management list **640** so that when memory resources run low, the oldest events (some of which may be immature correlation events) can be deleted from the fusion engine **22**.

In decision step **1415**, it is determined whether the raw event is being considered as an inbound attack. This step distinguishes whether the current AFAH correlation event includes the current raw event as an inbound or outbound attack. If the inquiry to decision step **1415** is negative, then the raw event is being considered as an outbound attack and the "no" branch is followed to step **1425**. If the inquiry to decision step **1415** is positive, then the raw event being considered is an inbound attack and the "yes" branch is followed to step **1420**.

In decision step **1420**, it is determined whether the raw event being considered as an inbound attack ia an integrity attack and occurs earlier than at least one event in the outbound attacks list of the current correlation event. The raw event being processed is known to be an integrity attack since it was added to the inbound attack list of the correlation event during the processing of step As indicated in the description of the Attack From Attacked Host event included in the earlier list of exemplary correlation event types, the AFAH event can be generated when an Integrity attack is seen against a host followed by a Confidentiality, Integrity, or Availability attack originating from that host. As indicated in the discussion of Figure 13, when the controller **655** loads a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 231

correlation rule **620** during system initialization, it associates the rule with all of the event types included in the event categories of interest to the rule.

In the case of the AFAH rule, the event categories of interest are Confidentiality, Integrity, and Availability attacks. The AFAH rule is therefore associated with all event types defined by the raw event classification database **635** as belonging to one of these three categories. Therefore, any raw event whose event type belongs to one of these three categories can be forwarded to routine **770** for processing. Since the definition of the AFAH event requires inbound attacks to be Integrity attacks, and some of the raw events forwarded to routine **770** can instead be Confidentiality or Availability attacks, the present decision step **1420** must verify that the raw event being considered as an inbound attack is an Integrity attack.

A further consideration results from the fact that the fusion engine **22** can receive raw events generated by multiple detectors, leading to the possibility that raw events can be received in non-chronological order (that is, a raw event with a later timestamp can be received before a raw event with an earlier timestamp). For this reason, routine **770** cannot assume that raw events will be received in chronological order, and the present decision step **1420** therefore determines whether the current raw event occurs earlier than at least one event in the outbound attacks list of the current correlation event. If the inquiry to decision step **1420** is negative, then the current correlation event is deemed not mature and the "no" branch is followed back to routine **785** of Figure **7**. If the inquiry to decision step **1420** is positive, then the current correlation event is deemed mature and the "yes" branch is followed to step **1427**.

In decision step **1425**, it is determined whether the raw event being considered as an outbound attack occurs later than at least one event in the inbound attacks list of the current correlation event. Unlike decision step **1420**, it is not necessary to determine whether the current raw event belongs to a particular category because (as described in the discussion of step **1420**) every raw event forwarded to routine **770** will be either a Confidentiality, Integrity, or Availability attack and will therefore meet the criteria for inclusion as an outbound attack in

IPR2024-00027
CrowdStrike Exhibit 1002 Page 232

any AFAH event. If the inquiry to decision step **1425** is negative, then the current correlation event is deemed not mature and the "no" branch is followed back to routine **785** of Figure **7**. If the inquiry to decision step **1425** is positive, then the current correlation event is deemed mature and the "yes" branch is followed to step **1427**.

In step **1427**, any outbound attacks in the current correlation event that occur earlier than the earliest inbound attack of the correlation event are removed from the list of outbound attacks. This is done because the definition of the AFAH correlation event requires that each outbound attack included in a mature AFAH correlation event must be preceded by at least one inbound attack.

In step **1430**, the correlation event is removed from the memory management list **640** so that the correlation event will not be subject to being erased by the memory management mechanisms. In this way, the correlation event that is removed will not be deleted from the fusion engine **22** since the correlation event is now deemed to be mature.

In step **1435**, the update time of the correlation event can be set to the most recent raw event's time stamp if the event source being read by the event reader **600** is either the event database **26** or the event log file **610**, in which case the fusion engine **22** is operating in a batch mode. Alternatively, the update time of the correlation event can be set to the current time of the system on which the fusion engine **22** is executing if the event source being read by the event reader **600** is the event collector **24**, in which case the fusion engine **22** is operating in a real-time mode.

In step **1440**, the correlation event is added to the mature event correlation list **650**. In step **1445**, the correlation event containing the two or more raw events is indicated as being mature by setting an internal parameter of the correlation event. The process then returns to step **775** of Figure **7**. In one exemplary embodiment (not shown), each correlation event may be assigned a priority status, similar to the priority status parameter **555** of raw events.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 233

Exemplary Rule Processing for Raw Event II Illustrated in Figures 5D, 5E, 5F

The following is the processing that would be carried out by an Attack From Attacked Host correlation rule **620** for raw event II as illustrated in Figures **5(d)**, **5(e)**, and **5(f)**. This discussion assumes that raw events I and II are both types of Integrity attacks and therefore qualify as inbound attacks according to the definition of the AFAH event, that raw event I occurs before raw event II, and that raw event II occurs before raw event III.

Referring back to Figure **7**, in step **745**, when raw event II is being considered as an inbound attack, its destination internet protocol address (3.3.3.3) would be used as a lookup key to retrieve an AFAH correlation event from the correlation event cache **665**. Assuming that in this case raw events are received in chronological order and therefore raw event III has not yet been processed by the fusion engine, there would be no AFAH correlation event indexed by the attacked internet protocol address 3.3.3.3, and therefore the "no" branch of decision step **745** would be taken and correlation event **513** would be created in step **750**. In step **755**, correlation event **513** would be stored in the correlation event cache **665**. In step **760**, raw event II would be associated with correlation event **513** by storing a reference to it in the inbound attacks list of correlation event **513**. In step **765** it would be determined that correlation event **513** is not already mature, so the "no" branch would be followed to step **770**.

Referring now to Figure **14**, in decision step **1415** the "yes" branch would be followed since raw event II is being considered as an inbound attack. In decision step **1420**, the "no" branch would be followed since there are no raw events in the outbound attacks list of the newly created correlation event **513**. To summarize this processing, when considered as an inbound event, raw event II is added to a newly created but still immature correlation event **513**.

Referring back to Figure **7**, in step **745**, when raw event II is being considered as an outbound attack, its source internet protocol address (2.2.2.2) would be used as a lookup key to retrieve an AFAH correlation event from the correlation event cache **665**. Assuming that in this case raw events are received in

IPR2024-00027
CrowdStrike Exhibit 1002 Page 234

chronological order and therefore raw event I has already been processed by the fusion engine, AFAH correlation event **511** would already be present in the correlation event cache **665** indexed by the attacked internet protocol address 2.2.2.2, and therefore the "yes" branch of decision step **745** would be taken to step **760**. In step **760**, raw event II would be associated with correlation event **511** by storing a reference to it in the outbound attacks list of correlation event **511**. In step **765** it would be determined that correlation event **511** is not already mature, so the "no" branch would be followed to step **770**.

Referring now to Figure **14**, in decision step **1415** the "no" branch would be followed since raw event II is being considered as an outbound attack. In decision step **1425**, the "yes" branch would be followed since the inbound attacks list of correlation event **511** already contains raw event I, and the timestamp of raw event II is later than that of raw event I. At this point, correlation event **511** has been determined to be mature and steps **1427** through **1445** would be followed to process the newly-mature correlation event **511**. To summarize this processing, when considered as an outbound attack, raw event II is added to existing correlation event **511** which becomes mature as a result.

To perform decision step **1425** in the above-described exemplary rule processing, the respective time stamps of the first-generated raw event I and the second-generated raw event II are compared. However, it is noted that since the raw events could originate from different detectors, there could be some variance in the time stamps provided for each raw event. That is, while the second-generated raw event II may occur after the first-generated raw event I, because of possible variances in the internal clocks of the detectors generating the raw events, it is foreseeable that the first-generated raw event I may have a later time stamp than the second-generated raw event II.

In other words, the internal clocks between respective detectors in neighboring intrusion detection systems may not be synchronized. In order to compensate for such a scenario, a tri-state comparison could be performed. That is, the fusion engine **22** and more specifically, the rules **620** may allow for the possibility that there may be some synchronization offsets so a determination can

IPR2024-00027
CrowdStrike Exhibit 1002 Page 235

be made if a first raw event came before another raw event. More specifically, when comparing the timestamps of two raw events generated by different detectors to determine whether one of the events precedes (or follows) the other, the result of the comparison can be yes, no, or maybe. The "maybe" result occurs when the timestamps of the two events are sufficiently close that uncertainty regarding the synchronization offsets of the two detectors makes it impossible to determine which event occurred first.

The fusion engine **22** could be configured to treat the "maybe" result either as a "yes" (in one configuration) or as a "no" (in an alternative configuration). In a preferred embodiment, the fusion engine **22** treats a "maybe" as a "yes" to maximize the chance that correlation event maturity criteria will be met (so that mature correlation events will be generated whenever it appears possible that their criteria might be met). When the fusion engine **22** compares the timestamps of two events generated by the same detector, then it can ignore any synchronization effects and perform a simple binary comparison between the timestamps of the two events.

Exemplary Computer-Implemented Process to Determine if Mature Correlation Events Timed Out

Referring now to Figure **15**, this Figure illustrates the computer-implemented process for routine **785** which determines whether any mature correlation events have stopped occurring. Step **1510** is the first step of routine **785** in which the current processing time is compared with the update times of the correlation events stored in the mature event list **650** (the update times of correlation events are set as described in step **1435** of figure **14**). For the purpose of this comparison, the definition of the current processing time depends on the mode in which the fusion engine **22** is operating. If it is operating in a batch mode (that is, the event source from which events are being read is either the event database **26** or the event log file **610**), then the current processing time is the timestamp of the last event that was read from the event source. Alternatively, if the fusion engine **22** is operating in real-time mode (that is, the event source from

IPR2024-00027
CrowdStrike Exhibit 1002 Page 236

which events are being read is the event collector **24**), then the current processing time is the current time of the system on which the fusion engine **22** is running.

In decision step **1515**, it is determined whether the difference between the current processing time and the update time of each correlation event exceeds a predetermined threshold. In other words, it is determined whether the mature correlation events contained within the mature event list **650** have become old or stale in that no computer activity or raw events have occurred for these correlation events over some period of time. If the inquiry to decision step **1515** is positive, then the "yes" branch is followed back to step **790** of Figure **7**. If the inquiry to decision step **1515** is negative, then the "no" branch is followed back to step **795** of Figure **7**.

It should be understood that the foregoing relates only to illustrative embodiments of the present invention, and that numerous changes may be made therein without departing from the spirit and scope of the invention as defined by the following claims.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 237

CLAIMS

What is claimed is:

1. A method for managing security information comprising the steps of:

receiving raw events from one or more data sources;

classifying the raw events;

storing the raw events;

assigning a ranking to each raw event;

identifying relationships between two or more raw events;

in response to identifying any relationships between two or more raw events, generating a mature correlation event message; and

displaying one or more mature correlation event messages on a console that describe relationships between raw events.

2. The method of claim 1, wherein each raw event comprises suspicious computer activity detected by one of an automated system and human observation.

3. The method of claim 1, wherein the step of receiving raw events from one or more data sources further comprises the step of receiving real-time raw events from one of intrusion detection system, a detector within an intrusion detection system, and a firewall.

4. The method of claim 1, wherein the step of receiving raw events from one or more data sources further comprises the step of receiving raw events from one of a file and database.

5. The method of claim 1, wherein the step of classifying the raw events further comprises the steps of:

identifying an event type parameter for each raw event;

comparing the event type parameter with an event type category of a list; and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 238

assigning each raw event to a corresponding event type category in the list.

6. The method of claim 1, wherein the step of assigning a ranking to each raw event further comprises the steps of:

comparing parameters of each raw event with information in a database; and

assigning additional parameters to each raw event relating to the environment of the raw event.

7. The method of claim 6, wherein the additional parameters comprise one of a priority status, a vulnerability status, a historical frequency value, a source zone value, a destination zone value, a detector zone value, and a text string.

8. The method of claim 1, wherein the step of assigning a ranking to each raw event further comprises the steps of:

identifying a priority status parameter of a raw event;

comparing each raw event to information contained in a context database;

changing the priority status parameter of a respective raw event if a match occurs in response to the comparison step; and

leaving the priority status in tact if a match does not occur in response to the comparison step.

9. The method of claim 1, wherein the step of identifying relationships between two or more raw events further comprises the steps of:

associating each raw event with a rule which corresponds with a type parameter of a raw event; and

applying one or more rules to groups of raw events having the same type parameter; and

determining if a computer attack or security breach has occurred based upon successful application of a rule.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 239

10. The method of claim 1, wherein the step of storing raw events further comprises the step of storing each raw event in a high speed memory device comprising random access memory (RAM).

11. The method of claim 1, further comprising the step of determining the intent of a computer attack based upon the type of mature correlation event generated.

12. The method of claim 1, further comprising the steps of:

creating a memory management list;

identifying a time stamp for each raw event; and

adding each raw event to the memory management list.

13. The method of claim 1, further comprising the step of creating a raw event tracking index that identifies one or more software components that are monitoring one or more raw events.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 240

14. A method for determining relationships between two or more computer events, comprising the steps of:

receiving a plurality of raw events having a first set of parameters;

creating raw event storage areas based upon information received from a raw event classification database;

storing each event in an event storage area based upon an event type parameter;

comparing each raw event to data contained in a context database;

adjusting a priority parameter or leaving the priority parameter in tact for each raw event in response to the comparison to the context database;

associate each raw event with a correlation event;

applying one or more rules to each event based upon the correlation event association; and

generating a mature correlation event message in response to a successful application of a rule.

15. The method of claim 14, wherein each raw event comprises suspicious computer activity detected by one of an automated system and human observation.

16. The method of claim 14, wherein the context database comprises any one of vulnerability values, computer event frequency values, and source and destination zone values.

17. The method of claim 14, wherein the raw event classification database comprises tables that include information that categorizes raw events based on any one of the following: how a raw event may impact one or more target computers, how many target computers that may be affected by a raw event, and how respective raw events gain access to one or more target computers.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 241

18. A security management system comprising:

a plurality of data sources;

an event collector linked to the plurality of data sources;

a fusion engine linked to the event collector, said fusion engine identifying relationships between two or more raw events generated by the data sources; and

a console linked to the event collector for displaying any output generated by the fusion engine.

19. The security management system of claim 18, further comprising a detector, the detector running in a kernel mode of a computer and the fusion engine running in a user mode of the computer.

20. The security management system of claim 18, further comprising a detector chip, and the fusion engine comprising software running on a computer.

21. The security management system of claim 18, further comprising a detector board, and the fusion engine comprising software running on a computer.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 242

22. A fusion engine comprising:

a controller;

an event reader for receiving raw events;

a classifier linked to the event reader for classifying the received raw events;

a raw event classification database linked to the classifier;

a context based risk-adjustment processor linked to the classifier, for adjusting priorities of raw events;

a context database linked to the context based risk-adjustment processor; and

a rule database, for determining if relationships exist between two or more events.

23. The fusion engine of claim 22, further comprising an event reporter, a mature event list, a memory management list, and a raw event tracking index .

24. The fusion engine of claim 22, wherein the context database comprises any one of vulnerability values, computer event frequency values, and source and destination zone values.

25. The fusion engine of claim 22, wherein the raw event classification database comprises tables that include information that categorizes raw events based on any one of the following: how a raw event may impact one or more target computers, how many target computers that may be affected by a raw event, and how respective raw events gain access to one or more target computers.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 243

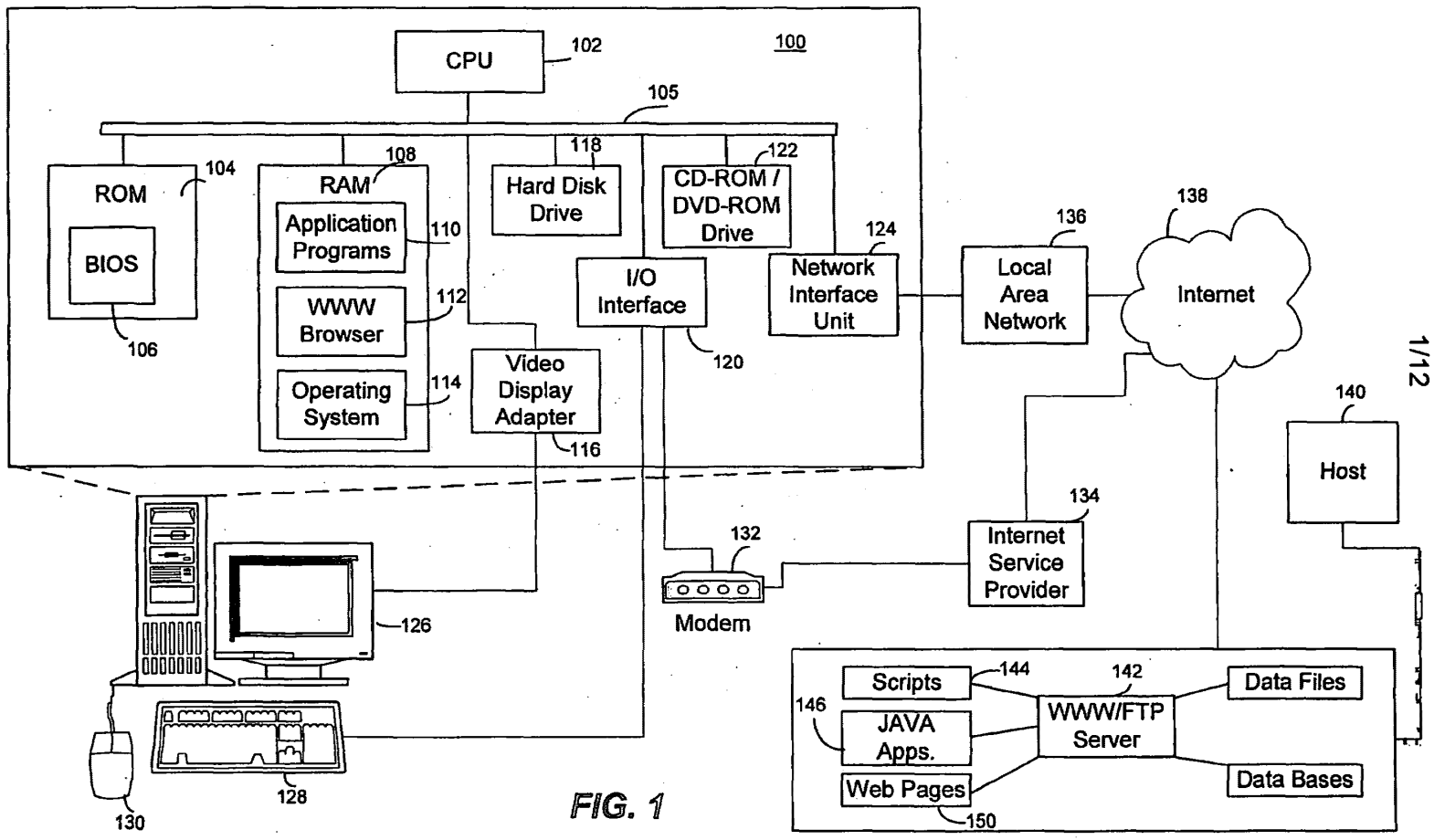

CPU
102
100
105
ROM
104
BIOS
106
RAM
108
Application Programs
110
WWW Browser
112
Operating System
114
Hard Disk Drive
118
CD-ROM / DVD-ROM Drive
122
I/O Interface
120
Network Interface Unit
124
Video Display Adapter
116
126
128
130
132
Modem
Local Area Network
136
Internet
138
Internet Service Provider
134
Host
140
Scripts
144
142
Data Files
146
JAVA Apps.
WWW/FTP Server
Web Pages
150
Data Bases

FIG. 1

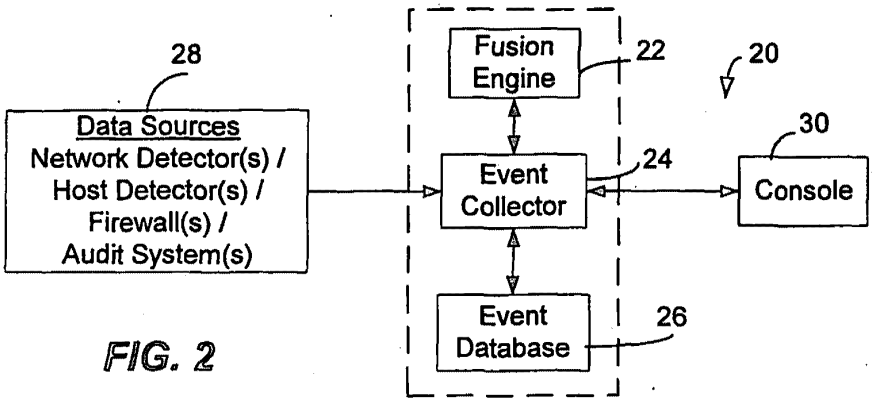


FIG. 2

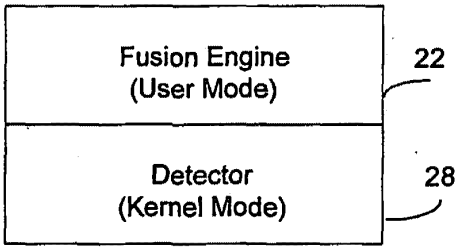


FIG. 3

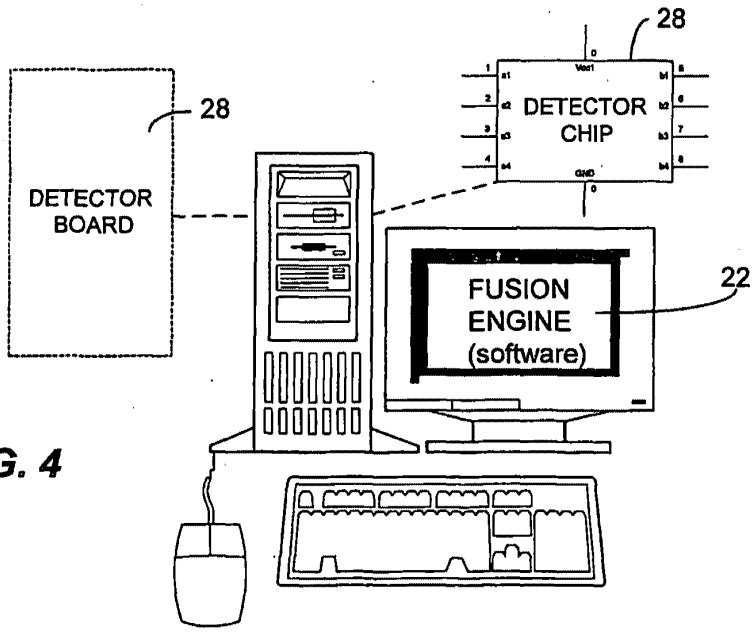


FIG. 4

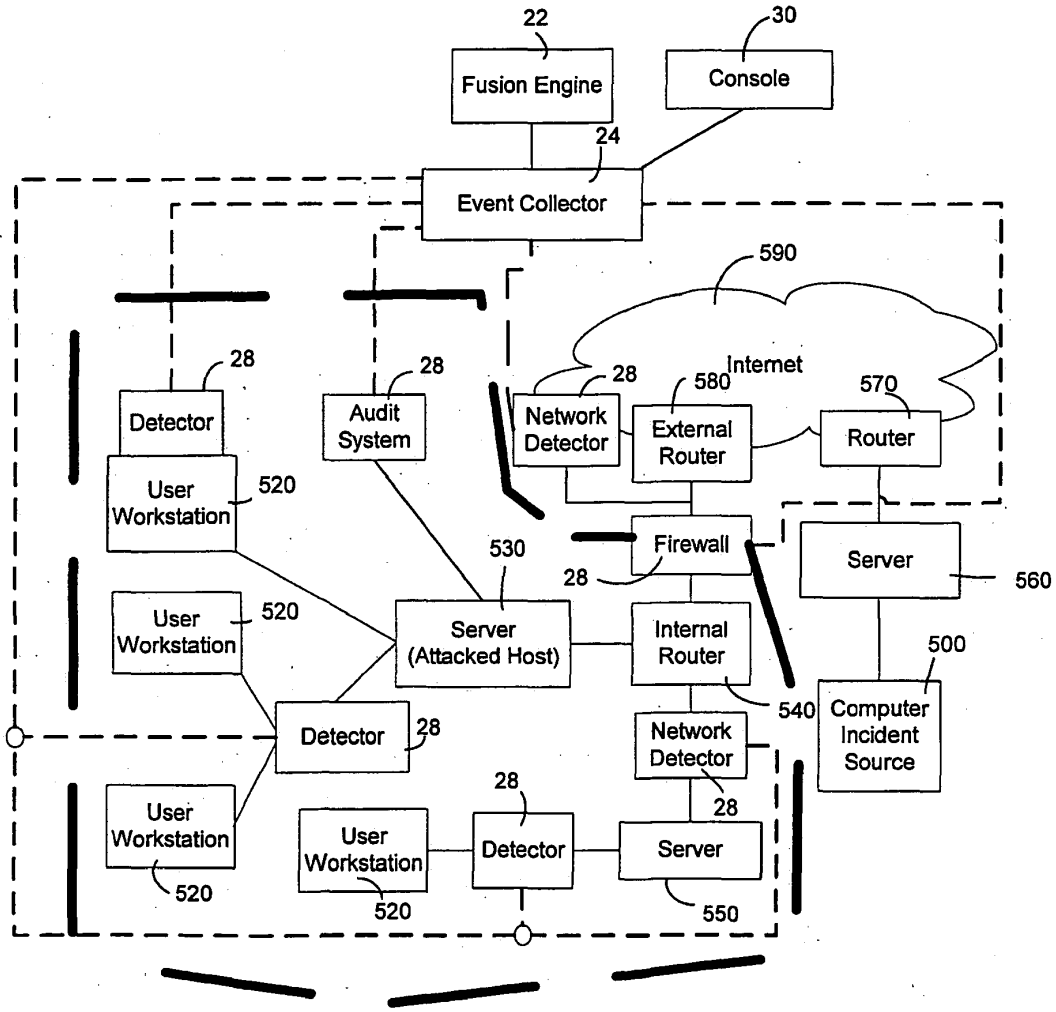

22
Fusion Engine
30
Console
24
Event Collector
590
Internet
28
Detector
28
Audit System
28
Network Detector
580
External Router
570
Router
User Workstation
520
Firewall
530
Server
560
28
User Workstation
520
Server (Attacked Host)
Internal Router
500
Detector
28
540
Computer Incident Source
Network Detector
28
28
User Workstation
520
User Workstation
520
Detector
Server
550

FIG. 5A

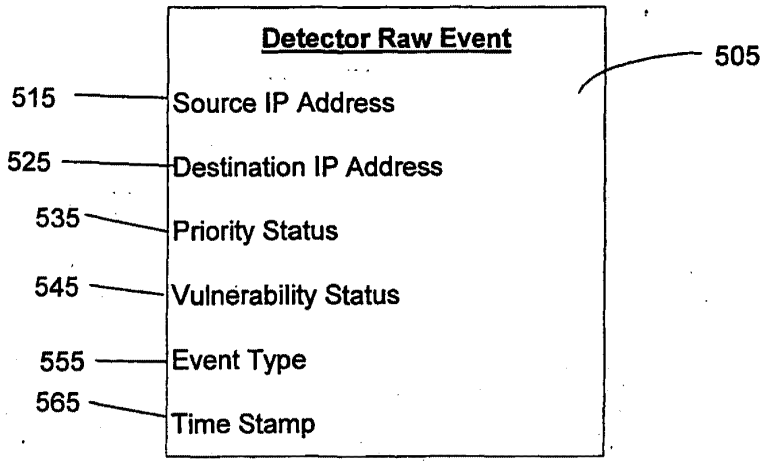

Detector Raw Event
505
515
Source IP Address
525
Destination IP Address
535
Priority Status
545
Vulnerability Status
555
Event Type
565
Time Stamp

FIG. 5B

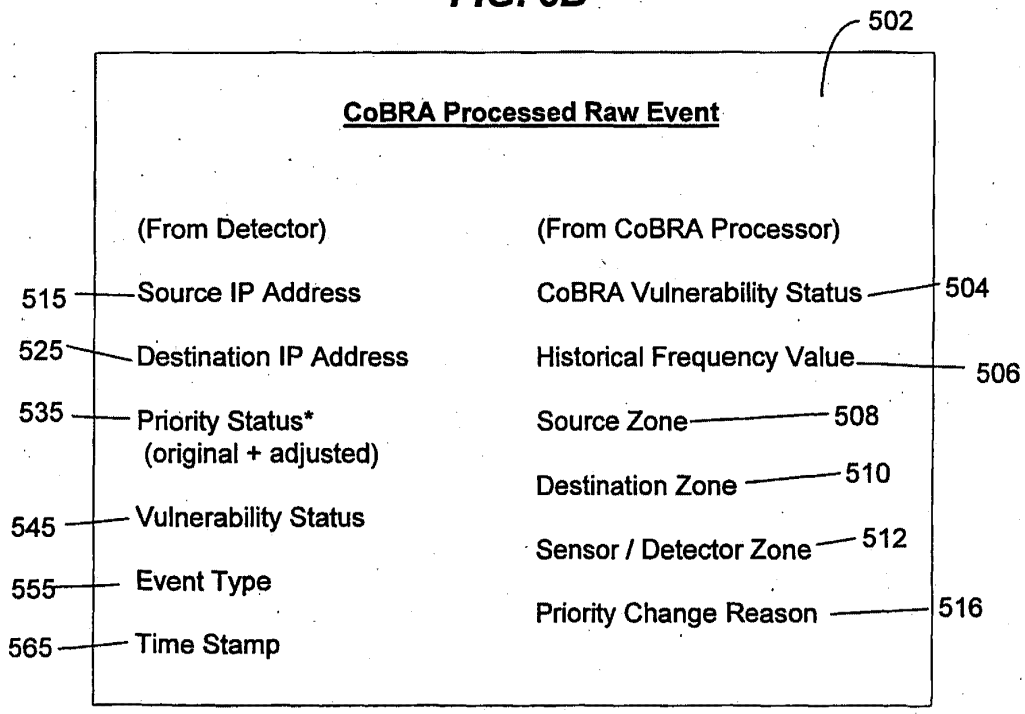

502
CoBRA Processed Raw Event
(From Detector)
(From CoBRA Processor)
515
Source IP Address
CoBRA Vulnerability Status
504
525
Destination IP Address
Historical Frequency Value
506
535
Priority Status*
(original + adjusted)
Source Zone
508
Destination Zone
510
545
Vulnerability Status
Sensor / Detector Zone
512
555
Event Type
Priority Change Reason
516
565
Time Stamp

FIG. 5C

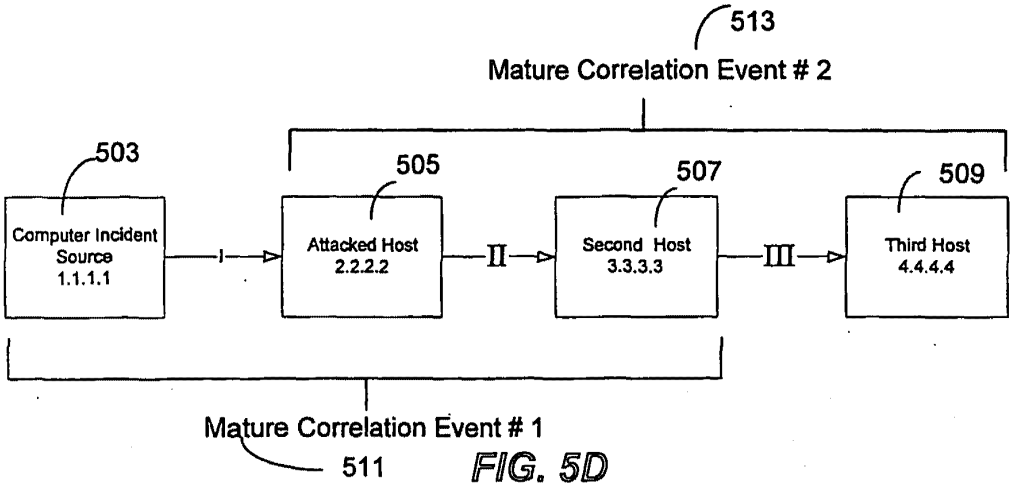


FIG. 5D

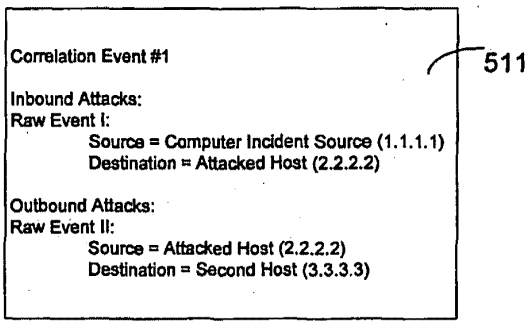


FIG. 5E

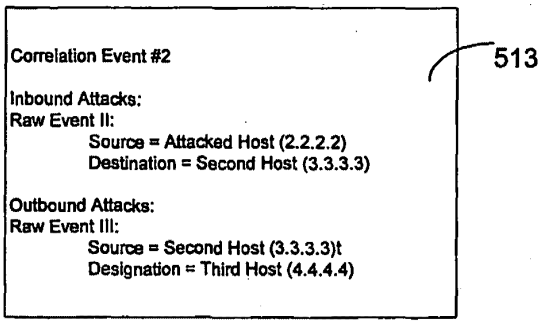


FIG. 5F

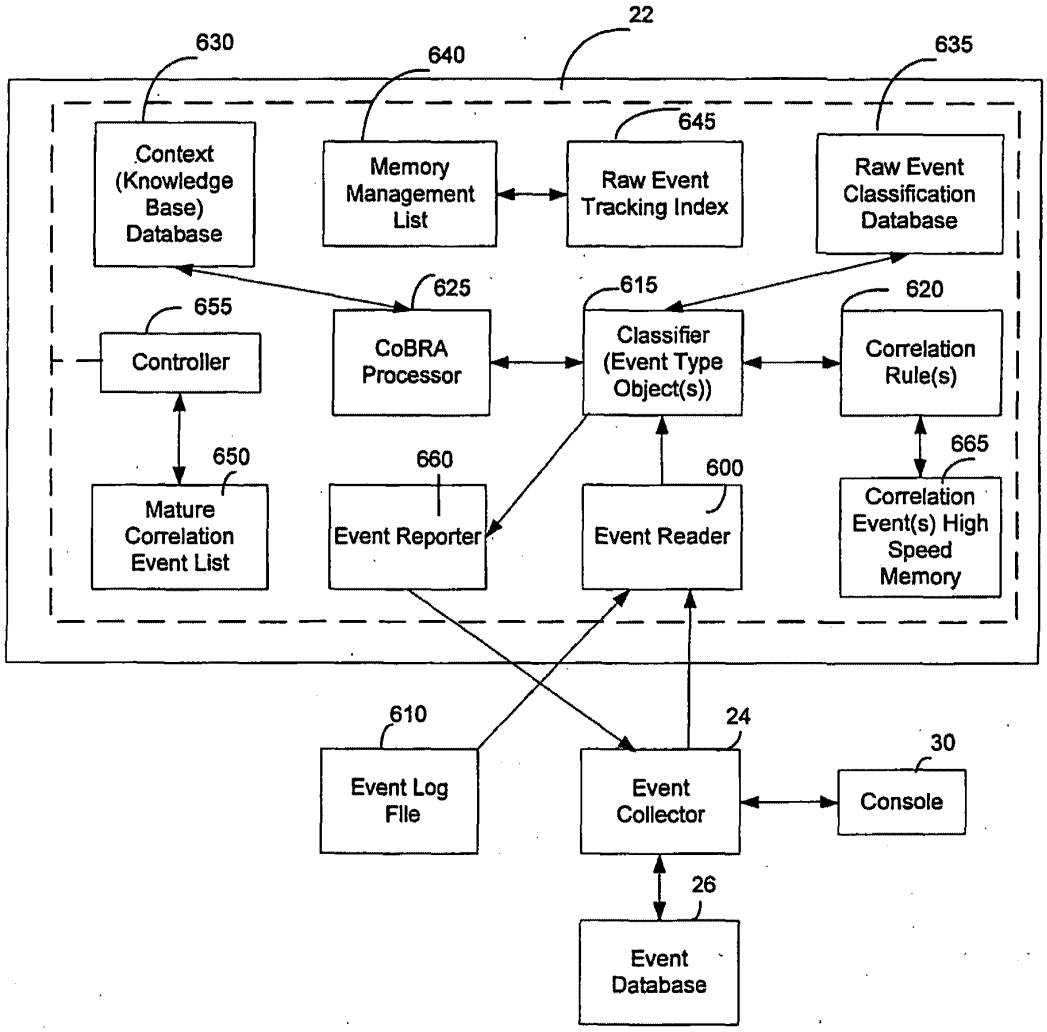

22
630
640
635
645
Context (Knowledge Base) Database
Memory Management List
Raw Event Tracking Index
Raw Event Classification Database
625
615
620
655
Controller
CoBRA Processor
Classifier (Event Type Object(s))
Correlation Rule(s)
650
660
600
665
Mature Correlation Event List
Event Reporter
Event Reader
Correlation Event(s) High Speed Memory
610
24
30
Event Log File
Event Collector
Console
26
Event Database

FIG. 6

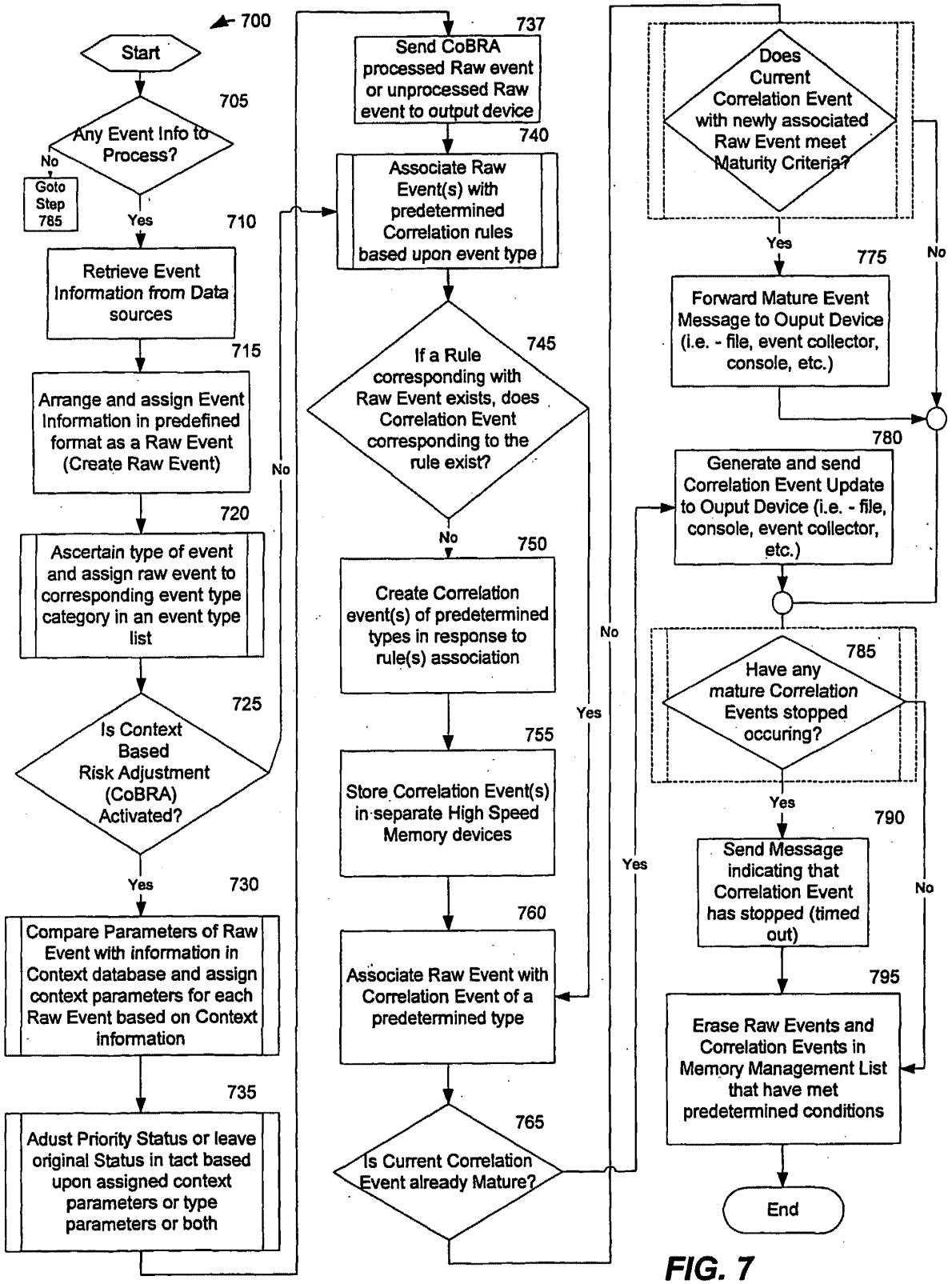

700
Start
705
Any Event Info to Process?
No
Goto Step 785
Yes
710
Retrieve Event Information from Data sources
715
Arrange and assign Event Information in predefined format as a Raw Event (Create Raw Event)
720
Ascertain type of event and assign raw event to corresponding event type category in an event type list
725
Is Context Based Risk Adjustment (CoBRA) Activated?
No
Yes
730
Compare Parameters of Raw Event with information in Context database and assign context parameters for each Raw Event based on Context information
735
Adust Priority Status or leave original Status in tact based upon assigned context parameters or type parameters or both
737
Send CoBRA processed Raw event or unprocessed Raw event to output device
740
Associate Raw Event(s) with predetermined Correlation rules based upon event type
745
If a Rule corresponding with Raw Event exists, does Correlation Event corresponding to the rule exist?
No
Yes
750
Create Correlation event(s) of predetermined types in response to rule(s) association
755
Store Correlation Event(s) in separate High Speed Memory devices
760
Associate Raw Event with Correlation Event of a predetermined type
765
Is Current Correlation Event already Mature?
No
Yes
Does Current Correlation Event with newly associated Raw Event meet Maturity Criteria?
Yes
No
775
Forward Mature Event Message to Ouput Device (i.e. - file, event collector, console, etc.)
780
Generate and send Correlation Event Update to Ouput Device (i.e. - file, console, event collector, etc.)
785
Have any mature Correlation Events stopped occuring?
Yes
No
790
Send Message indicating that Correlation Event has stopped (timed out)
795
Erase Raw Events and Correlation Events in Memory Management List that have met predetermined conditions
End

FIG. 7

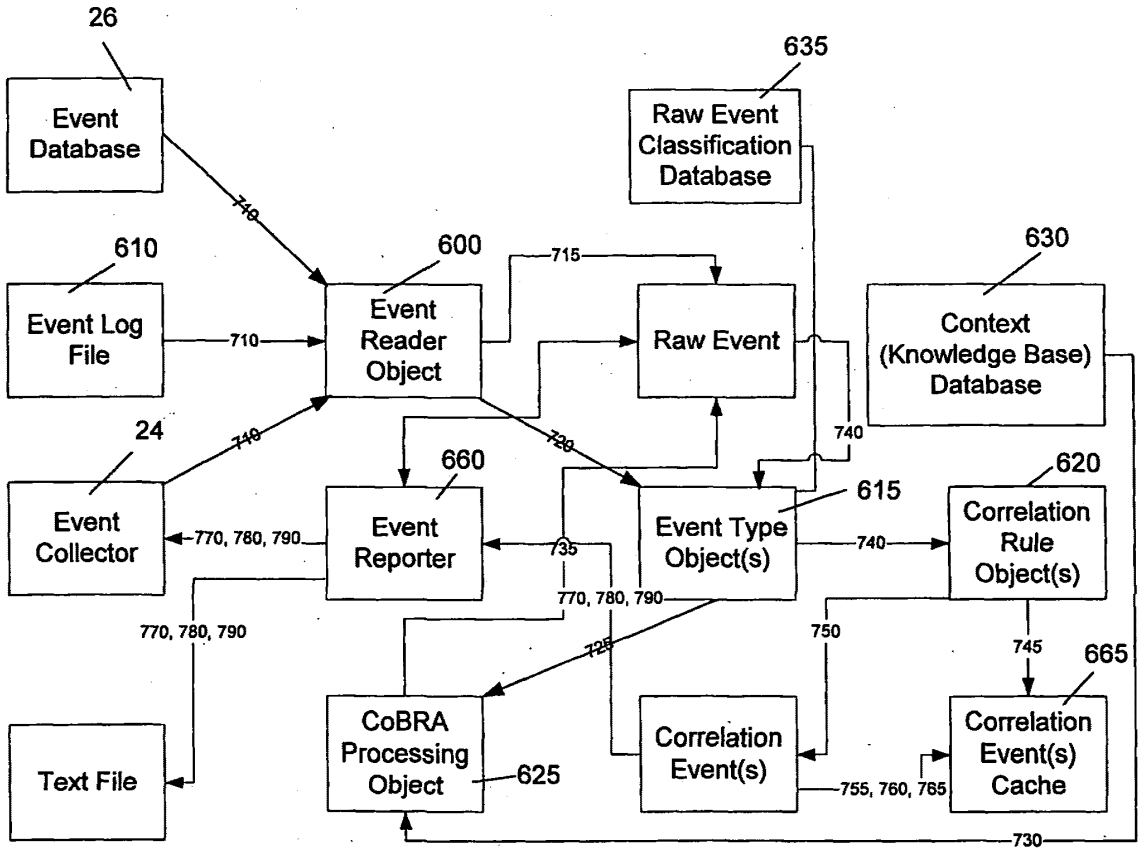

26
Event Database
635
Raw Event Classification Database
610
Event Log File
600
Event Reader Object
715
Raw Event
630
Context (Knowledge Base) Database
710
740
24
Event Collector
660
Event Reporter
735
615
Event Type Object(s)
620
Correlation Rule Object(s)
770, 780, 790
770, 780, 790
770, 780, 790
725
750
745
665
Text File
CoBRA Processing Object
625
Correlation Event(s)
Correlation Event(s) Cache
755, 760, 765
730

FIG. 8

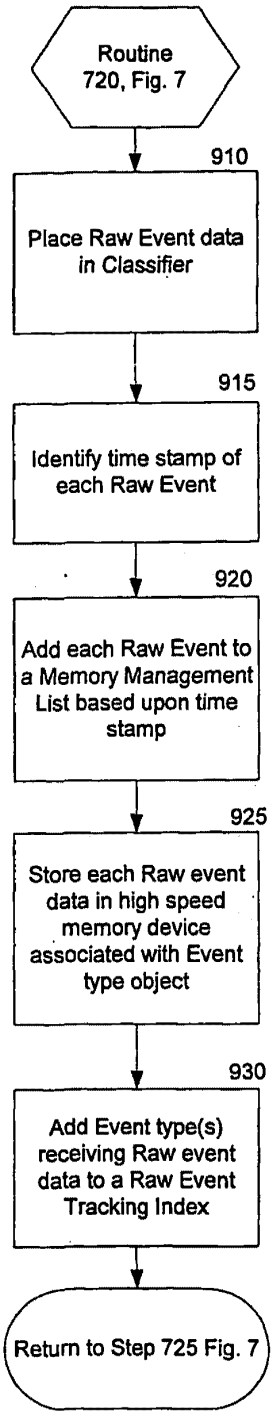

Routine 720, Fig. 7
910
Place Raw Event data in Classifier
915
Identify time stamp of each Raw Event
920
Add each Raw Event to a Memory Management List based upon time stamp
925
Store each Raw event data in high speed memory device associated with Event type object
930
Add Event type(s) receiving Raw event data to a Raw Event Tracking Index
Return to Step 725 Fig. 7

FIG. 9

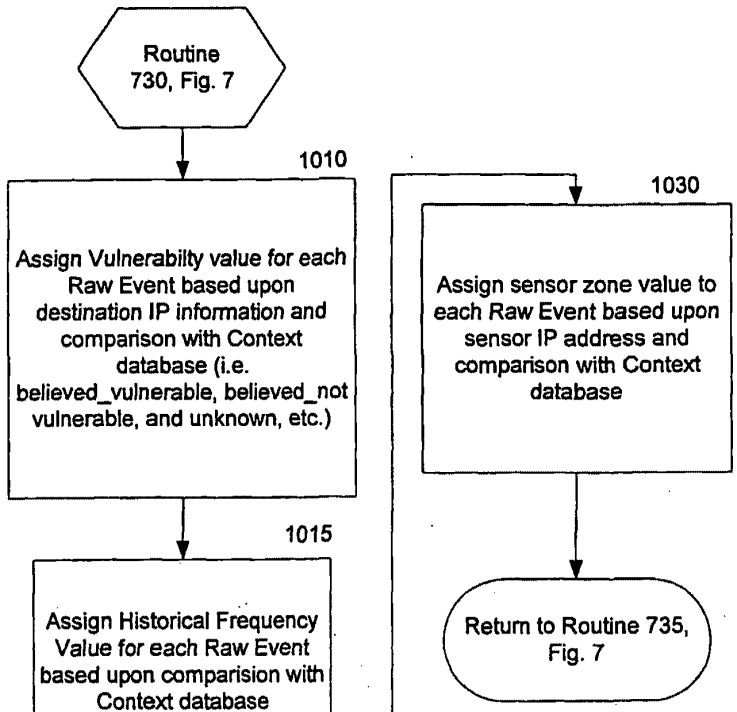

Routine 730, Fig. 7
1010
Assign Vulnerabilty value for each Raw Event based upon destination IP information and comparison with Context database (i.e. believed_vulnerable, believed_not vulnerable, and unknown, etc.)
1015
Assign Historical Frequency Value for each Raw Event based upon comparision with Context database
1020
Assign source zone value to each Raw Event based upon source IP address and comparison with Context database
1025
Assign destination zone value to each Raw Event based upon destination IP address and comparison with Context database
1030
Assign sensor zone value to each Raw Event based upon sensor IP address and comparison with Context database
Return to Routine 735, Fig. 7

FIG. 10

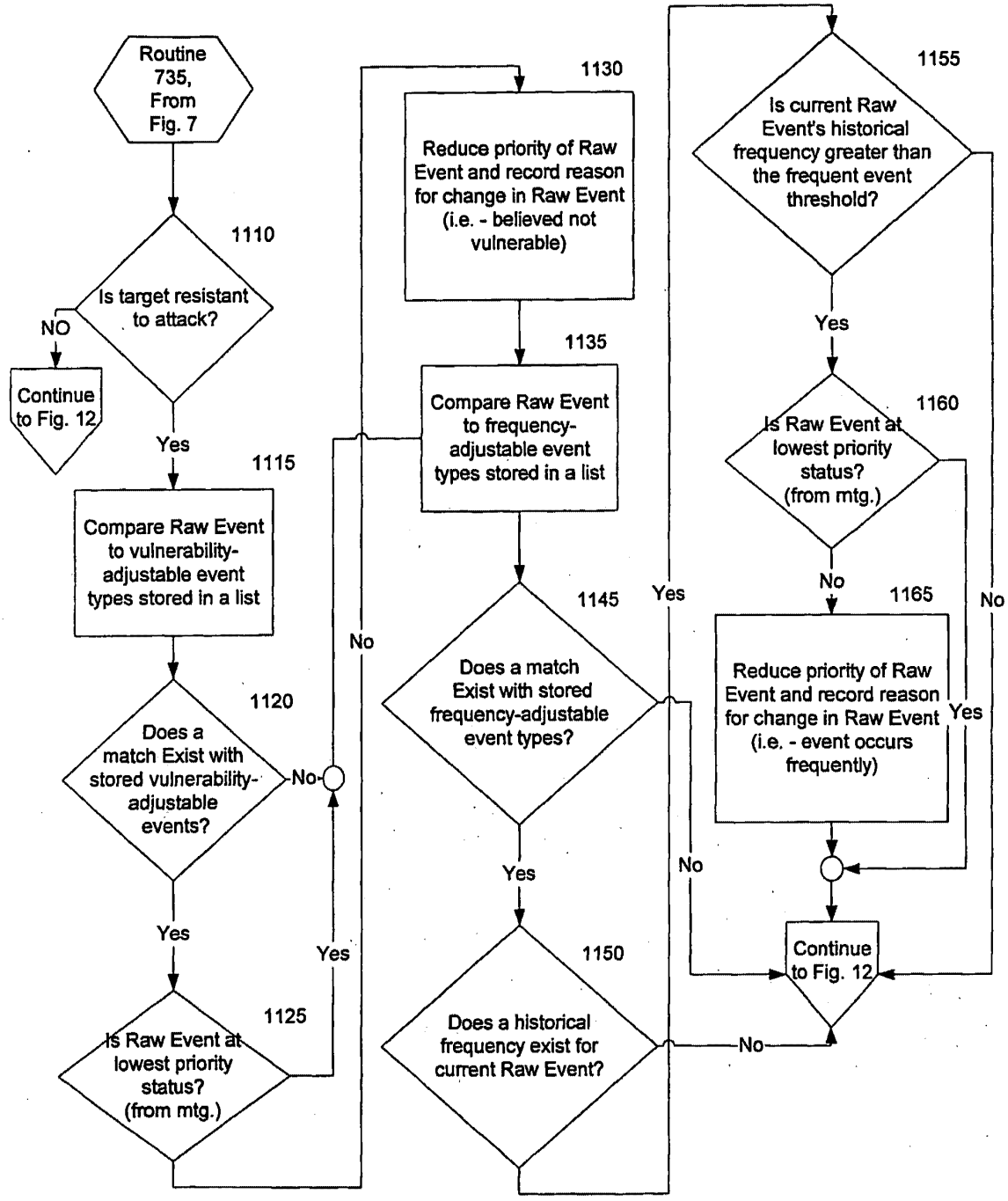

Routine 735, From Fig. 7
1110
Is target resistant to attack?
NO
Continue to Fig. 12
Yes
1115
Compare Raw Event to vulnerability-adjustable event types stored in a list
1120
Does a match Exist with stored vulnerability-adjustable events?
No
Yes
1125
Is Raw Event at lowest priority status? (from mtg.)
Yes
No
1130
Reduce priority of Raw Event and record reason for change in Raw Event (i.e. - believed not vulnerable)
1135
Compare Raw Event to frequency-adjustable event types stored in a list
1145
Does a match Exist with stored frequency-adjustable event types?
Yes
No
1150
Does a historical frequency exist for current Raw Event?
No
Yes
1155
Is current Raw Event's historical frequency greater than the frequent event threshold?
Yes
No
1160
Is Raw Event at lowest priority status? (from mtg.)
No
Yes
1165
Reduce priority of Raw Event and record reason for change in Raw Event (i.e. - event occurs frequently)
Continue to Fig. 12

FIG. 11

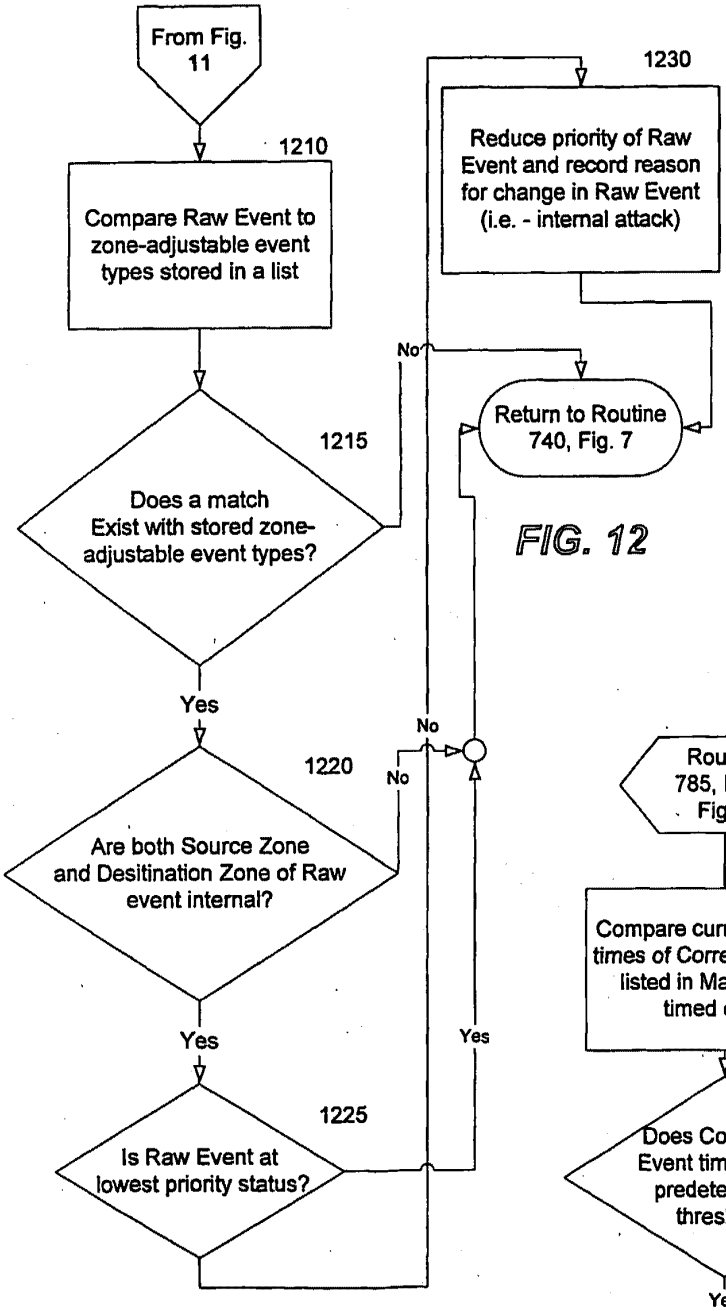


FIG. 12

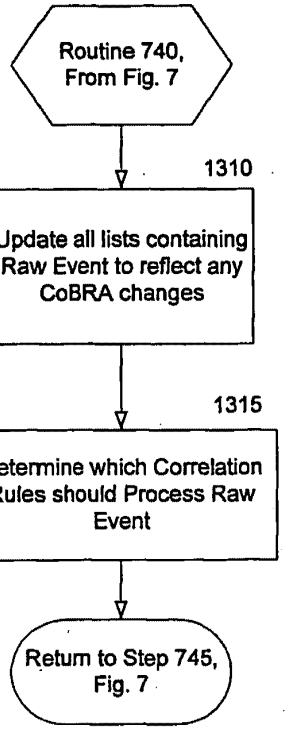


FIG. 13

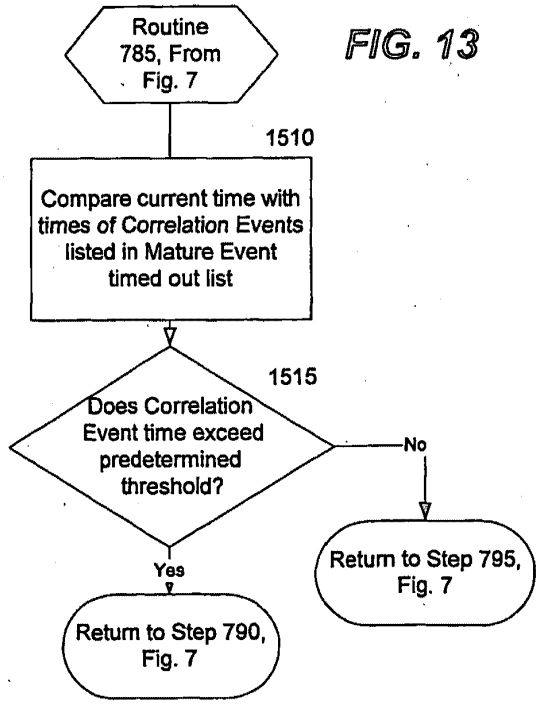


FIG. 15

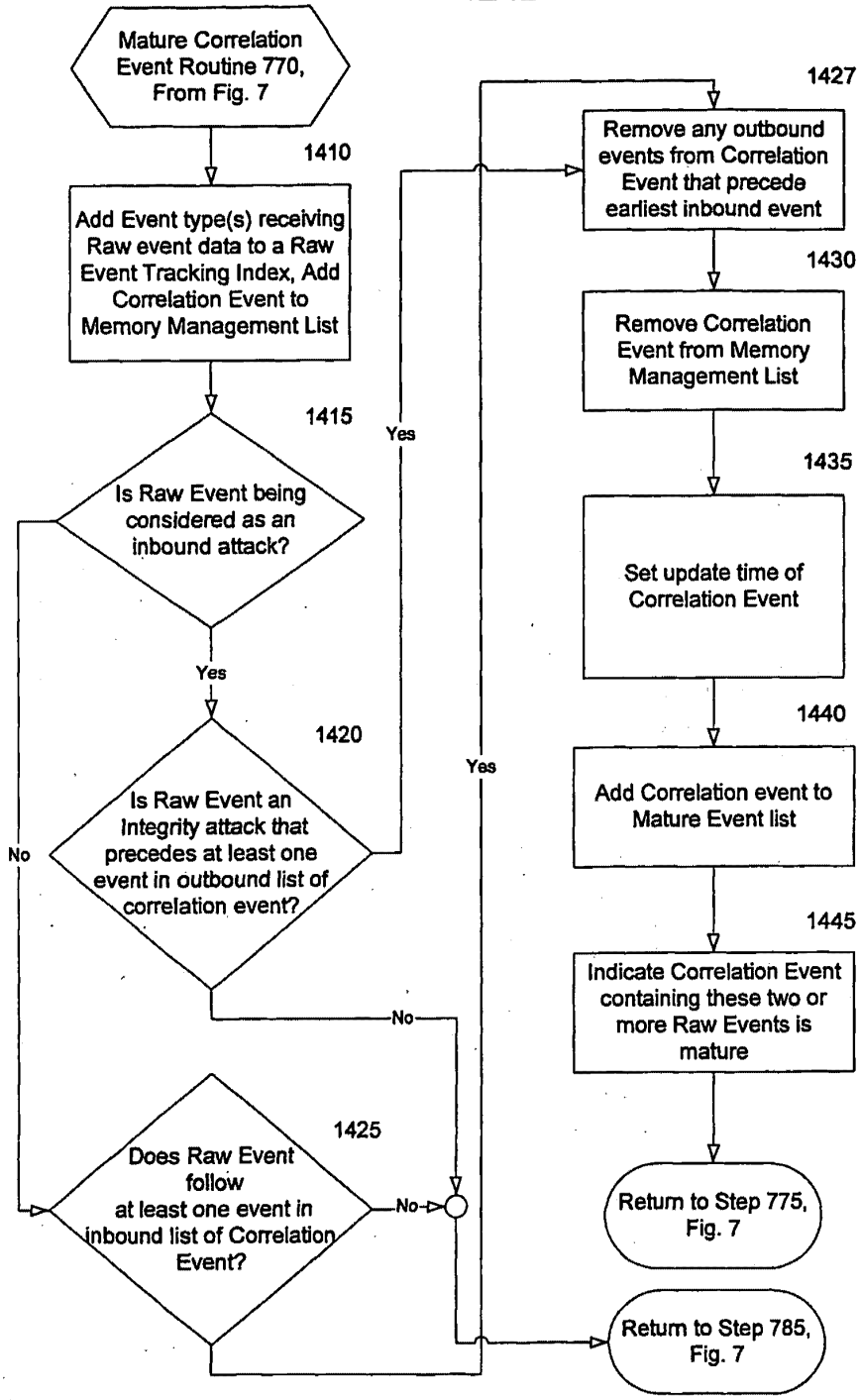

Mature Correlation Event Routine 770, From Fig. 7
1410
Add Event type(s) receiving Raw event data to a Raw Event Tracking Index, Add Correlation Event to Memory Management List
1415
Is Raw Event being considered as an inbound attack?
Yes
No
1420
Is Raw Event an Integrity attack that precedes at least one event in outbound list of correlation event?
Yes
No
1425
Does Raw Event follow at least one event in inbound list of Correlation Event?
No
1427
Remove any outbound events from Correlation Event that precede earliest inbound event
1430
Remove Correlation Event from Memory Management List
1435
Set update time of Correlation Event
1440
Add Correlation event to Mature Event list
1445
Indicate Correlation Event containing these two or more Raw Events is mature
Return to Step 775, Fig. 7
Return to Step 785, Fig. 7

FIG. 14

#5



PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Satterlee et al.

Serial No. **10/336,299**

Filed: **January 3, 2003**

For: **System and Method for the Managed Security Control of Processes on a Computer System**

Art Unit: **2131**

PETITION FOR EXTENSION OF TIME UNDER 37 C.F.R. § 1.136

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

Sir:

The above-identified Applicant respectfully requests a one-month extension of time within which to file a response to the Notice to File Missing Parts dated April 1, 2003, to expire July 1, 2003. A check in the amount of $110.00 is enclosed to cover the one-month extension of time fee.

Please charge any additional fees, or credit any overpayment, to Deposit Account 11-0980. A duplicate copy of this sheet is attached.

Respectfully submitted,

06/17/2003 MBERHE 00000123 10336299
02 FC:1251. 110.00 OP

Robert T. Neufeld
Reg. No. 48,394

King & Spalding LLP
191 Peachtree Street, N.E., 45th Floor
Atlanta, Georgia 30303
404.572.4600
K&S Docket: 05456.105038
3223418 v1

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Missing Parts, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on June 13, 2003.

Robert T. Neufeld, Reg. No. 48,394

OIPE JC109
JUN 16 2003
TRADEMARK OFFICE

#4

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)
	)
Satterlee et al.	)
	)
Serial No. **10/336,299**	)
	)
Filed: **January 3, 2003**	)
	)
For: **System and Method for the Managed Security Control of Processes on a Computer System**	)

TRANSMITTAL OF MISSING PARTS AND LATE FEE

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

Sir:

In response to the Notice of Missing Parts mailed April 1, 2003, the undersigned attorney for applicant hereby transmits:

1. an executed Declaration and Power of Attorney for William Frank Hackenberger;
2. two checks, one in the amount of $130.00 to cover the declaration surcharge for the late filing of these parts, and one in the amount of $110.00 to cover the Petition for Extension of Time; and
3. a duplicate copy of Notice to File Missing Parts of Nonprovisional Application.

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Missing Parts, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on June 13, 2003.

Robert T. Neufeld

Robert T. Neufeld, Reg. No. 48,394

OIPE JC109
JUN 16 2003
PATENT & TRADEMARK OFFICE

It is believed that the submission of these papers and this payment completes the filing of all necessary papers and it is therefore respectfully requested that the application be passed to the appropriate examining group for action on the merits.

Respectfully submitted,

Robert T. Neufeld
Reg. No. 48,394

King & Spalding LLP
191 Peachtree Street, N.E., 45th Floor
Atlanta, Georgia 30303
404.572.4600
K&S Docket: 05456.105038
3223432 v1

IPR2024-00027
CrowdStrike Exhibit 1002 Page 258

OIPE JC109
JUN 16 2003
PATENT & TRADEMARK OFFICE

DECLARATION AND POWER OF ATTORNEY Attorney's Docket No. **05456.105038 US**

In re Application of: **Thomas James Satterlee and William Frank Hackenberger**

As a below named inventor, I hereby declare that my residence, post office address and citizenship are as stated below next to my name. 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: **System and Method for the Managed Security Control of Processes on a Computer System**, the specification of which is attached hereto.

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 do not know and do not believe that the same was ever known or used by others in the United States of America before my or our invention thereof, or patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to the date of this application. I further state that the invention was not in public use or on sale in the United States of America more than one year prior to the date of this application. *I understand that I have a duty of candor and good faith toward the Patent and Trademark Office,* and I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 (a)-(d) of the foreign application(s) for patent or inventor's certificate listed below, and have also identified below any foreign application for patent or inventor's certificate disclosing subject matter in common with the above-identified specification and having a filing date before that of the application on which priority is claimed:

Application No.	Country	Filing Date	Priority Claimed Under 35 USC §119 (a)-(d)
			Yes_____ No_____

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

60/345,432	January 4, 2002		
(Application No.)	(Filing Date)	(Application No.)	(Filing Date)

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter disclosed and claimed in the present application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §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

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 statement 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 patents issuing thereon.

POWER OF ATTORNEY: The following are hereby appointed to prosecute this application and transact all business in the Patent and Trademark Office connected therewith: Sherry M. Knowles-33,052; W. Scott Petty-35,645; Clark G. Sullivan-36,942; Holmes J. Hawkins-38,913; Steven P. Wigmore-40,447; Charles E. Peeler-45,004; Joseph M. Bennett-Paris-47,226; A. Shane Nichols-43,836; William O. Isaacs II-44,165; Robert T. Neufeld-48,394; Stephanie D. Adams-47,378; Kristin Chrudimsky-51,561; Hoang Vo-47,158; James M. Hannon-48,565; Charles Thorpe, Jr.-48,782

IPR2024-00027
CrowdStrike Exhibit 1002 Page 259

OIPE JC109 JUN 16 2003 PATENT & TRADEMARK OFFICE

Send correspondence to: **King & Spalding, LLP** Customer No. 20786 Direct telephone calls at **(404) 572-4600**
191 Peachtree Street, N.E., 45th Floor
Atlanta, Georgia 30303 Robert T. Neufeld

Full name of inventor: William Frank Hackenberger	Citizenship: United States of America
Inventor's signature: William Frank Hackenberger	Date: 1/3/2003
Residence and Post Office Address: 707 University Avenue, Los Altos, CA 94022	

IPR2024-00027
CrowdStrike Exhibit 1002 Page 260

OIPE
JAN 21 2003
JC58
PATENT & TRADEMARK OFFICE

RCO
01-14-03

PATENTS #3

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)
	)
Satterlee et al.	)
	)
Serial No.: **10/336,299**	)
	)
Filed: **January 3, 2003**	)
	)
For: **System and Method for the Managed Security Control of Processes on a Computer System**	)

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
Washington, DC 20231

Sir:

The citation of information on the attached Form PTO-1449, "List of Art Cited by Applicant" is made pursuant to 37 C.F.R. §§ 1.56, 1.97, and 1.98. A copy of each cited item is enclosed.

The citation of this information does not constitute an admission of priority or that any cited item is available as a reference, or a waiver of any right the applicant may have under applicable statutes, Rules of Practice in patent cases, or otherwise.

Respectfully submitted,

Robert T. Neufeld
Reg. No. 48,394

King & Spalding LLP
45th Floor, 191 Peachtree Street, N.E.
Atlanta, GA 30303
404.572.4600
K&S Docket: 05456.105038 US
3088899 v1

I certify that this correspondence is being deposited with the United States Postal Service in an envelope addressed to: Commissioner for Patents, Washington, DC 20231, on January 15, 2003.

Robert T. Neufeld, Reg. No. 48,394

JAN 2 1 2003
PATENT & TRADEMARK OFFICE

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTORNEY DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038 US	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Satterlee et al.		

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	AA	5,586,260	12/17/1996	Wei-Ming Hu			2/12/1993
	AB	5,606,668	2/25/1997	Gil Shwed			12/15/1993
	AC	5,623,601	4/22/1997	Hung T. Vu			11/21/1994
	AD	5,764,887	6/9/1998	Kells et al.			12/11/1995
	AE	5,787,177	7/28/1998	James Leppek			8/1/1996
	AF	5,796,942	8/18/1998	Daniel Esbensen			11/21/1996
	AG	5,798,706	8/25/1998	Kraemer et al.			6/18/1996
	AH	5,815,574	9/29/1998	Michael S. Fortinsky			11/28/1995
	AI	5,828,833	10/27/1998	Belville et al.			8/15/1996
	AJ	5,832,208	11/3/1998	Chen et al.			9/5/1996
	AK	5,832,211	11/3/1998	Blakley et al.			11/13/1995
	AL	5,835,726	11/10/1998	Shwed et al.			6/17/1996
	AM	5,838,903	11/17/1998	Blakely et al.			11/13/1995
	AN	5,864,665	1/26/1999	Tran et al.			8/20/1996
	AO	5,875,296	2/23/1999	Shi et al.			1/28/1997
	AP	5,884,033	3/16/1999	Duvall et al.			5/15/1996
	AQ	5,964,839	10/12/1999	Johnson et al.			5/27/1998
	AR	5,983,270	11/9/1999	Abraham et al.			4/2/1997

EXAMINER	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.

FORM PTO-1449, Adapted

JAN 21 2003 JC58 PATENT AND TRADEMARK OFFICE

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO. 05458.1 CIP DIV	SERIAL NO. 10/336,299	FILING DATE January 3, 2003
APPLICANT Satterlee et al.		GROUP

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	BA	5,983,348	11/9/1999	Shuang Ji			9/10/1997
	BB	5,987,606	11/16/1999	Cirasole et al.			3/19/1997
	BC	5,987,611	11/16/1999	Gregor Freund			5/6/1997
	BD	6,041,347	3/21/2000	Harsham et al.			10/24/1997
	BE	6,061,795	5/9/2000	Dircks et al.			5/12/1997
	BF	6,085,224	7/4/2000	Richard H. Wagner			3/11/1997
	BG	6,088,803	7/11/2000	Tso et al.			12/30/1997
	BH	6,098,173	8/1/2000	Elgressy et al.			11/3/1998
	BI	6,119,234	9/12/2000	Aziz et al.			6/27/1997
	BJ						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.
	BL						
	BM						
	BN						
	BO						

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	BP		
	BQ		
	BR		

EXAMINER	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.

OIPE NU
JUN 25 2003
PATENT & TRADEMARK OFFICE
JC29

2131

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Satterlee et al.

Serial No.: **10/336,299**

Art Unit: **2131**

Filed: **January 3, 2003**

For: **System and Method for the Managed Security Control of Processes on a Computer System**

RECEIVED
JUN 26 2003
Technology Center 2100

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT

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

Sir:

The citation of information on the attached Form PTO-1449, "List of Art Cited by Applicant" is made pursuant to 37 C.F.R. §§ 1.56, 1.97, and 1.98. A copy of each cited item is enclosed.

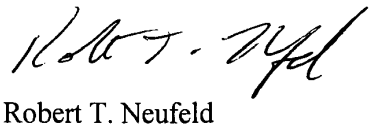
Pursuant to 37 CFR § 1.97(e), the undersigned hereby certifies that each item of information contained in this Supplemental Information Disclosure Statement was first cited in a communication from a foreign patent office in a counterpart foreign application not more than three (3) months prior to the filing of this Supplemental Information Disclosure Statement; or that no item of information contained in the Supplemental Information Disclosure Statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the undersigned after making reasonable inquiry, no item of information contained in the Supplemental Information Disclosure Statement was known to any individual

I certify that this correspondence is being deposited with the United States Postal Service in an envelope addressed to: Commissioner for Patents, P. O. Box 1450, Arlington, VA 22313-1450, on June 18, 2003.

Robert T. Neufeld, Reg. No. 48,394

designated in 37 CFR § 1.56(c) more than three (3) months prior to the filing of the Supplemental Information Disclosure Statement.

Respectfully submitted,



Robert T. Neufeld
Reg. No. 48,394

King & Spalding LLP
45th Floor, 191 Peachtree Street, N.E.
Atlanta, GA 30303
404.572.4600
K&S Docket: 05456.105038 US
3225318 v1

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : H04L 9/00 | A1

(11) International Publication Number: WO 93/25024

(43) International Publication Date: 9 December 1993 (09.12.93)

(21) International Application Number: PCT/US93/05029

(22) International Filing Date: 26 May 1993 (26.05.93)

(30) Priority data:
07/888,909 26 May 1992 (26.05.92) US

(71) Applicant: CYBERLOCK DATA INTELLIGENCE, INC. [US/US]; 906 Woodbrook Lane, Philadelphia, PA 19150 (US).

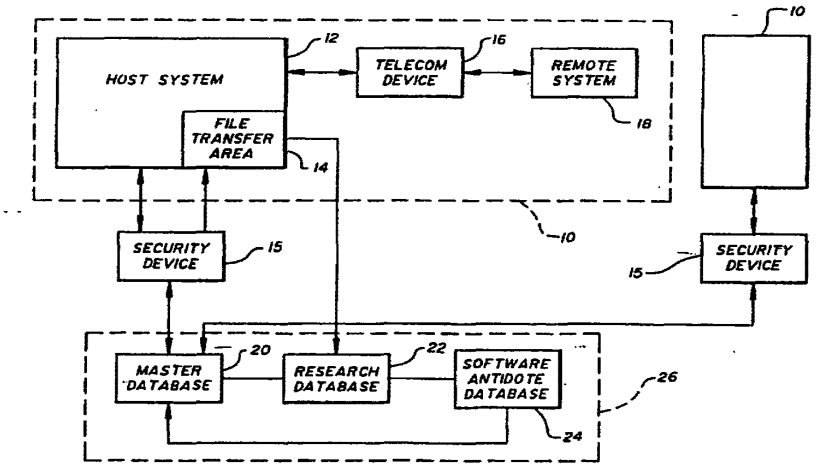
(72) Inventor: BRANHAM, Reginald, K. ; 906 Woodbrook Lane, Philadelphia, PA 19150 (US).

(74) Agent: HEIDELBERGER, Louis, M.; Reed Smith Shaw & McClay, 2500 One Liberty Place, 1650 Market Street, Philadelphia, PA 19103-7301 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: COMPUTER VIRUS MONITORING SYSTEM



(57) Abstract

A method and an apparatus for preventing the infection of computer systems (10) by computer viruses is disclosed. The computer virus monitoring system provides an external security device (15) that stores copies of the host boot sector and file allocation table, and electronic fingerprints of executable files on the host system disk. The external security device (15) monitors writes to the host disk and informs a network monitoring host (26) if the boot sector, file allocation table or electronic fingerprints are altered. The network monitoring host (26) responds to such an emergency by saving the status of the host system (12) and transferring corrective software to remove the virus. The computer virus monitoring system further provides for periodic scanning of the host files to detect and eradicate known viruses.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 266

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	**FR**	France	**MR**	Mauritania
AU	Australia	**GA**	Gabon	**MW**	Malawi
BB	Barbados	**GB**	United Kingdom	**NL**	Netherlands
BE	Belgium	**GN**	Guinea	**NO**	Norway
BF	Burkina Faso	**GR**	Greece	**NZ**	New Zealand
BG	Bulgaria	**HU**	Hungary	**PL**	Poland
BJ	Benin	**IE**	Ireland	**PT**	Portugal
BR	Brazil	**IT**	Italy	**RO**	Romania
CA	Canada	**JP**	Japan	**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	**SK**	Slovak Republic
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SN**	Senegal
CM	Cameroon	**LI**	Liechtenstein	**SU**	Soviet Union
CS	Czechoslovakia	**LK**	Sri Lanka	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**MC**	Monaco	**UA**	Ukraine
DK	Denmark	**MG**	Madagascar	**US**	United States of America
ES	Spain	**ML**	Mali	**VN**	Viet Nam
FI	Finland	**MN**	Mongolia		

COMPUTER VIRUS MONITORING SYSTEM

Field Of The Invention

This invention relates to detection of viruses or logic bombs in computer systems and for taking responsive action upon such detection.

Background Of The Invention

Security of computer systems is becoming increasingly important. Computer viruses and logic bombs ("viruses") are proliferating, and their potential impact is increasing with the increasing use of network systems. Efforts to address the problems posed by viruses have included backing up the contents of computer systems on magnetic tape or other media, and storing the backup in a safe location. Such systems are time consuming and expensive to implement, and only serve to mitigate the effects of a problem after a problem has occurred. Others have developed software programs, resident on a computer system to be protected, to monitor the system for viruses. Such protection systems have certain drawbacks: for instance, because they are resident on a system which may become infected, they are similarly subject to disruption or corruption. Such systems may also be cumbersome to update to respond to newly encountered viruses.

Summary Of The Invention

It is therefore an object of the invention to provide an improved computer virus monitoring system. In accordance with the invention, the system includes a monitoring computer which is coupled to the system to be protected. The monitoring computer monitors the boot sector and file allocation table of th disk drives of the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 268

host system to be protected, and creates and maintains identification data for each executable file. The monitor detects attempts to alter these files or attempts by a file to alter itself or another file and takes appropriate responsive action to protect the host system from unauthorized alteration. The monitor also includes means for scanning the host system to detect particular viruses, and to take appropriate protective action upon detection of a virus. In a particularly preferred embodiment, such monitors on a number of host systems are coupled by a communication network to facilitate analysis of viruses and distribution of antidotes therefor.

Other objects and features of the invention will be understood with reference to the following specification and claims and the drawings.

Brief Description Of The Drawings

Figure 1 is a block diagram showing the basic elements of a system in accordance with the invention.

Figure 2 is a flow diagram illustrating a virus protection method which may be implemented on the system of Figure 1.

Figure 3 is a flow diagram of a method of operating the host system of Figure 1 in accordance with the invention.

Figure 4 is a flow diagram illustrating a method of operation of the monitor of Figure 1 in accordance with the invention.

Figure 5 is a flow diagram illustrating a method of operation of the monitor in response to an emergency condition.

Figure 6 is a flow diagram illustrating a method of communication between monitoring system at a number of host locations and a central monitoring system.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 269

Figure 7 is a block diagram illustrating the basic elements of a preferred embodiment of a security device in accordance with the invention.

Detailed Description of the Invention

Figure 1 is a diagram showing the functional elements of the preferred embodiment of the system of the present invention. A computer network 10 which is to be protected (the "protected network") includes a host system 12 operating at a customer field site that communicates with one or more remote systems and communication nodes 18 via one or more telecommunication devices 16. In a typical application, host system 12 may be an IBM fileserver, an Apple fileserver, or other computer system. Telecommunications device 16 may be a modem, a network card, or a satellite transceiver, for example.

Security device 15 monitors the operation of host system 12 in three different modes to automatically detect viruses and prevent adverse effects. Security device 15 monitors the boot sector and file allocation table (FAT) of disk drives in host system 12. It creates and maintains identifying data as a "fingerprint" for each executable file. These files should never be altered; therefore, if an attempted alteration is detected, appropriate protective action may be taken. Finally, security device 15 scans host system 12 periodically to detect particular viruses.

Security device 15 is coupled to host 12 via a parallel connection to the standard architectural bus of host 12 in the preferred embodiment; however, security device 15 may be coupled to host 12 via a serial port in an alternate embodiment. Security device 15 is also coupled to a security network monitoring host 26, by a telephone line 19 or other communications link.

Desirably, a number of protected networks 10, each with its own security device 15, are connected in a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 270

network with a security network monitoring host computer system 26. Security network monitoring host 26 includes a master service database 20 to monitor standard operation of security devices 15. It also includes a research service database 22 which may called by a protected network 10 in case of an emergency. Because research service database 22 includes storage media such as disk drives, it may become infected by a virus. Therefore, research database 22 is kept separate from the master service database 20. Security network monitoring host system 26 also includes an antidote software service database 24, in which the characteristics of newly encountered viruses are stored and in which solutions to eradicate them are maintained. Antidote software may be transferred via master database 20 to security devices 15 in the monitoring network as needed.

In a system such as that shown in Figure 1, it is important for security device 15 to be as secure and reliable as possible. Therefore, security device 15 is preferably a stand alone unit without a disk drive or any other alterable storage device except RAM, to minimize the possibility of infection. Security devices 15 are provided with battery backup to maintain operation during a power failure. An emergency routine is called when security device 15 detects tampering, such as an unauthorized attempt to open its housing. Communications between security network monitoring host 26 and security devices 15 or protected networks 10 are desirably protected by requiring device-specific electronic identification numbers (EINs) to initiate communication.

Figure 2 is a flow diagram showing a method for operating a host system 12 and a security device 15 to provide virus protection for executable files. In this method, an electronic binary "fingerprint" of identifying data is created and assigned for ach executable file on host system disk drive pack. The identifying data is then after each write access monitored by security device 15.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 271

If the identifying data changes, then security device 15 will halt the protected system and initiate a call to the research database 22 at security network monitoring host 26.

The process of Figure 2 for monitoring the software of the protected system starts in step 50 and new software is added to the protected system in step 52. In step 54 the fingerprint, which can be conventionally generated as a check sum, a cyclic redundancy check (CRC) or other error detecting coding scheme, is created to provide an identifying code that is unique to the file. In a preferred embodiment, the fingerprint is generated by host system 12, although in alternate embodiments the fingerprint could be generated on security device 15 or on a dedicated circuit card. The fingerprint is saved on host system 12 in step 58, and on security device 15 in step 56. Steps 60 and 62 provide an ongoing process, for determining the status of the software by monitoring the fingerprint. The host fingerprint is read in step 60. In step 62 the host fingerprint is compared with the corresponding fingerprint in security device 15. If the host fingerprint has not changed, the process returns from step 62 to step 60 to continue the monitoring process. If the fingerprint has changed, then in step 64 security device 15 halts operation of host system 12. Security device 15 then initiates a call to the system operator of host system 12 in step 68, and in the preferred embodiment, a call to the research database 22 of security network monitoring host 26 in step 70. Also in response to detection in step 62 of a fingerprint change, security device 15 attempts in step 72 to destroy the virus which caused the fingerprint change.

The process of destroying the virus involves comparison of the contents of files stored on host 12 with characteristics of known viruses stored in security device 15 downloaded from master database 20 of network host 26. If characteristics of a known virus are detected, antidote

IPR2024-00027
CrowdStrike Exhibit 1002 Page 272

software, stored in antidote software database 24, is executed to destroy the virus in step 74. Upon successful detection and destruction of a virus, a report of this activity is generated in step 76, transmitted to the research database 22 of security network monitoring host 26 in step 78, and transmitted to the system operator of host system 12 in step 80. Step 82 returns the process to step 54, in which a new fingerprint is created, and then saved. Host system 12 may thereafter continue its operation free from the virus. If in step 74 the virus has not been successfully destroyed, a further call is initiated by security device 15 to research database 22 for further assistance in identification and destruction of the virus, while operation of host system 12 is disabled to prevent consequential damage from the virus.

Figure 3 is a flow diagram of the operation of host system 12 relating to security device 15. Initialization steps of host system 12 include porting the boot sector and FAT of host system 12 to security device 15 in steps 90 and 92 respectively, creating a fingerprint file in step 94 and porting the fingerprint file to security device 15 in step 96. In the host monitoring process, security device 15 monitors write access signals in step 102, indicating an instruction to write to a disk. When a write access signal is received, host system 12 provides information to security device 15 to use in tests to verify system integrity and virus free input. In step 104, host system 12 sends a write access interrupt to security device 15. In step 106, security device 15 reads its interrupt status to determine whether the emergency interrupt flag is set. If an emergency interrupt is indicated, the keyboard is locked and host 12 is halted in step 108. In step 110 the write access is examined to determine whether it seeks to write to an existing executable file; if so, in step 112 the keyboard is locked and host system 12 is halted. Otherwise, in step 114 the writ access is examined to determine whether it seeks to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 273

write a new executable file or batch. If so, in step 116 host 12 calls a scan routine from security device 15 and executes it. The scan routine examines the file to be written for viruses and the like. If the scan indicates the presence of a virus or the like, step 118 directs the process to step 120, where an emergency interrupt is sent to security device 15, and step 122, where the keyboard is locked and host system 12 is halted. If in step 118 no viruses or other problems have been detected, or if step 114 has determined that the write access is not for a new executable file, the file may be written to disk. In step 124, host system 12 updates the FAT on security device 15, creates new fingerprints in step 126, and updates the fingerprint data on security device 15 in step 128.

Host system 12 also waits for interrupt from security device 15 to run tests to verify the entire system is virus-free. This is done to determine at what time any system errors occur to help with data recovery in case that becomes necessary. Thus, in step 130 if a fingerprint interrupt is received from security device 15, host system 12 in step 132 ports its fingerprint data to security device 15. If host system 12 then receives a fingerprint emergency interrupt from security device 15 in step 134, it locks the keyboard and halts host system 12 in step 136; otherwise, control is passed to step 138 and host system 12 returns to its start status in 100. If in step 140 a scan interrupt is received from security device 15, host system 12 calls the scan routine from security device 15 and executes it in step 142. If no viruses or the like are detected in the scan, step 144 directs a return to the start status via step 138. If, however, the scan process results in the detection of a virus or the like, in step 146 an emergency interrupt is sent to security device 15 and host system 12 is halted and its keyboard locked in step 148.

Figure 4 is a flow diagram illustrating the operation of security device 15. Security device 15

IPR2024-00027
CrowdStrike Exhibit 1002 Page 274

initialization includes steps 150, 152, and 154, where boot sector, FAT, and fingerprints ("CRCs") of host 12 are copied and mapped into security device 15 memory.

The security device 15 monitoring process starts in step 156, and in step 158 security device 15 monitors the host until a write access interrupt is received from the host. In step 160, security device 15 compares the boot sector files stored in its memory with the boot sector files stored in host system 12. If these are not the same, step 162 directs the process to an emergency state in step 164. If the boot sector files are determined to be the same as in step 162, step 166 determines whether the write access seeks to write to a .EXE, .COM file, or the like. If so, security device 15 goes to an emergency state in step 168 and follows the emergency procedure shown in Figure 5. If not, step 170 determines whether the write access seeks to write a new executable file. If so, step 172 scans the file to be written to determine whether it contains any viruses. If the test is not passed, security device 15 is directed in step 174 to go to an emergency state in step 176. If the test in step 174 is passed, or if in step 170 the write access was not for a new executable file, the file may be written. In step 178, the FAT is updated on host system 12 and security device 15, and in step 180 fingerprint data for the file is created on the host 12 and updated fingerprint data is sent to security device 15.

In addition to monitoring initiated upon write accesses, security device 15 also automatically periodically monitors fingerprint data and scans for viruses. In step 182, if an internal clock in security device 15 indicates that it is time for a fingerprint check, in step 184 the fingerprint data stored in security device 15 is compared with the current fingerprint data of host system 12. If the fingerprint data is the same, step 194 directs the process to step 190 and a return to start

IPR2024-00027
CrowdStrike Exhibit 1002 Page 275

condition 154. If the fingerprint data is different, security device 15 goes to an emergency state in step 196.

In step 186, if security device 15 clock indicates that it is time to perform a scan, a scan is performed in step 188 to determine whether the contents of host system 12 contain any viruses or the like by comparison with identifying data stored in security device 15. If the scan detects no such code, step 192 directs the process to the start condition through step 190; if a virus is detected, security device 15 goes to an emergency state in step 198.

Figure 5 is a flow diagram illustrating emergency condition operation of security device 15, which starts at step 200. A security device 15 sends an emergency interrupt to host system 12 in step 202, instructing host system 12 in step 204 to close all files and return to the DOS prompt (or an equivalent prompt in a non-DOS system), and in step 206 to lock the keyboard and halt operation. This action isolates the virus in host system 12. Host system 12 remains locked until the proper unlock password is entered in step 208, in which event host system 12 will unlock the keyboard in step 210 and save the status to disk in step 212, display the status in step 214, and write the status to the printer in step 216.

In addition to sending an emergency interrupt to host system 12, in the preferred embodiment security device 15 is connected in a network, and in step 218 calls the monitoring host 26. In step 220, security device 15 identifies itself by sending a login code and the electronic identification number for the particular security device 15. If the electronic identification number is accepted, security device 15 will be permitted to login to the research database, a call is generated to the monitoring network system operator in step 222, and the status of the protected network 10 which generated the emergency condition is displayed in step 224. After the monitoring network system operator identifies and corrects

IPR2024-00027
CrowdStrike Exhibit 1002 Page 276

the problem which generated the emergency condition, the system operator resets security device 15 in step 222, and normal operation of protected network 10 resumes in step 228.

Figure 6 is a flow diagram illustrating the operation of network monitoring system 26. In step 250, network monitoring system 26 is responding to a connect attempt initiated by a security device 15, in step 218 of Figure 5. In step 252, network monitoring system 26 prompts security device 15 for its EIN and login code. In step 220, security device 15 sends its logon code and EIN to network monitoring system 26. In step 254, network monitoring system 26 checks the received EIN and logon code to verify that the communication was initiated by a legitimate security device 15, not by a computer hacker. To ensure the security of the system, a preferred embodiment of the invention uses a 128 bit EIN. In a preferred embodiment, network monitoring host 26 waits only a limited amount of time, approximately 9 seconds, to receive logon information after it detects a carrier signal on the telephone link. The combination of a 128 bit EIN and a limited time to respond effectively eliminates the possibility of a hacker logging onto the system. If network monitoring system 26 detects an invalid EIN or a time-out condition, it disconnects the telephone hookup in step 256. If network monitoring system 26 detects a valid EIN in step 254, the logon procedure is executed in step 258.

In step 260, network monitoring system 26 determines whether or not the call is an emergency call. If the call is an emergency call, the status of host system 12 is saved in research database 22 in step 272, displayed in step 274 and printed out in step 276, so that the virus can be isolated. In parallel with steps 272, 274 and 276, a call is generated by an emergency routine within host 12 to the system operator of network monitoring system 26 in step 278. In step 280, network

IPR2024-00027
CrowdStrike Exhibit 1002 Page 277

monitoring system 26 checks whether host system 12 has been reset. Once host system 12 has been reset, a logoff procedure is followed in step 282 and network monitoring system 26 disconnects telephone link 19 from security device 15 to research database 22 in step 284.

If, in step 260, network monitoring system 26 determines that the call is not an emergency call, it checks in step 262 whether the call is a report call. If the call is not a report call, the system operator is called in step 264 and network monitoring system 26 enters terminal mode. If the call is a report call, network monitoring system 26 creates a report file including the time and date in step 266, stores the report file in step 268 and prints the file in step 270. After the file is sent to the printer, network monitoring host 26 initiates a logoff procedure in step 282 and disconnects the telephone link from security device 15 in step 284.

Figure 7 is a block diagram of a preferred embodiment of security device 15. Security device 15 includes a processor 300 connected to a control bus 354, a data bus 356 and an address bus 358. Processor 300 communicates through buses 354, 356 and 358 to read only memory (ROM) 306, random access memory (RAM) 308, I/O interface 320, I/O interface 316 and liquid crystal display (LCD) driver 310. Processor 300 further includes an input from Clock 302 and from Reset Input 304. LCD driver 310 is coupled through LCD interface 312 to LCD 314. LCD 314 is used by security device 15 to display status information to a maintenance technician or an engineer, to aid in isolating a virus found in host system 12. Security device 15 includes ROM 306 and RAM 308 as storage devices. ROM 306 is utilized for storing the firmware which controls processor 300. RAM 308 is used by Processor 300 to store temporary information, including fingerprint files, boot sector information and file access table information. Security device 15 communicates with host 12 through I/O interface 320. I/O interface 320 is

IPR2024-00027
CrowdStrike Exhibit 1002 Page 278

preferably coupled to file transfer area 14 of host 12 via a parallel connection, although in another embodiment a serial port is utilized. Security device 15 is assigned a random address location in the memory space of host system 12, so that a Hacker cannot try to access RAM 308. Security device 15 is coupled to modem 318 through I/O interface 316. Security device 15 uses modem 318 to access the master database 20 of host of network monitoring system 26 through telephone line 19.

It is to be understood that the foregoing is merely illustrative of the principles of this invention, and that various modifications can be made by those skilled in the art without departing from the scope and spirit of the invention.

What Is Claimed Is:

1. A monitoring system for detecting software viruses in a host computer system having a host memory and a file storage memory, for storing an executable software file by means of a write access, said monitoring system comprising:

means for generating an electronic fingerprint of the executable software file when the file is written to the file storage memory by means of a first write access;

a security device coupled to the host computer, said security device comprising means for monitoring write accesses of the file storage memory, fingerprint memory means for storing a copy of the fingerprint and emergency interrupt response means for halting and locking the host computer in response to an emergency interrupt;

means for transferring a first copy of the fingerprint to the fingerprint memory of the security device and for transferring a second copy of the fingerprint to the host memory; and

comparison means for comparing the first copy of the fingerprint stored in fingerprint memory with the second copy of the fingerprint stored in the host memory when a subsequent write access of the executable software file occurs and for generating an emergency interrupt to the security device if the first and second copies are not identical.

2. The system of claim 1, wherein the host computer system includes the fingerprint generation means.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 280

3. The system of claim 1, wherein the security device includes the fingerprint generation means.

4. The system of claim 2, wherein the electronic fingerprint is a checksum code.

5. The system of claim 2, wherein the electronic fingerprint is a cyclic redundancy check code.

6. The system of claim 1, wherein the host computer includes the comparison means.

7. The system of claim 1, wherein the security device includes the comparison means.

8. A monitoring system for detecting software viruses in a protected network including a host computer coupled to a remote computer system, the host computer having a host memory and a host file storage memory for storing a software file by means of a write access, the monitoring system comprising:

means for generating an electronic fingerprint of the software file when the file is written to the file storage memory;

a security device coupled to the host computer, said security device comprising means for monitoring write accesses of the file storage memory, fingerprint memory means for storing a copy of the fingerprint and emergency interrupt response means for halting and locking the host computer in response to an emergency interrupt;

means for transferring a first copy of the fingerprint to the fingerprint memory of the security device and for transferring a second copy of the fingerprint to the host memory; and

comparison means for comparing the first copy of the fingerprint stored in fingerprint memory with th second copy of the fingerprint stored in the host memory when a subsequent write access of the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 281

software file occurs and for generating an emergency interrupt to the security device if the first and second copies are not identical.

9. The system of claim 8, further comprising:

means, in the security device, for storing a copy of the file allocation table of the host computer system; and

comparison means responsive to a write access of file storage memory, for comparing the copy of the file allocation table stored in the security device memory means with the file allocation table stored in the host system when a write access occurs and for generating an emergency if the copy of the file access table is not identical to the file allocation table stored in the host system.

10. The system of claim 9, further comprising:

means for copying the boot sector of the host computer system to the memory means in the security device; and

comparison means for comparing the copy of the boot sector stored in the security device memory with the boot sector stored in the host memory when a write access occurs and for generating an emergency interrupt if the copy of the boot sector is not identical to the boot sector of the host system.

11. The system of claim 8, further comprising a network monitoring system coupled to the security device and to the host system, for monitoring a plurality of security devices.

12. The system of claim 11, wherein the network monitoring system further comprises a research database for compiling performance characteristics for each

IPR2024-00027
CrowdStrike Exhibit 1002 Page 282

security device and protected network, and an antidote software database for storing software to overcome known computer viruses.

13. The system of claim 12, wherein the research database contains known computer virus code patterns.

14. The system of claim 13, further comprising means for scanning the host file storage memory to detect the known software viruses stored in the research database.

15. A method for protecting a host computer system including a file storage memory from infection by a software virus, comprising the steps of:

generating an electronic fingerprint for an executable software file being stored in the file storage memory;

storing a first copy of the electronic fingerprint in the host memory and storing a second copy of the electronic fingerprint in an external security device;

monitoring write accesses to the file storage memory;

comparing the first and second copies of the fingerprint whenever the write access occurs; and

halting and locking the host computer when the first and second copies of the fingerprint differ, indicating the possible presence of a virus, so that the virus cannot damage the host computer system.

16. The method of claim 15, further comprising the steps of:

transferring a copy of the boot sector of the host computer system to the security device memory;

periodically comparing the copy of the boot sector with the host boot sector to

IPR2024-00027
CrowdStrike Exhibit 1002 Page 283

determine whether the host boot sector has changed; and

if the host boot selector has changed, halting and locking the host computer system to prevent possible damage by a virus.

17. The method of claim 16 further comprising the steps of:

transferring a copy of the file allocation table of the host computer system to the security device memory:

periodically comparing the file allocation table with the copy of the file allocation table stored in the security device memory to determine whether its fingerprint has changed; and

if the file allocation table has changed, halting and locking the host computer system to prevent possible damage by a virus.

18. The method of claim 17, further comprising the steps of:

storing known computer virus code patterns in a first database;

storing antidote software for said known computer virus code patterns in a second database;

periodically scanning the file storage memory of the host computer system for said known computer virus code patterns; and

halting the host computer system and executing said antidote software if a virus is detected.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 284

SUBSTITUTE SHEET

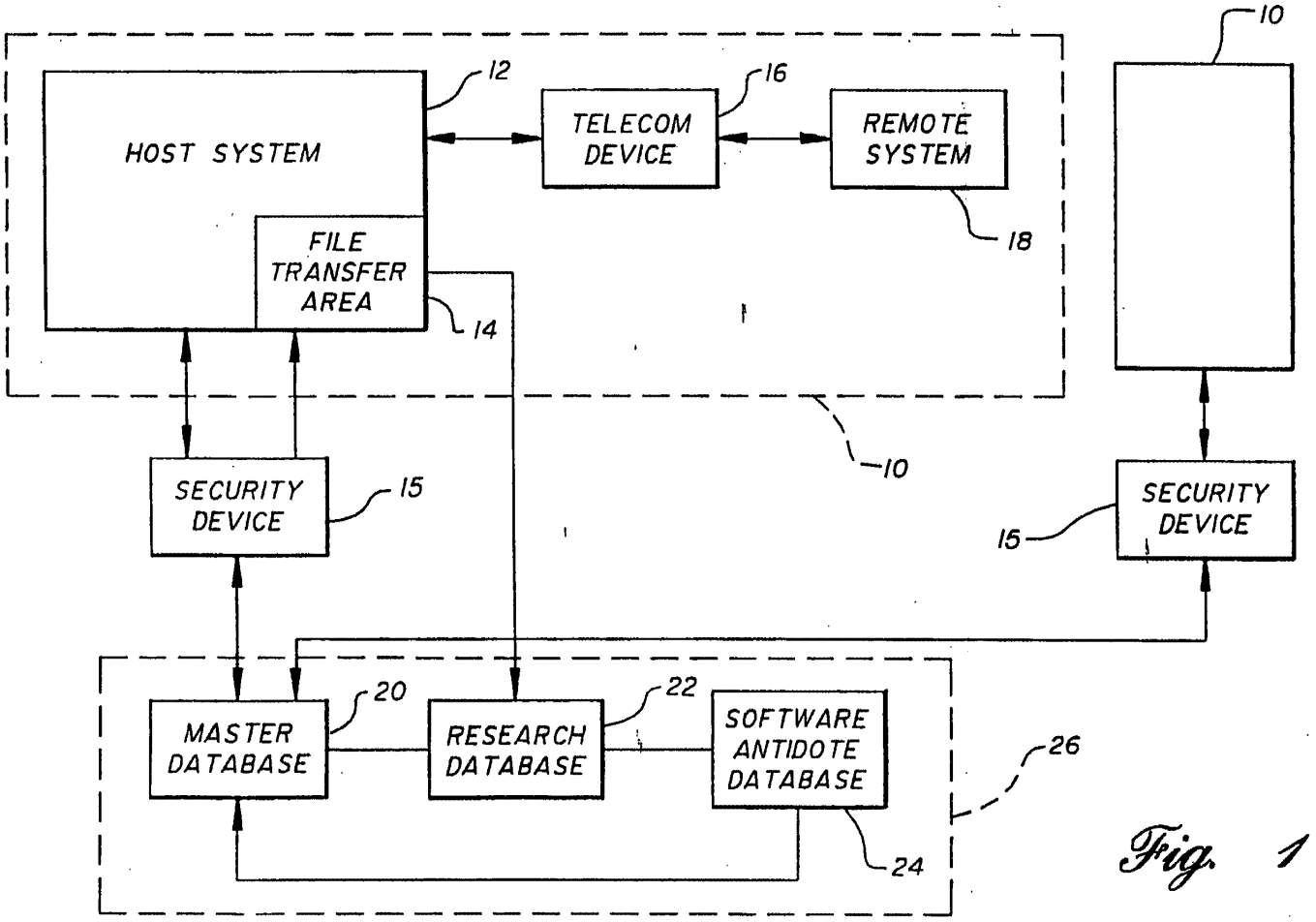


Fig. 1

SUBSTITUTE SHEET

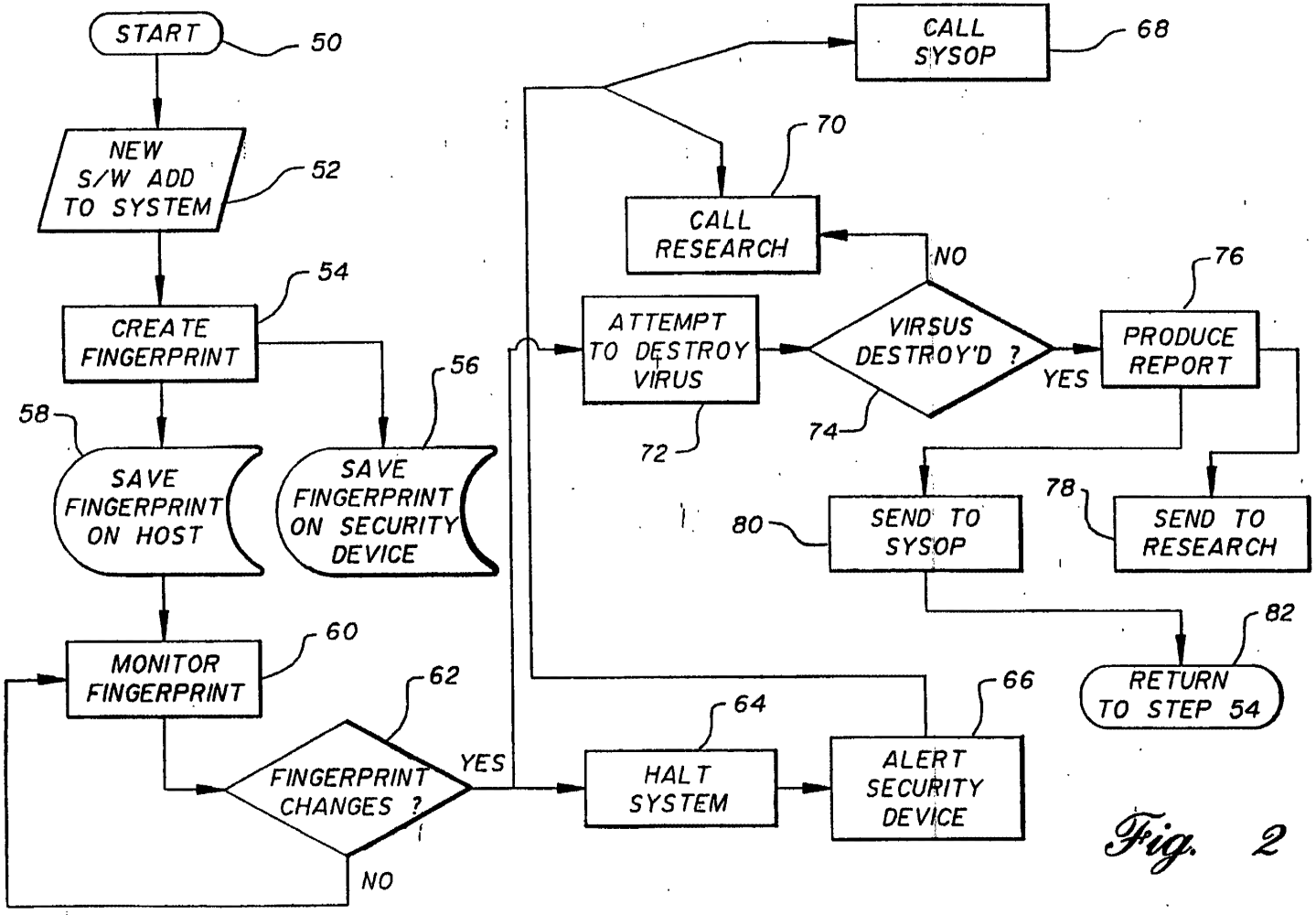


Fig. 2

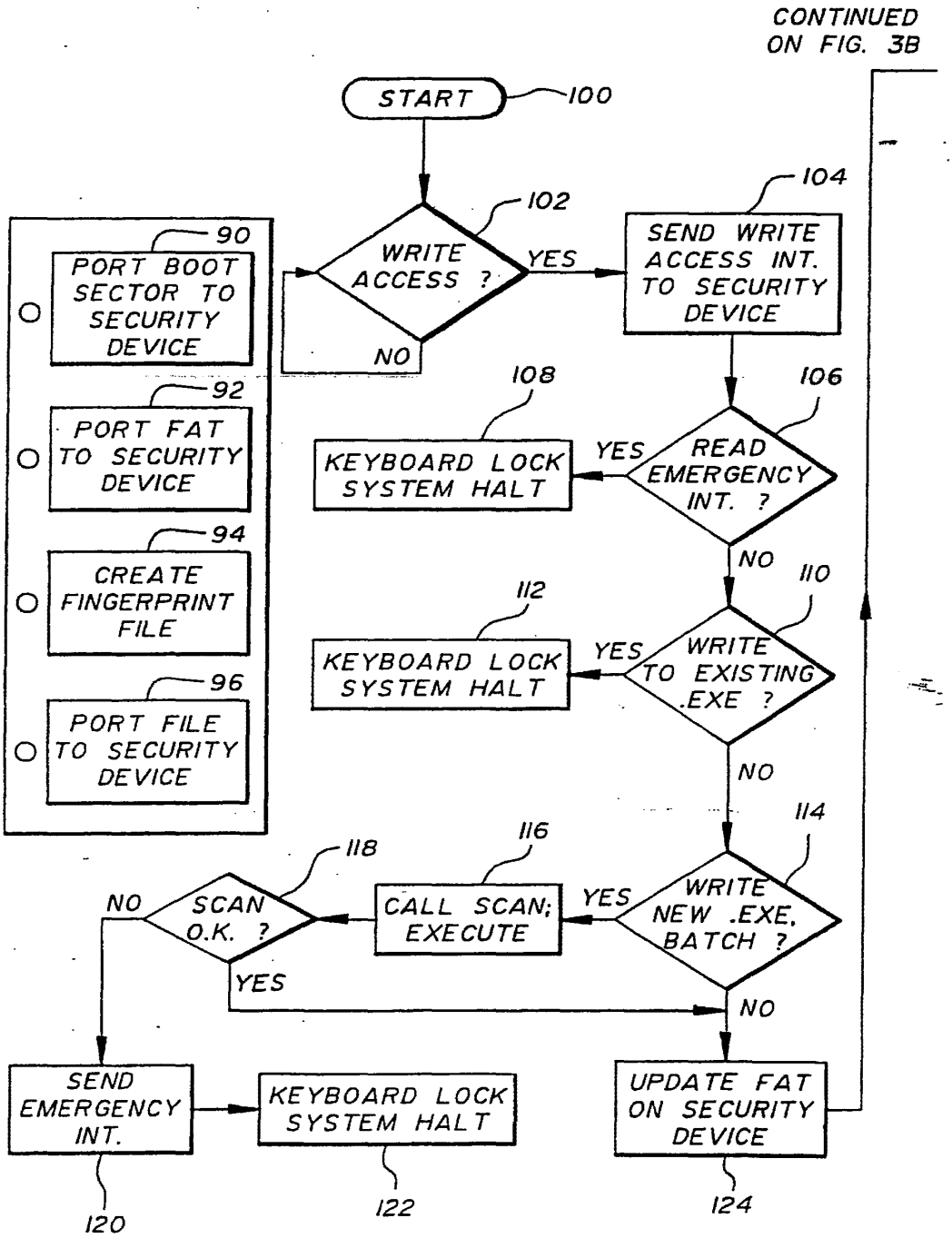


Fig. 3A

SUBSTITUTE SHEET

IPR2024-00027
CrowdStrike Exhibit 1002 Page 287

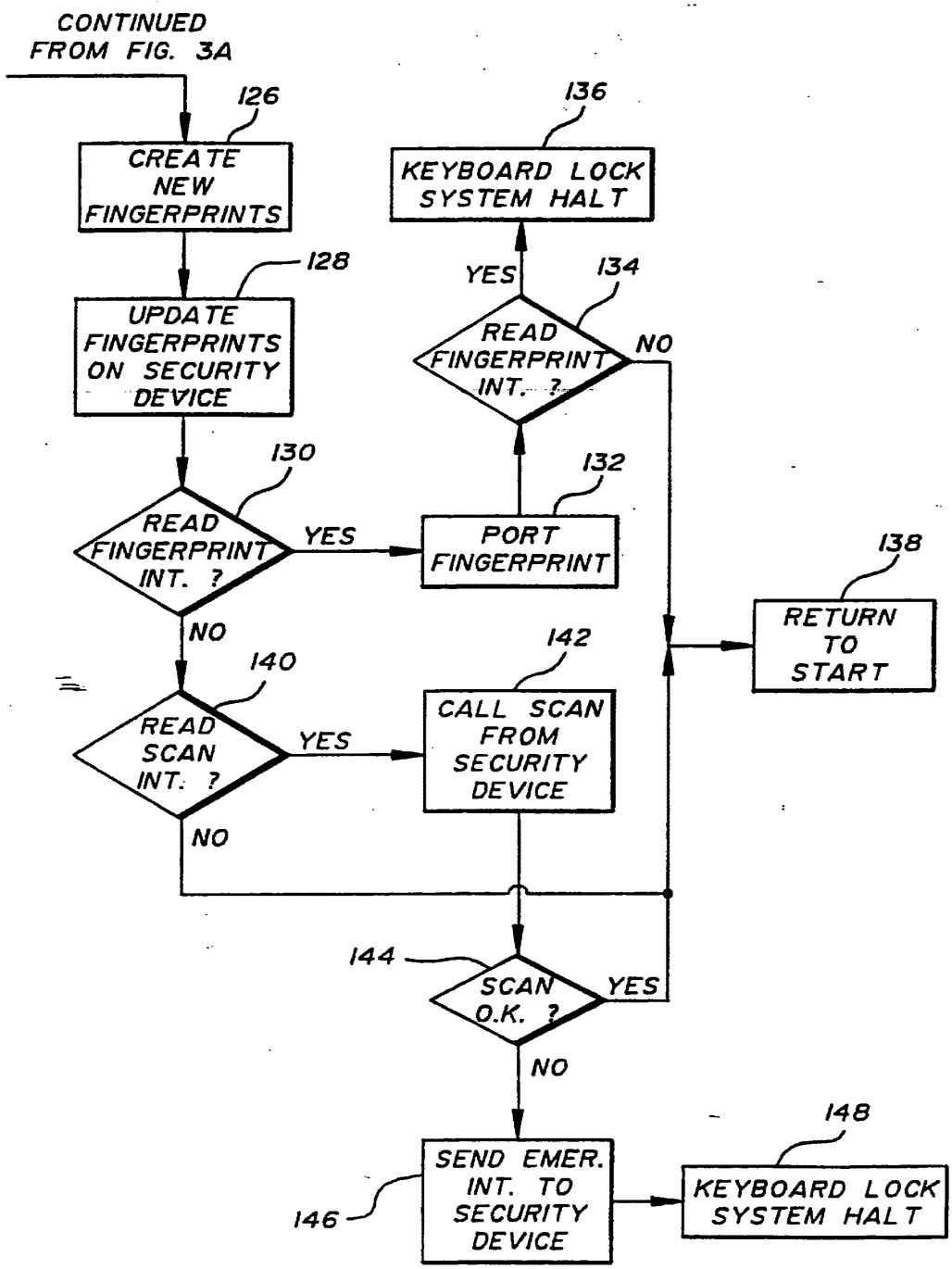


Fig. 3B

SUBSTITUTE SHEET

IPR2024-00027
CrowdStrike Exhibit 1002 Page 288

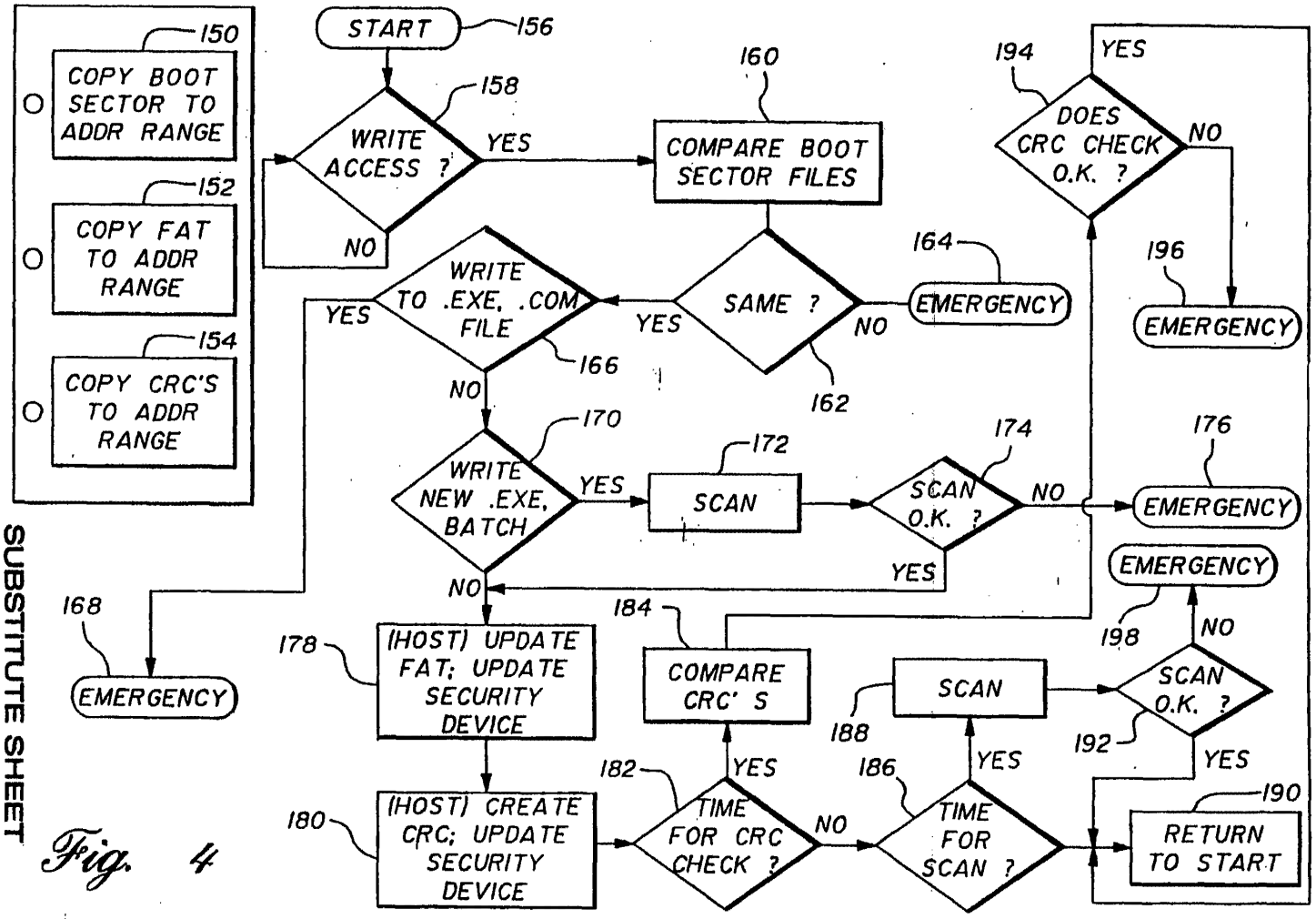


Fig. 4

SUBSTITUTE SHEET

SUBSTITUTE SHEET

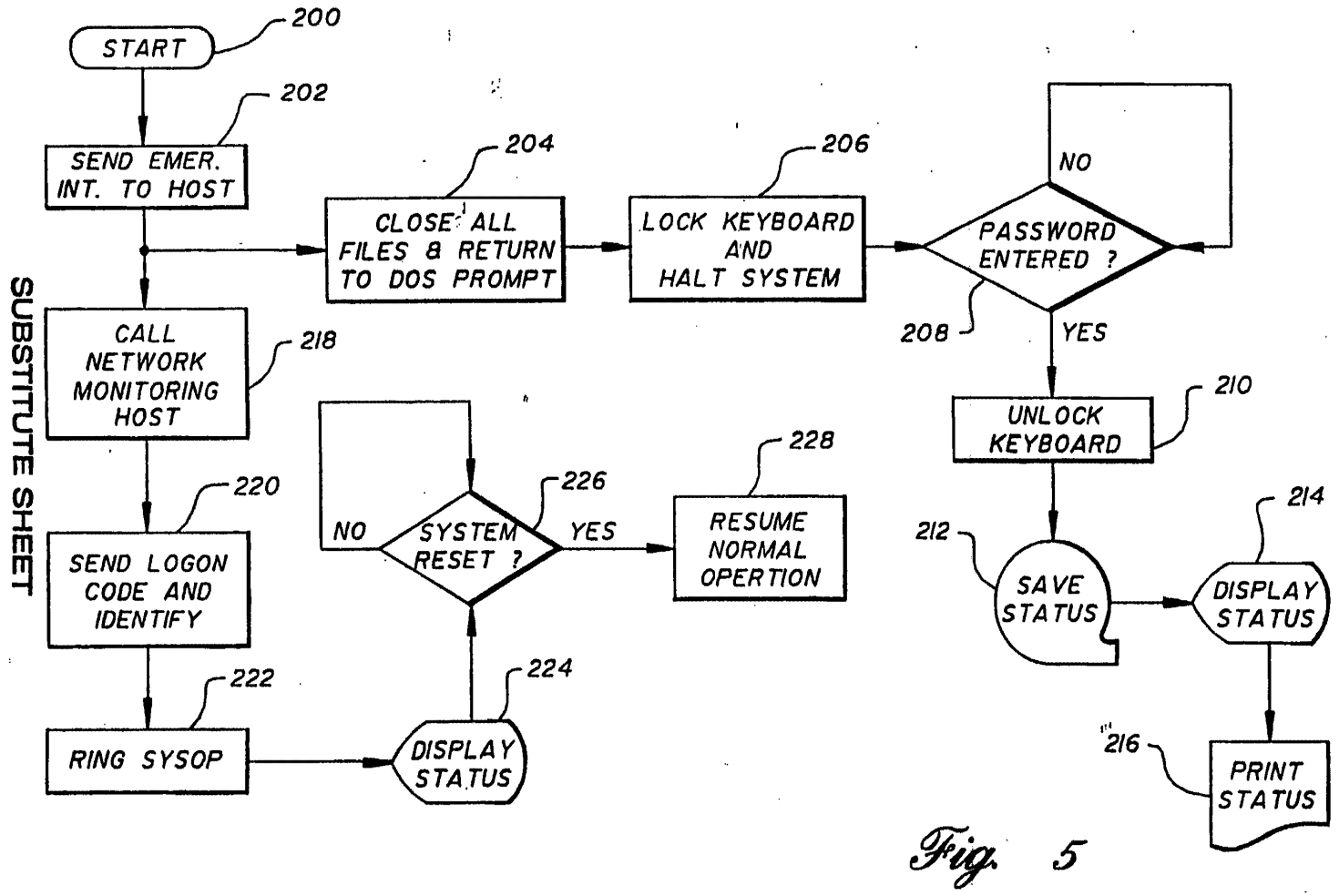


Fig. 5

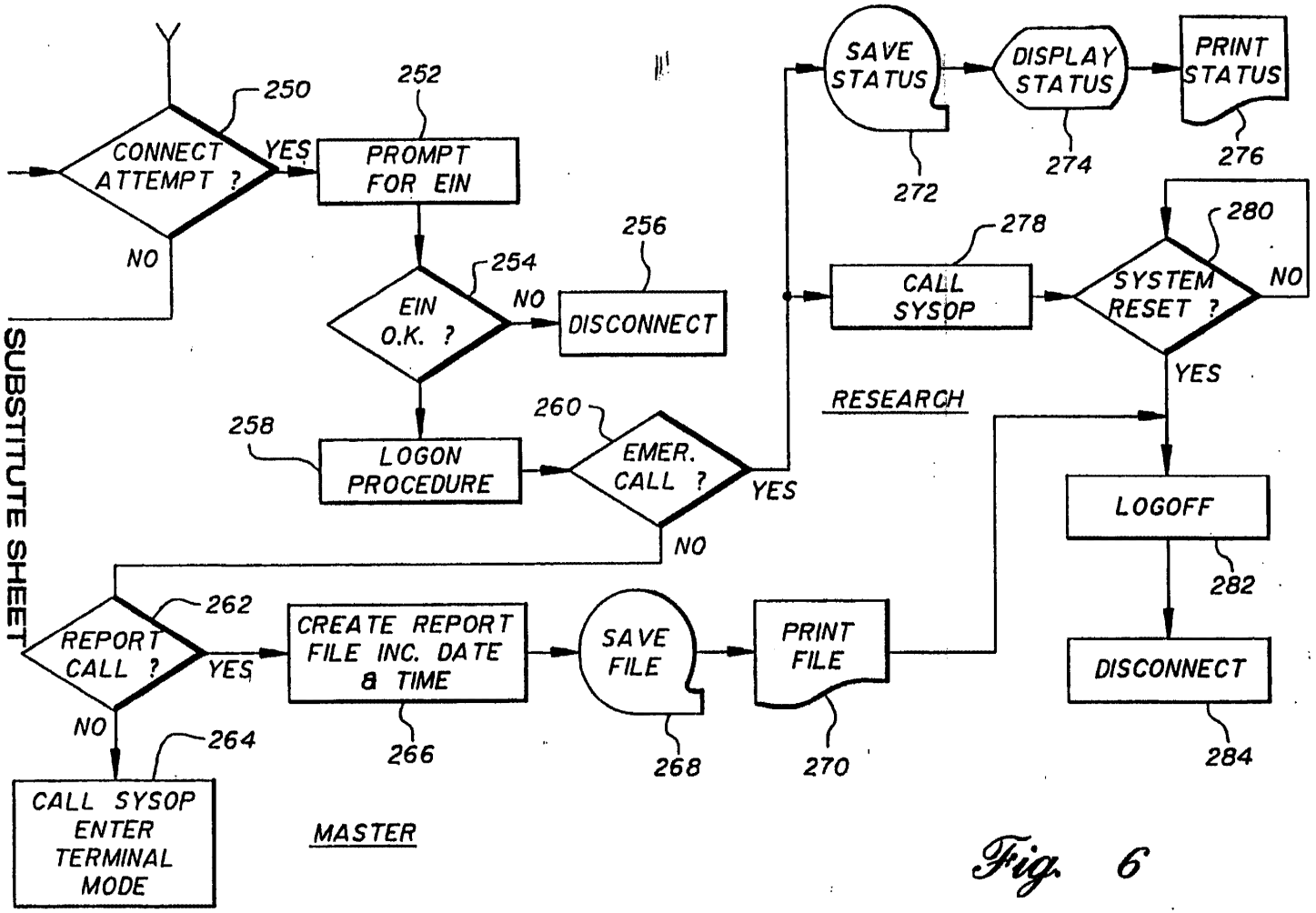


Fig. 6

IPR2024-00027
CrowdStrike Exhibit 1002 Page 291

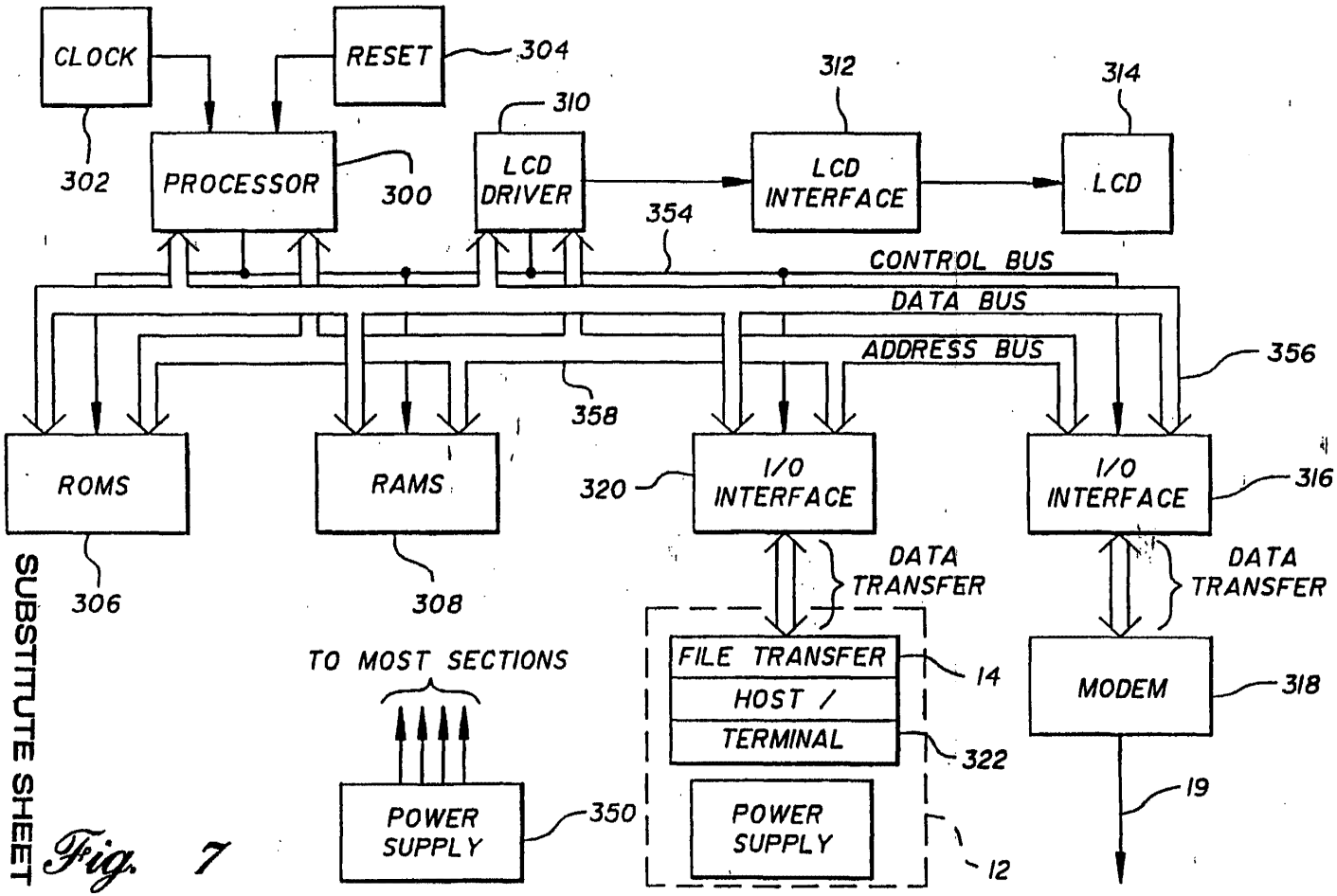


Fig. 7

BLOCK DIAGRAM OF SECURITY DEVICE

IPR2024-00027
CrowdStrike Exhibit 1002 Page 292

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/05029

A. CLASSIFICATION OF SUBJECT MATTER
IPC(5) :H04L 9/00
US CL :395/575
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)

U.S. : 380/24,18,4,25; 371/14,19,67.1

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 practicable, search terms used)

AUTOMATED PATENT SEARCH:
Search terms: Virus and (computer? or program? or microprocessor? or software)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US, A, 4,975,950 (Lentz) 04 December 1990, see Abstract; Figures 1 and 2; and col. 1, lines 1 - col. 4, lines 51.	1-18
P,Y	US, A, 5,121,345 (Lentz) 09 June 1992, see Abstract; Figures 1 and 2; and col. 1, lines 1 - col. 4, line 60.	1-18
P,A	US, A, 5,144,660 (Rose) 01 September 1992, see Abstract; col. 1- col. 2, line 17.	1-18
P,A	US, A, 5,163,088 (LoCascio) 10 November 1992, see Abstract; Figures 1 and 2; and col. 4, lines 39048.	1-18

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be part of particular relevance
"E" earlier document 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
09 JULY 1993	21 SEP 1993
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. NOT APPLICABLE	Authorized officer LY V. HUA Telephone No. (703) 305-9684

Form PCT/ISA/210 (second sheet)(July 1992)*

IPR2024-00027
CrowdStrike Exhibit 1002 Page 293

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/00155

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 11/30
US CL : 713/189, 200

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)

U.S. : 713/189, 200

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 practicable, search terms used)

West, Proquest, INSPECT, Internet, Derwint

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,359,659 A (ROSENTHAL) 25 Octorber 1994, Columns 9-14	1-40
A	US 6,230,288 B1 (KUO et. al.) 08 May 2001 Column 4-6	1-40
X	US 6,266,773 B1 (KISOR, et. al.) 24 July 2001 Columns 3-6	1-40
A	US 5,987,610 A (FRANCZEK et. al.) 16 November 1999, Column 5-6	1-40
A	WO 93/25024 (BRANHAM) 09 December 1993, pages 6-10	1-40

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"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
"A"	document defining the general state of the art which is not considered to be of particular relevance	"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
"E"	earlier document published on or after the international filing date	"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
"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)	"&"	document member of the same patent family
"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		

Date of the actual completion of the international search	Date of mailing of the international search report
09 APRIL 2003	15 MAY 2003
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer for GAIL HAYES

FORM PTO-1449, Adapted

JUN 25 2003 JC26 PATENT AND TRADEMARK OFFICE

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO. 05456.1[illegible]	SERIAL NO. 10/336,299	FILING DATE January 3, 2003
APPLICANT Satterlee et al.		GROUP 2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	AA	5,359,659	10/25/1994	Doren Rosenthal			6/19/1992
	AB	5,987,610	11/16/1999	Franczek et al.			2/12/1998
	AC	6,230,288 B1	5/8/2001	Kuo et al.			10/29/1998
	AD	6,266,773 B1	7/24/2001	Kisor et al.			12/31/1998
	AE						
	AF						
	AG						
	AH						
	AI						
	AJ						
	AK						

RECEIVED
JUN 26 2003
Technology Center 2100

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.
	AL	WO 93/25024	5/26/1993	PCT	Reginald K. Branham		
	AM						
	AN						
	AO						

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	AP		International Search Report dated May 15, 2003
	AQ		

EXAMINER	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.

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE



In re Application of:	)
	)
Satterlee et al.	)
	)
Serial No.: **Not Yet Assigned**	)
	)
Filed: **January 3, 2003**	)
	)
For: **System and Method for the Managed Security Control of Processes on a Computer System**	)

INFORMATION DISCLOSURE STATEMENT

Box Patent Application
Commissioner for Patents
Washington, DC 20231

Sir:

The citation of information on the attached Form PTO-1449, "List of Art Cited by Applicant" is made pursuant to 37 C.F.R. §§ 1.56, 1.97, and 1.98. A copy of each cited item is enclosed.

The citation of this information does not constitute an admission of priority or that any cited item is available as a reference, or a waiver of any right the applicant may have under applicable statutes, Rules of Practice in patent cases, or otherwise.

Respectfully submitted,

Robert T. Neufeld
Reg. No. 48,394

King & Spalding LLP
45th Floor, 191 Peachtree Street, N.E.
Atlanta, GA 30303
404.572.4600
K&S Docket: 05456.105038 US
3088899 v1

I certify that this correspondence is being deposited with the United States Postal Service as Express Mail with Express Mail #EU 468794582 US in an envelope addressed to: Commissioner for Patents, Box Patent Application Washington, DC 20231, on the date given below.

1/3/03
Robert T. Neufeld, Reg. No. 48,394

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

JC996 U.S. PTO 10/336299 01/03/03

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038 US	Not Yet Assigned	January 3, 2003
APPLICANT		**GROUP**
Satterlee et al.		

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	AA						
	AB						
	AC						
	AD						
	AE						
	AF						
	AG						
	AH						
	AI						
	AJ						
	AK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.
	AL						
	AM						
	AN						
	AO						

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	AP		NXi Communications, Inc.; *White Paper "NTS Security Issues;"* October 15, 2001; pp. 1 - 12
	AR		

EXAMINER	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.

OIPE JC37 MAR 24 2005 PATENT & TRADEMARK OFFICE

2131

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	
Thomas Satterlee et al.	Conf. No.: **7850**
Serial No.: **10/336,299**	Art Unit No.: **2131**
Filed: **January 3, 2003**	Examiner: **Unknown**
For: **System and Method for the Managed Security Control of Processes on a Computer System**	Docket: **05456.105038**

STATUS INQUIRY

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

Sir:

Applicant respectfully requests information as to the status of the above-referenced application.

The application was filed on January 3, 2003. The last communication received from the Patent Office in connection with the referenced application was the Notice of Publication of Application dated February 5, 2004. To date, no other communication from the Patent Office has been received. Applicant respectfully requests such status information.

Respectfully submitted,

Robert T. Neufeld
Reg. No. 48,394

King & SpaldingLLP
191 Peachtree Street, 45th Floor
Atlanta, Georgia 30303
Tel. (404) 572-4600

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on March 21, 2005.

Robert T. Neufeld, Reg. No. 48394

3718603_1.DOC

IPR2024-00027
CrowdStrike Exhibit 1002 Page 298

OIPE IAP2
JAN 23 2006
PATENT & TRADEMARK OFFICE

2131

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:		Art Unit:	**2131**
Thomas Satterlee et al.		Examiner:	**Unknown**
Serial No.:	**10/336,299**	Docket No.:	**05456.105038**
Filed:	**January 3, 2003**	Conf. No.:	**7850**
For:	**System and Method for the Managed Security Control of Processes of a Computer**		

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT ("IDS") WITH AUTHORIZATION TO CHARGE DEPOSIT ACCOUNT THE APPROPRIATE FEE TO CONSIDER THE IDS

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

Sir:

Pursuant to 37 C.F.R. §§ 1.97 and 1.98, Applicants hereby submit an Information Disclosure Statement ("IDS") for consideration by the Examiner. The patents, publications or other information submitted for consideration by the Patent Office are listed on Form PTO-1449, attached hereto. Submitted herewith is a legible copy of (i) each U.S. and foreign patent; (ii) each publication or that portion which caused it to be listed; and (iii) all other information or that portion which caused it to be listed.

Except as may be indicated below, all of the patents, publications or other information are in the English language, a copy of which is attached hereto and therefore, a concise explanation of each reference is not required. The following additional information is provided for the Examiner's consideration.

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on January 20, 2006.

3974805_1.DOC

Steven P. Wigmore, Reg. No. 40,447

IPR2024-00027
CrowdStrike Exhibit 1002 Page 299

To assist the Examiner in reviewing this IDS, the Applicants have conducted a key word search of each U.S. patent and U.S. patent application publication listed on the attached Form PTO-1449. The Applicants used the following search terms in his query of the U.S. patents and U.S. patent application publications: (1) "securit$"; (2) "(suspend$ or postpon$ or stopp$ or stop or stops)"; (3) "progra$"; (4) "(check$ or verif$ or compar$)", and (5) "execut$." The Applicants used Boolean logic when performing this search in which each term was linked by the logical connector "AND." A search of these U.S. patents and U.S. patent application publications using the aforementioned terms and Boolean search logic yielded the following results (42 U.S. Patents and 4 U.S. Patent Application Publications):

U.S. PATENTS
U.S. Patent No. 5,032,979 to Hecht et al.
U.S. Patent No. 5,440,723 to Arnold et al.
U.S. Patent No. 5,606,668 to Shwed
U.S. Patent No. 5,696,486 to Poliquin et al.
U.S. Patent No. 5,764,887 to Kells et al.
U.S. Patent No. 5,832,208 to Chen et al.
U.S. Patent No. 5,835,726 to Shwed et al.
U.S. Patent No. 5,864,665 to Tran
U.S. Patent No. 5,892,903 to Klaus
U.S. Patent No. 5,905,859 to Holloway et al.
U.S. Patent No. 5,931,946 to Terada et al.
U.S. Patent No. 5,974,457 to Waclawsky et al.
U.S. Patent No. 5,983,348 to Ji
U.S. Patent No. 5,987,611 to Freund
U.S. Patent No. 5,991,881 to Conklin et al.
U.S. Patent No. 6,009,467 to Ratcliff et al.
U.S. Patent No. 6,016,553 to Schneider et al.
U.S. Patent No. 6,070,244 to Orchier et al.
U.S. Patent No. 6,073,172 to Frailong et al.
U.S. Patent No. 6,092,194 to Touboul
U.S. Patent No. 6,119,165 to Li et al.
U.S. Patent No. 6,154,844 to Touboul et al.
U.S. Patent No. 6,167,520 to Touboul
U.S. Patent No. 6,199,181 to Rechef et al.
U.S. Patent No. 6,226,372 to Beebe et al.
U.S. Patent No. 6,272,641 to Ji
U.S. Patent No. 6,275,942 to Bernhard et al.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 300

U.S. PATENTS
U.S. Patent No. 6,289,457 to Bishop et al.
U.S. Patent No. 6,298,445 to Shostack et al.
U.S. Patent No. 6,324,647 to Bowman-Amuah
U.S. Patent No. 6,405,364 to Bowman-Amuah
U.S. Patent No. 6,460,141 to Olden
U.S. Patent No. 6,470,449 to Blandford
U.S. Patent No. 6,477,585 to Cohen et al.
U.S. Patent No. 6,496,858 to Frailong et al.
U.S. Patent No. 6,574,737 to Kingsford et al.
U.S. Patent No. 6,601,190 to Meyer et al.
U.S. Patent No. 6,606,744 to Mikurak
U.S. Patent No. 6,647,400 to Moran
U.S. Patent No. 6,661,904 to Sasich et al.
U.S. Patent No. 6,725,377 to Kouznetsov
U.S. Patent No. 6,912,676 to Gusler et al.

U.S. PATENT APPLICATIONS PUBLICATIONS
U.S. Patent Application Publication No. 2003/0037136 to Labovitz et al.
U.S. Patent Application Publication No. 2002/0138753 to Munson
U.S. Patent Application Publication No. 2002/0083331 to Krumel
U.S. Patent Application Publication No. 2001/0034847 to Gaul

The Applicants submit that these search terms cannot be interpreted as providing emphasis on these terms when the claims are interpreted and construed by both the Examiner and the courts. The Applicants merely selected these terms to identify the prior art that may be more relevant than prior art that do not contain these search terms.

The Applicants exerted this additional time and effort to help the Examiner so that if a patent is ever issued on this patent application, third parties will understand that the Applicants submitted the attached prior art on a good faith basis. The Applicants recognize that the attached prior art is voluminous but it is not the Applicants' intent to "bury" any patents that maybe more pertinent than other ones. The Applicants do not want to dismiss any reference that they are is aware of as cumulative in order to prevent third parties from arguing that the Applicants intentionally did not submit any prior art in an effort to deceive the Examiner and the Patent Office.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 301

Commissioner is authorized to charge any underpayment of fees to Deposit Account No. 11-0980.

If the Examiner has any questions concerning this IDS, he is requested to contact the undersigned. If it is determined that this IDS has been filed under the wrong rule, the Patent Office is requested to consider this IDS under the proper rule (with a petition if necessary) and charge the appropriate fee to Deposit Account No. 11-0980.

Respectfully submitted,

KING & SPALDING LLP

By_____

Steven P. Wigmore, Esq.
Reg. No. 40,447
Tel. (404) 572-4600

Enclosures:
1. Form PTO-1449
2. Foreign Patents
3. Publications
4. Postcard

IPR2024-00027
CrowdStrike Exhibit 1002 Page 302

OIPE IAP2 JAN 23 2006 PATENT AND TRADEMARK OFFICE

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	AA	5,696,486	12/9/97	Poliquin et al.	340		
	AB	6,353,385	3/5/02	Molini et al.	340		
	AC	5,353,393	10/4/94	Bennett et al.	345		
	AD	6,429,952	8/6/02	Olbricht	358		
	AE	5,748,098	5/5/98	Grace	370		
	AF	6,396,845	5/28/02	Sugita	370		
	AG	6,711,127	3/23/04	Gorman et al.	370		
	AH	5,734,697	3/31/98	Jabbarnezhad	379		
	AI	6,104,783	8/15/00	DeFino	379		
	AJ	6,226,372	5/1/01	Beebe et al.	379		
	AK	6,445,822	9/3/02	Crill et al.	382		
	AL	6,563,959	5/13/03	Troyanker	382		
	AM	6,618,501	9/9/03	Osawa et al.	382		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 303

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	AN	6,628,824	9/30/03	Belanger	382		
	AO	6,647,139	11/11/03	Kunii et al.	382		
	AP	6,661,904	12/9/03	Sasich et al.	382		
	AQ	6,668,082	12/23/03	Davison et al.	382		
	AR	6,668,084	12/23/03	Minami	382		
	AS	6,792,144	9/14/04	Yan et al.	382		
	AT						
	AU	6,278,886	8/21/01	Hwang	455		
	AV	6,889,168	05/03/05	Hartley et al.	702		
	AW	5,210,704	5/11/93	Husseiny	702		
	AX	5,842,002	11/24/98	Schnurer et al.	703		
	AY	5,854,916	12/29/98	Nachenberg	703		
	AZ	5,999,723	12/7/99	Nachenberg	703		
	BA	6,067,410	5/23/00	Nachenberg	703		
	BB	5,864,803	1/26/99	Nussbaum	704		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 304

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	BC						
	BD	6,584,454	6/24/03	Hummel, Jr. et al.	705		
	BE	5,675,711	10/7/97	Kephart et al.	706		
	BF	5,907,834	5/25/99	Kephart et al.	706		
	BG	5,649,185	7/15/97	Antognini et al.	707		
	BH						
	BI	5,878,420	3/2/99	de la Salle	707		
	BJ	5,899,999	5/4/99	De Bonet	707		
	BK	6,144,961	11/7/00	de la Salle	707		
	BL	6,161,109	12/12/00	Matamoros et al.	707		
	BM	6,282,546	8/28/01	Gleichauf et al.	707		
	BN	6,463,426	10/8/02	Lipson et al.	707		
	BO	6,647,400	11/11/03	Moran	707		
	BP	4,223,380	9/16/80	Antonaccio et al.	709		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 305

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	BQ	5,309,562	5/3/94	Li	709		
	BR	5,347,450	9/13/94	Nugent	709		
	BS	5,371,852	12/6/94	Attanasio et al.	709		
	BT	5,515,508	5/7/96	Pettus et al.	709		
	BU	5,539,659	7/23/96	McKee et al.	709		
	BV						
	BW	5,706,210	1/6/98	Kumano et al.	709		
	BX	5,745,692	4/28/98	Lohmann, II et al.	709		
	BY	5,790,799	8/4/98	Mogul	709		
	BZ						
	CA						
	CB						
	CC	5,922,051	7/13/99	Sidey	709		
	CD	5,715,395	02/03/98	Brabson et al.	709		
	CE	6,220,768	04/24/01	Barroux	709		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 306

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	CF						
	CG	5,974,237	10/26/99	Shurmer et al.	709		
	CH	5,974,457	10/26/99	Waclawsky et al.	709		
	CH						
	CJ	6,009,467	12/28/99	Ratcliff et al.	709		
	CK						
	CL	6,035,323	3/7/00	Narayen et al.	709		
	CM						
	CN	6,052,709	4/18/00	Paul	709		
	CO	6,070,190	5/30/00	Reps et al.	709		
	CP	6,073,172	6/6/00	Frailong et al.	709		
	CQ						
	CR	6,119,165	9/12/00	Li et al.	709		
	CS	6,195,687	2/27/01	Greaves et al.	709		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 307

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	CT	6,377,994	4/23/02	Ault et al.	709		
	CU	6,397,245	5/28/02	Johnson, II et al.	709		
	CV	6,415,321	7/2/02	Gleichauf et al.	709		
	CW	6,434,615	8/13/02	Dinh et al.	709		
	CX	6,438,600	8/20/02	Greenfield et al.	709		
	CY	6,453,345	9/17/02	Trcka et al.	709		
	CZ	6,453,346	9/17/02	Garg et al.	709		
	DA	6,484,203	11/19/02	Porras et al.	709		
	DB	6,496,858	12/17/02	Frailong et al.	709		
	DC	6,519,647	2/11/03	Howard et al.	709		
	DD	6,708,212	3/16/04	Porras et al.	709		
	DE	6,711,615	3/23/04	Porras et al.	709		
	DF	6,718,383	4/6/04	Hebert	709		
	DG	2002/0032793	3/14/02	Malan et al.	709		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	DH	2003/0037136	2/20/03	Labovitz et al.	709		
	DI	4,400,769	8/23/83	Kaneda et al.	710		
	DJ	5,511,184	4/23/96	Lin	710		
	DK	6,467,002	10/15/02	Yang	710		
	DL	5,991,856	11/23/99	Spilo et al.	711		
	DM	6,324,627	11/27/01	Kricheff et al.	711		
	DN	4,975,950	12/4/90	Lentz	713		
	DO	5,032,979	7/16/91	Hecht et al.	713		
	DP	5,121,345	6/9/92	Lentz	713		
	DQ	5,204,966	4/20/93	Wittenberg et al.	713		
	DR	5,274,824	12/28/93	Howarth	713		
	DS						
	DT	5,311,593	5/10/94	Carmi	713		
	DU						
	DV						
	DW						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

EXAMINER	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.

3974796_1.DOC

IPR2024-00027
CrowdStrike Exhibit 1002 Page 309

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	DX	5,557,742	9/17/96	Smaha et al.	713		
	DY						
	DZ						
	EA	5,623,600	4/22/97	Ji et al.	713		
	EB						
	EC	5,696,822	12/9/97	Nachenberg	713		
	ED						
	EE						
	EF						
	EG						
	EH						
	EI						
	EJ	5,812,763	9/22/98	Teng	713		
	EK						
	EL	5,822,517	10/13/98	Dotan	713		
	EM	6,851,057	02/01/05	Nachenberg	713		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	EN	5,826,013	10/20/98	Nachenberg	713		
	EO						
	EP						
	EQ						
	ER						
	ES	5,845,067	12/1/98	Porter et al.	713		
	ET	5,848,233	12/8/98	Radia et al.	713		
	EU						
	EV						
	EW	5,892,903	4/6/99	Klaus	713		
	EX	5,919,257	7/6/99	Trostle	713		
	EY	5,919,258	7/6/99	Kayashima et al.	713		
	EZ	5,925,126	7/20/99	Hsieh	713		
	FA	5,931,946	8/3/99	Terada et al.	713		
	FB	5,940,591	8/17/99	Boyle et al.	713		
	FC	6,886,102	04/26/05	Lyle	713		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 311

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	FD	5,961,644	10/5/99	Kurtzberg et al.	713		
	FE	5,978,917	11/2/99	Chi	713		
	FF						
	FG	5,983,350	11/9/99	Minear et al.	713		
	FH						
	FI						
	FJ						
	FK	5,991,881	11/23/99	Conklin et al.	713		
	FL	5,999,711	12/7/99	Misra et al.	713		
	FM	6,003,132	12/14/99	Mann	713		
	FN						
	FO	6,070,244	5/30/00	Orchier et al.	713		
	FP	6,081,894	6/27/00	Mann	713		
	FQ	6,493,752	12/10/02	Lee et al.	713		
	FR	5,905,859	05/18/99	Holloway et al.	713		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 312

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	FS						
	FT	6,088,804	7/11/00	Hill et al.	713		
	FU	6,092,194	7/18/00	Touboul	713		
	FV						
	FW						
	FX	6,122,738	9/19/00	Millard	713		
	FY	6,154,844	11/28/00	Touboul et al.	713		
	FZ	6,167,520	12/26/00	Touboul	713		
	GA	6,185,689	2/6/01	Todd, Sr. et al.	713		
	GB	6,205,552	3/20/01	Fudge	713		
	GC						
	GD						
	GE	6,272,641	8/7/01	Ji	713		
	GF						
	GG	6,275,942	8/14/01	Bernhard et al.	713		
	GH	6,279,113	8/21/01	Vaidya	713		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	GI	6,298,445	10/2/01	Shostack et al.	713		
	GJ	6,301,668	10/9/01	Gleichauf et al.	713		
	GK	6,314,520	11/6/01	Schell et al.	713		
	GL	6,321,338	11/20/01	Porras et al.	713		
	GM	6,324,647	11/27/01	Bowman-Amuah	713		
	GN	6,338,141	1/8/02	Wells	713		
	GO	6,347,374	2/12/02	Drake et al.	713		
	GP	6,357,008	3/12/02	Nachenberg	713		
	GQ						
	GR	6,408,391	6/18/02	Huff et al.	713		
	GS	6,460,141	10/1/02	Olden	713		
	GT	6,470,449	10/22/02	Blandford	713		
	GU	6,477,648	11/5/02	Schell et al.	713		
	GV	6,477,651	11/5/02	Teal	713		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	GW	6,487,666	11/26/02	Shanklin et al.	713		
	GX	6,499,107	12/24/02	Gleichauf et al.	713		
	GY	6,510,523	1/21/03	Perlman et al.	713		
	GZ	6,519,703	2/11/03	Joyce	713		
	HA	6,530,024	3/4/03	Proctor	713		
	HB	6,546,493	4/8/03	Magdych et al.	713		
	HC	6,574,737	6/3/03	Kingsford et al.	713		
	HD	6,578,147	6/10/03	Shanklin et al.	713		
	HE	6,681,331	1/20/04	Munson et al.	713		
	HF	6,691,232	2/10/04	Wood et al.	713		
	HG	6,704,874	3/9/04	Porras et al.	713		
	HH	6,725,377	4/20/04	Kouznetsov	713		
	HI	6,725,378	4/20/04	Schuba et al.	713		
	HJ	6,775,780	8/10/04	Muttik	713		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	HK	6,792,546	9/14/04	Shanklin et al.	713		
	HL	6,816,973	11/9/04	Gleichauf et al.	713		
	HM	6,839,850	1/4/05	Campbell et al.	713		
	HN	2001/0034847	10/25/01	Gaul, Jr.	713		
	HO	2002/0035698	3/21/02	Malan et al.	713		
	HP	2002/0083331	6/27/02	Krumel	713		
	HQ	2002/0083334	6/27/02	Rogers et al.	713		
	HR	2002/0138753	9/26/02	Munson	713		
	HS	2002/0144156	10/3/02	Copeland, III	713		
	HT	2003/0088791	5/8/03	Porras et al.	713		
	HU	2003/0212903	11/13/03	Porras et al.	713		
	HV	2004/0010718	1/15/04	Porras et al.	713		
	HW	4,672,609	6/9/87	Humphrey et al.	714		
	HX	4,773,028	9/20/88	Tallman	714		
	HY	5,398,196	3/14/95	Chambers	714		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	HZ	5,440,723	8/8/95	Arnold et al.	714		
	IA	5,452,442	9/19/95	Kephart	714		
	IB	5,649,095	7/15/97	Cozza	714		
	IC	5,765,030	6/9/98	Nachenberg et al.	714		
	ID	5,964,889	10/12/99	Nachenberg	714		
	IE	6,006,016	12/21/99	Faigon et al.	714		
	IF						
	IG	6,021,510	2/1/00	Nachenberg	714		
	IH	6,029,256	2/22/00	Kouznetsov	714		
	II						
	IJ	6,094,731	7/25/00	Waldin et al.	714		
	IK	6,108,799	8/22/00	Boulay et al.	714		
	IL	6,173,413	1/9/01	Slaughter et al.	714		
	IM	6,912,676	06/28/05	Gusler et al.	714		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	IN	6,199,181	3/6/01	Rechef et al.	714		
	IO						
	IP	6,314,525	11/6/01	Mahalingham et al.	714		
	IQ	6,324,656	11/27/01	Gleichauf et al.	714		
	IR	6,601,190	7/29/03	Meyer et al.	714		
	IS	2002/0032880	3/14/02	Poletto et al.	714		
	IT	5,454,074	9/26/95	Hartel et al.	715		
	IU	5,522,026	5/28/96	Records et al.	715		
	IV	6,271,840	8/7/01	Finseth et al.	715		
	IW	6,517,587	2/11/03	Satyavolu et al.	715		
	IX	6,535,227	3/18/03	Fox et al.	715		
	IY						
	IZ						
	JA	5,774,727	6/30/98	Walsh et al.	717		
	JB	5,872,978	2/16/99	Hoskins	717		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/336,299	January 3, 2003
APPLICANT		**GROUP**
Thomas Satterlee et al.		2131

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	JC						
	JD	6,118,940	9/12/00	Alexander, III et al.	717		
	JE	6,405,364	6/11/02	Bowman-Amuah	717		
	JF	6,606,744	8/12/03	Mikurak	717		
	JG	6,397,242	5/28/02	Devine et al.	718		
	JH	2002/0032717	3/14/02	Malan et al.	718		
	JI	6,477,585	11/5/02	Cohen et al.	719		
	JJ	6,721,806	4/13/04	Boyd et al.	719		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP	
Thomas Satterlee et al.	2131	BOX 1 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.
	JK						
	JL						
	JM						
	JN						
	JO	0 985 995	08/13/03	EP	International Business Machines Corp.		
	JP						
	JQ	WO 00/02115	01/13/00	PCT	PRC Inc.		
	JR	WO 99/00720	01/07/99	PCT	Telefonaktiebolaget LM Ericsson		
	JS	WO 99/50734	10/07/99	PCT	AT&T Corporation		
	JT	WO 98/41919	09/24/98	PCT	Trend Micro, Inc.		
	JU	WO 99/53391	10/21/99	PCT	Sun Microsystems, Inc.		
	JV	0 636 977	05/23/01	EP	Chambers		
	JW	WO 99/15966	04/01/99	PCT	Symantec Corporation		

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	JX		Essex, David, E-Sleuths Make Net Safe for E-Commerce, Computerworld, June 2000, pp. 1-2
	JY		Newman, David, Intrusion Detection Systems, Data Communications, 1998, pp. 1-9
	JZ		International Search Report for PCT/US02/17161 of December 31, 2002
	KA		Hyland, et al., Concentric Supervision of Security Applications: A New Security Management Paradigm Computer Security Applications Conference, 1998, pp. 59-68

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 320

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 1 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	KB		Koilpillai et al., Recon-A Tool for Incident Detection, Tracking and Response, Darpa Information Survivability Conference and Exposition, 2000, pp. 199-206
	KC		Alves-Foss, J., An Overview of SNIF: A Tool for Surveying Network Information Flow, Network and Distributed System Security, 1995, pp. 94-101
	KD		Mansouri-Samani et al., A Configurable Event Service for Distributed Systems Configurable Distributed Systems, 1996, pp. 210-217
	KE		International Search Report for PCT/US01/13769 of March 8, 2002
	KF		Jagannathan et al., System Design Document: Next-Generation Intrusion Detection Expert Systems (NIDES), Internet Citation, March 9, 1993, XP002136082, pp. 1-66
	KG		Koilpillai, Adaptive Network Security Management, DARPA NGI PI Conference, October 1998, pp. 1-27
	KH		Hiverworld Continuous Adaptive Risk Management, Hiverworld, Inc., 1999-2000, pp. 1-14
	KI		International Search Report for PCT/US02/04989of September 19, 2002
	KJ		International Search Report for PCT/US02/02917 of August 8, 2002
	KK		
	KL		
	KM		
	KN		

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 321

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 1 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	KO		
	KP		
	KQ		
	KR		
	KS		
	KT		
	KU		
	KV		Guha et al., Network Security via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposed Solution, IEEE, March 1996, pp. 603-610
	KW		Garg et al., High Level Communication Primitives for Concurrent Systems, IEEE, 1988, pp. 92-99
	KX		Hastings et al., TCP/IP Spoofing Fundamentals, IEEE, May 1996, pp. 218-224
	KY		Snapp, Signature Analysis and Communication Issues in a Distributed Intrusion Detection System, Master Thesis, University of California, Davis, California, 1991, pp. 1-40
	KZ		Guha et al., Network Security via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposed Solutions, IEEE, July 1997, pp. 40-48

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 322

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	BOX 1 / 4
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	LA		Djahandari et al., An MBone for an Application Gateway Firewall, IEEE, November 1997, pp. 72-81
	LB		Kim et al., Implementing a Secure Login Environment: A Case Study of Using a Secure Network Layer Protocol, Department of Computer Science, University of Alabama, June 1995, pp. 1-9
	LC		Satyanarayanan, Integrating Security in a Large Distributed System, Acm Transaction on Computer Systems, Vol. 7, No. 3, August 1989, pp. 47-280
	LD		
	LE		
	LF		
	LG		
	LH		
	LI		
	LJ		
	LK		

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 323

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 1 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	LV		
	LW		
	LX		Winn Schwartau, "e.Security™-Solving 'Dumb Days' With Security Visualization," e-Security, Inc., Naples, FL 34103, 2000.
	LY		Anita D'Amico, Ph.D., "Assessment of Open e-Security Platform™: Vendor-Independent Central Management of Computer Security Resources," Applied Visions, Inc., 1999.
	LZ		"e.Security™-Open Enterprise Security Management: Delivering an integrated, automated, centrally Managed Solution You Can Leverage Today and Tomorrow," e-Security, Inc., Naples, FL 34102, 1999.
	MA		"e.Security ™-Vision," e-Security, Inc., Naples, FL, 1999.
	MB		"e.Security™ -Administrator Workbench™," e-Security, Inc. Naples, FL, 1999.
	MC		"e.Security™-Fact Sheet," e-Security, Inc., Naples, FL, 1999.
	MD		"e.Security™-Open e-Security Platform™," e-Security, Inc. Naples, FL, 1999.
	ME		Babcock, "E-Security Tackles The Enterprise," July 28, 1999; Inter@ctive Week, www.Zdnet.com.
	MF		Kay Blough, "In Search of More-Secure Extranets," November 1, 1999, www.InformationWeek.com
	MG		Paul H. Desmond, "Making Sense of Your Security Tools," Software Magazine and Wiesner Publishing, www.softwaremag.com, 1999.
	MH		Kay Blough, "Extra Steps Can Protect Extranets," November 1, 1999, www. InformationWeek.com

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 324

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 1 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	MI		Sean Hao, "Software protects e-commerce - e-Security's product alerts networks when hackers attack," Florida Today, Florida.
	MJ		Scott Weiss, "Security Strategies - E-Security, Inc.," product brief, Hurwitz Group, Inc., March 24, 2000.
	MK		Sean Adee, CISA, "Managed Risk, Enhanced Response -The Positive Impact of Real-Time Security Awareness," Information Systems Control Journal, Vol. 2, 2000
	ML		"Reprint Review - The Information Security Portal - Open e-Security Platform Version 1.0", February 2000, West Coast Publishing, SC Magazine, 1999
	MM		"e.Security - "Introducing the First Integrated, Automated, and Centralized Enterprise Security Management System," white paper, e-Security, Inc., Naples, FL 34102, 1999
	MN		Ann Harrison, "Computerworld - Integrated Security Helps Zap Bugs," Feb. 21, 2000, Computerworld, Vol. 34, No. 8, Framingham, MA.
	MO		Shruti Daté, "Justice Department Will Centrally Monitor Its Systems For Intrusions," April 3, 2000, Post-Newsweek Business Information, Inc., www.gcn.com.
	MP		e.Security™ , website pages (pp 1-83), www.esecurityinc.com, e-Security, Inc., Naples, FL 34103, September 14, 2000.
	MQ		Peter Sommer, "Intrusion Detection Systems as Evidence," Computer Security Research Centre, United Kingdom.
	MR		Musman et al., System or Security Managers Adaptive Response Tool, DARPA Information Survivability Conference and Exposition, January 25, 2000, pp. 56-68
	MS		Gibson Research Corporation Web Pages, Shields Up! - Internet Connection Security Analysis, grc.com/default.htm, Laguna Hills, California, 2000

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 325

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 1 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	MT		Rouse et al., Design and Evaluation of an Onboard Computer-Based Information System fro Aircraft, IEEE Transactions of Systems, Man, and Cybernetics, Vol. SMC-12, No. 4, July/August 1982, pp. 451-463
	MU		Hammer, An Intelligent Flight-Management Aid for Procedure Execution, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-14, No. 6, November/December 1984, pp. 885-888
	MV		Mann et al., Analysis of User Procedural Compliance in Controlling a Simulated Process, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16, No. 4, July/August 1986
	MW		Todd, Signed and Delivered: An Introduction to Security and Authentication, Find Out How the Jave Security API Can Help you Secure your Code, Javaworld, Web Publishing, Inc., San Francisco, December 1, 1998, pp. 1-5
	MX		Arvind, Secure This. Inform, Association for Information and Image Management, Silver Spring, September /October 1999, pp. 1-4
	MY		Stevens, TCP/IP Illustrated, Vol. 1, 1994, pp. 247
	MZ		Lee et al., A Generic Virus Detection Agent on the Internet, IEEE, 30th Annual Hawaii International Conference on System Sciences, 1997, Vol. 4
	NA		Cutler, Inside Windows NT, 1993, Microsoft Press
	NB		Duncan, Advanced MS-Dos, 1986, Microsoft Press
	NC		McDaniel, IBM Dictionary of Computing, 1994, International Business Machines Corporation
	ND		Burd, Systems Architecture, 1998, Course Technology, Second Edition
	NE		Programmer's Guide PowerJ, 1997, Sybase

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 326

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	**GROUP**	**BOX 1 / 4**
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	NF		Swimmer et al., Dynamic detection and classification of computer viruses using general behavior patterns, 1995, Proceedings of the Fifth International Virus Bulletin Conference, Boston
	NG		Advanced Virus Detection Technology for the Next Millennium, August 1999, Network Associates, A Network Associates Executive White Paper, pp. 1-14
	NH		Enterprise-Grade Anti-Virus Automation in the 21st Century, June 2000, Symantec, Technology Brief, pp. 1-17
	NI		Kephart et al., Blueprint for a Computer Immune System, 1997, Retrieved from Internet, URL: http//www.research.ibm.com/antivirus/scipapers/kephart/VB97, pp. 1-15
	NJ		Richardson, Enterprise Antivirus Software, February 2000, Retrieved from Internet, URL: http://www.networkmagazine.com/article/nmg20000426S0006, pp. 1-6
	NK		Understanding and Managing Polymorphic Viruses, 1996, Symantec, The Symantec Enterprise Papers, Vol. XXX, pp. 1-13
	NL		Gong, JavaTM Security Architecture (JDK1.2), October 2, 1998, Sun Microsystems, Inc., Version 1.0, pp. i-iv, 1-62
	NM		Softworks Limited VBVM Whitepaper, November 3, 1998, Retrieved from the Internet, URL: http://web.archive.org/web/19981203105455/http://softworksltd.com/vbvm.html, pp. 1-4
	NN		Kephart, A Biologically Inspired Immune System for Computers, 1994, Artificial Life IV, pp. 130-139
	NO		International Search Report for PCT/US01/26804 of March 21, 2002
	NP		Kosoresow et al., Intrusion Detection via System Call Traces, IEEE Software, pp. 35-42, September/October 1997
	NQ		Veldman, Heuristic Anti-Virus Technology, Proceedings, 3rd International Virus Bulletin Conference, pp. 67-76, September 1993

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 327

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	BOX 1 / 4
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	NR		Symantec, Understanding Heuristics: Symantec's Bloodhound Technology, Symantec White Paper Series, Vol. XXXIV, pp. 1-14, September 1997
	NS		Nachenberg, A New Technique for Detecting Polymorphic Computer Viruses, A thesis submitted in partial satisfaction of the requirements for the degree Master of Science in Computer Science, University of California Los Angeles, pp. 1-127, 1995
	NT		Microsoft P-Code Technology, http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnarvc/html/msdn_c7pcode2.asp, pp. 1-6, April 1992
	NU		DJGPP COFF Spec, http://delorie.com/djgpp/doc/coff/, pp. 1-15, October 1996
	NV		Natvig, Sandbox Technology Inside AV Scanners, Virus Bulletin Conference, September 2001, pp. 475-488
	NW		Norman introduces a new technique for eliminating new computer viruses, found on Norman's website, file://c:/documents%20and%20settings\7489\local%20settings\temporary%20internet%20files\olk, pp. 1-2, published October 25, 2001, printed from website December 27, 2002
	NX		International Search Report for PCT/US01/19142 of January 17, 2003
	NY		Using the CamNet BBS FAQ, http://www.cam.net.uk/manuals/bbsfaq/bbsfaq.htm, January 17, 1997.
	NZ		Express Storehouse Ordering System, "Accessing ESOS through the Network", http://www-bfs.ucsd.edu/mss/esos/man3.htm, September 3, 1996.
	OA		NASIRE, NASIRC Bulletin #94-10, http://cs-www.ncsl.nist.gov/secalert/nasa/nasa9410.txt, March 29, 1994.
	OB		Packages in the net directory, http://linux4u.jinr.ru/usoft/WWW/www_debian.org/FTP/net.html, March 20, 1997.
	OC		Sundaram, An Introduction to Intrusion Detection, Copyright 1996, published at www.acm.org/crossroads/xrds2-4/intrus.html, pp. 1-12

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 328

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	BOX 1 / 4
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	OD		Samfat, IDAMN: An Intrusion Detection Architecture for Mobile Networks, IEEE Journal on Selected Areas in Communications, Vol. 15, No. 7, September 1997, pp. 1373-1380
	OE		INFO: Visual Basic Supports P-Code and Native Code Compilation (Q229415), http://support.micorsoft.com/support/kb/articles/Q229/4/15.ASP, pp. 1-2, April 28, 1999
	OF		International Search Report for PCT/US99/29117 of May 2, 2000

EXAMINER	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.

3974796_1.DOC

IPR2024-00027
CrowdStrike Exhibit 1002 Page 329

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	**GROUP**	**BOX 1 / 4**
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.
	OG	WO 99/13427	3/18/99	PCT	MCI Worldcom, Inc.		
	OH	WO 99/57626	11/11/99	PCT	Acquis Technology, Inc.		
	OI	WO 00/10278	2/24/00	PCT	Cryptek Secure Communications, LLC		
	OJ	WO 00/25214	5/4/00	PCT	Securecoft, Inc.		
	OK	WO 00/25527	5/4/00	PCT	Telefonaktiebolaget LM Ericsson		
	OL	WO 00/34867	6/15/00	PCT	Network Ice Corporation		
	OM	WO 02/101516	12/19/02	PCT	Intruvert Networks, Inc.		

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

EXAMINER	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.

3974796_1.DOC

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 2 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	ON		Nordin, U of MN OIT Security and Assurance, February 9, 2000
	OO		Internet Security Systems, RealSecure SiteProtector, SAFEsuite Decisions to SiteProtector Migration, August 8, 2003, Pages 1-42
	OP		Internet Security Systems, SAFEsuite Enterprise, SAFEsuite Decisions, 1998
	OQ		Internet Security Systems, SAFEsuite Enterprise, Recognizing the Need for Enterprise Security: An Introduction to SAFEsuite Decisions, August 1998, Pages 1-9
	OR		Internet Security Systems, SAFEsuite Decisions 2.6, Frequently Asked Questions, February 21, 2001, Pages 1-10
	OS		Internet Security Systems, SAFEsuite Decisions Version 1.0, User's Guide, 1998, Pages 1-78
	OT		Porras et al., Emerald: Event Monitoring Enabling Responses to Anomalous Live Disturbances, Computer Science Laboratory, SRI International, Menlo Park, CA, October 1997, Pages 353-365
	OU		Cisco Systems, Empowering the Internet Generation, 1998
	OV		Messmer, Start-Up Puts Hackers on BlackICE, Network World Fusion, http://www.nwfusion.com/cgi-bin/mailto/x/cgi, April 21, 1999, Page 1-2
	OW		NeworkICE Corporation, Can You Explain How Your Product Can Protect a Remote User with a VPN Client?, 1998-1999, Pages 1-2, http://www.webarchive.org/web/20000304071415/advice.networkice.com/advice/support/kb/q000003/default
	OX		Yasin, Start-Up Puts Network Intruders on Ice, http://www.internetweek.com/story/INW19990505S0001, May 5, 1999, Pages 1-2

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 331

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	BOX 2 / 4
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	OY		Morency, NetworkWorldFusion, http://nwfusion.com/cgi-bin/mailto/x.cgi, June 28, 1999, Pages 1-2
	OZ		Rogers, Network ICE Touts Security Wares, April 23, 1999, San Mateo, California, http://www.crn.com/showArticle.jhtml?articleID=18829106&flatPage=true, Pages 1-2
	PA		Rogers, Network ICE Signs Resellers, May 26, 1999, San Mateo, California, http://www.crn.com/showArticle.jhtml?articleID=18805302&flatPage=true, Pages 1-2
	PB		Internet Security Systems, I've Been Attacked! Now What?, August 27, 1999, http://www.iss.net/security_center/advice/Support/KB/q000033/default.htm, Pages 1-2
	PC		Internet Security Systems, What is the Format of "Attack-List.CSV"?, August 21, 1999, http://www.iss.net/security_center/advice/Support/KB/q000018/default.htm, Pages 1-2
	PD		Neumann et al., Experience with Emerald to Date, April 11-12, 1999, 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California, Pages 1-9
	PE		Lindqvist et al., Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST), May 9-12, 1999, Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, California, Pages 1-16
	PF		Kendall, A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems, June 1999, Department of Defense Advanced Research Projects Agency, Pages 1-124
	PG		Neumann, Computer Security and the U.S. Infrastructure, November 6, 1997, Congressional Testimony, Pages 1-11
	PH		Porras et al., Life Traffic Analysis of TCP/IP Gateways, November 10, 1997, Internet Society's Networks and Distributed Systems Security Systems Symposium, March 1998, http://www.sdl.sri.com/projects/emerald/live-traffic.html, Pages 1-16

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 332

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	**GROUP**	**BOX 2 / 4**
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	PI		Raynaud et al., Integrated Network Management IV, 1995, Proceedings of the 4th International Symposium on Integrated Network Management, Pages 1-2 and 5-16
	PJ		Heberlein et al., A Method to Detect Intrusive Activity in a Networked Environment, October 1-4, 1991, 14th National Computer Security Conference, Washington, D.C., Pages 362-363 and 365-371
	PK		Ko et al., Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-Based Approach, 1997, Proceedings of the 1997 IEEE Symposium on Security and Privacy, Pages 175-187
	PL		Crosbie et al., Active Defense of a Computer System Using Autonomous Agents, Technical Report No. 95-008, February 15, 1995, Purdue University, West Lafayette, Indiana, Pages 1-14
	PM		Mansouri-Samani et al., Monitoring Distributed Systems, November 1993, IEEE Network, Pages 20-30
	PN		Jakobson et al., Alarm Correlation, November 1993, IEEE Network, Pages 52-59
	PO		Anderson et al., Next-Generation Intrusion Detection Expert (NIDES), A Summary, May 1995, SRI International, Pages 1-37
	PP		Veritas Software, Press Release, Robust Enhancements in Version 6.0 Maintain Seagate WI as the De Facto Standard for Software Distribution, October 6, 1997, Press Releases, Pages 1-4, http://216.239.39.104/search?q=cache:HS9kmK1m2QoJ:www.veritas.com/us/aboutus/pressroom/199....
	PQ		Yasin, Network-Based IDS are About to Stop Crying Wolf, Security Mandate: Silence False Alarms, April 9, 1999, http://lists.jammed.com/ISN/1999/04/0021.html, Pages 1-3
	PR		Internet Security Systems, Press Release, ISS Reports Record Revenues and Net Income for Second Quarter, July 19, 1999, http://bvlive01.iss.net/issEn/delivery/prdetail.jsp?type=Financial&oid=14515, Pages 1-5
	PS		LaPadula, State of the Art in CyberSecurity Monitoring, A Supplement, September 2001, Mitre Corporation, Pages 1-15

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 333

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	**GROUP**	**BOX 2 / 4**
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	PT		Balasubramaniyan et al., An Architecture for Intrusion Detection Using Autonomous Agents, June 11, 1998, Purdue University, West Lafayette, Indiana, Pages 1-4, http://gunther.smeal.psu.edu/images/b9/f3/bb/9e/ba7f39c3871dcedeb9abd0f70cb84607/1.png
	PU		Crosbie et al., Active Defense of a Computer System Using Autonomous Agents, February 15, 1995, Technical Report No. 95-008, Purdue University, West Lafayette, Indiana, Pages 1-14
	PV		Crosbie et al., Defending a Computer System Using Autonomous Agents, March 11, 1994, Technical Report No. 95-022, Purdue University, West Lafayette, Indiana, Pages 1-11
	PW		Denning, An Intrusion-Detection Model, February 1987, IEEE Transactions on Software Engineering, Vol. SE-13, No. 2, Pages 1-17
	PX		Lunt, A Survey of Intrusion Detection Techniques, 1993, Computers & Security, 12 (1993), Pages 405-418
	PY		Porras et al., Penetration State Transition Analysis A Rule-Based Intrusion Detection Approach, 1992, Pages 220-229
	PZ		Javitz et al., The NIDES Statistical Component: Description and Justification, SRI International, Menlo Park, California, SRI Project 3131, March 7, 1994
	QA		Lindqvist et al., Detecting Computer and Network Misuses Through the Production-Based Expert System Toolset (P-BEST), October 25, 1998, Pages 1-20
	QB		Javitz et al., The SRI IDES Statistical Anomaly Detector, SRI Internationa, Menlo Park, California, May 1991, IEEE Symposium on Security and Privacy, Pages 1-11
	QC		Porras et al., Live Traffic Analysis of TCP/IP Gateways, November 10, 1997, SRI International, Menlo Park, California, Pages 1-16

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 334

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	**GROUP**	**BOX 2 / 4**
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	QD		Porras et al., Live Traffic Analysis of TCP/IP Gateways, December 12, 1997, SRI International, Menlo Park, California, Proceedings of the 1998 ISOC Symposium on Network and Distributed Systems Security, Pages 1-13
	QE		Information & Computing Sciences: System Design Laboratory: Programs: Intrusion Detection, SRI International, http://www.sdl.sri.com/programs/intrusion/, June 17, 2004, Pages 1-2
	QF		Lindqvist et al., eXpert-BSM: A Host-based Intrusion Detection Solution for Sun Solaris, SRI International, Menlo Park, California, December 10-14, 2001, Proceedings of the 17th Annual Computer Security Applications Conference, Pages 1-12
	QG		Almgren et al., Application-Integrated Data Collection for Security Monitoring, October 10-12, 2001, SRI International, Menlo Park, California, Pages 1-15
	QH		Debar et al., Research Report: A Revised Taxonomy for Intrusion-Detection Systems, October 25, 1999, IBM Research, Switzerland, Pages 1-23
	QI		Porras et al., Emerald: Event Monitoring Enabling Responses to Anomalous Live Disturbances, Computer Science Laboratory, SRI International, Menlo Park, CA, December 18, 1996, Pages 1-3.
	QJ		Frequently-Asked Questions about RealSecure, Pages 1-14, http://web.archive.org/web/19970721183227/iss.net/prod/rs_faq.html, May 30, 1997
	QK		Cisco Systems, Inc., Empowering the Internet Generation, 1998.
	QL		Internet Security Systems, Inc., RealSecure Release 1.2 for UNIX, A User Guide and Reference Manual, 1997.
	QM		Internet Security Systems, Inc., Real-time attack recognition and response: A solution for tightening network security, January 1997, Pages 1-13.
	QN		Internet Security Systems, Inc., SAFEsuite Decisions User Guide, Version 1.0, 1998, Pages 1-78

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 335

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 2 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	QO		NetworkICE Corporation, ICEcap Administrator's Guide, Version 1.0 BETA, 1999, Pages 1-142.
	QP		Debar, Herve et al., A Neural Network Component for an Intrusion Detection System, 1992, Pages 240-250.
	QQ		SRI International, A Prototype IDES: A Real-Time Intrusion-Detection Expert System, August 1987, Page 1-63.
	QR		SRI International, Requirements and Model for IDES-A Real-Time Intrusion-Detection Expert System, August 1985, Pages 1-70.
	QS		SRI International, An Intrusion-Detection Model, November 1985, Pages 1-29.
	QT		Dowell et al., The ComputerWatch Data Reduction Tool, Proc. of the 13th National Computer Security Conference, Washington, D.C., October 1990, Pages 99-108.
	QU		Fox et al., A Neural Network Approach Towards Intrusion Detection, July 2, 1990, Pages 125-134.
	QV		Garvey et al., Model-Based Intrusion Detection, October 1991, Pages 1-14.
	QW		Ilgun et al., State Transition Analysis: A Rule-Based Intrusion Detection Approach, March 1995, Pages 181-199.

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 336

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	BOX 2 / 4
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	QX		Javitz et al., The SRI IDES Statistical Anomaly Detector, May 1991, Pages 1-11.
	QY		SRI International, The NIDES Statistical Component Description and Justification, March 7, 1994, Pages 1-46.
	QZ		Karen, Oliver, PC Magazine, The Digital Doorman, November 16, 1999, Page 68.
	RA		Liepins et al., Anomaly Detection: Purpose and Frameowrk, 1989, Pages 495-504.
	RB		Lindqvist et al., Detecting Computer and Network Misuse Through the Production-Bases Expert System Toolset (P-BEST), October 25, 1998, Pages 1-20.
	RC		Lunt, Teresa, A survey of intrusion detection techniques, 1993, Pages 405-418.
	RD		Lunt, Teresa, Automated Audit Trail Analysis and Intrusion Detection: A Survey, October 1988, Pages 1-8.
	RE		Porras et al., Penetration State Transition Analysis A Rule-Based Intrusion Detection Approach, 1992, Pages 220-229.
	RF		Sebring et al., Expert Systems in Intrusion Detection: A Case Study, October 1988, Pages 74-81.
	RG		Shieh et al., A Pattern-Oriented Intrusion-Detection Model and Its Applications, 1991, Pages 327-342.
	RH		Smaha, Stephen, Haystack: An Intrusion Detection System, 1988.
	RI		Snapp, Steven Ray, Signature Analysis and Communication Issues in a Distributed Intrusion Detection System, 1991, Pages 1-40.
	RJ		Porras et al., EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances, October 1997, Pages 353-365.

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 337

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	BOX 2 / 4
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	RK		Lunt et al., Knowledge-Based Intrusion Detection, March 1989, Pages 102-107.
	RL		Lunt et al., An Expert System to Classify and Sanitize Text, December 1987, Pages 1-5.
	RM		Tener, William, Computer Security in the Age of Information, AI & 4GL: Automated Detection and Investigation Tools, 1989, Pages 23-29.
	RN		Teng et al., Adaptive Real-time Anomaly Detection Using Inductively Generated Sequential Patterns, 1990, Pages 278-284.
	RO		Vaccaro et al., Detection of Anomalous Computer Session Activity, 1989, Pages 280-289.
	RP		Winkler, J.R., A UNIX Prototype for Intrusion and Anomaly Detection in Secure Networks, 1990, Pages 115-124.
	RQ		Boyen et al. Tractable Inference for Complex Stochastic Process, July 24-26, 1998.
	RR		Copeland, Observing Network Traffic – Techniques to Sort Out the Good, the Bad, and the Ugly, 2000, Pages 1-7.
	RS		Goan, Terrance, Communications of the ACM, A Cop on the Beat Collecting and Appraising Intrusion Evidence, July 1999, Pages 47-52.
	RT		Heberlein et al., A network Security Monitor, 1990, Pages 296-304.
	RU		Jackson et al., An Expert System Applications for Network Intrusion Detection, October 1991, Pages 1-8.
	RV		Lankewicz et al., Real-Time Anomaly Detection Using a Nonparametric Pattern Recognition Approach, 1991, Pages 80-89.

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 338

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	BOX 2 / 4
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	RW		Lippmann et al., Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation, 1999.
	RX		Munson et al., Watcher: The Missing Piece of the Security Puzzle, December 2001.
	RY		Pearl, Judea, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, September 1988.
	RZ		Porras et al., Live Traffic Analysis of TCP/IP Gateways, December 12, 1997, Pages 1-13.
	SA		EMERALD TCP Statitical Analyzer 1998 Evaluation Results, http://www.sdl.sri.com/projects/emerald/98-eval-estat/index.html, July 9, 1999, Pages 1-15.
	SB		Staniford-Chen, GrIDS-A Graph Based Intrusion Detection System for Large Networks, October 1996.
	SC		Tener, William, Discovery: An Expert System in the Commercial Data Security Environment, December 1986, Pages 45-53.
	SD		Valdes et al., Adaptive, Model-Based Monitoring for Cyber Attack Detection, 2000, Pages 1-19.
	SE		SRI International, Blue Sensors, Sensor Correlation, and Alert Fusion, October 4, 2000.
	SF		Valdes et al., Statistical Methods for Computer Usage Anomaly Detection Using NIDES, January 27, 1995, Pages 306-311.
	SG		Wimer, Scott, The Core of CylantSecure, http://www.cylant.com/products/core.html, 1999, pages 1-4.
	SH		Zhang et al., A Hierarchical Anomaly Network Intrusion Detection System using Neural Network Classification, February 2001.

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 339

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	BOX 2 / 4
Thomas Satterlee et al.	2131	

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	SI		Cisco Secure Intrusion Detection System 2.1.1 Release Notes, http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/csids3/nr211new.htm, June 10, 2003, pages 1-29.
	SJ		Linux Weekly News, http://lwn.net/1998/0910shadow.html, September 8, 1998, pages 1-38.
	SK		Cracker Tracking: Tighter Security with Intrucsion Detection, http://www.bvte.com/art/9805/sec20/art1.htm, May 1998, pages 1-8.
	SL		Cisco Systems, Inc., Newtork RS: Intrusion Detection and Scanning with Active Audit Session 1305, 1998.
	SM		Business Security Advisor Magazine, Intrusion Detection Systems: What You Need to Know, http://advisor.com/doc/0527, September 1998, pages 1-7.
	SN		Garvey et al., An Inference Technique for Integrating Knowledge from Disparate Sources, Multisensor Integration and Fusion for Intelligenct Machines and Systems, 1995, pages 458-464.
	SO		Power et al., CSI Intrusion Detection System Resource, July 1998, pages 1-7.
	SP		Cisco Systems, Inc., NetRanger User's Guide Version 2.1.1, 1998.
	SQ		Internet Security Systems, Real-Time Attack Recognition and Response: A Solution for Tightening Network Security, http://www.iss.net, 1997, Pages 1-13
	SR		Network ICE Corporation, Network ICE Product Documentation, Pages 1-3, http://www.web.archive.org/web/20011005080013/www.networkice.com/support/documentation.html, July 6, 2004
	SS		Network ICE Corporation, Network ICE Documentation, Page 1, http://www.web.archive.org/web/19991109050852/www.networkice.com/support/docs.htm, July 6, 2004
	ST		Network ICE Corporation, Network ICE Press Releases, Page 1, http://www.web.archive.org/web/19990903214428/www.netice.com/company/pressrelease/press.htm, July 7, 2004

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 340

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	**GROUP**	**BOX 2 / 4**
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	SU		Network ICE Corporation, Network ICE Press Releases, Page 1, http://www.web.archive.org/web/20000304074934/www.netice.com/company/pressrelease/press.htm, July 7, 2004
	SV		Brentano et al., An Architecture for Distributed Intrusion Detection System, Department of Energy Computer Security Group, 14th Annual Conference Proceedings, Pages (17)25-17(45), May 1991
	SW		Staniford-Chen et al., GrIDS-A Graph Based Intrusion Detection System for Large Networks, University of California, Davis, California, 19th National Information Systems Security Conference, 1996, Pages 1-10
	SX		Ricciulli et al., Modeling Correlated Alarms in Network Management Systems, SRI International, Menlo Park, California, , Proceedings of the Conference on Communication Networks and Distributed System Modeling and Simulation, 1997, Pages 1-8
	SY		Porras et al., Emerald: Event Monitoring Enabling Responses to Anomalous Live Disturbances, SRI International, Menlo Park, California, 20th National Information Systems Security Conference, October 1997, Pages 1-24
	SZ		Porras et al., A Mission-Impact Based Approach to INFOSEC Alarm Correlation, SRI International, Menlo Park, California, October 2002, Pages 1-33
	TA		Phrack 55 Download (234 kb, 1999-09-09), http://www.phrack.org/show.php?p=55&a=9, Pages 1-6
	TB		Porras et al., A Mission-Impact-Based Approach to INFOSEC Alarm Correlation, SRI International, Menlo Park, California, October 2002, Pages 1-19
	TC		Bace, An Introduction to Intrusion Detection and Assessment for System and Network Security Management, 1999, Pages 1-38

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 341

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	**GROUP**	**BOX 2 / 4**
Thomas Satterlee et al.	2131	

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	TD		Hunteman, Automated Information System - (AIS) Alarm System, University of California, Los Alamos National Laboratory, 20th National Information System Security Conference, October 1997, Pages 1-12
	TE		Janakiraman et al., Indra: A Peer-to-Peer Approach to Network Intrusion Detection and Prevention, Proceedings of the 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003, Pages 1-5
	TF		Innella, Intrusion Detection Systems (IDS), Navy Information Assurance, June 28, 2004, Pages 1-4, http://www.infosec.navy.mil/ps/?t=infosecprodsservices/infosecprodsservices.tag&bc=/infosecprodsservices/b...
	TG		Curry, Intrusion Detection Systems, IBM Emergency Response Service, Coast Laboratory, http://www.cerias.purdue.edu/about/history/coast_resources/idcontent/ids.html, July 19, 2004, Pages 1-28
	TH		Lunt et al., Knowledge-Based Intrusion Detection, SRI International, Menlo Park, California, Conference on AI Systems in Government, Washington, D.C., March 1989, Pages 102-107
	TI		A. Information Assurance BAA 98-34 Cover Sheet, SRI International, November 1, 2000, Pages 2-35
	TJ		NetScreen Products, FAQ, http://www.netscreen.com/products/faq.html, February 28, 2003, Pages 1-6
	TK		Miller, A Network Under Attack: Leverage Your Existing Instrumentation to Recognize and Respond to Hacker Attacks, NetScout Systems, Westford, MA, April 2003, Pages 1-8
	TL		Technology Brief: Intrusion Detection for the Millennium, Internet Security Systems, 2000, Pages 1-6
	TM		Weiss et al., Analysis of Audit and Protocol Data using Methods from Artificial Intelligence, Siemens AG, Munich, Germany, Proc. of the 13th National Computer Security Conference, Washington, D.C., October 1990, Pages 109-114
	TN		Snapp et al., DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and an Early Protype), University of California, Davis California, , Proc. 14th National Computer Security Conference, Washington, D.C., October 1991, Pages 167-176

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 342

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP	
Thomas Satterlee et al.	2131	BOX 2 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	TO		Internet Security Systems, Inc., SAFEsuite Enterprise Edition, Project "Gatekeeper" 1.0, Market Requirements Document, March 30, 1998, pp. 1-12
	TP		Internet Security Systems, SAFEsuite Decisions, 2000, pp. 1-2
	TQ		Internet Security Systems, Dynamic Threat Protection, Presse-Roundtable, Munich, Germany, April 10, 2003, pp. 1-63
	TR		Internet Security Systems, Preliminary ODM 1.0 Functional Specification, August 1997, pp. 1-7
	TS		Internet Security Systems, Inc., Scanner-ISSDK Interface, Design Specification, June 2, 2000, Draft 1.07, pp. 1-7
	TT		RealSecure, Adaptive Network Security Manager Module Programmer's Reference Manual, pp. 1-74
	TU		Advanced Concept Technology Demonstrations (ACTD), 2005, pp. 1-28
	TV		Frank, Sounding the Alarm, September 6, 1999, Federal Computer Week, pp. 1-2
	TW		Crumb, Intrusion Detection Systems to be Integrated at AFRL, Air Force Research Laboratory, News@AFRL, Fall 1999, pp. 1
	TX		Temin, Case Study: The IA: AIDE System at Two, 15th Annual Computer Security Applications Conference, December 6-10, 1999, Phoenix, Arizona, pp. 1-26
	TY		Spink, Automated Intrusion Detection Environment (AIDE), Intrusion Detection Sensor Fusion, September 7, 1999, pp. 1-25
	TZ		Frincke et al., A Framework for Cooperative Intrusion Detection, 21st National Information Systems Security Conference, October 6-9, 1998, Crystal City, Virginia, pp. 1-20

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 343

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 2 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	UA		Anderson et al., In Athena's Camp: Preparing for Conflict in the Information Age, An Exploration of Cyberspace Security R&D Investment Strategies for DARPA: The Day After-in Cyberspace II, Chaper Eleven, 1996, pp. 253-271
	UB		Valdes et al., SRI International, Probabilistic Alert Correlation, 2001, pp. 1-15
	UC		Bass, Multisensor Data Fusion for Next Generation Distributed Intrusion Detection Systems, April 28, 1999, Iris National Symposium, pp. 1-6
	UD		Perrochon et al., Enlisting Event Patterns for Cyber Battlefield Awareness, No Date, Stanford University, pp. 1-12
	UE		Perrochon, Using Context-Based Correlation in Network Operations and Management, July 2, 1999, Stanford University, pp. 1-20
	UF		Perrochon, Real Time Event Based Analysis of Complex Systems, June 1998, pp. 1-9
	UG		Luckham et al., Complex Event Processing in Distributed Systems, August 18, 1988, Stanford University, pp. 1-26
	UH		Pettigrew, US Southcom United States Southern Command's Information Sharing Projects, Summer 1999, IAnewsletter, Vol. 3, No. 1, pp. 1-24
	UI		Levitt et al., CMAD IV, Computer Misuse & Anomaly Detection, Session Summaries, November 12-14, 1996, Proceedings of the Fourth Workshop on Future Directions in Computer Misuse and Anomaly Detection, Monterey, California, pp. 1-86
	UJ		Cheung et al., The Design of GrIDS: A Graph-Based Intrusion Detection System, January 26, 1999, University of California, pp. 1-51

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 344

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP	
Thomas Satterlee et al.	2131	BOX 2, 3 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	UK		Cohen et al., Report of the Reporting and Analysis Track, Detection of Malicious Code, Intrusions, and Anomalous Activities Workshop, February 22-23, 1999, pp. 1-47
	UL		Garofalakis et al., Network Mining and Analysis: The Nemesis Project, Bell Labs, Lucent Technologies, No Date, pp. 1-12
	UM		RealSecure ICEcap Manager User Guide Version 3.6, Internet Security Systems, Inc., 1998-2002, pp. 1-228
	UN		Cuppens, Cooperative Intrusion Detection, Date Unknown, pp. 1-10
	UO		Mukherjee et al., Network Intrusion Detection, IEEE Network, May/June 1994, pp. 26-41
	UP		Machlis, Net Monitoring Tools Gain Real-Time Alerts, April 14, 1997, http://www.computerworld.com, pp. 1-12
	UQ		OmniGuard/ITA Intruder Alert, AXENT Technologies, Inc., November 17, 2004, http://www.web.archive.org, pp. 1-10
	UR		NetRanger Overview, Chapter 1, Date Unknown, pp. 1-16
	US		Sutterfield, Large-Scale Network Intrusion Detection, 1997, WheelGroup Corporation, pp. 1-10
	UT		Kumar et al., An Application of Pattern Matching in Intrusion Detection, Technical Report No. CSD-TR-94-013, June 17, 1994, Purdue University, pp. 1-55
	UU		Huang et al., A Large-Scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis, Date Unknown, The Boeing Company, pp. 1-12
	UV		Perrochon et al., Event Mining with Event Processing Networks, Date Unknown, Stanford University, pp. 1-4

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 345

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP	
Thomas Satterlee et al.	2131	**BOX 3 / 4**

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	UW		Gruschke, Integrated Event Management: Event Correlation Using Dependency Graphs, presented at DSOM 1998, University of Munich, pp. 1-12
	UX		Bass, Intrusion Detection System and Multisensor Data Fusion, April 2000, Communications of the ACM, Vol. 43, No. 4, pp. 99-105
	UY		Bass et al., A Glimpse into the Future of ID, Date Unknown, Usenix, pp. 1-10
	UZ		LaPadula, State of the Art in Anomaly Detection and Reaction, July 1999, Mitre Corporation, pp. 1-36
	VA		Rationalizing Security Events with Three Dimensions of Correlation, Date Unknown, NetForensics, Tech Brief, pp. 1-6
	VB		Jou et al., Design and Implementation of a Scalable Intrusion Detection System for the Protection of Network Infrastructure, Date Unknown, MCNC, pp. 1-15
	VC		Caldwell, Event Correlation: Security's Holy Grail?, August 1, 2002, GuardedNet, pp. 1-21
	VD		Metcalf et al., Intrusion Detection System Requirements, September 2000, Mitre Corporation, pp. 1-33
	VE		Jou et al., Architecture Design of a Scalable Intrusion Detection System for the Emerging Network Infrastructure, Technical Report CDRL A005, April 1997, MCNC, pp. 1-42
	VF		Security Manager for UNIX Systems Version 3.2.1 User's Guide, February 1998, Internet Security Systems, pp. 1-162
	VG		RealSecure Release 1.2 for UNIX A User Guide and Reference Manual, 1997, Internet Security Systems, Inc., pp. 1-92

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 346

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP	
Thomas Satterlee et al.	2131	BOX 3 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	VH		Internet Scanner SAFE SAFEsuite 4.0 User Guide and Reference Manual, 1996, Internet Security Systems, Inc., pp. 1-158
	VI		Internet Scanner 3.3 User Guide and Reference Manual, 1996, Internet Security Systems, Inc., pp. 1-119
	VJ		Landwehr et al., Newsletter of the IEEE Computer Society's TC on Security and Privacy Electronics, Electronic CIPHER, November 25, 1997, Issue 25, pp. 1-34
	VK		20th National Information Systems Security Conference, October 6-10, 1997, Baltimore, Maryland, pp. 1-44
	VL		EMERALD Alert Management Interface User's Guide Version 1.2, December 6, 2000, SRI International, pp. 1-11
	VM		Anderson et al., Detecting Unusual Program Behavior Using the Statistical Component of the Next-Generation Intrusion Detection Expert System (NIDES), May 1995, SRI International, pp. 1-89
	VN		Lunt et al., Detecting Intruders in Computer Systems, 1993 Conference on Auditing and Computer Technology, SRI International, pp. 1-17
	VO		Network ICE Products - ICEcap, Date Unknown, pp. 1-2
	VP		Forlanda, The Secrets to Driving on BlackICE, January 12, 2000, Network ICE, pp. 1-35
	VQ		BlackICE User's Guide Version 1.0 Beta Draft, 1999, Network ICE Corporation, pp. 1-59
	VR		ICEcap Administrator's Guide Version 1.0 Beta Draft, 1999, Network ICE Corporation, pp. 1-25
	VS		Shulak et al., ICEcap Advanced Administration Guide Version 3.0, 2001, Internet Security Systems Corporation, pp. 1-51

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 347

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP
Thomas Satterlee et al.	2131

BOX 3 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES *(Including Author, Title, Date, Pertinent Pages, Etc.)*

	VT		"Real Secure, OS Sensor User Guide," Version 5.0, ©Internet Security Systems, Inc. 199?-2000; September 2000, pgs. 1-64.
	VU		"Real Secure, User's Guide," Version 3.0, © 1992-1998, Internet Security Systems, Inc., pgs. 1-128.
	VV		"System Security Scanner, User Guide," Version 1.6, © 1996-1998, Internet Security Systems, Inc., pgs. 1-164.
	VW		"Real Secure™, Network Engine User Guide," Version 3.2.1, © 1999 by Internet Security Systems, Inc., pgs. 1-38.
	VX		"Real Secure™, User Guide," Version 3.2.1, © 1999 by Internet Security Systems, Inc., pgs. 1-38
	VY		"Real Secure™, Manager for HP OpenView User Guide," Version 1.3, © 1999 by Internet Security Systems, Inc., pgs. 1-48.
	VZ		"Database Scanner, User Guide," Version 2.0, © 1992-1999, Internet Security Systems, Inc., pgs. 1-112.
	WA		"Database Scanner™, User Guide," Version 4.0, © 2000 by Internet Security Systems, Inc., pgs. 1-122.
	WB		"Database Scanner™, User Guide," Version 3.0.1, , © 1999 by Internet Security Systems, Inc., pgs. 1-164.
	WC		"Real Secure™, Network Sensor User Guide," Version 5.0, © 2000 by Internet Security Systems, Inc., pgs. 1-42.
	WD		"Real Secure, Server Sensor User Guide," Version 5.5, © Internet Security Systems, Inc. 2000, pgs. 1-56.
	WE		"Internet Scanner™, User Guide," Version 6.0, Copyright © 1999 by Internet Security Systems, Inc., pgs. 1-182.
	WF		"Internet Scanner™, User Guide," Version 6.1, © 2000 by Internet Security Systems, Inc., pgs. 1-226.
	WG		"Internet Scanner™, User Guide," Version 5.6, © 1992-1998, Internet Security Systems, Inc., pgs. 1-162.

EXAMINER	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.

3974796_1.DOC

IPR2024-00027
CrowdStrike Exhibit 1002 Page 348

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003

APPLICANT	GROUP	
Thomas Satterlee et al.	2131	BOX 4 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	WH		"Internet Scanner™, User Guide," Version 5.3, © 1992-1998, Internet Security Systems, Inc. pgs. 1-173.
	WI		"Real Secure, Console User Guide," Version 5.5, © 199?-2000, Internet Security Systems, Inc., pgs. 1-162.
	WJ		"Internet Scanner™, User Guide," Version 5.8, © 1999 by Internet Security Systems, Inc., pgs. 1-148.
	WK		"SAFEsuite Decisions, User Guide," Version 1.0, © 1992-1998, Internet Security Systems, Inc., pgs. 1-88.
	WL		"Real Secure™, Console User Guide," Version 5.0, © 2000 by Internet Security Systems, Inc., pgs. 1-114.
	WM		"SAFEsuite DECISIONS, User Guide," Version 2.5, © 2000 by Internet Security Systems, Inc., pgs. 1-194.
	WN		"System Scanner, User Guide," Version 1.7, © 1992-1998, Internet Security Systems, Inc., pgs. 1-248.
	WO		"Sytem Scanner, User Guide," Version 1.0, © 1996-1998, Internet Security Systems, Inc., pgs. 1-140.
	WP		"System Scanner™, User Guide," Version 4.0, © 1999 by Internet Security Systems, Inc., pgs. 1-178.
	WQ		Internet Security Systems, Inc., "Introduction to RealSecure Version 5.0, The Industry's Only Integrated Host-Based and Network-Based Intrusion Detection System", August 22, 2000, pp. 1-47
	WR		Internet Security Systems, Inc., "RealSecure Network Sensor and Gigabit Network Sensor Policy Guide Version 7.0", August 2003, pp. 1-86
	WS		Internet Security Systems, Inc., "RealSecure Console User Guide Version 3.1", June 1999, pp. 1-98
	WT		Internet Security Systems, Inc., "RealSecure Version 2.0", April 1998, pp. 1- 154
	WU		Internet Security Systems, Inc., "Enhanced Dynamic Threat Protection via Automated Correlation and Analysis", an ISS White Paper, 2002, pp. 1-14

EXAMINER	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.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 349

FORM PTO-1449, Adapted

LIST OF INFORMATION DISCLOSED BY APPLICANT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.	FILING DATE
05456.105038	10/366,299	January 3, 2003
APPLICANT	GROUP	
Thomas Satterlee et al.	2131	BOX 4 / 4

U.S. PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUBCLASS	FILING DATE IF APPROPRIATE
	WV	6,871,284	03/22/05	Cooper et al.	713		
	WW	5,872,915	02/16/99	Dykes et al.	713		

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	DATE	COUNTRY	NAME	TRANSLATION YES	TRANSLATION NO.

OTHER REFERENCES (*Including Author, Title, Date, Pertinent Pages, Etc.*)

	WX		Internet Security Systems, Inc., "RealSecure Site Protector Comparison Guide for Internet Scanner 7.0 Reporting Version 1.2", March 2003, an ISS Tech Note, pp. 1-15
	WY		Internet Security System, Inc., "RealSecure Site Protector Comparison Guide for ICEcap Manager Version 1.5", January 2002, an ISS Technical White Paper, pp. 1-27
	WZ		Internet Security Systems, Inc., "RealSecure SiteProtector Security Fusion Module 2.0 Frequently Asked Questions", March 2003, pp. 1-8
	XA		Internet Security Systems, Inc., "RealSecure SiteProtector Console User Reference Guide Version 2.0 Service Pack 1", March 14, 2003, Chapter 3, pp. 1-27
	XB		Internet Security Systems, Inc., "Proventia Gateway and Network Protection", July 2003, pp. 1-12
	XC		Farley, Internet Security System, Inc., "ID Fusion Technology, A First-Generation Approach", April 29, 1999, National Security Framework Forum, pp. 1-8
	XD		Farley, "RealSecure Fusion Engine Concepts", an ISS White Paper, June 30, 1999, pp. 1-23
	XE		McGraw et al., "Secure Computing with Java: Now and the Future", 1997, http://java.sun.com/security/javaone97-whitepaper.html, pgs. 1-20

EXAMINER	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.

3974796_1.DOC

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06K 7/00 — A2

(11) International Publication Number: **WO 99/13427**

(43) International Publication Date: 18 March 1999 (18.03.99)

(21) International Application Number: PCT/US98/19039

(22) International Filing Date: 11 September 1998 (11.09.98)

(30) Priority Data:
08/928,851 12 September 1997 (12.09.97) US

(71) Applicant *(for all designated States except US)*: MCI WORLDCOM, INC. [US/US]; 515 East Amite Street, Jackson, MS 39201 (US).

(72) Inventors; and

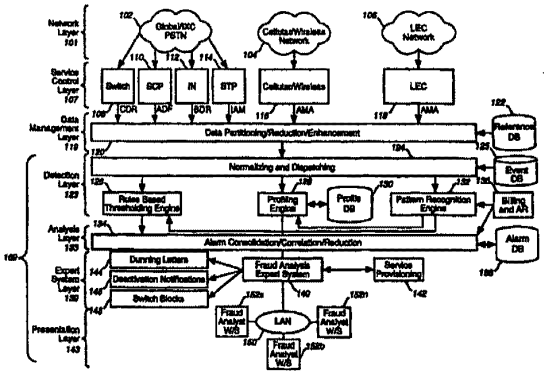
(75) Inventors/Applicants *(for US only)*: GAVAN, John [US/US]; 1624 N. Cascade Avenue, Colorado Springs, CO 80907 (US). PAUL, Kevin [US/US]; 660C Autumn Crest Circle, Colorado Springs, CO 80919 (US). RICHARDS, Jim [US/US]; 2613 N. 103 Court, Omaha, NE 68134 (US). VAN ARKEL, Hans [US/US]; 4125 Brigadoon Lane, Colorado Springs, CO 80909 (US). HERRINGTON, Cheryl [US/US]; 19086 Doewood Circle, Monument, CO 80132 (US). MAHONE, Saralyn [US/US]; 7220 Janice Place, Colorado Springs, CO 80920 (US). CURTIS, Terril, J. [US/US]; 660 W. Capistrano Avenue, Pueblo West, CO 81007 (US). WAGNER, James, J. [US/US]; 2796 Landers Drive, Larkspur, CO 80118 (US). DALLAS, Charles, A. [US/US]; 4945 Old Farm Circle W., Colorado Springs, CO 80917 (US).

(74) Agent: GROLZ, Edward, W.; Scully, Scott, Murphy & Presser, 400 Garden City Plaza, Garden City, NY 11530 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: SYSTEM AND METHOD FOR DETECTING AND MANAGING FRAUD



(57) Abstract

A system, method and computer program product for processing event records. The present invention includes a detection layer, an analysis layer, an expert systems layer and a presentation layer. The layered system includes a core infrastructure and a configurable, domain–specific implementation. The detection layer employs one or more detection engines, such as, for example, a rules–based thresholding engine and a profiling engine. The detection layer can include an AI–based pattern recognition engine for analyzing data records, for detecting new and interesting patterns and for updating the detection engines to ensure that the detection engines can detect the new patterns. In one embodiment, the present invention is implemented as a telecommunications fraud detection system. When fraud is detected, the detection layer generates alarms which are sent to the analysis layer. The analysis layer filters and consolidates the alarms to generate fraud cases. The analysis layer preferably generates a probability of fraud for each fraud case. The expert systems layer receives fraud cases and automatically initiates actions for certain fraud cases. The presentation layer also receives fraud cases for presentation to human analysts. The presentation layer permits the human analysts to initiate additional actions.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 351

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		

System and Method for Detecting and Managing Fraud

Cross-Reference to Related Applications

This patent application is related to the following commonly owned, co-pending U.S. utility patent application:

"Network Information Concentrator," Serial Number 08/426,256, Attorney Docket Number 1643/0012, incorporated herein by reference.

Background of the Invention

Field of the Invention

The present invention relates to processing event records, such as, for example, telecommunications network event records.

Related Art

As the telecommunications industry rapidly grows, telecommunications fraud also grows. In the United States alone, telecommunication fraud is estimated to have cost $3 billion in 1995. Telecommunications service providers have experienced difficulty in keeping up with new methods of fraud. As soon as service providers implement new systems to detect current methods of fraud, criminals innovate new methods.

Current methods of fraud are targeted at all types of services. Such services and corresponding fraud include use of calling cards, credit cards, customer premise equipment (CPE), including private branch exchanges (PBX), dial 1+, 800 inbound, and cellular calls. In addition, international dialing is a frequent target of fraud because of its high price of service. Subscription fraud, where a customer subscribes to a service, such as 800 or Dial 1, and then never pays, is also a frequent target of fraud.

Existing methods of detecting fraud are based primarily on setting predetermined thresholds and then monitoring service records to detect when a threshold has been exceeded. Parameters for such thresholds include total number of calls in a day, number of calls less than one minute in duration, number of calls more than 1 hour in duration, calls to specific telephone numbers, calls to specific countries, calls originating from specific telephone numbers, etc. Many parameters can be used to tailor a particular thresholding system for certain customers or services.

These thresholds must be manually programmed, which is labor intensive and time consuming. Moreover, these thresholds are generally subjective and not directly based upon empirical data. In addition, manually programmed thresholds are static and thus do not adjust to changing patterns of fraud. They are therefore easy for criminals to detect and circumvent. Also, since such thresholds must be

IPR2024-00027
CrowdStrike Exhibit 1002 Page 354

set conservatively in order to detect most fraud, they are frequently exceeded by non-fraudulent calls, contributing to high rates of false alarms.

When a threshold is exceeded, an alarm is triggered and presented to an analyst, who must then analyze the alarm to determine if it properly reflects fraud. The analyst must query many sources of data, such as customer payment history and service provisioning data, to assess the probability of fraud. The analyst must also assess several different alarms and correlate them to determine if a case of fraud is spanning across services. This manual process of analyzing and correlating is time consuming, labor intensive, highly subjective and prone to error.

When it is determined that fraud has occurred, the analyst must then select an appropriate action and then initiate it. Such actions can include deactivating a calling card or blocking an ANI (Automatic Number Identifier) from originating calls.

Because current systems of fraud management are rigid and generally not configurable for other service providers or industries, new rules, algorithms, routines, and thresholds must constantly be re-programmed.

What is needed is a configurable system, method and computer program product for detecting and automatically acting upon new and evolving patterns and that can be implemented in a variety of applications such as, for example, telecommunications fraud, credit card and debit card fraud, data mining, etc.

Summary of the Invention

The present invention is a system, method and computer program product for processing event records. The present invention includes a detection layer for detecting certain types of activity, such as, for example, thresholds and profiles, for generating alarms therefrom and for analyzing event records for new patterns.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 355

The present invention also includes an analysis layer for consolidating alarms into cases, an expert systems layer for automatically acting upon certain cases and a presentation layer for presenting cases to human operators and for permitting human operator to initiate additional actions.

The present invention combines a core infrastructure with configurable, user-specific, or domain-specific, implementation rules. The core infrastructure is generically employed regardless of the actual type of network being monitored. The domain-specific implementation is provided with user specific data and thus provides configurability to the system.

The domain-specific implementation can include a user-configurable database for storing domain-specific data. The user-configurable database can include one or more databases including, for example, flat files databases, object oriented databases, relational database, etc. User-configurable data can include conversion formats for normalizing records and dispatch data for specifying which fields of normalized network event records are to be sent to different processing engines.

In one embodiment, the present invention is implemented as a telecommunications fraud detection system in which the detection layer receives network event records from a telecommunications network and detects possible fraudulent use of the telecommunications network. In another embodiment, the present invention is implemented in a credit card and/or debit card fraud detection system. In yet another embodiment, the present invention is implemented in a data mining system or a market analysis system.

Regardless of the implementation-specific embodiment, event records can come from a variety of sources. Thus, event records are preferably normalized event records prior to acting upon them. Normalized event records are dispatched to one or more processing engines in the detection layer, depending upon the specific embodiment employed. The normalizing and dispatching functions include a core infrastructure and a configurable, domain-specific implementation.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 356

The detection layer can employ a plurality of detection engines, such as, for example, a thresholding engine, a profiling engine and a pattern recognition engine. One or more of the detection engines can enhance event records prior to acting upon them. Enhancement can include accessing external databases for additional information related to a network event record. For example, in a telecommunications fraud detection system, enhancement data can include, for example, bill paying history data for a particular caller.

A thresholding engine constantly monitors normalized event records to determine when thresholds have been exceeded. When a threshold is exceeded, an alarm is generated. In a telecommunications fraud detection implementation, thresholding can be based on pre-completion call data, as well as conventional post-call data.

The thresholding engine includes a core infrastructure and a configurable, domain-specific implementation. The core infrastructure includes configurable detection algorithms. The domain-specific implementation includes user-specific thresholding rules. The rules can be easily tailored for specific uses and can be automatically updated, preferably with updates generated by a pattern recognition engine. Thus, the domain-specific implementation of the thresholding engine can employ complex thresholding rules that compare and aggregate various data and network event records. The underlying core infrastructure provides scalability to the configurable domain-specific implementation.

A profiling engine constantly monitors normalized event records to determine when a departure from a standard profile has occurred. When a departure from a profile is detected, a corresponding alarm is generated. In a telecommunications fraud detection implementation, profiling can be based on pre-completion call data, as well as conventional post-call data.

The profiling engine includes a core infrastructure and a configurable, domain-specific implementation. The domain-specific implementation provides user-specific profiles. The profiles can be easily tailored for specific uses and can

IPR2024-00027
CrowdStrike Exhibit 1002 Page 357

be automatically updated, preferably with updates that are generated by a pattern recognition engine. The core infrastructure provides scalability to the configurable domain-specific implementation.

A pattern recognition engine preferably employs artificial intelligence to monitor event records and to determine whether any interesting or unusual patterns develop. In a telecommunications fraud detection implementation, interesting or unusual patterns can indicate fraudulent or non-fraudulent use of the telecommunications network. The pattern recognition engine uses the new patterns to dynamically update both a rules database for parametric thresholding and a profile database for profile analysis.

The pattern recognition engine includes a core infrastructure and a configurable, domain-specific implementation. The core infrastructure includes an AI pattern analysis processor for analyzing records and a call history database for storing a history of prior records. The actual contents of the call history database are developed from actual use of the system and are, thus, part of the domain-specific implementation.

By implementing AI for pattern recognition, thresholds are dynamic and can be adjusted in accordance with changing patterns of fraud. Patterns and thresholds are based on real-time event data, as well as historical data derived from external sources. In addition, pattern recognition data is fed to the profiling engine, which can then establish profiles that represent normal and fraudulent calling patterns. Varying departures from these profiles will trigger an alarm. In a telecommunications fraud detection implementation, a probability of fraud is calculated for each alarm.

The analysis layer receives alarms from the detection layer and performs several analysis functions to generate cases. In a fraud detection implementation, the analysis layer correlates alarms generated from common network incidents, builds cases of suspected fraud from individual alarms and prioritizes cases according to their probability of fraud so that there are likely to be fewer false

IPR2024-00027
CrowdStrike Exhibit 1002 Page 358

positives at the top of the priority list than at the bottom. The analysis layer includes a core infrastructure and a configurable, domain-specific implementation.

The analysis layer employs a fraud case builder to correlate multiple alarms that are generated by one or more detection layer engines. For example, a single event can violate one or more thresholding rules while simultaneously violating one or more profiling rules. The alarms can be consolidated into a single fraud case which lists each violation. The fraud case builder can also correlate over time. Thus, an event subsequent to the event listed above can be related to earlier events.

For example, a telephone call that is charged to a particular credit card can violate a threshold rule pertaining to the length of a call. A subsequent call which is charged to that same credit card can violate the same rule or other thresholding rules or profiles. The fraud case builder can correlate all such calls into a fraud case indicating all of the violations associated with the credit card. Depending on the implementation layer analysis rules, the fraud case builder can also generate additional fraud cases based upon the calling number, the called number, etc.

The domain-specific implementation of the analysis layer includes a configurable informant for retrieving data from external systems for use by an enhancer. A configuration database indicates the data necessary for enhancement. Preferably, the configuration database is a user-configurable database including one or more databases such as, for example, flat files databases, object oriented databases, relational database, etc. The domain-specific implementation also includes rules for analyzing alarms. The rules are user specific and can be tailored as necessary.

The expert system layer receives cases from the analysis layer, performs automated analysis of cases and automates decision support functions. The expert system layer includes a prioritizer for prioritizing cases, such as fraud

IPR2024-00027
CrowdStrike Exhibit 1002 Page 359

cases, for example, and an informant for retrieving additional data from external systems. The informant interfaces with external systems in formats native to the external systems. The expert system layer informant is similar to the informants that are employed by the detection and the analysis layers. External systems provide data that can be used in determining whether a fraud case is so obvious that automatic action, such as terminating an account, is warranted.

The expert system layer includes an enforcer for interfacing with external action systems. For example, in a fraud detection implementation, when the prioritizer determines that automatic action is required to stop a fraudulent activity, the enforcer sends necessary commands to one or more external action systems which will implement the action. The enforcer includes a configurable, domain-specific implementation that includes user-specific interfacing protocols for interfacing with external action systems in formats native to the external systems.

The expert system layer includes a core infrastructure and a configurable, configurable, domain-specific implementation. The domain-specific implementation includes prioritization rules for use by the prioritizer for prioritizing cases. These rules are generally user-specific and are typically based on prior experience. The domain-specific implementation also includes action rules for use by the prioritizer to determine what action to take on fraud cases.

The presentation layer receives cases for presentation to and analysis by human operators. Human operators can initiate action independent of any action automatically initiated by the expert system layer. The presentation layer includes a core infrastructure and a configurable, domain-specific implementation.

The present invention is scalable, configurable, distributed and redundant and can be implemented in software, firmware, hardware, or any combination thereof. The present invention employs Artificial Intelligence (AI) and Expert System technologies within a layered logical systems architecture. Thus,

IPR2024-00027
CrowdStrike Exhibit 1002 Page 360

configurability of detection criteria, portability to multiple enterprises and ability to detect new methods of fraud are all enhanced. In addition, dynamic thresholds and automated analysis are provided.

Further features and advantages of the present invention, as well as the structure and operation of various embodiments of the present invention, are described in detail below with reference to the following drawings.

Brief Description of the Figures

The accompanying drawings, which are incorporated herein and form part of the specification, illustrate the present invention and, together with the description, further serve to explain the principles of the invention and to enable a person skilled in the pertinent art to make and use the invention.

The present invention is described with reference to the accompanying Figures, wherein:

FIG. 1 is a block diagram of a multi-layer event record detection and processing system, including a detection layer, an analysis layer, an expert system layer and a presentation layer, implemented as a telecommunications fraud detection system;

FIG. 2 is a high-level process flowchart illustrating a method for detecting and acting upon fraud in a telecommunications system;

FIG. 3 is a block diagram of a distributed architecture for implementing the present invention;

FIG. 4 is a process flowchart expanding upon step 214 of FIG. 2, illustrating a rules-based thresholding process, a profiling process and a pattern recognition process;

FIG. 5A is a block diagram of a rules-based thresholding engine that can be used in the detection layer of the present invention;

IPR2024-00027
CrowdStrike Exhibit 1002 Page 361

FIG. 5B is a high-level block diagram of a feature vector that can be used to represent features associated with data records;

FIG. 5C is a detailed block diagram of the feature vector illustrated in FIG. 5B;

FIG. 5D is a block diagram of an alternative embodiment of the rules-based thresholding engine depicted in FIG. 5A;

FIG. 6 is a block diagram of a profiling engine that can be used in the detection layer of the present invention;

FIG. 7 is a block diagram of an artificial intelligence-based pattern recognition engine that can be used in the present invention;

FIG. 8 is a process flowchart illustrating a process for analyzing alarms generated by the rules-based thresholding process and the profiling process of FIG. 4 and for generating cases therefrom;

FIG. 9 is a block diagram of the analysis layer of FIG. 1;

FIG. 10 is a process flowchart illustrating a method for prioritizing fraud cases and for taking appropriate action upon certain of those prioritized fraud cases;

FIG. 11 is a block diagram of the expert system layer of FIG. 1;

FIG. 12 is a block diagram of the presentation layer of FIG. 1; and

FIG. 13 is a block diagram illustrating a relationship between a core infrastructure and a user-specific, or domain-specific, implementation of the present invention.

The present invention will now be described with reference to the accompanying drawings. In the drawings, like reference numbers typically indicate identical or functionally similar elements. Additionally, the left-most digit(s) of a reference number typically identifies the drawing in which the reference number first appears.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 362

Detailed Description of the Preferred Embodiments

Table of Contents

I. Overview

II. Example Environment

III. Processing Event Records

A. Detection Layer
1. Normalization and Dispatching
2. Rules-Based Thresholding
3. Profiling
4. Pattern Recognition

B. Analysis Layer

C. Expert System Layer

D. Presentation Layer

IV. Conclusions

I. Overview

The present invention is a configurable system, method and computer program product for detecting and automatically acting upon new and evolving patterns in, for example, telecommunications fraud detection, data mining, market analysis (i.e., to segment strong potential customers and to detect consumer purchasing patterns), etc. The present invention is a multi-layered, scalable, configurable, distributed and redundant system that can be implemented in software, firmware, hardware or any combination thereof. The present invention can be implemented as a portable software suite. The multiple layer architecture includes a detection layer for detecting thresholds, profiles and patterns for

IPR2024-00027
CrowdStrike Exhibit 1002 Page 363

generating alarms, an analysis layer for analyzing alarms and for consolidating alarms into cases, an expert system layer for acting upon cases and a presentation layer for presenting cases to human users.

Referring to FIG. 13, the invention includes a core infrastructure 1310 that permits each layer to be implemented in a variety of applications without alteration. The invention also includes a configurable, rules-based, user-specific, or domain-specific, implementation 1312 that permits each layer to be tailored for specific applications.

Domain-specific implementation 1312 permits the present invention to be configured for use in a variety of applications such as, for example, telecommunications fraud, credit card and debit card fraud, data mining, etc. Core infrastructure 1310 and domain-specific implementation 1312 can be implemented as software, firmware, hardware or any combination thereof.

Core infrastructure 1310 includes elements that are required, regardless of the actual implementation environment. Domain-specific implementation 1310 includes user-specific data and functions such as thresholding rules, profiles, pattern recognition rules, alarm correlation and reduction rules, fraud case prioritization and action rules, presentation parameters and external system interacting parameters. Core infrastructure 1310 can be used in a variety of applications, without redesign. Domain-specific implementation 1312 permits substantial tailoring of the system to user-specific situations.

Domain-specific implementation 1312 includes configurable rules to provide flexibility. Configurable rules include event recognition rules, event normalization rules, enhancement rules, detection rules, analysis rules, prioritization rules, presentation rules, and dispatching rules, including provisioning rules.

Event recognition rules specify how to recognize incoming events. Event normalization rules specify how to normalize the events. Provisioning rules specify how to map sets of detection rules to events, such as, for example, by

IPR2024-00027
CrowdStrike Exhibit 1002 Page 364

customer or by product. Enhancement rules specify how to derive new information from existing information, such as how to derive product identification from an incoming event.

Detection rules specify how to generate alarms in reaction to events of suspected fraud. Detection rules can include thresholding rules and profiles of use. Detection rules also specify alarm priorities for alarms generated in the detection layer, based on the type of rule violated.

Analysis rules further prioritize alarms and specify how to correlate alarms into fraud cases. Prioritization rules specify how to prioritize cases for automatic actions. Presentation rules specify how to display information to users. Dispatching rules, which include provisioning rules, are used for provisioning data to a rule set, partitioning data and deciding which engine to send data to.

Rules can be created or modified and take effect while the system is running. When a rule is created or modified, it will be applied to new events that arrive at the system. Generally, rules are not applied to previously received events. When a rule is deleted, its deletion does not affect values or entities that were generated from the rule. For example, deleting an alarm type definition rule does not affect existing alarms.

Domain-specific implementation 1312 also includes configurable values. These can include, but are not limited to, one or more of the following values.

A database time-out variable specifies a maximum amount of time to wait for a response from a database. For example, if a request for data is sent to an external database, the system will wait only for the time-out period. Configurable rules will determine what event to take upon failure to receive a response within the time-out period.

An expected volume of data from a data management layer variable specifies the number of messages which can be received and the time period over which to measure this expected number of messages.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 365

A time-out wait period for data from a data management layer variable specifies the maximum time to wait for a message from the data management layer before sending a network alarm.

A maximum age for arriving events variable specifies the maximum time between event creation and event arrival from a data management layer. This variable can be used to increment a counter of old events. A maximum number of old events variable specifies the number of events older than the maximum age for arriving events. Typically, a network management message is generated whenever this variable is exceeded.

A maximum number of invalid events variable specifies the maximum number of invalid events which can be received from a data management layer. Typically, a network management message is generated whenever this variable is exceeded.

A case-high priority threshold variable specifies a priority level above which cases are monitored if they go unprocessed. A maximum case unprocessed time variable specifies the maximum time that a case above the case-high priority threshold can go unprocessed before being reportable.

A rules performance measurement time period variable specifies a time period over which rules performance will be measured. This variable is typically used for reporting purposes.

A variety of purge time variables specify time periods for storing a variety of data items. Data items can include invalid events, valid and normalized events, alarms, cases determined to be fraudulent, cases determined to be other than fraudulent and actions taken.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 366

II. Example Environment

The present invention can be configured for a variety of applications, such as, for example, telecommunications fraud detection, credit card and debit card fraud detection, data mining, marketing analysis, etc. The present invention is described below as implemented as a telecommunications fraud detection system. The examples that are described herein are provided to assist in the description of the present invention, not to limit the present invention.

Telecommunications systems can include any of a variety of types of telecommunications networks. Several such telecommunications networks are depicted in network layer 101 of FIG. 1. Network layer 101 can include a Global/Inter-Exchange Carrier (IXC) public switch telephone network (PSTN) 102 which can include conventional IXC networks with domestic and global coverage, such as those of MCI Telecommunications and British Telecom (BT). A variety of services can be supported by such networks. Network layer 101 can also include cellular and wireless networks 104, which offer conventional analog and digital cellular services. Local exchange networks (LEC) 106, such as those operated by Regional Bell Operating Companies (RBOCs), independent local telephone companies, and Competitive Access Providers (CAPs) can also be included.

A service control layer 107 offers and manages various telecommunications services and creates service records which contain data representing each instance of a particular service offering. For example, service control layer 107 can support Global/Inter-Exchange Carrier PSTN 102 with a plurality of switches 108 for issuing Call Detail Records (CDR) for each voice and data call it processes. In addition, a plurality of service control points (SCP) 110 can be used to provide data and intelligence for enhanced services, such as virtual network or 800 call routing. SCPs issue records for each transaction they

IPR2024-00027
CrowdStrike Exhibit 1002 Page 367

process. These records are referred to herein as Application Data Field (ADF) records.

Intelligent networks (IN) 112 can be provided for enhanced service offerings, such as operator services. Components from INs 112 can issue records, commonly referred to as Billing Detail Records (BDR), and intelligent services bad billed number (BBN) records, for such services. In addition, completed IN operator assisted calls can be sent from a network information concentrator (NIC) as enhanced operator service records (EOSR), which include an ISN BDR which is matched with a switch format (EOSR).

Signal transfer points (STP) 114 can be employed for signaling networks, referred to as SS7 networks, which use these signal transfer points (STP) 114 for processing call signaling messages. STPs 114 issue messages, such as, for example, Initial Address Messages, which contain data pertaining to a call being carried by the IXC network.

Service control layer 107 can also employ cellular service control components 116 for issuing standard AMA records for cellular calls handled by cellular network 104. Service control layer 107 can include LEC service control components 118 of a LEC network 106 for issuing AMA records for local calls and the local exchange of long-distance calls.

A single call can traverse multiple networks and generate multiple call records. ADFs, BDRs and IAMs can be issued prior to a call completing. CDRs, EOSRs and AMAs are issued after a call completes.

A data management layer 119 collects the various service records from the service control components and processes them to produce network event records that can be used by a variety of systems. Processing of data to produce network event records can include partitioning data among various distributed processors, reducing data by eliminating redundancy and consolidating multiple records for the same call, and enhancing data by augmenting records with pertinent data from external systems.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 368

Data management layer 119 can be implemented in a variety of manners and can include a data partitioning, reduction and enhancement component 120. Preferably, component 120 is a Network Information Concentrator (NIC), as specified and claimed in co-pending U.S. application, Serial Number 08/426,256, incorporated herein by reference in its entirety. NIC 120 can utilize one or more reference databases 122 to provide external data for enhancement. External reference data can include customer identification codes, service type codes and network element codes.

Typically, each of the telecommunications networks within network layer 101 can handle or process any of a variety of types of calls such as calling card calls, credit card calls, customer premise equipment (CPE) calls, dial 1+ calls, toll free 800 calls and cellular calls. They can also include credit card and debit card transactions. Each of these types of calls are subjected to fraudulent use. Thus, each of the telecommunications networks within network layer 101 are affected by fraud.

III. Processing Event Records

The present invention provides a multi-layered system, method and computer program product for detecting and acting upon data patterns and thresholds. When implemented as a telecommunications fraud detection system for detecting and acting upon fraud in one or more telecommunications networks, the present invention detects fraud by comparing network event records with thresholding rules and profiles. Violations result in the generation of alarms. Multiple alarms are correlated into fraud cases based on common aspects of the alarms, thus reducing the amount of analysis which must be performed on suspected incidents of fraud.

The system automatically acts upon certain cases of detected fraud to reduce losses stemming therefrom. In addition, live analysts can initiate

IPR2024-00027
CrowdStrike Exhibit 1002 Page 369

additional actions. In a parallel operation, calling patterns are analyzed via network event records to discern new methods or patterns of fraud. From these newly detected methods of fraud, new thresholds and profiles are automatically generated for protecting the telecommunications system.

Referring to FIG. 1, the present invention is illustrated as implemented as a fraud detection system 169. The present invention includes a detection layer 123, an analysis layer 133, an expert systems layer 139 and a presentation layer 143. In FIG. 2, a high-level process flowchart illustrates a method of detecting and handling fraud in a telecommunications system, such as that shown in FIG. 1. The process can be performed with software, firmware, hardware, or any combination thereof.

The process begins at step 210, where service records are generated by service control layer 107 for calls handled by the telecommunications systems in network layer 101. Service record formats and the data contained therein vary according to the type of call and the network equipment which handle a particular call, as described above. Because a single call can traverse multiple networks, a single call can generate multiple call records.

In step 212, service records are processed by data management layer 119 to generate network event records. This processing includes partitioning data among various distributed processors, reducing data by eliminating redundancy and consolidating multiple records for the same call, and enhancing data by augmenting records with pertinent data from external systems.

In step 214, network event records are analyzed by detection layer 123 for possible fraud. Step 214 is further detailed in the flowchart of Figure 4, as described below.

Detection layer 123 specifies and executes tests to detect fraudulent use of network layer 101 services. Detection layer 123 is part of the infrastructure and is scalable and distributed with a configurable component to allow for customization in accordance with user requirements. Preferably, detection layer

IPR2024-00027
CrowdStrike Exhibit 1002 Page 370

123 includes three classes of processing engines, which are three distinct but related software processes, operating on similar hardware components. Preferably, these three classes of engines include a rules-based thresholding engine 124, a profiling engine 128 and a pattern recognition engine 132. These scalable and distributed engines can be run together or separately and provide the system with unprecedented flexibility.

A normalizing and dispatching component 124 can be employed to normalize network event records and to dispatch the normalized records to the various processing engines. Normalization is a process or processes for converting variously formatted network event records into standardized formats for processing within detection layer 123. Preferably, the normalizing process is dynamic in that the standardized formats can be varied according to the needs of the user.

Dispatching is a process which employs partitioning rules to pass some subset of the normalized network event records to particular paths of fraud detection and learning. Thus, where a particular processing engine requires only a subset of the available information, time and resources are conserved by sending only the necessary information.

Rules-based thresholding engine 124 constantly reads real-time network event records from network information concentrator 120 and compares these records to selected thresholding rules. If a record exceeds a thresholding rule, the event is presumed fraudulent and an alarm is generated. Thresholding alarms are sent to analysis layer 133.

Profiling engine 128 constantly reads real-time network event records from network information concentrator 120 and from other possible data sources which can be specified in the implementation layer by each user architecture. Profiling engine 128 then compares event data with appropriate profiles from a profile database 130. If an event represents a departure from an appropriate profile, a probability of fraud is calculated based on the extent of the departure

IPR2024-00027
CrowdStrike Exhibit 1002 Page 371

and an alarm is generated. The profiling alarm and the assigned probability of fraud are sent to an analysis layer 133.

Preferably, in step 214, network event records are also analyzed in real-time by an artificial intelligence-based pattern recognition engine 132. This AI analysis will detect new fraud profiles so that threshold rules and profiles are updated dynamically to correspond to the latest methods of fraud.

Pattern recognition engine 132 permits detection layer 123 to detect new methods of fraud and to update the fraud detecting engines, including engines 126 and 128, with new threshold rules and profiles, respectively, as they are developed. In order to detect new methods of fraud and to generate new thresholds and profiles, pattern recognition engine 132 operates on all network event records including data from network information concentrator 120 through all other levels of the system, to discern anomalous call patterns which can be indicative of fraud.

Pattern recognition engine 132 collects and stores volumes of event records for analyzing call histories. Utilizing artificial intelligence (AI) technology, pattern recognition engine 132 analyzes call histories to learn normal patterns and determine if interesting, abnormal patterns emerge. When such an abnormal pattern is detected, pattern recognition engine 132 determines if this pattern is to be considered fraudulent.

AI technology allows pattern recognition engine 132 to identify, using historical data, types of patterns to look for as fraudulent. Pattern recognition engine 132 also uses external data from billing and accounts receivable (AR) systems 136 as references to current accumulations and payment histories. These references can be applied to the pattern recognition analysis process as indicators to possible fraud patterns.

Once pattern recognition engine 132 has established normal and fraudulent patterns, it uses these results to modify thresholding rules within the thresholding engine 226. For example, pattern recognition 132 can determine

IPR2024-00027
CrowdStrike Exhibit 1002 Page 372

that credit card calls to a specific country, which exceed 50 minutes in length, are fraudulent 80% of the time. Pattern recognition engine 132 can then modify a thresholding rule within thresholding engine 126 which will generate an alarm if event data is received which reflects that particular pattern. Thus, by dynamically modifying threshold rules, the system is able to keep up with new and emerging methods of fraud, thereby providing a key advantage over conventional parametric thresholding systems for fraud detection.

Similarly, once normal and fraudulent patterns have been established by pattern recognition engine 132, pattern recognition engine 132 updates the profiles within the profile database 130. This allows profiles to be dynamically modified to keep up with new and emerging methods of fraud.

In step 216, alarms are filtered and correlated by analysis layer 133. For example, suppose a threshold rule generates an alarm if more than ten credit card calls charged to a single credit card are made within a one hour time frame. Suppose also that another threshold rule generates an alert if more than one call is charged to a particular credit card at the same time. If ten calls are placed within one hour using the same single credit card, and the ninth and tenth calls were made simultaneously (two different callers using the same credit card number) then two alarms would be generated at the same time. One alarm for exceeding ten calls per hour and one for exceeding one call per card at a time.

A correlation scheme for step 216 might combine the above two alarms into a single fraud case indicating that a particular credit card has exceeded two different threshold rules. In addition, if a pattern recognition engine is employed, a new threshold rule can be generated to cause an alarm to be generated in the event of any future attempted use of the identified credit card.

Alarms which are generated by the detection layer 123 are sent to the analysis layer 133. Analysis layer 133 analyzes alarm data and correlates different alarms which were generated from the same or related events and consolidates these alarms into fraud cases. This reduces redundant and

IPR2024-00027
CrowdStrike Exhibit 1002 Page 373

cumulative data and permits fraud cases to represent related fraud occurring in multiple services. For example, different alarms can be received for possibly fraudulent use of calling cards and cellular phones. The correlation process within analysis layer 133 can determine that fraudulent use of a credit card over a cellular phone is occurring. An alarm database 138 stores alarms received from the detection layer for correlation.

Analysis layer 133 prioritizes the fraud cases according to their probability of fraud so that there are likely to be fewer false positives at the top of the priority list than at the bottom. Thus, fraud cases which are generated due an occasional exceeding of a threshold by an authorized user or by an abnormal calling pattern by an authorized user, such as calling from a new country while on business travel, are typically assigned a low priority. Preferably, analysis layer 133 employs artificial intelligence algorithms for prioritization. Alternatively, detection layer 123 rules can be customized to prevent such alarms in the first place.

In a preferred embodiment, analysis layer 133 includes a software component 134 which performs the consolidation, correlation, and reduction functions. Software component 134 makes use of external data from billing and AR systems 136 in the correlation and reduction processes. Preferably, alarm database 138 resides on the same hardware as software component 134.

In step 218, consolidated fraud cases are sent to expert system layer 139 for automatically executing one or more tasks in response to certain types of fraud cases. Thus, in the example above, automatic action can include notifying the responsible credit card company of the suspected fraud so that they can take fraud-preventive action. In addition, any pending calls can be terminated if such functionality is supported by the network.

Preferably, expert system layer 139 includes a fraud analysis expert system 140, which applies expert rules to determine priorities and appropriate actions. An off-the shelf expert system can be used. Preferably, however, a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 374

customized expert system is employed and is programmed using a rules-based language appropriate for expert systems, such as, for example, CLIPS.

Typically, algorithms for step 218 are designed to identify certain cases of suspected fraud which are so egregious that automatic responses are appropriate. However, these algorithms can also be designed to withhold automatic responses where, for example, the suspected fraud is not so flagrant or not too potentially expensive or where certain mitigating circumstances exist, such as, for example, a history of similar activity which were later accounted for or honored by the customer.

Expert system 140 includes interfaces to several external systems for the purpose of performing various actions in response to detected fraud. These can include a system 144 for issuing dunning letters, a system 146 for issuing deactivation notifications and a system 148 for instituting switch-based ANI blocks. Expert system 140 can include an interface to a service provisioning system 142 for retrieving data relating to services provided to a customer and for initiating actions to be taken on a customer's service. Expert system 140 can employ artificial intelligence for controlling execution of automated or semi-automated actions.

Regardless of whether automatic responses are generated, it is important to provide all cases of suspected fraud to live operators so that they can take some action for which the automated system is not capable. Thus, in step 220, all fraud cases, included those which caused automatic responses in step 218 and those which did not are sent to a presentation layer 143 for presentation to human analysts.

Presentation layer 143 preferably includes a plurality of workstations 152a-152n connected to the each other and to expert system 140 via a local area network LAN 150, a wide area network (WAN), or via any other suitably interfacing system. Thus, throughout the remainder of this document, where LAN 150 is referred to, it should be understood that a WAN or any other suitable

IPR2024-00027
CrowdStrike Exhibit 1002 Page 375

interfacing system can be substituted. Workstations 152a-152n can be conventional personal computers and can operate as clients having specific software which provides a graphical user interface (GUI) to the system.

Fraud data that has been collected and processed by the detection, analysis and expert system layers can thus be presented to human analysts via workstations 152a-152n via LAN 150. Presentation layer 143 also allows for human analysts operating from workstations 152a-152n to initiate actions to be taken in response to detected fraud. Such actions are executed through interfaces to various external systems. Presentation layer 143 can include a customized, flexible scripting language which forms part of the infrastructure component of the system.

In the example above, if the automated system does not trigger an automatic action, a live operator could, nevertheless, contact the credit card issuer or owner to inform them that the credit card is being used to make two simultaneous credit card calls and that the credit card has been used to make more than ten calls within one hour. If the credit card issuer or owner indicates that the calls are authorized and that the billing will be honored, no further action can be taken. The live operator can even input data into the system so that, in step 214, threshold rules or profiles are altered for this particular credit card so that similar use of the card in the future will not generate alarms.

Alternatively, where the credit card issuer or owner indicates that the calls are not authorized, the live operator can take action to immediately disconnect the calls, if they are still in progress and if the monitored network supports such functionality. Furthermore, the live operator can input data into the system so that, in step 214, alarms are immediately generated if even a single call is attempted to be billed to that credit card. The live operator can also input data into the system so that, in step 218, any alarms generated by that credit card generate an immediate automatic response including terminating the attempted

IPR2024-00027
CrowdStrike Exhibit 1002 Page 376

call. Alternatively, the operator can initiate deactivation of the card so that calls based on the card are blocked before any substantial analysis is performed.

Referring to FIG. 3, a preferred physical architecture of fraud detection system 169 is a client/server based system which operates on a LAN, WAN or other system, which can be LAN 150. Preferably, the logical components that make up detection layer 123, analysis layer 133 and expert system layer 139 operate on one or more servers such as servers 310a-310n. These logical components preferably include a rules-based thresholding engine 126, a profiling engine 128, a profile database 130, a pattern recognition engine 132, an alarm consolidation, correlation and reduction unit 134, an alarm database 138 and a fraud analysis expert system 140.

Network event records 314 are provided to servers 310a-310n from data management layer 119 which preferably includes the previously described network information concentrator 120. External systems 316 can provide data for enhancement of internal processes and data in the system. External systems 316 can include a billing and accounts receivable component 136 and various other systems utilized by the detection layer. Fraud analyst workstations 152a-152n provide an interface between fraud detection system 160 and human analysts.

A. Detection Layer

Referring to FIG. 4, a process flowchart illustrates various processes that can be performed by detection layer 123. These processes include a rules-based thresholding process (steps 412-420), a profiling process (steps 422-428 and 420) and a pattern recognition process (steps 430-438). Systems for implementing the processes of FIG. 4 are provided in FIGS. 5A, 5D, 6, and 7. In one preferred embodiment, the system is implemented in a distributed processing architecture. For example, the system can be implemented on a plurality of server components 310a-310n.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 377

1. Normalization and Dispatching

Normalization is a process for converting network event records of various formats to standardized formats that are recognized by each of the multiple paths of fraud detection and learning. Preferably, the normalization process is dynamic in that the standardized formats can be varied according to the implementation.

Dispatching is a process for forwarding normalized network event records to particular paths of fraud detection and learning in detection layer 123. Dispatching includes provisioning rules and partitioning rules. Provisioning rules determine which rule set or detection method within detection layer 123 will receive data. Partitioning rules balance loads among multiple processors assigned to a rule set or detection method.

Referring to FIGS. 5A, 5D, 6, and 7, normalizer and dispatcher 124 is provided for normalizing and dispatching network event records 501, sent from data management layer 119. Network event records include those created when a call completes, such as a CDR, EOSR and AMA, and those created during a call and received by the Data Management Layer 119 prior to completion of the call, such as an IAM, BDR, BBN and ADF. Operation of normalizer and dispatcher 124 is described with reference to the flowchart of FIG. 4.

In step 408, network event records 501 are received by normalizer and dispatcher 124. Normalizer and dispatcher 124 preferably includes a core infrastructure 1310 and a user-specific or domain-specific implementation 1312. A normalizer 502 converts network event records 501 from the various formats of the data management layer into standardized formats required by the detection layer.

In step 410, normalizer 502 uses configuration data 504 to convert the network event records 501 into normalized event records 506. Configuration data 504 is user-dependent and thus is part of domain-specific implementation 1312.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 378

Normalized network event records 506 are then sent to dispatcher 508 which employs user-specific dispatching rules to pass normalized network event records 506 to appropriate detection-layer engines 126, 128 and 132. In one embodiment, dispatcher 508 provides normalized network event records 506a, which are sent to rules-based thresholding engine 126, normalized network event records 506b, which are sent to profiling engine 128, and normalized network event records 506c, which are sent to pattern recognition engine 132.

Normalizer 502 also stores network event records in an event database 125 for use by one or more engines within detection layer 123. Events are preferably stored for a period of time that can be designated by the user. A storage period can, for example, be set to twenty four hours.

Dispatcher 508 allows the fundamental process being performed by normalizer 502 to be configured for any enterprise by programming data specific to the requirements of the enterprise into the dispatcher 508. Normalizer 502, configuration data 504 and dispatcher 508 also permit quick and easy updates to be made to the normalizer process whenever modifications to downstream detection processes are made.

2. Rules-Based Thresholding

Thresholding is a process whereby network event records are compared to threshold rules. In a telecommunications fraud detection system, where network event records represent telephone calls, network event records are compared to threshold rules to determine whether the network event records represent possible fraudulent use of a telecommunications network.

Referring to the process flowchart of FIG. 4, steps 412-420 represent a rules-based thresholding process. In FIG. 5A, details of rules-based thresholding engine 126 are provided for implementing the thresholding process of steps 412-

IPR2024-00027
CrowdStrike Exhibit 1002 Page 379

420. In describing the elements and process of thresholding engine 126, the following terms are used:

An event, represented by a normalized event record 506a, generates a feature in thresholding engine 126. Features are defined below. In a telecommunication fraud detection implementation, an event is typically a telephone call.

A generating event is an event that caused a feature to be generated. A generating event is typically the most recent event of a series of events that are measured to calculate a feature value.

A contributing event is an event that contributed to a feature, but did not cause the feature to be generated. While a feature can consist of a single event, it typically consists of one generating event and several contributing events. Every event that is received is at first a generating event, in that when it is received, it generates the calculation of a feature. The event can then become a contributing event for calculation of other features.

A key is an event field, such as ANI, calling card number, call destination, etc. A key is used to identify an event type.

A feature is information that is used by the threshold detector 520 to determine whether evidence of fraud exists. A feature can include, for example, the number of calls made with a certain calling card number in a two hour period. A feature can also be, for example, an NPA of a call. Features are calculated by enhancer 510 in accordance with enhancement rules 512, and with data from one or more events.

There are generally two types of features, single event features and multiple event features. Single event features are derived from a single generating event and include no contributing events. For example, a single event feature can be a call made from a hot ANI. Multiple event features are derived from a generating event and zero or more contributing events. Multiple event features are the result of measurements performed on a key for a period of time

IPR2024-00027
CrowdStrike Exhibit 1002 Page 380

such as, for example, a measurement of the number of calls made with certain calling card number within a period of time.

Referring to FIG. 5B, features can be represented by feature vectors such as, for example, feature vector 518. A feature vector, such as feature vector 518, includes one or more feature vector segments 532, 534, 536, 537, etc., for identifying a collection of features. In a preferred embodiment, enhancer 510 generates feature vectors 518 and passes them to threshold detector 520.

Referring to FIG. 5C, a preferred embodiment of feature vector 518 is provided where each feature vector segment 532-536 preferably includes a feature name field, or key feature field, 538-542, respectively, for identifying a particular key feature. For example, a key feature field might indicate a particular calling card number, an ANI, a credit card account number, a calling number, etc.

Each feature vector segment 532-536 includes a key value field 544-548, respectively, for providing a value for associated feature name fields 538-542. For example, if feature name field 538 identifies a particular calling card number as the key feature represented by feature vector segment 532, feature value field 544 might provide the number of calls made within the past two hours with that calling card. Similarly, if feature name field 540 identifies a particular ANI, feature value field 546 might provide the number of calls made within the past two hours from that ANI.

Generating event fields 550-554 identify a generating event for each feature vector segment 532-544. Recall that a generating event is an event that caused a feature to be generated. A generating event is typically the most recent event which is counted in a value field 544-548. For example, where value field 544 specifies that four calls were made in the past two hours with a particular calling card, the most recent of those four calls is the generating event for feature 532.

Contributing event fields 556-560 and 562-566 represent events which contributed to features vector segments 532 and 536, respectively. Using the

IPR2024-00027
CrowdStrike Exhibit 1002 Page 381

example above, where value field 544 specifies four calls within the past two hours and where generating event field 550 represent the fourth one of those calls, contributing event fields 556-560 represent the prior three calls. Note that feature vector segment 534 includes no contributing events. Feature vector segment 534, therefore, represents a single event feature such as, for example, a hot ANI call.

Although generating event fields 550-554 can identify different generating events, feature vector segments 532-536 are, nevertheless related through one or more common aspects of their generating events and/or contributing events.

For example, a feature vector segment 532 can represent calling card calls made within the past two hours with a particular calling card so that generating event 550 and contributing events 556-560 represent calls made within, for example, the past two hours with that calling card. In addition, feature vector segment 534 can represent a hot ANI and generating event 552 can identify a single instance of a call made from the hot ANI. Feature vector segments 532 and 534 are said to be related if one or more of generating event 550 and contributing events 556-560 identify a call that is made with the calling card represented by feature 532, from the hot ANI represented by generating event 552.

Referring back to FIG. 5A, thresholding engine 126 preferably employs a core infrastructure 1310 and a domain-specific implementation 1312. Core infrastructure 1310 includes an enhancer and threshold detection component 509 for applying configurable rules to network event records 506a and for generating alarms 526 in the event that one or more network event records 506a violate one or more configurable rules.

Enhancer and threshold detection component 509 includes at least one enhancer component 510 and at least one or more threshold detector component 520. Enhancer component 510 receives network event records 506a and generates a feature vector for identifying one or more key features associated with event records 506a. Key features can include, for example, ANI, credit card

IPR2024-00027
CrowdStrike Exhibit 1002 Page 382

number, calling number, etc. A network event record that triggers generation of a feature vector is referred to as a generating event.

Domain-specific implementation 1310 includes enhancement rules and configuration database 512 and a threshold detection rules database 522. Databases 512 and 522 include rules that can be created, deleted or modified according to evolving needs of a user. Changes to thresholding rules 522 can even be executed while the system is running. When a thresholding rule is created or modified, it will be applied to new events that arrive at the system.

With reference to FIG. 4, in step 412, normalized event records 506a are received by enhancer component 510 within thresholding engine 126. Each event, when it is received, is treated as a generating event, in that it generates the calculation of a feature vector. Threshold enhancer component 510 augments normalized event records 506a to produce feature vectors 518. Enhancement rules specify the fields which are to be kept, omitted, formatted and augmented. Augmentation can provide additional data which enable threshold detector 520, to apply appropriate threshold detection rules. Enhancement rules and configuration data 512 specifies data which is required for enhancement, where to find this data, and how to add it to event records. Thresholding enhancer configuration data 512, analogous to normalizer configuration data 504, provides modularity to the system for both ease of configurability and portability to fraud detection of other enterprises.

Threshold enhancer component 510, based on instructions from enhancement rules and configuration data 512, can request enhancement data from an informant 514. Informant 514 provides a communication interface to one or more external systems 516 from which additional data needed by the enhancer 510 can be retrieved. External systems 516 can include customer order entry systems and network engineering systems.

Based on a request that informant 514 receives from threshold enhancer component 510, informant 514 sends a query to an appropriate external system

IPR2024-00027
CrowdStrike Exhibit 1002 Page 383

516, using a communications protocol of the external system 516. Informant 514 receives the requested data and supplies it to enhancer 510.

Informant 514 thus provides modularity that allows adding and removing various external systems 516 by simply modifying interfaces within informant 514. Thresholding engine 126 can thus be interfaced to a variety of external systems 516 and can be easily reconfigured for fraud detection systems of other enterprises.

When enhancer 510 receives an event record 506a, enhancer 510 determines the type of event based on a key. For example, if the event is a calling card call, a key can be the calling card number field of the event record. Enhancer 510 looks up a rule set, based on provisioning, in enhancement rules and configuration data 512, for that event type.

A rule set includes one or more rules for specifying how to calculate features for an event type. A generating event can trigger the calculation of one or more features. A rule defines a feature and requests that feature be calculated using a certain measurement type. The resulting value of the feature is placed in a feature vector 518.

Multiple measurement types can be performed by enhancer 510, in accordance with specifications in the enhancement rules and configuration data 512. Each measurement type includes an algorithm used to calculate the value of a feature. For example, measurement types can include, but are not limited to, any of the following measurement types:

1) simple count: counts events in a given time period (i.e., number of calls in the past two hours);

2) field count: counts events that meet a criteria for a certain value of an event field (i.e., number of calls with ANI = 202-555-1234). Enhancer 510 looks up field in event. If the field value = a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 384

specified value, then Enhancer 510 adds the event to a list that will be counted;

3) set count: counts events that meet a criteria for a set of values of a field so that, if a field in an event has a value that is a member of a set (as defined by an enhancement rule), then enhancer 510 counts the event (i.e., number of calls originating in Texas, New Mexico, Arizona, or California);

4) sum: sums a certain field from one or more events in a given time period (i.e., sum the duration of all calls made in past 2 hours);

5) simultaneous: count calls (with certain criteria) that were made at same time, defined by minimum call duration overlap or maximum time separation (i.e., count all calls made with calling card number = *nnn,* that overlap by at least 2 seconds or that are made within 10 seconds of duration of another call made with same calling card number); and

6) geographic velocity: simultaneous calls over a distance. Rule will provide a minimum time needed between calls, based on physical distance between points of call origination. For example, if a call made with a certain calling card in a first city is placed less than 4 hours after another call made with the same calling card in a second city, and the second city is more than 4 hours in traveling time from the first city, then add both events to list that will be counted.

To count events using any of these measurement types, enhancer 510 places each event in a list, the events being sequenced in time. Each rule specifies a time period in which to include events for a measurement. To perform a count, enhancer 510 begins with the most recent event (the generating event) in the list, and counts events in the list backward in time, until the time period is

IPR2024-00027
CrowdStrike Exhibit 1002 Page 385

covered. As a new generating event is received, enhancer 510 begins its count later in the list. This represents a sliding time window.

For example, suppose a rule specifies a field count, such as "count all events in the past two hours in which the ANI equals a certain value." When a first event meeting the ANI criteria is received, it is a generating event, and causes enhancer 510 to retrieve this particular rule from enhancement rules and configuration data 512, using the ANI as a key.

Enhancer 510 places this event in a list of events that meet the same criteria. Enhancer 510 then counts all events in this list going back two hours from the generating event. Other events are contributing events. If another event meeting the same criteria is received, with a time stamp 5 minutes later than the first event, then this second event becomes a generating event. Enhancer 510 counts all events in the list going back two hours from this second event. The first event becomes a contributing event to the second event. The two hour time window slides forward by 5 minutes.

The six measurement types that are described above have the following common functions:

1) they each perform a measurement for a specific key (i.e., calling card number 202-555-1234);
2) they each analyze all events which have the specific key and apply an algorithm to each event within the given time period (the sliding time window);
3) they each return a feature value for the given time period, representing a time window (the time window is set by the most recent event, and goes backward in time from there); and
4) they are all persistent.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 386

Continuing with the process of FIG. 4, enhancer 510 performs a measurement on a generating event, in accordance with a rule that is part of a rule set that is read from enhancement rules and configuration data 512. Each generating event causes enhancer 510 to apply a rule set, a rule set including one or more rules.

The results of a measurement is a feature. Recall that a feature includes a measured feature value, a generating event, and zero or more contributing events. For performance reasons, each event in a feature is represented by an event identifier, which points to an actual event record in event database 125. A single generating event can result in one or more features. Enhancer 510 creates a feature vector, and places each feature in the feature vector.

The following example is provided for a rule which can be, for example:

If calling card number = 123456789, then
create feature vector;
calculate simple count: number of calls made in past 2 hours;
calculate set count: number of calls from *(list of pay phone ANIS)* made in past 2 hours;
calculate simultaneous: number of calls made within 10 seconds of another call, in past 2 hours;

Suppose that enhancer 510 receives an event 506a representing a call made with calling card number = 123456789. This is a generating event. Enhancer 510 uses the calling card number as a key, and retrieves the above rule from enhancement rules and configuration data 512.

Enhancer 510 then performs the rule by creating a feature vector 518. Enhancer 510 then reads a list of all events from event DB 125. Beginning with the generating event, enhancer 510 goes back two hours and counts all events

IPR2024-00027
CrowdStrike Exhibit 1002 Page 387

representing calls made with that calling card number. This is the simple count. Enhancer 510 goes back two hours and counts all calls made with that calling card number, from a pay phone ANI. This is the set count. Likewise, enhancer 510 performs the simultaneous count. The result of each count, along with identifiers for each event that was counted, is added as a feature to the feature vector 518.

Now enhancer 510 can also include in the feature vector a threshold for a feature. Thresholds are included if called for by an enhancement rule, and are provided by enhancement rules and configuration data 512. They are placed in the feature vector as a feature. For example, feature 534 can represent a threshold for feature 532. A threshold can be a value for measurement (i.e., "5" calls) or a statement of truth (i.e., if ANI = 202-555-1234). A statement of truth is equivalent to a default value of 1 (i.e., if 1 call with ANI = 202-555-1234).

A feature does not include a threshold for itself. It is simply a measured value. A threshold for a feature can be included in the feature vector as a feature in itself. But it is not necessary for every feature to have a threshold stated as another feature. Enhancer 510 does not perform any comparisons, it simply performs the measurements and creates the feature vector. Comparison of feature values with thresholds, to determine if a threshold has been exceeded, is performed by threshold detector 520. Thresholds can also be obtained from threshold detection rules 522 by the threshold detector 520, as part of the process of determining if a threshold has been exceeded. This is described with reference to step 414, below.

In step 414, threshold rules are applied events to determine whether fraud exists. In the preferred embodiment, feature vectors 518 are received by threshold detector 520 for application of threshold rules.

Threshold detector 520 is responsible for determining if evidence of fraud exists in a feature vector. Threshold detector 520 employs threshold detection

IPR2024-00027
CrowdStrike Exhibit 1002 Page 388

rules 522 to perform comparisons of feature values against thresholds. Threshold detection rules 522 specify how comparisons are to be made.

A threshold for a feature can be included as another feature in the feature vector, or can be obtained from the threshold detection rules 522. A threshold is generally a value for a measurement, which can be unity for default. A unity-valued threshold is useful for true/false statements of a feature. For example, if an ANI has been designated as a source of fraudulent calls, any call from that ANI is considered evidence of fraud. A threshold comparison is simply made by identifying a single event that contains that ANI.

Each feature vector segment 532, 534, 536, and 537 is a feature, and each contains a feature value. Threshold detection rules specify to the threshold detector 520 how to perform comparisons to determine if evidence of fraud exists. Threshold detection rules can include, for example, the following types of rules:

1) if A > "5", create evidence (threshold for A is a value obtained from Threshold Detection Rules 522);
2) if A > B, create evidence (threshold for A is another feature in the feature vector);
3) if A > B #and# B > C, create evidence (can be complex statements); and
4) if D, create evidence (threshold is unity; useful for items like a hot ANI or a stolen calling card number).

If an explicit value for a threshold is not given, it is assumed to be unity.

Referring to FIG. 5D, in a preferred embodiment, enhancer and threshold detector component 509 includes two sets of enhancer and threshold detector pairs 570 and 572. One pair, which can be pair 570, can be dedicated to analyzing single-event features while the other pair 572 can be dedicated to analyzing multiple event features.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 389

Generally, enhancers 510 generally perform complex calculations, as needed for the various measurement types, while threshold detector 520 generally performs simple comparisons. Thus, for added performance, as illustrated in enhancer and threshold detector pair 570, one or more threshold detectors 520 can be provided with two or more enhancers 510. Such an arrangement provides a more adequate and uniform throughput of data.

Threshold detection rules 522 can be created and modified dynamically, in real-time, while thresholding engine 126 is executing. Preferably, rules can be modified in two ways, corresponding with two general formats of rules.

In a first general format, a rule can be a general statement and can reference specific values in a table. For example, a rule can look like "If number of calls from pay phone ANI > *nnn*, create evidence", in which *nnn* is a pointer to a specific value in a table. Rules in this format can be dynamically modified or created by modifying or creating the specific values in the table.

In a second general format, a rule can be hard coded with specific values. For example, a rule can look like "If number of calls from pay phone ANI > 10, create evidence". Rules in this format can be dynamically modified or created by modifying or creating the rule itself.

Threshold detection rules 522 can vary according to the enterprise employing the engine. Preferably, threshold rules stored in database 522 can be dynamically modified without taking the engine off-line. Threshold detection rules 522 can be automatically created and modified both automatically by external pattern recognition engine 132 and manually by human analysts. In one embodiment, where rules are automatically generated by pattern recognition engine 132, threshold detection rules database 522 is updated automatically by pattern recognition engine 132.

Alternatively, where rules are automatically generated by pattern recognition engine 132, threshold detection rules database 522 is updated manually by human analysts. In this alternative embodiment, pattern recognition

IPR2024-00027
CrowdStrike Exhibit 1002 Page 390

engine 132 is used to detect new patterns of fraud, but instead of automatically creating a threshold detection rule, it notifies a human analyst and suggests a new rule. The human analyst can then enter the suggested rule or create a new rule.

A variety of types of rules can be implemented in threshold detection rules 522. For example, threshold detection rules 522 can include, but are not limited to, rules for generating one or more of the following types of alarms.

A long duration (LD) alarm is generated if a single completed call's duration exceeds a duration threshold, X. LD thresholds can be set by product for call type categories such as targeted international, non-targeted international and domestic. The call type category is determined from the international indicator on the normalized event.

An originating, terminating combination duration (OTCD) alarm is generated if a completed call originating from X and terminating to Y has a duration that exceeds Z. The duration threshold Z for a combination X and Y can be set by product. The origination X and termination Y can be specified as NPA-NXX, NPA or country code. There is no hierarchy required for applying the most specific threshold. Users could, however, implement a hierarchy if desired. One event can generate more than one OTCD alarm.

An originating, terminating combination attempt (OTCA) alarm is generated for a single attempt that originates from X and terminates to Y. OTCA alarms can be set by product. The origination X and termination Y can be specified as NPA-NXX, NPA or country code. There is no hierarchy for applying the most specific origination and termination combination. One event can generate more than one OTCA alarm.

A hot originating attempt (HOA) alarm is generated when a call attempt originates from X. The origination X is contained in a list of pre-defined numbers.

A hot terminating attempt (HTA) alarm is generated when a call attempt terminates to X. The termination X is contained in a list of pre-defined numbers.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 391

A hot originating completion (HOC) alarm is generated when a completed call originates from X. The origination X is contained in a list of pre-defined numbers.

A hot terminating completion (HTC) alarm is generated when a completed call terminates to X. The termination X is contained in a list of pre-defined numbers.

A deactivation hot originating attempt (DHOA) alarm is generated when a call attempt originates from a number from which a recently deactivated card had originated. The measure of how recently the card must have been deactivated is a time parameter T.

A deactivation hot termination attempt (DHTA) alarm is generated when a call attempt terminates to a number to which a recently deactivated card had terminated. The measure of how recently the card must have been deactivated is a time parameter T.

A deactivation hot originating completion (DHOC) alarm is generated when a completed call originates from a number from which a recently deactivated card had originated. The measure of how recently the card must have been deactivated is a time parameter T.

A deactivation hot termination completion (DHTC) alarm is generated when a completed call terminates to a number to which a recently deactivated card had terminated. The measure of how recently the card must have been deactivated is time T.

A pin hacking origination (PHO) alarm is generated when X number of attempts from the same origination fail PIN validation within time T. The number of attempts X is accumulated across all PIN validated products based on infodigit. Call attempts with infodigits indicating pay phones are screened out of the count.

A pin hacking billed number (PHBN) alarm is generated when X number of attempts on the same billed number fail PIN validation within time T. The

IPR2024-00027
CrowdStrike Exhibit 1002 Page 392

billed number is calculated by dropping the last four digits from the BTN, i.e., dropping the 4 digit PIN. The number of attempts X is accumulated across all PIN validated products based on infodigit. Call attempts with infodigits indicating pay phones are screened out of the count.

A simultaneous international (SI) alarm is generated when X number of completed international calls using the same authcode /BTN overlap in time by at least 2 minutes within a sliding window of time T. The X number of international calls is specified by product. The sliding window of time T within which simultaneity is checked can not exceed the purge time for normalized events. An international call is determined from the international indicator in the normalized event.

A simultaneous domestic (SD) alarm is generated when X number of completed domestic calls using the same authcode/BTN overlap in time by at least 2 minutes within a duration T of the generating event. The X number of domestic calls is specified by product. The duration T, within which simultaneity is checked, can not exceed the purge time for normalized events. A domestic call is determined from the international indicator in the normalized event.

A simultaneous all (SA) alarm is generated when X number of completed calls using the same authcode/BTN overlap in time by at least two minutes within a duration T of the generating event. The X number of calls is specified by product. The duration T within which simultaneity is checked can not exceed the purge time for normalized events. This alert includes both international and domestic calls.

A geographic velocity check is a check for a pair of calls using the same authcode-plus-PIN/BTN originating from locations between which it would be impossible for a caller to travel during the interval between the calls. Geographic velocity check alarms can be calculated either by specifying time for combinations of originations and terminations or by specifying a

IPR2024-00027
CrowdStrike Exhibit 1002 Page 393

latitude/longitude for each country or NPA and a maximum travel speed and performing a time calculation.

A geographic velocity check international (GVCI) alarm is generated when, for X number of pairs of international call completions using the same authcode-plus-PIN/BTN, each pair occurs within a time interval T1, each pair is not simultaneous and each pair occurs within a sliding window of time T2. The X number of pairs of calls is specified by product.

The interval T1 for a pair of calls is determined by the pair's combination of the originating ANI's NPAs and/or country codes. Determining whether the calls are made within a given interval is calculated from the difference between the terminating time of the first call and the originating time of the second call.

The sliding window of time T2 within which the geographic velocity check is performed cannot exceed the purge time for normalized events. An international call is determined from the international indicator in the normalized event.

Similarly, a geographic velocity check domestic (GVCD) alarm is generated when X number of pairs of domestic call completions, using the same authcode-plus-PIN/BTN, would have been impossible for a single caller to make. A geographic velocity check all (GVCA) is generated when X number of pairs of call completions, regardless of domestic or international classification, using the same authcode-plus-PIN/BTN, would have been impossible for a single caller to make.

For certain types of calls, such as toll free or pay to dial calls for example, switch blocks can be configured to block such calls to a set of countries. There are instances, however, where a country block on a switch fails. Thus a failed country block (FCB) alarm can be generated if a call is made from a switch with a block in place to a blocked country. This type of alarm makes use of data indicating, by switch identification, the blocked country codes.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 394

A completed call interval (CCI) alarm is generated when one or more completed calls on an authcode-plus-PIN/BTN exceed one or more thresholds. Thresholds can include cumulative call minutes, completed call counts and cumulative dollars. Thresholds can be based on targeted international, non-targeted international and domestic call type categories. Call type category is determined from an international indicator on the normalized event. CCI alerts can include detection rules comparable to current FMS ceiling and interval alarms.

In step 416, when threshold detector 520 receives a feature vector, it reads through threshold detection rules 522 and applies any rules that are designated for any features included in the feature vector. For example, in FIG. 5C, if a feature vector 518 includes features such as, for example, features 538, 540 and 544, threshold detector 520 will apply any rule that includes comparisons for those features.

If the network event record does not exceed the threshold, no further action is necessary and the process stops at step 418. If, however, the network event record exceeds or violates the threshold rule, evidence, or an indication of possible fraud, is generated.

An indication, or evidence, of fraud preferably includes at least a record containing a priority indicator, account name, and a suspicious event set. Actual contents of evidence are defined by implementation of the infrastructure. Suspicious event sets include a union of all events from the event sets of features that resulted in evidence of fraud. In step 420, if an indication, or evidence, of fraud is generated, threshold detector 520 generates an alarm 526, which is passed to analysis layer 133, preferably with the evidence.

Recall that all events that are received by normalizer 502 are preferably stored in event DB 125 for a period of time that can be designated by the user. A storage period can, for example, be set to twenty four hours. In addition, events that are identified in a suspicious event set can be kept longer for analysis

IPR2024-00027
CrowdStrike Exhibit 1002 Page 395

purposes. Suspicious events can be stored in event DB 125 with in indication of suspicion or can be stored in a separate database, so that they are not purged after the typical storage period.

3. Profiling

Profiling is a process whereby normalized dispatched network event records are compared to profiles representing normal and fraudulent use of one or more telecommunications networks. Profiling can require historical network event records to help determine whether a current network event record, fits a profile of fraudulent use of a telecommunications network.

Referring to FIG. 4, steps 422-428 and 420 represent a preferred process of profiling. In FIG. 6, details of profiling engine 128 are provided for implementing the profiling process of steps 422-428 and 420.

The process begins in step 422, where normalized event records 506b are sent to profiling engine 128 and are received by profiling enhancer 624. Profiling enhancer 624 provides additional data for the profiling processor 634 that will enable the application of appropriate profile detection rules 636. Profiling enhancer 624 operates similar to thresholding enhancer 508 except that profiling enhancer 624 can use a different configuration data component 626 since different types of data can be needed to create an enhanced event record 632 for profiling. Preferably, enhancer component 510 and 625 have similar fundamental structures, yet operate differently through the use of specific configuration data 626 and 512. Configuration data 626 specifies which data is required, where to find it and how to add it to event records. Enhancer 624, based on instructions from configuration data 626, then requests this data from an informant 628.

Informant 628 provides a communication interface to each of several external systems 630 from which additional data needed by the enhancer 624 will be retrieved. Informant 628, similar to threshold informant 514, is used to retrieve

IPR2024-00027
CrowdStrike Exhibit 1002 Page 396

required data from external systems 630. Again, the use of modular configuration data 626 and informant 628 components provides the current invention with ease of configurability and portability. Based on a request it receives from the enhancer 624, informant 628 sends a query to the appropriate external system 630 using the communications protocol of the external system 630. Informant 628 receives the requested data and supplies it to enhancer 624.

Enhancer 624 augments normalized event records 606b using data received from informant 628, and thus creates an enhanced event record 632.

In step 424, a profiling processor 634 receives enhanced event records 632 for applying against one or more profiles. Using certain parameters from the enhanced event record 632, the profiling process 634 selects an appropriate profile detection rule 636, several of which are kept in a database. Rules 636 determine which profile from the profile database 130 the event should match up to.

Profiles can be created in any of a number of ways. Profiles can be created as user, product, geographical, or global profiles. Profile database 130 can be an object database, a set of tables, a set of real numbers representing weighting factors for a neural network (as is used in AI technology), or other various forms. Preferably, profiles representing both normal and fraudulent patterns are stored.

In a preferred embodiment, profile development and profile matching employs Artificial Intelligence (AI) technology. Although several AI systems exist for such a purpose, the preferred embodiment utilizes statistical-based algorithms (rather than rules-based) to process volumes of known fraudulent and normal patterns received by pattern recognition engine 132 in order to build fraudulent and normal profiles. Preferably, an AI-based profiling processor 634 also trains itself to formulate profile rules 636 that allows it to match events to profiles and detect departures from normal profiles.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 397

In step 426, profiling component 634 retrieves an appropriate profile from the profile database 130, and compares the event with the profile. If an event falls within the selected profile or profiles, testing stops at step 428. If, however, a departure from the selected profile is detected, profiling component 634 generates an alarm 638 in step 420. Preferably, a probability of fraud is calculated based on the significance and degree of departure and is expressed as a percentage or weighting factor. Preferably, at least step 426 is performed with the aid of Artificial Intelligence (AI) technology. Alarm 638 is then sent to analysis layer 133.

4. Pattern Recognition

Pattern recognition is a process whereby network event records are analyzed to learn and to identify normal and potentially fraudulent patterns of use in a telecommunications network. Referring to FIG. 4, steps 430-438 represent a preferred process for pattern recognition and for updating threshold rules and profiles. In FIG.7, details of pattern recognition engine 132 are provided for implementing the pattern recognition process of FIG. 4. Pattern recognition engine 132 can receive feedback from other layers and can employ components which self-teach fraudulent and non-fraudulent patterns from such feedback.

In step 420, normalized event records 506c are sent to pattern recognition engine 132 where they are received by a pattern recognition enhancer 740. Pattern recognition enhancer 740 operates in much the same manner as the thresholding and profiling enhancers 510 and 624, respectively, except that enhancer 740 employs a different configuration data component 742. Also similar to the thresholding and profiling processes, enhancer 740 uses an informant 744 for data retrieval from external systems 746. This data is used to enhance normalized event records 506c to create enhanced event records 748.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 398

In step 432, enhanced event records 748 are sent to an update and store component 750, which maintains a call history database 752. Update and store component 750 enters each record 748 into the call history database 752. Call history database 752 contains volumes of call data that can be analyzed by a pattern analysis processor 754.

In step 434, pattern analysis processor 754 analyzes call histories from call history database 752 to determine whether any interesting patterns emerge. Interesting patterns include patterns which can be fraudulent and patterns which can be non-fraudulent. Non-fraudulent patterns recognition is important to minimize processing of non-fraudulent information.

If an interesting pattern is detected, pattern analysis processor 754 determines whether it is a fraudulent or non-fraudulent pattern. To accomplish this, pattern analysis processor 754 uses artificial intelligence technology to train itself in identifying fraudulent patterns. By analyzing volumes of events from call history 752, an AI-based pattern analysis processor 754 first determines normal patterns and then looks for deviations that can be identified as fraudulent. Processor 754 then detects emerging patterns of such deviations and identifies them as fraudulent patterns.

There are various AI systems available for such a purpose. Examples include tree-based algorithms that obtain discrete outputs, neural networks, and statistical-based algorithms that use iterative numerical processing to estimate parameters. Such systems are widely used for pattern recognition. By utilizing an AI system for pattern analysis 754, both normal and fraudulent patterns can be identified from the volumes of data stored in the call history database 752.

In step 436, pattern analysis processor 754 uses the results of step 434 to modify threshold detection rules 522 via thresholding interface 756. By recognizing fraudulent patterns, certain thresholding rules can be updated to reflect the most recent patterns of fraud. For example, pattern analysis processor 754 can detect a fraudulent pattern emerging with calling card calls made on

IPR2024-00027
CrowdStrike Exhibit 1002 Page 399

weekends to a certain country from certain states in the U.S. It can then update a thresholding rule to generate an alarm 526 whenever such a call is made.

In step 438, pattern analysis processor 754 uses the results of step 434 to modify profiles in profile database 130 via profiling interface 758. Pattern analysis processor 754 feeds known fraudulent and normal patterns to profiling processor 634. Using AI technology, profiling processor 634 processes these patterns to build profiles that are identified as fraudulent or normal. Thus, through use of AI-based pattern recognition, the invention allows for fraud detection to keep pace with the most current schemes of fraud.

The processes of threshold detection, profiling and pattern recognition are described as being performed substantially in parallel primarily to reduce the processing time. The processes can, however, be performed one after another or as some combination of parallel and non-parallel processing.

B. Analysis Layer

Referring to FIG. 8, a a process flowchart is provided for a preferred alarm analysis process in which alarms generated in step 420 are analyzed by analysis layer 133 to consolidate and correlate the alarms into fraud cases. Upon case creation or the addition of a new alarm to a case, a case priority is calculated or recalculated. Case priority is calculated from configurable prioritization rules that can make use of any field of the case. Prioritization rules order cases so that there are likely to be fewer false positives at the top of the priority list. Prioritized cases are presented to an expert system and to a human analyst.

Referring to FIG. 9, details of analysis layer 133 are provided for implementing the process of FIG.8. Analysis layer 133 consolidates alarms by examining various features of each alarm and correlating those that are possibly related to the same fraudulent activity. Analysis layer 133 consolidates correlated alarms to build "cases" of fraud, thus reducing the total number of data that must

IPR2024-00027
CrowdStrike Exhibit 1002 Page 400

be further examined. A case of fraud consists of alarms that can span several types of services, but which are possibly related to the same events or callers. Preferably analysis layer 133 includes a core infrastructure portion and a user-specific implementation portion.

Alarms are consolidated into cases according to analysis rules, or case types. Cases include alarms that have some aspect in common. Alarms can be consolidated into one or more cases depending on the analysis rules used. Each alarm must belong to at least one case. If an alarm is created that matches an existing non-closed case, the alarm will be added to the case. Otherwise, a new case is created for the alarm.

In step 810, alarms 526 from thresholding engine 126 and alarms 638 from profiling engine 128 are received by an alarm enhancer 902. Alarms 526 and 638 represent instances of possible fraud and designate fraud service type such as cellular, calling card, etc. Alarms 638 from profiling engine 128 are preferably accompanied by probability of fraud.

In step 812, these alarms are enhanced. Alarm enhancer 902 is similar to the detection layer enhancers 510, 624 and 740. Enhancer 902 augments alarms 526 and 638 to produce enhanced alarms 910. Configuration data 904 specifies additional information which can be needed and how that information should be added, based on the type of alarm received. Additional information can be retrieved from billing, accounts receivable, order entry, or various other external systems 908. For example, informant 906 can access an accounts receivable system to determine that "this ANI has a past due account of $1000". Similarly, informant 906 can access an order entry system to determine that "this calling card number was deactivated two months ago". Informant 906 communicates with various external systems 908 to retrieve the requested information. Alarm enhancer 902 then adds this information to alarms 526 and 638 and produces enhanced alarms 910.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 401

In step 814, enhanced alarms 910 are sent to a fraud case builder 912 for correlating and consolidating various alarms into related fraud cases. This correlation is based on common aspects of alarms. An example of such a common aspect is "alerts having the same calling card number".

Correlation is governed by analysis rules 914, which can be programmed and kept in a rules database. Rules 914 can use the probability of fraud which was assigned by profiling processor 634 as a parameter. For example, a rule can state "only build cases from alarms having greater than 50% probability of fraud and which are generated for the same account".

In operation, fraud case builder 912 receives an enhanced alarm 910 and determines whether there is an existing case or cases in which to place the alarm. Fraud case builder 912 looks for features which the alarm can have in common with existing cases, using analysis rules 914. If no existing case is appropriate, fraud case builder 912 creates a new case for the alert.

Fraud case builder 912 attempts to prevent duplication of cases. Fraud case builder 912 also attempts to prevent corruption of cases, which otherwise might occur due to the distributed platform of the invention. For example, in this parallel processing environment, multiple instances of the same process, such as updating a case, can occur. If two analysts are attempting to update the same case with either identical data or different data, case builder 912 attempts to ensure that the case reflects both data if the data are different, and attempts to ensure that duplicate cases are not created if both data are identical. Fraud case builder 912 can employ a case-locking mechanism to accomplish this goal.

A primary objective of the analysis layer is to reduce the amount of data that an analyst must examine. Thus, although an alarm can go into more than one fraud case, overall reduction of data can still be achieved. For example, an alarm for a calling card call can designate both the card number and the ANI. The alert can then be placed in a first case of consolidated alarms based on a common card number, and in a second case of consolidated alarms based on a common ANI.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 402

However, overall reduction of alerts will generally be achieved as the number of alarms that are consolidated exceeds the number of alerts that are placed in more than one case.

In step 816, fraud case builder 912 outputs fraud cases 916 to expert system layer 139 for further analysis and action.

C. Expert System Layer

Referring to FIG. 10, a process flowchart is provided for analyzing fraud cases and for automatically acting upon certain cases. The process analyzes cases by prioritizing them, adding additional information relevant to each case and performing automated actions for some cases. Since some actions are performed automatically, the action rules which trigger automatic actions are applied only to cases where a very high probability of fraud exists or where the potential cost of fraud is significant. In Fig. 11, details of expert system layer 139 are provided for implementing the process of FIG. 10.

In step 1010, fraud cases 916 are sent from analysis layer 133 and received by a prioritizer 1102. Prioritizer 1102 enhances fraud cases 916, assigns priority to them and determines whether any actions should be automatically executed.

In step 1012, fraud cases 916 are enhanced. To enhance cases, prioritizer 1102 uses configuration data 1104, an informant 1106, and external systems 1108. Configuration data 1104 specifies any additional information which can be needed for fraud cases 916, where to find the data and how to add the data to a fraud case. Informant 1106, serves as a communication interface to external systems 1108. Informant 1106 sends a query to an appropriate external system 1108 using a communications protocol of the external system 1108. Informant 1106 receives the requested data and supplies it to prioritizer 1102. Prioritizer

IPR2024-00027
CrowdStrike Exhibit 1002 Page 403

1102 then adds the information to a fraud case, creating an enhanced fraud case 1114.

In step 1014, enhanced fraud cases are prioritized. To assign priority to cases, prioritizer 1102 employs prioritization rules which are kept as part of configuration data 1104. Prioritization rules are based on experienced analysts' rules for determining and prioritizing fraud cases. These rules are programmed into configuration data 404 as logical algorithms. Prioritization rules can also employ parameters retrieved from external systems such as, "how much has customer's past revenue been." Such parameters are useful for determining the potential cost of a fraud case, which is influential in determining priority.

In step 1016, prioritized fraud cases are analyzed and appropriate actions are specified based on action rules. In determining whether to initiate action on a case, prioritizer 1102 uses action rules which are kept as part of configuration data 1104. An action is a response to suspected fraud and involves an external system. Examples of actions include card deactivation or activation or use-range privilege modifications. Actions are categorized as automatic, semi-automatic, user initiated, mail initiated or manual. Automatic and semi-automatic actions are initiated by an expert system, following pre-defined rules. Other actions are typically initiated by human analysts.

Semi-automatic actions are initiated by an expert system under pre-specified conditions. For example, under a pre-specified condition of excessive back-logs, the expert system can automatically deactivate high-priority fraud cases.

User initiated actions are performed upon user requests. Actions can include activations and deactivations of accounts such as telephone accounts, credit card account and debit card accounts.

Electronic mail initiated action are performed upon a request received from external groups, such as a customer service group, through electronic mail

IPR2024-00027
CrowdStrike Exhibit 1002 Page 404

scripts. Actions can include activations and deactivations of accounts such as telephone accounts, credit card account and debit card accounts.

Manual actions are initiated by users external to the system and executed independent of the system. The external user can request that the system record that the action was performed.

Action rules are based on experienced analysts' rules for specifying actions to take on a fraud case. Actions can include deactivating a calling card number, placing a switch block on an ANI or sending out a notice to a customer. Action rules can be programmed as logical algorithms and can consider such parameters as priority (i.e., "for any case over priority level N, deactivate account") and type of service (i.e., cellular. calling card, dial 1). Action rules can include data retrieved from external systems.

Action rules form part of the implementation layer. Action rules for manual actions can reference special handling instructions for acting upon suspected fraud on certain customer accounts. For example, special instructions might indicate to contact a customer fraud investigation unit rather than contacting card holder whose card is suspected of fraudulent activity.

Action rules are programmed into configuration data 404 as logical algorithms to be applied to enhanced cases. These actions can be based on priority (i.e., "for any case over priority level N, deactivate account"), type of service (i.e., cellular. calling card, dial 1), or enhanced data retrieved from external systems 1108.

In step 1018, an enforcer 1110 executes actions specified in a step 1016 interfacing with various external action systems 1112. External action systems can include switch interface systems for switch-based ANI blocks, order entry systems for account deactivations, network control systems for calling card deactivations, electronic mail system for e-mail notices to customers or internal personnel, customer service centers, print centers for mailing dunning letters and various other systems. Enforcer 1110 preferably resides on servers 310a ... 310n.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 405

In operation, enforcer 1110 receives a request for action from prioritizer 1102 and interfaces with an appropriate external action system 1112 to execute the action. External action systems 1112 can include switch interface systems for switch-based ANI blocks, order entry systems for account deactivations, network control systems for calling card deactivations, electronic mail system for e-mail notices to customers or internal personnel, customer service centers, print centers for mailing dunning letters, or various other systems.

Since these actions are performed automatically, the action rules that trigger them are preferably applied only to cases where a high probability of fraud exists or where the potential cost of the fraud is significant. Prioritizer 1102, along with the prioritization rules and action rules that are kept as configuration data 1104, perform as an expert system, applying expert rules to data records to determine what actions to perform. An off-the-shelf expert system can be used. However, it is preferable to program a customized expert system using a logic-based language appropriate for expert systems, such as CLIPS.

Cases in which an automated action is not warranted are sent to presentation layer 143 for further examination and potential action from a human analyst. Also sent as part of an enhanced case 414 to presentation layer 143 are data not useful to the automated expert system layer 139, such as text notes added to a case by a human analyst working at workstation 152a-152n.

D. Presentation Layer

Fraud cases 1114, including those which warrant automatic action in the expert system layer 139 and those which do not, are sent to presentation layer 143 for examination and potential action by a human analyst at workstations 152a-152n.

Referring to FIG. 12, details of a presentation layer 143 are provided for interfacing the detection layer 123 and analysis layer 123 with human analysts

IPR2024-00027
CrowdStrike Exhibit 1002 Page 406

working at workstations 152a ... 152n which are connected to LAN 150. Preferably, fraud cases 1114 include data generated in upper layers such as probability of fraud, information retrieved from external systems, and any actions already taken. Presentation layer 143 preferably allows human analysts to retrieve and add additional data and to perform actions similar to those performed by the expert system layer 139.

Enhanced fraud cases 1114 from expert system layer 139 are received by case enhancer 1202. Similar to data enhancers of the upper layers, case enhancer 1202 uses configuration data 1204, an informant 1206, and external systems 1208 to augment enhanced fraud cases 1114 with additional information relevant for presentation to an analyst.

A presentation interface 1210 serves as an interface to workstations 152a ... 152n, providing data for graphical presentation to the analyst. Fraud cases are presented according to presentation rules 1212, which are programmed as logical algorithms into a database and are configurable. Presentation interface 1210 employs an informant 1214 and external systems 1216 to retrieve additional information. However, this is not automatic, as in the upper layers. Rather informant 1214 retrieves data from external systems 1216 based on commands from analysts at work stations 152a ... 152n. For example, an analyst can view a case and decide that a customer's payment history is needed prior to taking any action. The analyst, via a workstation 152a ... 152n, sends a command to presentation interface 1210 requesting this data. Presentation interface 1210 then instructs informant 1214 to retrieve this data from an external accounts receivable system 1216.

Presentation interface 1210 uses enforcer 1218 for performing actions. This is not automatic, as it was in the expert system layer 139. Rather, enforcer 1218 performs actions based on commands from analysts. For example, an analyst can decide that a switch block on an ANI is needed. The analyst, via a workstation 152a ... 152n, sends a command to presentation interface 1210

IPR2024-00027
CrowdStrike Exhibit 1002 Page 407

requesting a switch-based ANI block. Presentation interface 1210 then instructs enforcer 1218 to execute the command. Enforcer 1218 interfaces with an external action, switch interface system 1220 to implement the ANI switch block. Other external action systems 1220 can be similar to those employed by expert system layer 239.

IV. Conclusions

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 408

What Is Claimed Is:

1. A method for detecting fraud in a telecommunications system, comprising the steps of:

(1) performing a plurality of types of fraud detection tests on network event records;

(2) generating fraud alarms upon detection of suspected fraud by any of the fraud detection tests;

(3) correlating the fraud alarms into fraud cases based on common aspects of the fraud alarms; and

(4) automatically responding to certain of the fraud cases.

2. The method of claim 1, wherein step (1) comprises the steps of:

(a) normalizing the network event records from any of a variety of formats into standardized formats;

(b) dispatching normalized network event records to at least one fraud detecting engine; and,

(c) parallel processing the dispatched portions in the plurality of fraud detecting engines.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 409

3. The method of claim 2, wherein step (c) comprises the step of enhancing the normalized dispatched network event records prior to testing for fraud.

4. The method of claim 1, wherein step (1) comprises the steps of:

(a) selecting a threshold rule from a plurality of threshold rules stored in a threshold rule database; and

(b) determining whether a network event record violates the selected threshold rule.

5. The method of claim 4, further comprising the step of:

(c) updating the threshold rule database during run-time.

6. The method of claim 5, wherein step (c) comprises the steps of:

(i) analyzing the network event records to identify new methods of fraud;

(ii) generating new threshold rules for detecting the new methods of fraud; and

(iii) updating the threshold rule database with the new threshold rules.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 410

7. The method of claim 5, wherein step (c) comprises the steps of:

(i) analyzing the network event records to identify new methods of fraud using artificial intelligence;

(ii) generating new threshold rules for detecting the new methods of fraud using artificial intelligence; and

(iii) updating the threshold rule database with the new threshold rules.

8. The method of claim 1, wherein step (1) comprises the steps of:

(a) selecting a profile from a plurality of profiles stored in a profile database; and

(b) determining whether a network event record violates the profile;

wherein step (2) comprises the step of generating an alarm and a probability of fraud based upon the extent of the departure if a network event record violates the profile.

9. The method of claim 8, further comprising the step of:

(d) updating the profile database during runtime.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 411

10. The method of claim 9, wherein step (d) comprises the steps of:

(i) analyzing the network event records to identify new methods of fraud;

(ii) generating new profiles representative of the new methods of fraud; and

(iii) updating the profile database with the new profiles.

11. The method of claim 10, wherein steps (ii) and (iii) are performed via artificial intelligence.

12. The method of claim 1, wherein step (3) comprises the step of :

(a) prioritizing the fraud cases to indicate a probability of fraud associated with each of the fraud cases.

13. The method of claim 12, wherein step (a) comprises the step of enhancing the fraud alarms with data prior to correlating.

14. The method of claim 12, wherein step (a) comprises the steps of:

(i) retrieving data from an external system; and

(ii) enhancing the fraud alarms with the retrieved data prior to correlating.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 412

15. The method of claim 1, further comprising the steps of:

(5) presenting the fraud cases to live operators; and

(6) manually responding to certain of the fraud cases.

16. A multi-layer fraud detection system for a telecommunications system, the telecommunication system comprising a network layer having at least one telecommunications network, a service control layer for managing the network layer and for generating service records containing data representing instances of telecommunications in the network layer, and a data management layer for receiving the service records from various components and processes of the service control layer and for reducing data by eliminating redundancy and consolidating multiple records into network event records, the multi-layer fraud detection system comprising:

a detection layer to receive network event records from the data management layer, to test the network event records for possible fraud and to generate alarms indicating incidences of suspected fraud;

an analysis layer to receive alarms generated by the detection layer and to consolidate the alarms into fraud cases; and

an expert system layer to receive fraud cases from the analysis layer and to act upon certain of the fraud cases.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 413

17. The multi-layer fraud detection system of claim 16, wherein said detection layer comprises a core infrastructure and a configurable, domain-specific implementation.

18. The multi-layer fraud detection system of claim 17, wherein said detection layer further comprises at least one fraud detection engine having a core infrastructure and a configurable, domain-specific implementation.

19. The multi-layer fraud detection system of claim 18, wherein said detection layer further comprises:

a network event normalizer to convert network event records from any of a plurality of formats into standardized formats for processing by said at least one fraud detection engine; and

a dispatcher to dispatch portions of said normalized network event records to said at least one fraud detection engine.

20. The multi-layer fraud detection system of claim 18, wherein said at least one fraud detection engine comprises a rules-based thresholding engine.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 414

21. The multi-layer fraud detection system of claim 18, wherein said at least one fraud detection engine comprises:

a configurable enhancer that augments event records with additional data; and

a configurable informant to interface said enhancer to an external system and to retrieve additional data from the external system.

22. The multi-layer fraud detection system of claim 21, further comprising:

means for interfacing said informant with the external system in a format native to the external system; and

a rules database comprising instructions for processing the enhanced event records to detect fraud.

23. The multi-layer fraud detection system of claim 22, wherein:

said at least one fraud detection engine includes a rules-based thresholding engine; and

said rules database comprises threshold rules for use by said rules-based thresholding engine.

24. The multi-layer fraud detection system of claim 22, wherein:

said at least one fraud detection engine includes a profiling engine; and

said rules database comprises profiles for use by said profiling engine.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 415

25. The multi-layer fraud detection system of claim 18, wherein said detection layer further comprises a pattern recognition engine that learns new patterns of fraud and that generates updates for said at least one fraud detection engine.

26. The multi-layer fraud detection system of claim 16, wherein said analysis layer comprises a core infrastructure and a configurable, domain-specific implementation.

27. The multi-layer fraud detection system of claim 26, wherein said analysis layer further comprises:

a configurable alarm enhancer to augment fraud alarms with data;

a configurable informant to interface said alarm enhancer to an external system and to retrieve additional data from the external system; and

a configurable fraud case builder to consolidate fraud alarms that are generated by said detection layer.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 416

28. The multi-layer fraud detection system of claim 27, wherein said user-specific implementation layer of said analysis layer further comprises:

means for interfacing said informant with the external system in a format native to the external system; and

an analysis rules database comprising instructions for said fraud case builder for filtering and correlating fraud alarms into fraud cases according to at least one common attribute.

29. The multi-layer fraud detection system of claim 28, wherein said at least one common attribute is one of the following attributes:

ANI;

originating switch;

credit card number;

DNIS;

destination country;

originating geographic area;

originating area code; and

calling equipment type.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 417

30. The multi-layer fraud detection system of claim 16, wherein said expert system layer comprises a core infrastructure and a configurable, domain-specific implementation.

31. The multi-layer fraud detection system of claim 30, wherein said expert system layer domain-specific implementation comprises:

a configurable prioritizer that generates enhanced fraud cases, prioritizes the enhanced fraud cases and directs actions on external action systems for certain of the prioritized, enhanced fraud cases;

a configurable informant that interfaces said alarm enhancer to an external system and that retrieves the additional data from the external system; and

a configurable enforcer that interfaces said prioritizer to an external action system and that directs execution of actions by the external action system based upon commands that are generated by the prioritizer.

32. The multi-layer fraud detection system of claim 31, wherein said user-specific implementation layer of said expert system layer includes a configuration database, and wherein said configuration database comprises:

means for interfacing said informant with the external system in a format native to the external system; and

prioritizing rules for use by the prioritizer.

33. The multi-layer fraud detection system of claim 16, further comprising:

a presentation layer that receives prioritized fraud cases from said expert system layer and that presents the prioritized fraud cases to live operators, wherein said presentation layer includes a core infrastructure and a configurable, domain-specific implementation.

34. The multi-layer fraud detection system of claim 33, wherein said domain-specific implementation of said presentation layer comprises:

a configurable case enhancer that enhances prioritized fraud cases with additional data;

a configurable presentation interface that distributes the enhanced, prioritized fraud cases to one or more workstations and that sends

IPR2024-00027
CrowdStrike Exhibit 1002 Page 419

action commands generated at the workstations to an external action system;

a configurable first informant that interfaces said case enhancer to a first external system and that retrieves data from the first external system;

a configurable second informant that interfaces said presentation interface to a second external system and that retrieves data from the second external system, based upon commands generated at the workstations; and

a configurable enforcer that interfaces the workstations, via said presentation interface, to the external action system and that directs execution of actions by the external action system based upon commands generated at the workstations.

35. The multi-layer fraud detection system of claim 34, wherein the first and second external systems are each a part of the same external system.

36. The multi-layer fraud detection system of claim 34, wherein said user-specific implementation layer of said presentation layer further comprises:

means for interfacing said informant with the first external system in an interfacing format that is native to the first external system; and

configurable presentation rules to direct presentation of enhanced, prioritized fraud cases at the workstations.

37. A method of detecting fraud in a telecommunications system, comprising the steps of:

(1) determining whether a network event record violates a selected threshold rule;

(2) generating an alarm when the network event record violates the selected threshold rule;

(3) determining whether the network event record deviates from a selected profile; and

(4) generating an alarm when the network event record deviates from the selected profile.

38. The method of claim 37, further comprising the step of:

(5) performing steps (1) and (3) in parallel.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 421

39. The method of claim 37, further comprising the steps of:

(5) analyzing a history of network event records to identify normal patterns of use in a telecommunications system; and

(6) analyzing a history of network event records to identify fraudulent patterns of use in a telecommunications system.

40. The method of claim 37, further comprising the steps of:

(5) analyzing a history of network event records to identify normal patterns of use in a telecommunications system, using artificial intelligence; and

(6) analyzing a history of network event records to identify fraudulent patterns of use in a telecommunications system, using artificial intelligence.

41. The method of claim 40, further comprising the step of:

(7) generating at least one of the following types of updates when a fraudulent pattern of use is identified:

updates for a threshold rules database; and

updates for a profile database.

42. A system for processing event records, comprising:

a scalable core infrastructure that can be implemented in more than one application; and

a configurable, domain-specific implementation including configurable rules.

43. The system according to claim 42, wherein said core infrastructure is implemented as part of a telecommunications fraud detection system and said configurable, domain-specific implementation comprises:

thresholding rules for testing telecommunications network event records; and

profiles for comparison to telecommunications network event records.

44. The system according to claim 42, wherein said core infrastructure is implemented as part of a credit card fraud detection system and said configurable, domain-specific implementation comprises:

thresholding rules for testing credit card event records; and

profiles for comparison to credit card event records.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 423

45. The system according to claim 42, wherein said core infrastructure is implemented as part of a data mining system and said configurable, domain-specific implementation comprises:

thresholding rules for testing data mining event records; and

profiles for comparison to data mining event records.

46. The system according to claim 42, wherein said core infrastructure is implemented as part of a consumer purchasing pattern analysis system and said configurable, domain-specific implementation comprises:

thresholding rules for testing consumer purchasing event records; and

profiles for comparison to consumer purchasing event records.

47. The system according to claim 42, further comprising:

a detection layer that detects and normalizes event records, dispatches event records to one or more detection engines and generates alarms when an event record meets a condition;

an analysis layer that receives alarms from said detection layer and that consolidates the received alarms into cases based upon common traits of the alarms; and

an expert system layer that receives cases from said analysis layer and that acts upon certain cases.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 424

48. The system according to claim 47, said detection layer further comprising:
means for generating feature vectors for representing multiple occurrences of an event feature.

49. The system according to claim 42, further comprising:
a presentation layer that receives cases from the detection layer, presents the received cases to human analysts, receives commands from human analysts and sends instructions to external action systems to take actions based upon the commands from human analysts.

50. A computer program product comprising a computer useable medium having computer program logic stored therein, said computer program logic for enabling a computer system to process event records, wherein said computer program logic comprises:
means for enabling the computer to test event records;
means for enabling the computer to generate alarms upon certain tested conditions;
means for enabling the computer to correlate the alarms into cases based on common aspects of the alarms; and
means for enabling the computer to respond to certain of the cases.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 425

51. The computer program product according to claim 50, further comprising:

means for enabling the computer to present the cases to live operators; and

means for enabling the computer to permit the live operators to manually initiate responses to certain of the cases.

52. The computer program product according to claim 50, further comprising:

user configurable means for enabling the computer to process user-specific types of event records; and

core infrastructure means for enabling the computer to process event records for a variety of types of event records.

53. A method for processing event records, comprising the steps of:

(1) testing event records;

(2) generating alarms upon certain tested conditions;

(3) correlating the alarms into cases based on common aspects of the alarms; and

(4) generating responses to certain of the cases.

54. The method according to claim 53, further comprising the steps of:

(5) presenting the cases to live operators; and

(6) permitting the live operators to manually initiate responses to certain of the cases.

IPR2024-00027
CrowdStrike Exhibit 1002 Page 427

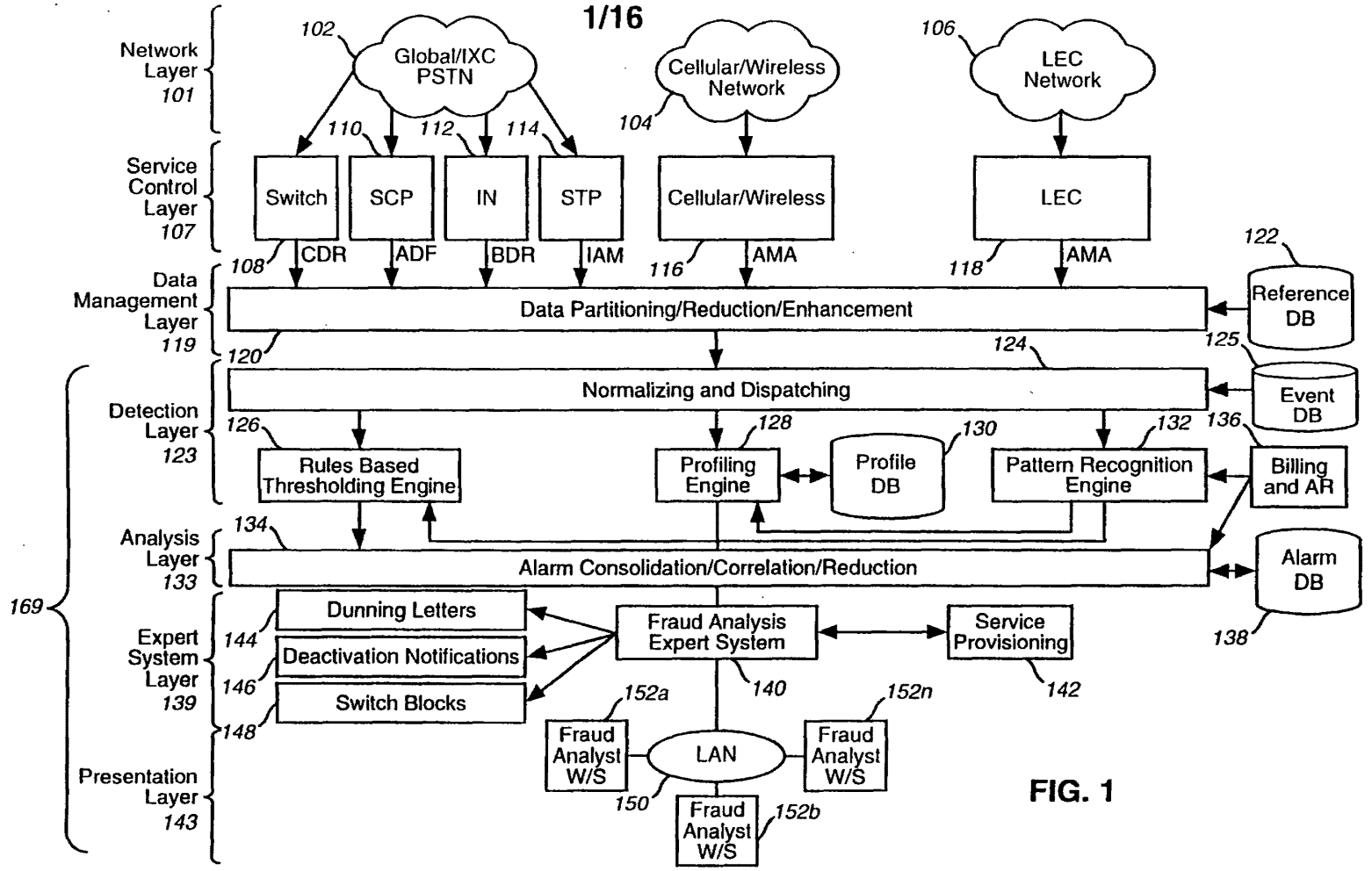

1/16
Network Layer 101
Service Control Layer 107
Data Management Layer 119
Detection Layer 123
Analysis Layer 133
Expert System Layer 139
Presentation Layer 143
169
102
Global/IXC PSTN
104
Cellular/Wireless Network
106
LEC Network
110
112
114
Switch
SCP
IN
STP
Cellular/Wireless
LEC
108
CDR
ADF
BDR
IAM
116
AMA
118
AMA
122
Reference DB
125
Event DB
136
Billing and AR
Data Partitioning/Reduction/Enhancement
120
124
Normalizing and Dispatching
126
Rules Based Thresholding Engine
128
Profiling Engine
130
Profile DB
132
Pattern Recognition Engine
134
Alarm Consolidation/Correlation/Reduction
Alarm DB
138
Dunning Letters
144
Deactivation Notifications
146
Switch Blocks
148
Fraud Analysis Expert System
140
Service Provisioning
142
152a
Fraud Analyst W/S
LAN
150
152n
Fraud Analyst W/S
Fraud Analyst W/S
152b

FIG. 1

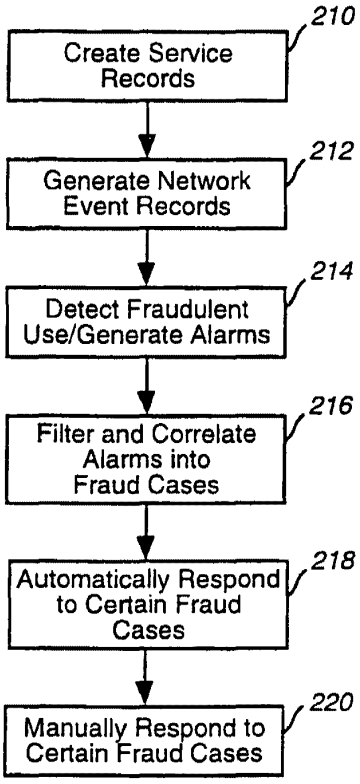


FIG. 2

IPR2024-00027
CrowdStrike Exhibit 1002 Page 429