FILED
December 09, 2022
CLERK, U.S. DISTRICT COURT
WESTERN DISTRICT OF TEXAS
BY: lad
DEPUTY

UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
WACO DIVISION

WEBROOT, INC., OPENTEXT INC.,	§	
	§	
Plaintiffs,	§	
v.	§	
AO KASPERSKY LAB	§	**W-22-CV-00243-ADA-DTG**
TREND MICRO, INC.	§	**W-22-CV-00239-ADA-DTG**
SOPHOS LTD.	§	**W-22-CV-00240-ADA-DTG**
CROWDSTRIKE HOLDINGS, INC.	§	**W-22-CV-00241-ADA-DTG**
FORCEPOINT, LLC	§	**W-22-CV-00342-ADA-DTG**

SECOND AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Plaintiffs Webroot, Inc., ("Webroot") and Open Text, Inc., ("OpenText") (collectively "Plaintiffs") allege against Forcepoint LLC ("Forcepoint" or "Defendant") the following:

1. This case involves patented technologies that helped to revolutionize and have become widely adopted in the fields of malware detection, network security, and endpoint protection. Endpoint protection involves securing endpoints or entry points of end-user devices (*e.g.*, desktops, laptops, mobile devices, etc.) on a network or in a cloud from cybersecurity threats, like malware.

2. Before Plaintiffs' patented technologies, security platforms typically relied on signatures (*i.e.*, unique identifiers) of computer objects (*e.g.*, computer programs) that were analyzed and identified as "bad" by teams of threat researchers. This approach required antivirus companies to employ hundreds to thousands of threat analysts to review individual programs and determine if they posed a threat.

3. The "bad" programs identified by researchers were compiled into a library and uploaded to an antivirus software program installed on each endpoint device. To detect threats, a

IPR2023-00784
Open Text Inc. EX-2012

resource intensive "virus scan" of each endpoint device was conducted. These virus scans could take hours to complete and substantially impact productivity and performance.

4. Despite substantial investments in resources and time, the conventional systems still were unable to identify and prevent emerging ("zero-day") threats from new or unknown malware. New threats persisted and were free to wreak havoc until a team of threat analysts could identify each one and upload these newly identified threats to an update of the "bad" program library. The updated "bad" program library, including signatures to identify new threats as well as old, then had to be disseminated to all of the endpoint computers, which required time and resource consuming downloads of the entire signature library to every computer each time an update was provided.

5. By the early-to-mid 2000s, new threats escalated as network connectivity became widespread, and programs that mutate slightly with each new copy (polymorphic programs) appeared. These events, and others, rendered the traditional signature-based virus scan systems entirely ineffective for these modern environments.

6. Plaintiffs' patented technology helped transform the way malware detection and network security is conducted, reducing, and often even eliminating the shortcomings that plagued signature-based endpoint security products that relied on human analysts.

7. Instead of relying on human analysts, Plaintiffs' patented technology enabled the automatic and real-time analysis, identification, and neutralization of previously unknown threats, including new and emerging malware, as well as advanced polymorphic programs.

8. Plaintiffs' patented technology uses information about the computer objects being executed—including, for example, information about the object's behavior and information collected from across a network—along with machine learning technology and novel system

architectures—to provide security systems that are effective in identifying and blocking new security threats in real-time in real-world, commercial systems.

9. Plaintiffs' patented technology further includes new architectures that efficiently and effectively distribute workloads across the network, new techniques for enabling safe and secure browsing even of potentially malicious websites, advanced identification, and classification of potentially harmful Internet resources, among other technologies.

10. Plaintiffs' patented technology makes the new security platforms and techniques better at detecting malware by for example reducing false positives/negatives and enabling the identification and mitigation of new and emerging threats in near real-time. These improvements are accomplished while at the same time reducing the resource demands on the endpoint computers and network appliances.

11. Plaintiff Webroot has implemented this patented technology in its security products like Webroot SecureAnywhere AntiVirus, which identifies and neutralizes unknown and undesirable computer objects in the wild in real-time.

12. Over the years, Plaintiff Webroot has also received numerous accolades and awards for its products and services. For example, Webroot has received 22 PC Magazine Editor's Choice Awards, including "Best AntiVirus and Security Suite 2021." That same year, Webroot also received the Expert Insights Best-of-Endpoint Security award.

13. Plaintiffs currently own more than 70 patents describing and claiming these various security innovations, including U.S. Patent No. 8,726,389 (the "'389 Patent"), U.S. Patent No. 10,599,844 (the "'844 Patent"), U.S. Patent No. 8,438,386 (the "'386 Patent"), U.S. Patent No. 9,413,721 (the "'721 Patent"), U.S. Patent No. 10,025,928 (the "'928 Patent"), U.S. Pat. No. 11,409,869 (the "'869 Patent"), U.S. Patent No. 8,201,243 (the "'243 Patent"), and U.S. Patent

No. 8,719,932 (Exhibits 1 – 8, respectively).

14. Plaintiffs' patented technology represents such a vast improvement on the traditional malware detection and network security systems that it has become a widely adopted and accepted approach to providing endpoint security in real-time.

15. Defendant Forcepoint is a direct competitor of Plaintiffs and provides security software that, without authorization, implements Plaintiffs' patented technologies. Forcepoint's infringing security products and services include, but are not limited to, Forcepoint's Next-Generation Firewall, Web Appliance or Secure Web Gateway, Web Security, Forcepoint One, Advanced Classification Engine, Threat Seeker Intelligence, Advanced Malware Detection, and Remote Browser Isolation, as well as products that include any of the above products or modules, or that include the same functionality described herein (collectively, "Accused Products").

16. Plaintiffs bring this action to seek damages for and to ultimately stop Defendant's continued infringement of Plaintiffs' patents, including in particular the '389, '844, '386, '721, '928,'869, '243, and '932 Patents (collectively the "Asserted Patents"). As a result of Forcepoint's unlawful competition in this District and elsewhere in the United States, Plaintiffs have lost sales and profits and suffered irreparable harm, including lost market share and goodwill.

NATURE OF THE CASE

17. Plaintiffs brings claims under the patent laws of the United States, 35 U.S.C. § 1, *et seq.*, for the infringement of the Asserted Patents. Defendant has infringed and continue to infringe each of the Asserted Patents under at least 35 U.S.C. §§271(a), 271(b) and 271(c).

THE PARTIES

18. Plaintiff Webroot, Inc., is the owner by assignment of each of the Asserted Patents.

19. Webroot has launched multiple cybersecurity products incorporating its patented

technology, including for example Webroot SecureAnywhere and Evasion Shield.

20. Webroot is a registered business in Texas with multiple customers in this District. Webroot also partners with several entities in this District to resell, distribute, install, and consult on Webroot's products.

21. Plaintiff Open Text Inc. holds an exclusive license to the Asserted Patents. OpenText is registered to do business in the State of Texas.

22. OpenText is a Delaware corporation and maintains three business offices in the state of Texas, two of which are located in this District, including one in Austin and another in San Antonio. Over 60 employees work in OpenText's Austin office, including employees in engineering, customer support, legal and compliance teams, IT, and corporate development. The Austin office also hosts an OpenText data center. OpenText is in the computer systems design and services industry. OpenText sells and services software in the United States.

23. Defendant Forcepoint LLC is a Delaware limited liability company with a principal place of business in this District at 10900 Stonelake Blvd #350, Austin, TX 78759. Forcepoint is registered to do business in the State of Texas

JURISDICTION & VENUE

24. This action arises under the Patent Laws of the United States, 35 U.S.C. § 1, *et seq*. The Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

25. This Court has personal jurisdiction over Forcepoint because Forcepoint's principal place of business is located in the State of Texas and because Forcepoint regularly conducts business in the State of Texas and in this District, including operating systems, using software, providing services, and/or engaging in activities in Texas and in this District that infringe one or more claims of the Asserted Patents.

26. Defendant Forcepoint has, either directly and through its extensive network of partnerships including those with local IT service providers, purposely and voluntarily placed its infringing products and/or provided services into the stream of commerce with the intention and expectation that they will be purchased and used by customers in this District, as detailed below. (*See, e.g.*, Forcepoint, *Find a Partner*, https://www.forcepoint.com/partners/find-a-partner.)

27. Venue is proper in this District pursuant to 28 U.S.C. §§ 1391(b) and (c) and 28 U.S.C. § 1400(b) because Forcepoint resides in this District, has a regular and established place of business in this District, and has committed acts of infringement within this District.

28. On information and belief, Forcepoint has employees in this District that have relevant knowledge regarding the Accused Products, including for example how they are marketed and sold to customers, what additional services are provided to customers based on the Accused Products, and how the products operate.

29. Forcepoint's operations in this District include client outreach and sales for each of the Accused Products. Forcepoint also provides technical support to partners and customers located in this District, including from its office in this District.

30. On information and belief, Forcepoint uses and/or tests the Accused Products in this District, including at its office in this District.

31. Forcepoint further sells, offers for sale, advertises, makes, installs, maintains, and/or otherwise provides security software, appliances, and services, including the Accused Products, the use of which infringes the Asserted Patents in this District. Forcepoint performs these infringing acts directly in this District.

32. Forcepoint also performs these infringing acts through other entities such as resellers, managed service provides, and cybersecurity experts located in this District, including

for example, through its "partners." (Forcepoint, *Find a Partner*, https://www.forcepoint.com/partners/find-a-partner.).

33. As further detailed below, Defendant engages in activities within this judicial district that infringe (directly or indirectly) the Asserted Patents, either literally or under the doctrine equivalents, including the provision of, use, operation, sales, offering for sale, installation, maintenance, and advertising of the Accused Products. Forcepoint also infringes (directly or indirectly) the Asserted Patents by using, offering for sale, selling, installing, maintaining, operating, providing instructions, and/or advertising the Accused Products within this District, either literally or under the doctrine of equivalents.

34. End-users and partner customers infringe the Asserted Patents at least by using and operating, in whole and in part, the Accused Products with this District.

35. Defendant Forcepoint encourages and induces third parties including partners and customers to use the Accused Products in an infringing way at least by making Forcepoint's security software, appliances, and services available for download or purchase, widely advertising those products and services, providing instructions for using, installing, and maintaining those products, providing technical support to users, and/or engaging in activities that aid and abet infringement of the Asserted Patents by end-users. (*See, e.g.*, Forcepoint, *Welcome to Forcepoint Hub*, https://support.forcepoint.com/s/login/?ec=302&startURL=%2Fs%2F).

36. Defendant Forcepoint also contributes to the infringement of its customers and end users of the Accused Products by offering to sell or selling with the United States or importing into the United States the Accused Products, which are for use in practicing, and under normal operation practice, methods claimed in the Asserted Patents, constituting a material part of the inventions claimed, and not a staple article or commodity of commerce suitable for substantial

non-infringing uses. Indeed, the Accused Products and the example functionality discussed below have no substantial non-infringing uses but instead are specifically designed to practice the Asserted Patents.

37. Defendant's infringement adversely impacts Plaintiffs and their employees who live in this district, as well as Plaintiffs' partners and customers who live and work in and around this judicial district. On information and belief, Defendant actively targets and offers Accused Products to customers served by Plaintiffs, including in particular customers/end-users in this District.

PLAINTIFFS' PATENTED INNOVATIONS

38. Plaintiff Webroot, and its predecessors were all pioneers and leading innovators in developing and providing modern end point security protection, including "community-based" signatureless threat detection process using AI-driven behavior analysis across the entire network to provide "zero-day" protection against unknown threats.

39. The Asserted Patents discussed below capture technology, features, and processes that reflect these and other innovations, and improve on traditional anti-Malware and network security systems.

U.S. Patent No. 8,726,389

40. The '389 Patent generally discloses and claims systems and processes related to real-time and advanced classification techniques. Plaintiff Webroot owns by assignment the entire right, title, and interest in and to the '389 Patent. Webroot has granted Plaintiff OpenText an exclusive license to the '389 Patent.

41. The '389 Patent is entitled "Methods and Apparatus for Dealing with Malware," was filed on July 8, 2012, and was duly and legally issued by the United States Patent Office on

May 13, 2014. The ’389 Patent claims priority to Foreign Application No. 0513375.6 (GB), filed on June 30, 2005. A true and correct copy of the ’389 Patent is attached as Exhibit 1.

42. Malware detection systems in use at the time the ’389 Patent was filed identified malware by maintaining a database of signatures identifying known bad objects (*i.e.*, malware). The signature for an object was conventionally made by creating a hash or checksum corresponding to the object file, which uniquely identifies that object. The signature of each object was then compared to the database to look up whether it matches known malware.

43. If the signature of the object is not found in the database, it is assumed safe or alternatively, the whole file is sent for further investigation by a human analyst. The process of further investigation was typically carried out manually or “semimanually” by subjecting the file to detailed analysis, for example by emulation or interpretation, which can take days given the human involvement that is typically required. (*See*, *e.g.*, Exhibit 1, ’389 Patent, 2:9-17.)

44. This approach had significant drawbacks, including that it required considerable effort by the providers of such systems to identify and analyze new malware and generate signatures of objects that are found to be bad after human analysis. Large vendors of anti-malware packages typically employed *thousands* of human analysts to identify and analyze objects and keep the database of signatures of bad objects reasonably up to date.

45. However, as the volume of network traffic increases, the task of keeping up with identifying suspect objects and investigating whether or not they are bad becomes practically impossible. It can take days to subject a suspicious file to detailed analysis given the human involvement, and a considerable period of time elapses before a new file is classified as safe or as malware. Thus, the human analysis introduces a time delay where users are exposed and

unprotected from the risks posed by previously unidentified malware. (*See* Exhibit 1, '389 Patent, 2:9-23, 2:63-67.)

46. By contrast, the methods and systems disclosed and claimed in the '389 Patent perform automatic, sophisticated review (*e.g.*, "pattern analysis") of the actual attributes of a software object or process and the behavior engaged in by, or associated with, that object or process on computers connected to a network.

47. This review enables a determination of "the nature of the object," (*e.g.*, whether it is malicious or not based on review of the object, its behaviors or the activities associated with the object), without requiring a detailed manual analysis of the code of the object itself or relying exclusively on whether it has a signature that matches an extensive database of known malicious "signatures." (*See* Exhibit 1, '389 Patent, 3:14-24.) This provides a significant improvement to the operation of the computer network because monitoring behavior or other information about the object or process, rather than code or signature matching, allows the system to rapidly determine the nature of the object (*e.g.*, malware), without requiring a detailed manual analysis of the code of the object itself as in conventional anti-virus software. (*See* Exhibit 1, '389 Patent, 3:14-24.)

48. The approaches in the '389 Patent are generally focused on receiving *information about the behavior* of objects or processes on remote computers at a base computer. This information is analyzed automatically by, for example, mapping the behavior and attributes of objects known across the community in order to identify suspicious behavior and to identify malware at an early stage. (*See* Exhibit 1, '389 Patent, 11:5-26.) This approach allows, among other advantages, the number of human analysts needed to be massively reduced. It also improves the computer network by reducing the latency involved with identifying new threats and responding to objects exhibiting new, potentially malevolent behavior.

49. The techniques disclosed and claimed in the ’389 Patent is necessarily rooted in computer technology—in other words, the identification of malicious computer code in computer networks is fundamentally and inextricably a problem experienced with computer technology and networks—and addresses this fundamental computer technology problem with a computer technology solution. Furthermore, the ’389 patent improves the technical functioning of the computer network using techniques—such as analyzing behavioral information about or associated with computer objects and processes—to improve network security by identifying malware more quickly and with less resources. These technical improvements address identified weaknesses in conventional systems and processes. (*See, e.g.*, Exhibit 1, ’389 Patent, 3:14-24.)

50. In particular, the ’389 Patent describes and claims deploying an unconventional “event” based model that classifies a particular object as malicious or safe by analyzing real-time data sent by remote computers on the events, or actions, that a particular software “object,” and other objects deemed similar to it, initiate or perform on those computers. (*See* Exhibit 1, ’389 Patent, 3:14-55.) This information is collected from across the network, correlated and used for subsequent comparisons to new or unknown computer objects to identify relationships between the correlated data and the new or unknown computer objects. The objects may be classified as malware based on this comparison.

51. Through continuous aggregate analysis of events involving computer objects as they occur across networks, the methods and systems described and claimed in the ’389 Patent maintain up-to-date information about computer objects (including malicious objects) seen across the network, identify relationships between those previously identified objects and any new or unknown objects, and make malware determinations based on those relationships. “For example, a new object that purports to be a version of notepad.exe can have its behavior compared with the

behav[io]r of one or more other objects that are also known as notepad.exe … In this way, new patterns of behav[io]r can be identified for the new object." (*Id.* at 10:58-65.)

52. The methods and systems described and claimed in the '389 Patent can rapidly determine "the nature of the object," (*e.g.*, whether it is malicious or not) based on information such as the behavior of the object or effects the object has, without requiring "detailed analysis of the object itself as such" (manually reviewing the object's code) or reliance on matching an extensive database of known malicious "signatures." (*Id.* at 3:14-24.)

53. The '389 Patent describes and claims systems and methods that necessarily address issues unique to computer networks and computer network operation; namely the identification of "bad" software (*e.g.*, malware, viruses, etc.). This patent provides unique network security enhancement that solves the technical problem of rapidly identifying newly arising and emerging malware by reviewing information about the object and processes (*e.g.*, the behaviors and events associated with software objects and processes running on computers within the network).

54. The systems and methods described and claimed in the '389 Patent improve the operation of computer networks by identifying malicious objects in real-time and taking action to remove or eliminate the threat posed by the malware object or process once it has been identified. The described and claimed techniques provide a technological solution to a technological problem—the inability of conventional code or signature matching solutions to identify new or unknown malware objects or processes at or near the runtime of the objects or processes themselves without the extensive delay and resource use associated with traditional systems.

U.S. Patent Nos. 10,599,844 and 11,409,869

55. The '844 Patent is entitled "Automatic Threat Detection of Executable Files Based on Static Data Analysis," was filed May 12, 2015 and was duly and legally issued by the USPTO

on March 24, 2020. A true and correct copy of the '844 Patent is attached as Exhibit 2. Plaintiff Webroot owns by assignment the entire right, title, and interest in and to the '844 Patent. Webroot has granted Plaintiff OpenText an exclusive license to the '844 Patent.

56. The '869 Patent, entitled "Automatic Threat Detection of Executable Files Based on Static Data Analysis," is a continuation of the '844 Patent and claims priority to the application of the '844 Patent. The '869 Patent was filed on February 14, 2020 and was duly and legally issued by the USPTO on August 9, 2022. A true and correct copy of the '869 Patent is attached as Exhibit 6. Plaintiff Webroot owns by assignment the entire right, title, and interest in and to the '869 Patent. Webroot has granted Plaintiff OpenText an exclusive license to the '869 Patent.

57. The '844 and '869 Patents address and improve upon conventional approaches to malware detection in computer networks and computer network operation. Every day, an uncountable number of new executable files are created and distributed across computer networks. Some of those files are unknown, and malicious. It is, thus, vital to accurately and immediately diagnose those files for any potential threat, while also efficiently using resources (*e.g.*, processing power). (*See* Exhibit 2, '844 Patent, 1:7-13.)

58. Conventional approaches for diagnosing potential malware threats were costly and time consuming, making it difficult to realistically address zero-day threats for all of the files entering a system. These "[a]pproaches to detecting threats typically focus[ed] on finding malicious code blocks within a file and analyzing the behavior of the file." (*See* Exhibit 2, '844 Patent, 2:15-17.) Encrypted files would be decrypted then disassembled to extract the code for analysis, typically by traditional anti-virus software based on signature matching. (*Id.* at 2:15-20) If the code was malware, investigating its behavior involved running the code on the system, which put the system at risk. (*Id.* at 2:20-23.)

59. Another approach for protecting against potential threats from unknown executable files involved wavelet decomposition to determine software entropy. (*See* WBRT007115 at 7491, '844 Patent Prosecution History, April 24, 2019 Applicant Remarks, at 8.) Wavelet decomposition is a process where an original image is decomposed into a sequence of new images, usually called wavelet planes. (*Id.*) In this method, each data file in a set of data files is split into random, non-overlapping file chunks of a fixed length. (*Id.*) Those file chunks are then represented as an entropy time-series, which measures the time it takes for each chunk to decompose. (*Id.*) Said differently, this approach measured how much time it took a data file to decompose. (*Id.*) Once the file decomposition rate, or entropy time-series, had been calculated, that rate would be compared to decomposition rates of "known bad" files to identify files that contain malware. (*Id.* at WBRT007492.) This process required significant computing resources—typically taking hours to complete—and was not sufficiently accurate in identifying malware.

60. The '844 and '869 Patents significantly improve upon and address shortcomings associated with these prior approaches. The Patents describe and claim methods and systems that detect threats in executable files without the need to decrypt or unpack those executable files by extracting "static data points inside of the executable file without decrypting or executing the file," generating "feature vectors" from those static data points, selectively turning on or off features of the feature vector, and then evaluating the feature vector to determine if the file is malicious. (*See, e.g.*, Exhibit 2, '844 Patent, 1:20-21; cl. 1.) The described systems and methods enable accurate and efficient identification of malware without the need to distinguish between encrypted files and non-encrypted files (*id*. at 6:58-59), thereby significantly increasing efficiency and reducing processing resources required to analyze each potentially malicious computer object. By using this unconventional approach to determine whether a file executable on a computer poses a threat, the

’844 and ’869 Patents improve on the operation of the computer network associated with the computer by enhancing security, including by increasing detection of new threats, reducing the error rates in identifying suspicious files, and improving efficiency in detecting malicious files. (*See* Exhibit 2, ’844 Patent, 2:46-56.)

61. The ’844 and ’869 Patents describe and claim techniques that employ a learning classifier (*e.g.*, a machine-learning classifier) to determine whether an executable file is malicious, for example by using the classifier to classify data into subgroups and identify and analyze specific data points to which those subgroups correspond. (*See* Exhibit 2, ’844 Patent, 4:33-41, 7:40-8:1.) The described and claimed techniques also selectively turn on or off features for evaluation by the learning classifier. (*See id*. at 7:57-66.) Doing so accelerates analysis and reduces false positives by testing those features of a file likely to be relevant to a determination of its maliciousness. For example, the learning classifier “may detect that the file does not contain ‘legal information’,” such as “timestamp data, licensing information, copyright information, etc.” (*See id.* at 7:66-8:5.) In this example, given the lack of legal protection information in the file, the learning classifier would “adaptively check” the file for additional features that might be indicative of a threat,” while “turn[ing] off,” and thus not use processing time unnecessarily checking features related to an evaluation of “legal information.” (*Id*. at 8:5-10.)

62. Second, the ’844 and ‘869 Patents describe and claim techniques that use character strings extracted from within the executable file to generate a feature vector and then evaluate that feature vector using support vector processing to classify executable files. (*See* Exhibit 2, ’844 Patent, 9:2-11.) The classifier provides, for example, the ability to leverage the indicia of “benign” files, which use “meaningful words” in certain data fields, versus “malicious” files, which leave such fields empty or full of “random characters,” to build meaningful feature vectors that are

analyzed to make faster and more identifications of malware (*See, e.g.*, Exhibit 2, '844 Patent, 9:2-18.)

63. The '844 and '869 Patents are thus directed to specific solutions to problems necessarily rooted in computer technology, namely, the determination whether a file executable on a computer poses a threat. The '844 and '869 Patents improved upon the accuracy and efficiency of malware detection. (*See* Exhibit 2, '844 Patent, 2:15-45.)

64. By using some or all of the unconventional techniques described above to determine whether a file executable on a computer poses a threat, the '844 and '869 Patents address a problem necessarily involving computers and improve upon the operation of computer networks. In particular, the '844 and '869 Patents achieve a number of technical advantages over conventional approaches to malware detection including, for example:

- enhanced security protection including automatic detection of threats, reduction or minimization of error rates in identification and marking of suspicious behavior or files (*e.g.*, cut down on the number of false positives),
- ability to adapt over time to continuously and quickly detect new threats or potentially unwanted files/applications,
- improved efficiency in detection of malicious files, and
- improved usability and interaction for users by eliminating the need to continuously check for security threats.

(*See* Exhibit 2, '844 Patent, 2:15-57.)

U.S. Patent No. 8,438,386

65. U.S. Patent No. 8,438,386, entitled "System and Method for Developing a Risk Profile for an Internet Service," was filed on February 21, 2010, and claims priority to two provisional applications, application numbers 61/171,264 and 61/241,389, filed on April 21, 2009 and September 10, 2009 respectively. The United States Patent Office duly and legally issued the '386 Patent on May 7, 2013. A true and correct copy of the '386 Patent is attached as Exhibit 3. Plaintiff Open Text owns by assignment the entire right, title, and interest in and to the '386 patent. Open Text has granted Plaintiff Webroot an exclusive license to the '386 patent.

66. The '386 Patent generally is directed to a method for controlling access to an Internet resource by determining that resource's reputation and risk. The '386 Patent discloses and claims inventive techniques that significantly improve on prior art tools for preventing access to Internet resources. Preventing access to computer network resources is, by its very nature, necessarily rooted in computer technology and overcomes a problem specific to computer networks (namely, problems arising with accessing malicious Internet resources).

67. Before the filing of the '386 Patent, conventional methods for preventing users from accessing malicious code on webpages involved "'Content Filtering' or "Security solutions, such as antivirus products." (Exhibit 3, '386 Patent, 1:30-46.) Content filtering is where "Web sites are organized into categories and requests for Web content are matched against per-category policies and either allowed or blocked." (*Id.* at 1:30-40.) Security solutions like anti-virus products, by contrast, "examine file or Web page content to discover known patterns or signatures that represent security threats to users, computers, or corporate networks. These focus not on the subject matter of a site but look for viruses and other 'malware that are currently infecting the site.'" (*Id.* at 1:41-46.) These approaches, however, are reactive rather than predictive; for example, they focus on

"accessing known infected sites" and "identif[ying] and distribut[ing]" signatures of known malware. (*Id.* at 1:46-50.)

68. Unlike these prior art systems (*e.g.*, signature matching systems) which cannot predictively prevent access, the '386 Patent improved on prior-art security solutions by assigning risk profiles to internet resources that have not been previously characterized before granting or allowing access. (*Id.* at 1:23-24.) The patent explains that a "predictive security assessment for an Internet resource is provided based on known facts about the Internet resource, which is more secure than relying only on knowledge of previously experienced security attacks." (*Id.* at 9:42-45.) Techniques described in the '386 Patent system include the use of samples of internet resources that have varying degrees of risk and, using a reputational model, predicts an internet resource's relative degree of risk. (*Id.* at 4:34-36.)

69. For example, the methods and systems described and claimed in the '386 Patent system may predict the relative degree of risk by generating a reputation index based on factors that include location, behavior and legitimacy. (*Id.* at 6:10-25.) When an end-user of a Local Area Network ("LAN") transmits a request for an internet resource, the system analyzes the "reputation index" for the requested resource and determines whether the reputation index is at or above a threshold value established for the LAN before granting or denying access to the resource. (*Id.* at 1:26.) In doing so, behavioral and other information about the internet resource may be used to proactively block or allow access, as opposed to systems that employ a reactive approach that relies on, for example, signature matching.

70. By assessing an internet resource's risk in a predictive way—*e.g.*, before infections are isolated and signatures are identified and distributed—a technological solution that addresses weaknesses in prior systems is provided. As the '386 Patent explains, "[a] predictive security

assessment for an Internet resource is provided . . . , which is more secure than relying only on knowledge of previously experienced security attacks." (*Id.* at 9:42-45.) In other words, systems and methods described in the '386 Patent can protect against threats from internet resources that have not been previously characterized as malicious or included in pre-existing signature files.

71. The techniques described and claimed in the '386 Patent solve problems in the field of computers and network security. As discussed above, traditional virus detection methods focused on reactive approaches—blocking web resources that included signatures of known viruses or malware, and internet resource management tools were limited to categorically blocking certain types of web resources. These limitations in prior systems were overcome by, for example, predicting the potential risk that requested internet resources presented to a user's computing device and the network on which the user was operating. Whereas categorically blocking websites of a certain type or websites that were infected with known signatures of viruses and malware left systems vulnerable to unknowns, assessing the "reputation" of an internet resource based on, for example, location, legitimacy and/or behavioral factors, the '386 Patent provided methods and systems that effectively and efficiently determined whether to block or allow a user's access to a web resource in a predictive manner. These approaches can address zero-day threats in an efficient and effective way. Moreover, the specific factors and combinations of factors described and claimed in the '386 Patent (*e.g.*, "country of an internet protocol address block," "top-level domain," and "script block count") are necessarily rooted in computer technology to overcome a problem specifically arising in the realm of computer networks. (*See, e.g.*, Exhibit 3, '386 Patent, cl. 1.) The '386 Patent system improves on prior, manual, and reactive approaches with an unconventional, automated, and predictive approach that improves computer functionality.

U.S. Patent No. 9,413,721

72. The ’721 Patent is entitled “Methods and Apparatus for Dealing with Malware,” was filed on February 13, 2012, and duly and legally issued by the United States Patent Office on February 5, 2013. A true and correct copy of the ’721 Patent is attached as Exhibit 4. Plaintiff Webroot owns by assignment the entire right, title, and interest in and to the ’721 Patent. Webroot has granted Plaintiff OpenText an exclusive license to the ’721 Patent.

73. The systems and methods described and claimed in the ’721 Patent are directed to improved techniques for detecting and classifying malware, a technological problem fundamentally and inextricably associated with computer technology and computer networks. The ’721 Patent explains that prior anti-malware products used signature matching to detect malware, either locally or at a central server. (Exhibit 4, ’721 Patent, 1:37-2:14.) The local anti-malware product suffered from delays in identifying new malware threats and obtaining signatures for them so they could be blocked. (*Id.* at 1:37-55.) Central servers stored signatures in the cloud. (*Id.* at 56-57.) But only signature or very basic information was sent to the central server for matching. (*Id.* at 1:67-2:2.) If the object was unknown, a copy had to be sent to the central server for investigation by a human, a time consuming and laborious task. (*Id.* at 2:5-7.) In a network environment, it was unrealistic for a human to investigate each new object due to the high volume of incursions that take place over a network. (*Id.* at 2:7-10.) Thus, under these approaches, “malevolent objects may escape investigation and detection for considerable periods of time.” (*Id.* at 2:10-13.)

74. To address these shortcomings, the ’721 Patent describes and claims unconventional, novel distributed system architectures, such as remote computers that may be allocated to “threat” servers, with “central” servers sitting behind them. (Exhibit 4, ’721 Patent, 9:16-57.) These enhanced computer architectures provide a technical solution to the technical

problem of detecting and classifying malware in a computer network environment, thus improving network security while identifying and classifying malware threats in real-time without delays engendered by use of human analysts. (*See, e.g.*, Exhibit 4, '721 Patent, 1:60-2:7.)

75. In particular, the '721 Patent describes and claims embodiments that may include three-tiered architectures of remote computers, threat servers, and a central server that provides a technical enhancement to the computer network itself (improving upon the two-tiered architectures of traditional systems having only remote computers and a central server) by enabling the central server to keep, for example, a master list "of all data objects, their metadata and behaviour seen on all of the remote computers" and propagate it back to the threat servers. (Exhibit 4, '721 Patent 12:28-54.) This novel network architecture improves the operation of the computer network over traditional networks because, for example and as described in the '721 Patent, "[t]his scheme has been found to reduce workload and traffic in the network by a factor of about 50 compared with a conventional scheme." (*Id.* at 12:55-57.)

76. Further, "by being able to query and analyze the collective view of an object, i.e., its metadata and behaviours, across all agents [] that have seen it, a more informed view can be derived, whether by human or computer, of the object. In addition, it is possible to cross-group objects based on any of their criteria, i.e. metadata and behaviour." (*Id.* at 18:17-22.) Thus, embodiments enable better malware identification than conventional systems (*e.g.*, using human analysis) in addition to providing an efficiency benefit. The patent explicitly notes that "the work in processing the raw data [] is too large of a task to be practical for a human operator to complete." (*Id.* at 18:50-52.)

77. The systems and methods described and claimed in the '721 Patent provide further technical improvements. For example, the information collected at the central server may include

additional information about the object being classified as well as a count associated with the number of times that the first computer object has been seen to the central server. (*Id.* at cl. 1.) As explained above, using information about the object (such as behavior information) being classified, embodiments described and claimed in the '721 Patent provide an approach that is more effective than traditional code or signature matching techniques for classifying objects as malicious. (*Id.* at 1:54-2:14.)

78. Prior methods of classifying malware had technical drawbacks when used on a distributed network. For example, a distributed network that required each server to maintain rules for determining what is malware required each server to deal with huge amounts of largely common data. (Exhibit 4, '721 Patent, 12:20-24.) It was also generally impractical to store the required data on each server because, for example, there were problems determining whether or not the data—which is both massive and constantly changing—is common and up-to-date in real-time. (*Id.* at 12:24-27.) The three-tiered architectures described and claimed in embodiments of the '721 Patent provide a technical solution for distributed computer networks by, *inter alia*, reducing the workload across the network. (*Id.* at 12:28-59.)

79. Accordingly, the '721 Patent discloses and claims, among other things, an unconventional technological solution to the inherently computer-network centric technical issue of identifying malware in computer systems. The solutions implemented by the '721 Patent provides a specific and substantial improvement over prior malware classification systems, for example by introducing novel computer network architecture elements combined in an unconventional manner. These approaches improve the function and working of malware detection services by, for example, utilizing multiple threat servers and central servers and performing the analysis and communication carried out by each type of server in an unconventional and efficient

manner. These elements and their combination represent a marked improvement in the functioning of computer systems utilized to identify and detect malware in computers networks.

U.S. Patent No. 10,025,928

80. U.S. Patent No. 10,025,928 ("the '928 Patent") entitled "Proactive Browser Content Analysis" was filed on October 3, 2012, claims priority to provisional application 61/542,693 filed October 3, 2011, and was duly and legally issued by the United States Patent Office on July 17, 2018. A true and correct copy of the '928 Patent is attached as Exhibit 5. Plaintiff Webroot owns by assignment the entire right, title, and interest in and to the '928 Patent. Webroot has granted Plaintiff OpenText an exclusive license to the '928 Patent.

81. The '928 Patent sets forth techniques for, *inter alia*, "controlling pestware or malware or other undesirable or unwanted applications and/or instructions." ('928 Patent, 1:19-21.) The '928 Patent describes and claims methods and systems that guard against malware by, for example, intercepting browser engine requests and modifying data received from the web server for rendering and display by the browser engine. The '928 Patent warns that "source[s] of malware … end up or [are] strategically placed at [a] webserver … [so that] the malware source 106 may generate a malware object in a variety of forms including in a scripting language." ('928 Patent, 3:21-26.)

82. In one embodiment, the '928 Patent describes a protection agent 116, which "ha[s] full control over what the browser engine 118 'sees.'" ('928 Patent, 4:6-10, 4:21-22.) Protection agent 116 intercepts the browser's requests for web content. ('928 Patent, 4:9.) "[I]f the request does not appear to be a request for malicious content," then protection agent 116 forwards the request to the web server and, in return, "receives [the requested] content … from the website." ('928 Patent, 6:19-22.) After receiving the web content, protection agent 116 analyzes the webpage

for threats. For example, "all the links in the webpage, all the pictures in the webpage, and any scripts in the webpage … at a low level of granularity." ('928 Patent, 6:51-54.) If a threat is found in the webpage action may be taken. For example, "[I]f a malicious script is found, it may be commented out [changed, enhanced, and removed] before the content is handed to the browser engine." ('928 Patent, 6:54-58.) "[A]fter performing all of its processing, removing, and/or adding any code as needed, the protection agent feeds the HTML content back to the browser engine" as if the browser engine were "'speaking to an actual web server." ('928 Patent, 4:14-21.)

83. The '928 Patent provides technical benefits that improve the functioning of the computer system, including enabling safer browsing without affecting the browsing experience. For example, the protection agent can analyze webpages for any signs of malware and delete any instances found. The web browser seamlessly may communicate with a protection agent in such a way that the browser receives communications from the protection agent as if the protection agent were the web server. In addition, methods and systems claimed in the '928 Patent provided technical benefits that improve the performance of computer systems. For example, while prior methods simply made high-level modifications to the content after it had been displayed through a Browser Helper Object," embodiments of the '928 Patent may modify the content at the protocol level, requiring "virtually no performance overhead." In fact, in addition to there being virtually no performance overhead, "in many cases, there is actually a performance improvement." ('928 Patent, 4:49-55.)

84. The systems and methods described and claimed in the '928 Patent are fundamentally rooted in computer technology—in fact, they are processes only performed within a networked computer environment. The techniques described in the '928 Patent address problems necessarily rooted in computer technology. For example, the '928 Patent guards against computer

malware distributed through webpages accessed over a computer network, a problem affecting internet traffic.

85. The ’928 Patent further describes and claims unconventional techniques for guarding against malware. For example, the ’928 Patent describes embodiments that annotate instances of possible malware—*e.g.*, when the protection agent assembles and modifies the webpage. While prior methods, for example, simply “make high-level modifications to the content after it has been displayed through a Browser Helper Object,” the described and claimed methods may do so at the protocol level, which requires “virtually no performance overhead.” (’928 Patent, 4:49-55.) Moreover, by modifying the webpage at the protocol level, embodiments of the ’928 Patent offer a novel means to prevent malware from ever reaching a user’s browser, without the need to account for differences between browsers. For example, the ’928 Patent explains:

> [M]odifying content at the [protocol] level … is very different than the prior approaches … [after] a page has already been parsed and rendered by a browser engine. This prior approach is problematic because it allows the browser engine to potentially execute malicious scripts or perform malicious actions while … parsing and rendering the code. And in addition, because the annotations are added after rending, the annotation process must account for the rendering differences (e.g., differences in how and where content is displayed) that different browsers (e.g., Firefox, Safari, Chrome, Internet Explorer, etc.) exhibit.”

(‘928 Patent, 7:11-20.)

Kernel-Level API Monitoring Patents
U.S. Patent Nos. 8,201,243 and 8,719,932

86. The ’243 and ’932 Patents are part of the same patent family. Plaintiff Webroot owns by assignment the entire right, title, and interest in and to the ’243 and ’932 Patents.

87. The ’243 Patent, entitled “Backwards Researching Activity Indicative of Pestware,” was filed on April 20, 2006, and was duly and legally issued by the USPTO on June 12, 2012. A true and correct copy of the ’243 Patent is attached as Exhibit 7.

88. The ’932 Patent, also entitled “Backwards Researching Activity Indicative of Pestware,” was filed on June 6, 2012, and was duly and legally issued by USPTO on May 6, 2014. The ’932 Patent claims priority to the ’243 Patent. A true and correct copy of the ’932 Patent is attached as Exhibit 8.

89. At the time the ’243 and ’932 Patents were filed, conventional pestware-detection-and-removal solutions struggled to stay ahead of evolving pestware because they relied on “definitions of known pestware to search for and remove files on a protected system.” (Exhibit 10, ’243 Patent, 1:63-65.) “Th[os]e definitions were often slow and cumbersome to create,” and “it was often difficult to locate the pestware in order to create the definitions,” in the first place. (*Id.*at 1:65-67.) Furthermore, relying on definitions made it difficult to distinguish wanted pestware from unwanted pestware. (*Id.* at 1:60-62.)

90. To overcome these and other shortcomings, the ’243 and ’932 Patents describe and claim novel “systems and methods for discovering the source of activity that is indicative of pestware.” (*Id*. at 2:17-19.) For example, the ’243 and ’932 Patents describe and claim the use of a kernel-mode driver to monitor the behavior of processes running on a computer, for example monitoring API calls, to detect pestware. (*Id.* at 4:24-29.) The inventors found that this approach provided technical advantages, such as the ability to “intercept” “when a process (*e.g.*, the pestware process) attempt[ed] to spawn … another pestware process or alter a registry entry, … before it [wa]s carried out by an operating system of the protected computer.” (*Id.* at 5:46-50.)

91. Moreover, the ’243 and ’932 Patents describe and claim techniques that enable the identification of suspected pestware as well as externally networked sources that, for example, may be the source of the pestware on the system. This approach provides benefits, such as identifying the source of malicious malware, which can be used to identify new malware (*e.g.*,

based on a known source of malware) as well as to block potential malware based on a known malicious source.

ACCUSED PRODUCTS

92. Forcepoint's products that practice one or more claims of the Asserted Patents include, but are not necessarily limited to, Forcepoint's Next-Generation Firewall, Web Appliance, Web Security or Secure Web Gateway, Advanced Classification Engine, Threat Seeker Intelligence, Advance Malware Detection, Forcepoint One, and Remote Browser Isolation, as well as products that include any of the above products or modules, or that include the same functionality described herein.

93. Forcepoint's Next-Generation Firewall ("NGFW") is an SD-WAN network device that provides Forcepoint's customers with "consistent security, performance, and operations across physical, virtual, and cloud systems." Forcepoint's NGFW provides application-layer exfiltration protection, "selectively and automatically whitelist[ing] or blacklist[ing] network traffic originating from applications on PCs, laptops, servers, file shares, and other endpoint devices based on highly granular endpoint contextual data." (*See* https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_ngfw_en_0_0.pdf.)

94. Forcepoint's Web Appliances "are preconfigured to eliminate vulnerabilities from unnecessary software, open ports, default logins, and more, easing your deployment and enhancing your security." (Forcepoint, *Appliance*, https://www.forcepoint.com/appliance.) They are designed "to simplify the deployment of redundant, load-balanced clusters, increasing the throughput of your implementation." (https://www.forcepoint.com/appliance.)

95. Forcepoint's Web Security (or Secure Web Gateway, "SWG"), which on

information and belief is offered as a standalone product or as part of Forcepoint One, is a product that "provides robust protection through content aware defenses and cloud app discovery and monitoring, reducing risks to sensitive data for both on premise and mobile users." (*See, e.g.*, Forcepoint, *Forcepoint Web Security*, https://forcepoint.drift.click/brochure_secure_web_gateway.) Forcepoint Web Security employs Forcepoint's Advanced Classification Engine ("ACE") as the decision engine that identifies the nature and format of the digital artifacts being analyzed. (*See* https://forcepoint.drift.click/brochure_secure_web_gateway.) Forcepoint Web Security additionally employs Forcepoint's ThreatSeeker Intelligence to provide real-time security updates that block advanced threats, malware, phishing attacks, lures, and scams.

96. Forcepoint's ACE is a suite of cyber threat prevention and detection analytics embedded in Forcepoint products. (*See* Forcepoint, *Forcepoint Advanced Classification Engine (ACE)*, https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_ace_en.pdf.) As shown in the picture below, ACE combines many different threat analysis approaches into one detection engine.



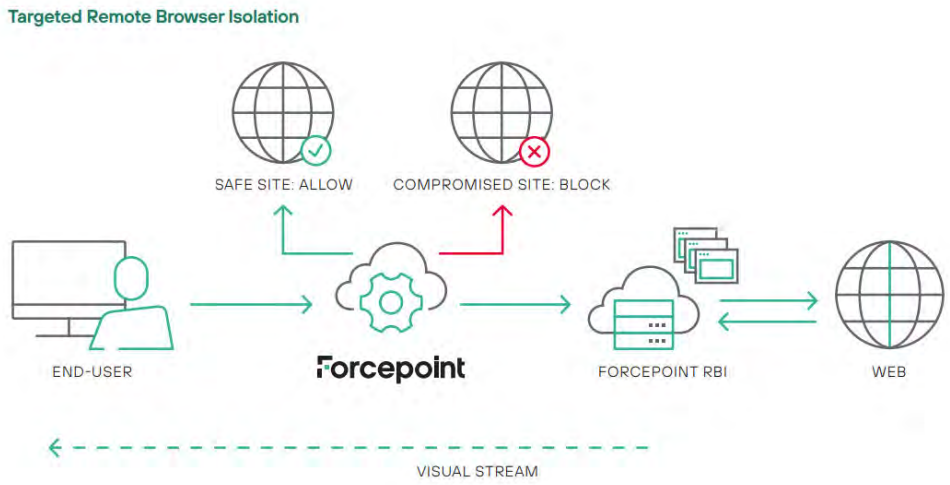
(*See* https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_ace_en.pdf.) Using these various methods, Forcepoint's ACE provides "detailed, real-

time categorization of content to enable a rich picture of the content surrounding cyber behavior." (*See* https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_ace_en.pdf.)

97. Forcepoint's ThreatSeeker Intelligence is responsible for collecting and aggregating threat data, using ACE to analyze the data. (*See* https://forcepoint.drift.click/brochure_secure_web_gateway.) ThreatSeeker Intelligence provides Forcepoint's products with "the core collective security intelligence." (*See* https://forcepoint.drift.click/brochure_secure_web_gateway.)

98. Forcepoint's Advance Malware Detection ("AMD") provides visualized malware reporting. Using Advanced Malware Detection, system administrators can view their threat exposure with detailed correlated incident information. (*See* Forcepoint, *Advanced Malware Detection*, https://www.forcepoint.com/product/advanced-malware-detection.)

99. Forcepoint's Remote Browser Isolation "provides users with safe access to uncategorized sites and known bad sites when necessary by using Forcepoint Web Security with Remote Browser Isolation." By isolating the browser from the end user's desktop, Forcepoint's Remote Browser Isolation prevents threats from reaching the user's browser.



(*See* Forcepoint, *Remote Browser Isolation*, https://www.forcepoint.com/sites/default/files/

resources/solution_brief/solution-brief-remote-browser-isolation-en_0_0_0_0_0_0_0_0_0_0_0_0_0.pdf.)

FIRST CAUSE OF ACTION
(INFRINGEMENT OF THE ’389 Patent)

100. Webroot realleges and incorporates by reference the allegations of the preceding paragraphs of this Complaint.

101. Forcepoint has infringed and continues to infringe one or more claims of the ’389 Patent in violation of 35 U.S.C. § 271 in this judicial district and elsewhere in the United States and will continue to do so unless enjoined by this Court. The Accused Products, including products that include features such as Forcepoint’s Advanced Classification Engine and ThreatSeeker Intelligence, at least when used for their ordinary and customary purposes, practice each element of at least claim 1 of the ’389 Patent as demonstrated below.

102. For example, claim 1 of the ’389 Patent recites:

> a method of classifying a computer object as malware, the method comprising:
>
> at a base computer, receiving data about a computer object from a first remote computer on which the computer object or similar computer objects are stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured or runs on the first remote computer, said information including at least an identity of an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed;
>
> at the base computer, receiving data about the computer object from a second remote computer on which the computer object or similar computer on which the computer object or similar computer objects are stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured, or runs on the second remote computer, said information including at least an identity of an object initiating the event, the

event type, and an identity of an object or other entity on which the event is being performed;

storing, at the base computer, said data received from the first and second remote computers;

correlating, by the base computer, at least a portion of the data about the computer object received from the first remote computer to at least a portion of the data about the computer object received from the second remote computer;

comparing, by the base computer, the correlated data about the computer object received from the first and second remote computers to other objects or entities to identify relationships between the correlated data and the other objects or entities; and

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

103. To the extent the preamble is limiting, the Accused Products perform *a method of classifying a computer object as malware*. For example, Forcepoint's Advanced Classification Engine ("ACE") in combination with Forcepoint's ThreatSeeker Intelligence "identif[ies] malware, phishing, spam, and other risks to [an] enterprise." (*See* Forcepoint, *Forcepoint Advanced Classification Engine (ACE) Solution Brief*, https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_ace_en.pdf.)

104. The Accused Products perform a method that includes the step of *at a base computer, receiving data about a computer object from a first [and second] remote computer on which the computer object or similar computer objects are stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured or runs on the first remote computer, said information including at least an identity of an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed*. Forcepoint's ThreatSeeker Intelligence resides on a base computer and receives threat data about computer objects from multiple remote computers. For

example, "ThreatSeeker Intelligence aggregates threat intel from ACE engines, firewalls, and endpoints deployed around the world to provide telemetry back to those devices." (*See* Forcepoint, *Forcepoint Advanced Classification Engine (ACE) Solution Brief*, https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_ace_en.pdf.) The figure below illustrates how ThreatSeeker receives information from Forcepoint products.



(*See* Forcepoint, *Forcepoint Advanced Classification Engine (ACE) Solution Brief*, https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_ace_en.pdf.)

105. ThreatSeeker Intelligence also receives "threat telemetry" and "threat intelligence," which includes collecting content from "web pages, documents, executable, scripts, streaming, media, emails, mobile applications, and other Internet traffic," as well as information from Forcepoint's Advanced Classification Engine (ACE). For example, "ACE inspects traffic content and usage patterns using up to eight different defense assessment areas for identifying malware, phishing, spam, and other risks" to provide threat data including behavioral information. (*Id.*)

106. Moreover, as explained above, Forcepoint's ThreatSeeker Intelligence aggregates data from sources of information, including endpoints, firewalls, and Threatpoint's ACE. That information about the object is used to detect threats. For example, ACE assesses objects according to identifying information from their code, *i.e.*, scripts and iframe tags, url classification of

websites, and how data is structured. This information is provided to the ThreatSeeker Intelligence, which "allow[s] threat intelligence [to,] from one attack vector[,] … influence analytics applied to another attack vector." (*See* Forcepoint, *Forcepoint Advanced Classification Engine (ACE) Solution Brief,* https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_ forcepoint_ace_en.pdf.)

107. The Accused Products perform a method that includes *storing, at the base computer, said data received from the first and second remote computers*. As explained above, "Forcepoint's ThreatSeeker Intelligence aggregates threat intel from ACE engines, firewalls, and endpoints." (*Id.*) Aggregating threat intelligence and telemetry involves and includes storing the information. For example, data collected is further stored and analyzed by Forcepoint X-Labs. "ThreatSeeker informs ACE with directly actionable updates by continually collecting content and new trends, and this data allows X-Labs researchers to further optimize data models and analytics on an ongoing basis." (*Id.*)

108. The Accused Products perform a method that includes *correlating, by the base computer, at least a portion of the data about the computer object received from the first remote computer to at least a portion of the data about the computer object received from the second remote computer*. As explained above, Forcepoint's ThreatSeeker Intelligence aggregates data from endpoints, firewalls, and Threatpoint's ACE. That aggregated information is then correlated so that the information may be used, for example, to assist in identifying zero-day threats as well as threats identified by other sources. For example, Forcepoint's ACE, in connection with Forcepoint's ThreatSeeker Intelligence, performs file analysis such that "[o]bserved behavior is correlated with known threats to provide valuable information for even zero-day threats, all in real time." (*See* Forcepoint, *Cybersecurity Intelligence (CSI) Tools*,

https://www.forcepoint.com/services/cybersecurity-intelligence-csi-tools.) This "allow[s] threat intelligence gained from one attack vector to influence analytics applied to another attack vector." (*See* Forcepoint, *Forcepoint Advanced Classification Engine (ACE) Solution Brief*, https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_ace_en.pdf.)

109. The Accused Products perform a method that includes *comparing, by the base computer, the correlated data about the computer object received from the first and second remote computers to other objects or entities to identify relationships between the correlated data and the other objects or entities*. As explained above, Forcepoint ACE, in connection with Forcepoint's ThreatSeeker Intelligence, correlates intelligence and telemetry from attacks and compares that correlated information to new and previously identified threats for analysis, including on information and belief, to determine relationships between the correlated data and any potential threats (other objects or entities) "to detect previously unknown malware." For example, Forcepoint advertises that its ACE shares what it learns, "allowing threat intelligence gained from one attack vector to influence analytics applied to another attack vector." (*See* Forcepoint, *Forcepoint Advanced Classification Engine (ACE) Solution Brief*, https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_ace_en.pdf.)

110. Moreover, Forcepoint's ACE, in connection with Forcepoint's ThreatSeeker Intelligence, performs file analysis such that "[o]bserved behavior is correlated with known threats to provide valuable information for even zero-day threats, all in real time." (*See* Forcepoint, *Cybersecurity Intelligence (CSI) Tools*, https://www.forcepoint.com/services/cybersecurity-intelligence-csi-tools.)

111. The Accused Products perform a method that includes *classifying, by the base computer, the computer object as malware on the basis of said comparison*. As explained above, Forcepoint's ThreatSeeker Intelligence, alone or in combination with Forcepoint's ACE, "identif[ies] malware, phishing, spam, and other risks to [an] enterprise." (*Id.*)

112. Each claim in the '389 Patent recites an independent invention. Neither claim 1, described above, nor any other individual claim is representative of all claims in the '389 Patent.

113. Forcepoint has been aware of the '389 Patent since at least the filing of this Complaint. Further, Plaintiffs have marked their products with the '389 Patent, including on its web site, since at least July 2020.

114. Forcepoint directly infringes at least claim 1 of the '389 Patent, literally or under the doctrine of equivalents, by performing the steps described above. For example, on information and belief, the Forcepoint performs a method in an infringing manner as described above by running the Accused Products to protect its own computer and network operations. On information and belief, the Forcepoint performs a method in an infringing manner in testing the operation of the Accused Products and corresponding systems. As another example, Forcepoint performs each of the method steps when providing or administering services to third parties, customers, and partners using the Accused Products.

115. Defendant's directly infringing activities in this District, Texas and elsewhere in the United States include at least making and using the inventions in the claims identified above. Defendant makes and uses the claimed inventions in connection with at least the following activities involving the Accused Products (*e.g.*, software and corresponding systems): (1) operating and maintaining data centers that process information; (2) employing the Accused Products in connection with, and to protect, its own computer and network operations and/or the

operations of affiliates, (3) using software that includes the Accused Products, (4) testing the operation of the Accused Products and corresponding systems, (5) collecting and gathering information and data, and (6) providing services to others including installation, maintenance, update, verification, processing, support, and other services.

116. In addition, Defendant directly infringes the above identified claims because it directs and controls the performance of the above-identified claimed inventions by others. As an example, Defendant 1) directs and controls the performance and operation of the Accused Products or components thereof on customers', partners', and end users' devices; and 2) establishes the manner of the operation of the Accused Products. Defendant further directs and controls by means that include contractual agreements such as its End User License Agreements, its continued monitoring, verification and access to the Accused Products, its continued collection of information from the Accused Products, its programming of the software used in the Accused Products, and/or its provision of information, data, and instructions that control the operation of the system (for example, updates to algorithms, malware definitions, and other system and module code). Each of these claimed steps is performed automatically, without the user taking an action or initiating the process.

117. Defendant also actively induces direct infringement of the above identified patent claims by others with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). Defendant's partners, customers, and end users (collectively, "Third Party Users") of the Accused Products and corresponding systems and services directly infringe at least by making and using the inventions in the above-identified claims, for example when they install, operate and use the Accused Products and corresponding systems. Defendant has had knowledge of the asserted patent and of

the specific allegations of infringement with respect to the asserted patent at least as of the filing data of the complaint.

118. Defendant encourages and induces Third Party Users with specific intent to use the Accused Products in a manner that infringes the asserted claims of the '389 Patent at least by offering, selling, licensing and providing the Accused Products which infringe the above-identified claims of the asserted patent when installed and operated by Third Party Users. Defendant further induces infringement of the above-identified claims by Third Party Users by engaging in activities that include: 1) making its security services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users, and 2) marketing, advertising, promotion, installation, support, and distribution activities involving the Accused Products, including its security software and services in the United States; 3) recommending and selling the Accused Products and providing technical support for the installation, implementation, integration, and ongoing operation of the Accused Products; 4) providing installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Products remain operational for each customer through ongoing technical support. *See, e.g.*, http://www.websense.com/content/support/library/web/v85/web_help/web_help.pdf; https://www.websense.com/content/support/library/amd-op/v10/install/manager-install.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/brochure_forcepoint_email_security_en_0.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf; http://www.websense.com/content/support/library/data/v87/help/help.pdf. For example, Defendant shares instructions, guides, and manuals, which advertise and instruct Third Party Users

on how to use the software in a manner that infringes the above-identified claims of the asserted patent *Id*. Defendant further provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that deploy and use the Accused Products in an infringing manner. *Id*.

119. Further, Defendant induces infringement by Third Party Users of the Accused Products by requiring each user to enter into one or more contractual agreements with Defendant including an End User License Agreement, which obligates each customer to perform certain actions in order to use the Accused Products and to use the Accused Products in a manner that infringes the above-identified claims. In order to receive the benefit of Defendant and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the above-identified claims of the asserted patent.

120. Defendant also contributes to the infringement of the Accused Products by Third Party Users by providing within the United States or importing the Accused Products into the United States, which are for use in practicing, and under normal operation practice, the claimed invention in an infringing manner, constituting a material part of the claimed invention, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. For example, the Accused Products and the example functionality described below have no substantial non-infringing uses but are specifically designed to practice the '389 Patent.

121. The Accused Products have no substantial non-infringing uses because the accused functionality is an integral part of the Accused Products and must be performed for the Accused Products to perform their intended purpose. For example, without compiling and analyzing data

about the behavior of an object running on one or more remote computers, the Accused Products could not timely detect processes (running objects) that pose as threats. *See, e.g.*, http://www.websense.com/content/support/library/web/v85/web_help/web_help.pdf; https://www.websense.com/content/support/library/amd-op/v10/install/manager-install.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/brochure_forcepoint_email_security_en_0.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf; http://www.websense.com/content/support/library/data/v87/help/help.pdf. These processes are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the asserted claims of the '389 Patent, that functionality could not be performed.

122. The accused functionality, including the accused behavioral threat analysis functionality described below, itself has no substantial non-infringing uses because the components, modules and methods identified are a necessary part of that functionality. For example, without the ACE and ThreatSeeker platforms that compile and correlate data from processes that run on the endpoints protected by the Accused Products, the Accused Products could not assess which processes on any given endpoint constitute a threat. *See, e.g.*, WBR_FRP001321, https://www.forcepoint.com/sites/default/files/resources/solutionbrief/solution_brief_forcepoint_ace_en.pdf. These process are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and

shown below, or without the system and components identified below that practice the asserted claims of the '389 Patent, that functionality could not be performed.

123. In addition, the products, systems, modules, and methods provided by Defendant constitute a material part of the invention—indeed, it provides all the components, modules, and features that perform the claimed methods and systems. For example, the Accused Products and corresponding functionality constitute a material part of the inventions claimed because such analysis is integral to the processes identified above (such as "*correlating, by the base computer, at least a portion of the data about the computer object received from the first remote computer to at least a portion of the data about the computer object received from the second remote computer*"), as required by the asserted claims of the '389 Patent. None of these products are staple goods—they are sophisticated and customized cyber security and malware detection products and systems.

124. On information and belief, the infringing actions of each partner, customer, and/or end-user of the Accused Products are attributable to Forcepoint. For example, on information and belief, Forcepoint directs and controls the activities or actions of its partners in connection with the Accused Products by contractual agreement or otherwise requiring partners to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Forcepoint further directs and controls the operation of devices executing the Accused Products by programming the software which, when executed by a customer or end user, perform the method steps of at least claim 1 of the '389 Patent.

125. Plaintiffs have suffered and continues to suffer damages, including lost profits, as a result of Defendant's infringement of the '389 Patent. Defendant is therefore liable to Plaintiffs

under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiffs for Defendant's infringement, but no less than a reasonable royalty.

126. Plaintiffs will continue to suffer irreparable harm unless this Court preliminarily and permanently enjoins Defendant, its agents, employees, representatives, and all others acting in concert with Defendant from infringing the '389 Patent. On information and belief, Plaintiffs have lost potential customers, business opportunities, and goodwill in the community. Plaintiffs will continue to suffer these harms absent an injunction.

127. Defendant's infringement of the '389 Patent, is knowing and willful. Defendant acquired actual knowledge of the '389 Patent at least when Plaintiffs filed this lawsuit and had constructive knowledge of the '389 Patent from at least the date Plaintiffs marked its products with the '389 Patent and/or provided notice of the '389 Patent on its website.

128. On information and belief, despite Defendant's knowledge of the Asserted Patents and Plaintiffs' patented technology, Defendant made the deliberate decision to sell products and services that they knew infringe these patents. Defendant's continued infringement of the '389 Patent with knowledge of the '389 Patent constitutes willful infringement.

129. Plaintiffs' allegations of infringement, indirect infringement, and willful infringement with respect to this patent are further set forth in Plaintiffs' Disclosure of Asserted Claims and Preliminary Infringement Contentions Pursuant to the Court's Standing Order Governing Patent Proceedings served July 12, 2022.

SECOND CAUSE OF ACTION
(INFRINGEMENT OF THE '844 PATENT)

130. Plaintiffs reallege and incorporate by reference the allegations of the preceding paragraphs of this Complaint.

131. Forcepoint has infringed and continues to infringe one or more claims of the ’844 Patent in violation of 35 U.S.C. § 271 in this judicial District and elsewhere in the United States and will continue to do so unless enjoined by this Court. The Accused Products, including features of the Forcepoint Advanced Malware Detection Appliance (as well as any other products that include the features described below), when used for their ordinary and customary purposes, practice each element of at least claim 1 of the ’844 Patent, as demonstrated below.

132. Claim 1 of the ’844 Patent recites:

> 1. A computer-implemented method comprising:
>
> extracting a plurality of static data points from an executable file without decrypting or unpacking the executable file, wherein the plurality of static data points represent predefined character strings in the executable file;
>
> generating a feature vector from the plurality of static data points using a classifier trained to classify the plurality of static data points based on a collection of data comprising known malicious executable files, known benign executable files, and known unwanted executable files, wherein the collection of data comprises at least a portion of the plurality of static data points, and
>
> wherein one or more features of the feature vector are selectively turned on or off based on whether a value of one or more static data points from the plurality of extracted static data points is within a predetermined range; and
>
> evaluating the feature vector using support vector processing to determine whether the executable file is harmful.

133. The Accused Products perform each element of the method of claim 1 of the ’844 Patent. To the extent the preamble is construed to be limiting, the Accused Products perform a *computer-implemented method*, as further explained below. For example, the Accused Products “perform[] a combination of static and behavioral analysis to detect and prevent the entry of known malware and brand-new exploits.” The “Forcepoint Advanced Malware Detection Appliance is an on-premises, automated malware analysis framework developed for organizations needing to add detection and prevention against stealthy and advanced threats to their existing Forcepoint Web

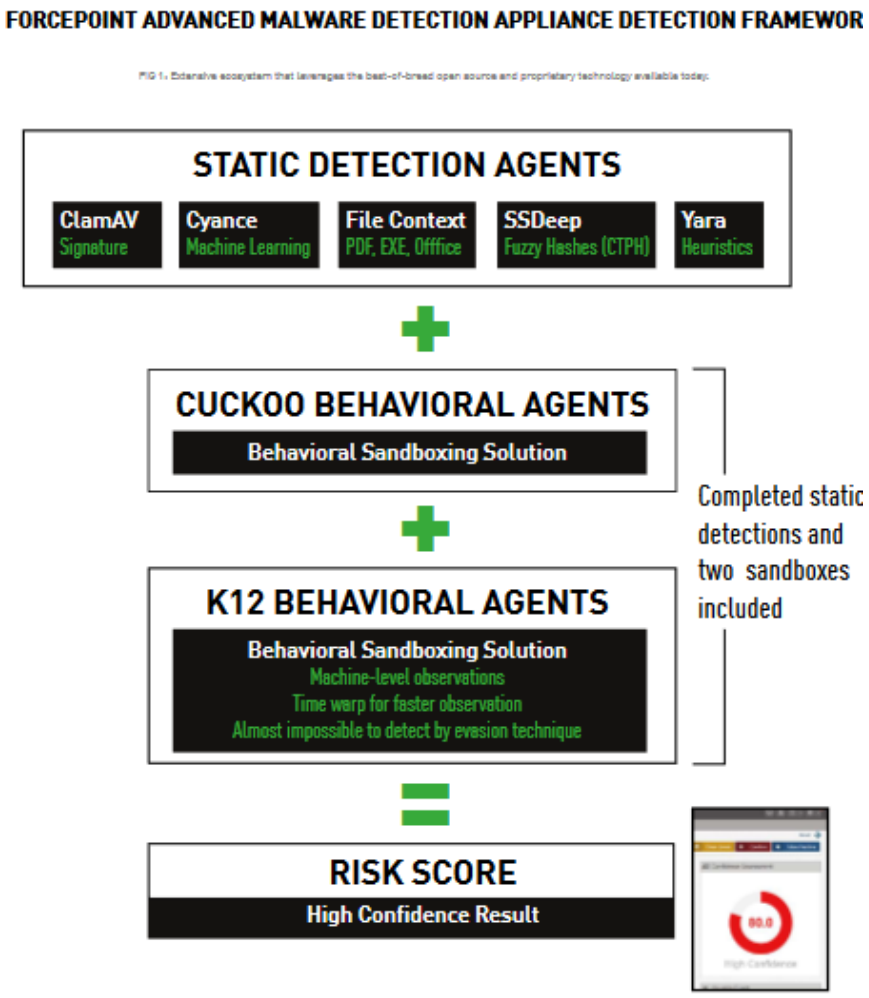
and Email Security solutions" including processing "files through seven distinct static analytic agents and a dual-sandboxing process." The Accused Products perform an analysis that includes integrating and using static analysis, including by integrating and using its partner Cylance's static analysis, in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy." On information and belief, this analysis includes the use of "models" that are "deliver[ed]" to the appliance and then used to "predict whether a file is valid or malicious."

For years, Security and Risk (SR) professionals made major investments in signature-based defenses of email, network and endpoint security solutions. The methodology of these solutions has proven itself ineffective against today's evasive malware being developed by highly sophisticated and well-funded adversaries. As a response, SR professionals are turning to Automated Malware Analysis (AMA) technologies in order to arm themselves against zero day and Advanced Persistent Threats (APTs) attacking their organizations. AMA tools automate the unique skill set of malware analysis traditionally done only by highly qualified manual practitioners. Due to the shortage of this expertise, manual processes have been replaced with automation that performs a combination of static and behavioral analysis to detect and prevent the entry of known malware and brand-new exploits.

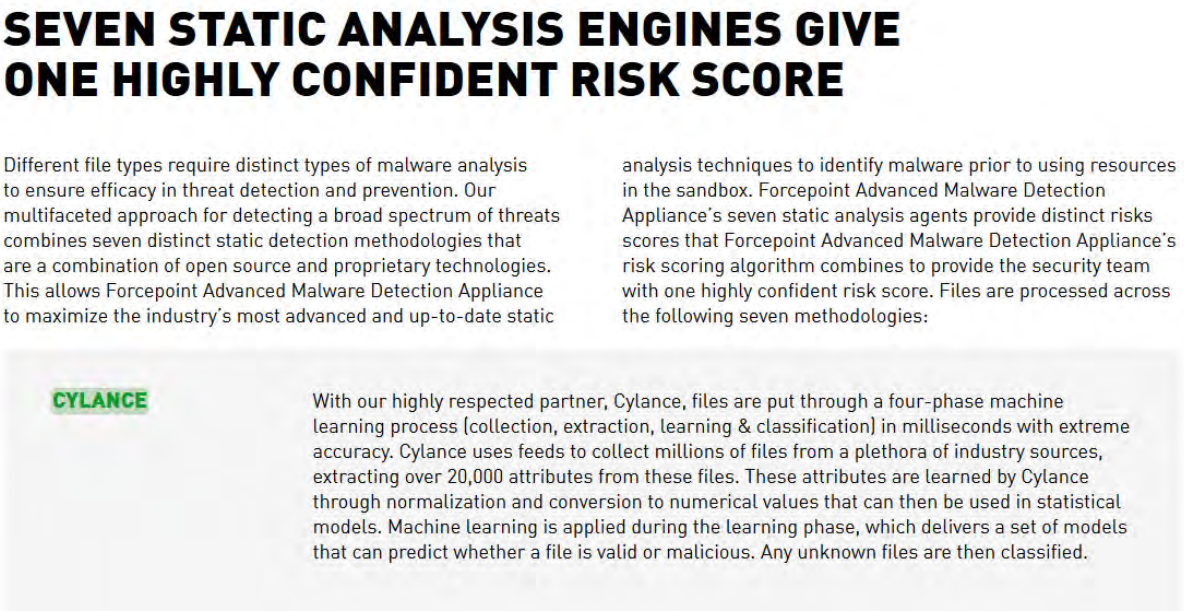
Forcepoint Advanced Malware Detection Appliance 3.4

Forcepoint Advanced Malware Detection Appliance is an on-premises, automated malware analysis framework developed for organizations needing to add detection and prevention against stealthy and advanced threats to their existing Forcepoint Web and Email Security solutions. Forcepoint Advanced Malware Detection Appliance framework's unmatched efficacy processes files through seven distinct static analytic agents and a dual-sandboxing process. Its ecosystem analyses malware behavior with a combination of best-of-breed open source and Forcepoint proprietary static and dynamic technologies. Unique to the market is the defense-grade anti-evasion technology within Forcepoint's proprietary ThINK sandbox, stopping malware typically capable of circumventing commercially available sandboxes.

(*See* https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at page 2.)



(*See* https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at page 3.)



SEVEN STATIC ANALYSIS ENGINES GIVE ONE HIGHLY CONFIDENT RISK SCORE

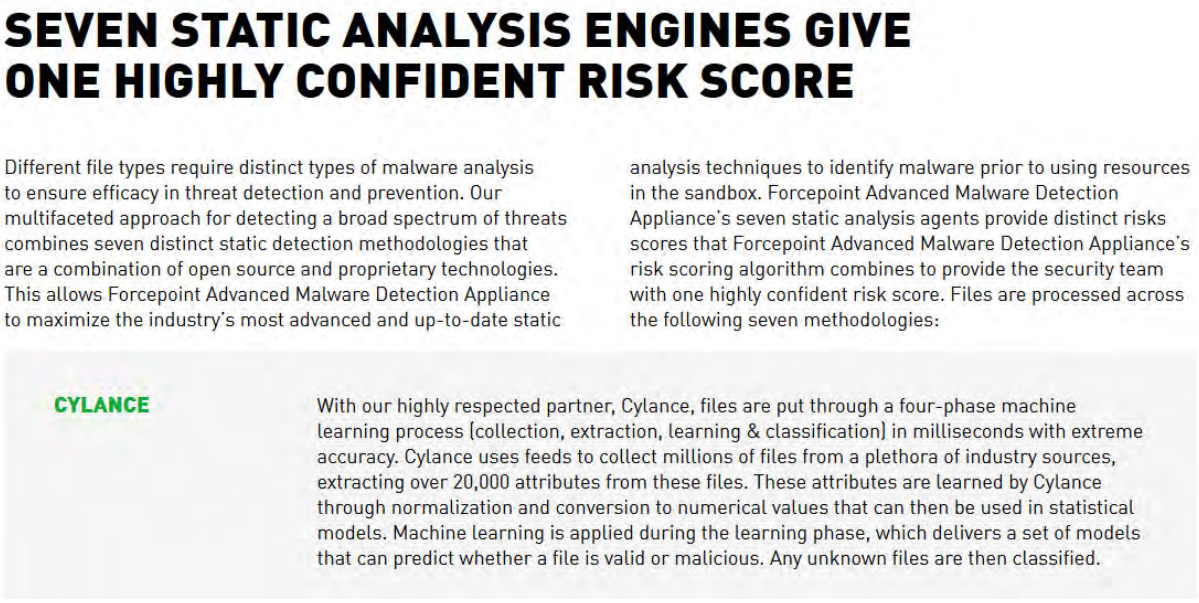
Different file types require distinct types of malware analysis to ensure efficacy in threat detection and prevention. Our multifaceted approach for detecting a broad spectrum of threats combines seven distinct static detection methodologies that are a combination of open source and proprietary technologies. This allows Forcepoint Advanced Malware Detection Appliance to maximize the industry's most advanced and up-to-date static analysis techniques to identify malware prior to using resources in the sandbox. Forcepoint Advanced Malware Detection Appliance's seven static analysis agents provide distinct risks scores that Forcepoint Advanced Malware Detection Appliance's risk scoring algorithm combines to provide the security team with one highly confident risk score. Files are processed across the following seven methodologies:

CYLANCE

With our highly respected partner, Cylance, files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy. Cylance uses feeds to collect millions of files from a plethora of industry sources, extracting over 20,000 attributes from these files. These attributes are learned by Cylance through normalization and conversion to numerical values that can then be used in statistical models. Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified.

(*See* https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at page 4.)

134. The Accused Products perform a method that includes *extracting a plurality of static data points from an executable file without decrypting or unpacking the executable file, wherein the plurality of static data points represent predefined character strings in the executable file*. For example, the Accused Products use "seven distinct static detection methodologies" including Cylance's static analysis, in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy."



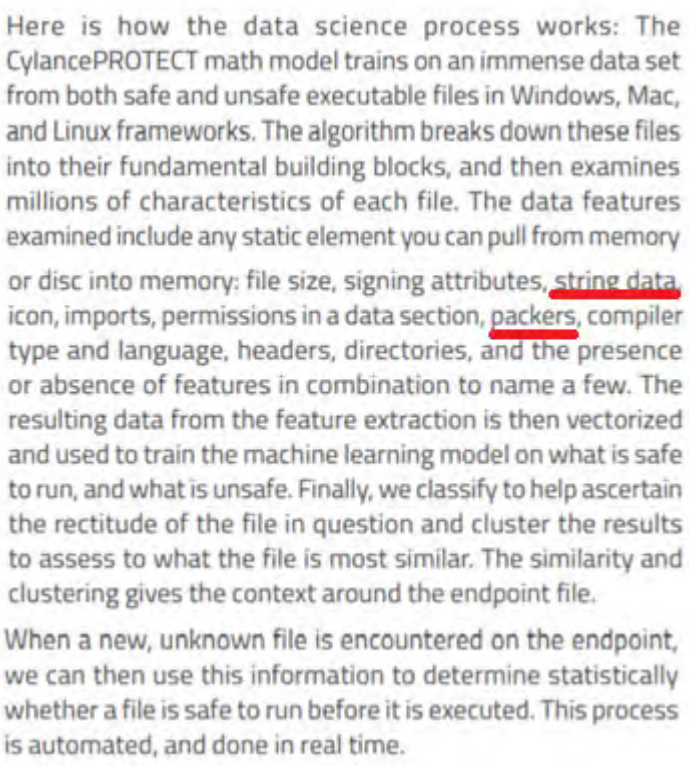
SEVEN STATIC ANALYSIS ENGINES GIVE ONE HIGHLY CONFIDENT RISK SCORE

Different file types require distinct types of malware analysis to ensure efficacy in threat detection and prevention. Our multifaceted approach for detecting a broad spectrum of threats combines seven distinct static detection methodologies that are a combination of open source and proprietary technologies. This allows Forcepoint Advanced Malware Detection Appliance to maximize the industry's most advanced and up-to-date static analysis techniques to identify malware prior to using resources in the sandbox. Forcepoint Advanced Malware Detection Appliance's seven static analysis agents provide distinct risks scores that Forcepoint Advanced Malware Detection Appliance's risk scoring algorithm combines to provide the security team with one highly confident risk score. Files are processed across the following seven methodologies:

CYLANCE

With our highly respected partner, Cylance, files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy. Cylance uses feeds to collect millions of files from a plethora of industry sources, extracting over 20,000 attributes from these files. These attributes are learned by Cylance through normalization and conversion to numerical values that can then be used in statistical models. Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified.

135. The Accused Products implement Cylance's static analysis—*e.g.* through the use of a set of models delivered to the appliance. That analysis examines data features that include "any static element you can pull from memory or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination." Further, "[w]hen a new, unknown file is encountered on the endpoint, we can then use this information to determine statistically whether a file is safe to run before it is executed."

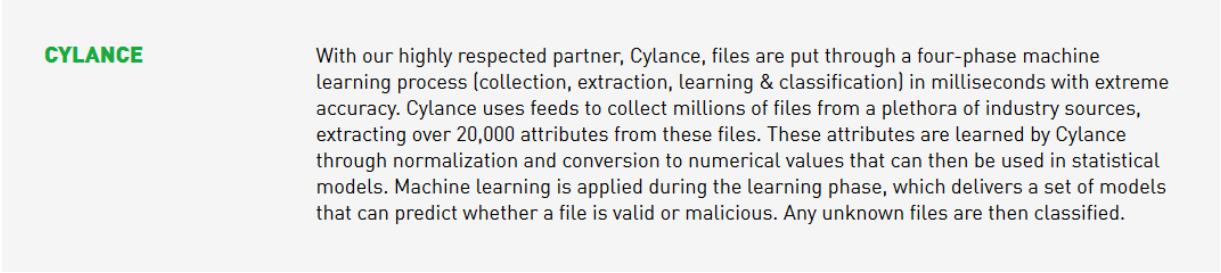


Here is how the data science process works: The CylancePROTECT math model trains on an immense data set from both safe and unsafe executable files in Windows, Mac, and Linux frameworks. The algorithm breaks down these files into their fundamental building blocks, and then examines millions of characteristics of each file. The data features examined include any static element you can pull from memory or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination to name a few. The resulting data from the feature extraction is then vectorized and used to train the machine learning model on what is safe to run, and what is unsafe. Finally, we classify to help ascertain the rectitude of the file in question and cluster the results to assess to what the file is most similar. The similarity and clustering gives the context around the endpoint file.

When a new, unknown file is encountered on the endpoint, we can then use this information to determine statistically whether a file is safe to run before it is executed. This process is automated, and done in real time.

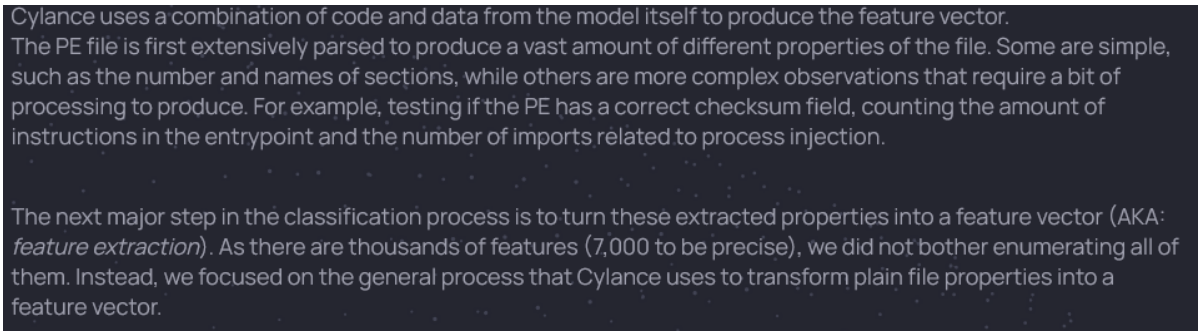
(*See* https://www.blackberry.com/content/dam/cylance/documents/pdf/pdf-feature-focus-protect-malware-control.pdf at page 02; *see also*, blackberry.com/us/en/products/unified-endpoint-security/cylance-is-now-blackberry ("Cylance is now part of Blackberry Cybersecurity").)

136. The Accused Products perform a method that includes *generating a feature vector from the plurality of static data points using a classifier trained to classify the plurality of static data points based on a collection of data comprising known malicious executable files, known benign executable files, and known unwanted executable files, wherein the collection of data comprises at least a portion of the plurality of static data points*. As explained above, the Accused Products implement Cylance's static analysis in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification)." "Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified."



(*See* https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf.)

137. Additionally, the resulting data "from the feature extraction is . . . vectorized and used to train the machine learning." On information and belief, Cylance's static analysis similarly generates a feature vector from the static data points during the classification process. (*See* https://www.blackberry.com/content/dam/cylance/documents/pdf/pdf-feature-focus-protect-malware-control.pdf at page 02.) For example, the Accused Products, implementing the Cylance static analysis, uses "any static element you can pull from memory or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination." During the classification process, the Accused Products generate a "feature vector" using "extracted properties" from the file:

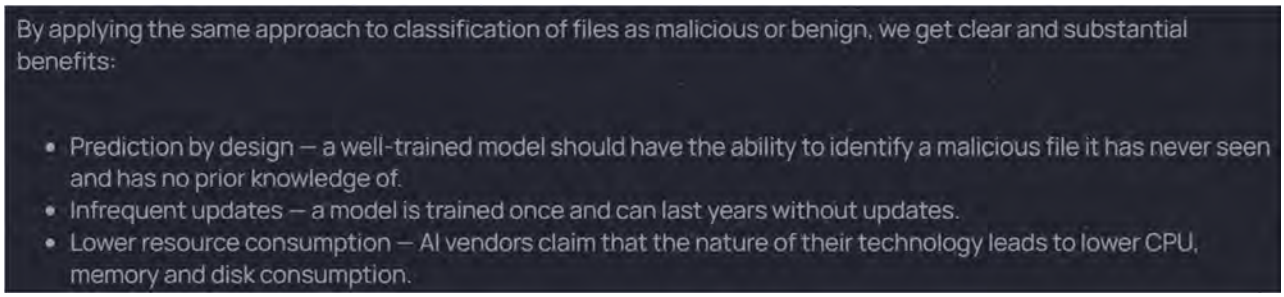


(*See* https://skylightcyber.com/2019/07/18/cylance-i-kill-you/.)

138. As explained above, the Accused Products implement Cylance's static analysis (*e.g.*, through the use of a set of models provided to the appliance), which uses a "math model [that] trains on an immense data set from both safe and unsafe executable files in Windows, Mac, and Linux frameworks. The algorithm breaks down these files into their fundamental building blocks, and then examines millions of characteristics of each file."

> Here is how the data science process works: The CylancePROTECT math model trains on an immense data set from both safe and unsafe executable files in Windows, Mac, and Linux frameworks. The algorithm breaks down these files into their fundamental building blocks, and then examines millions of characteristics of each file. The data features examined include any static element you can pull from memory

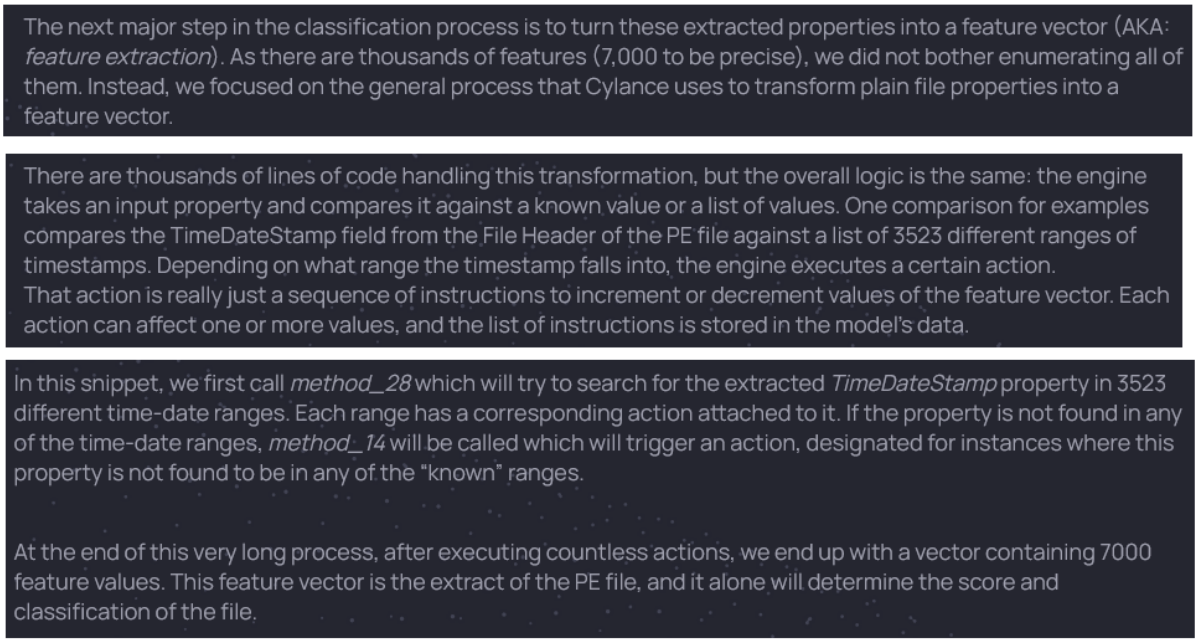
(*See* https://www.blackberry.com/content/dam/cylance/documents/pdf/pdf-feature-focus-protect-malware-control.pdf.)



(*See* https://skylightcyber.com/2019/07/18/cylance-i-kill-you/ at page 04.)

139. The Accused Products perform a method that includes *wherein one or more features of the feature vector are selectively turned on or off based on whether a value of one or more static data points from the plurality of extracted static data points is within a predetermined range*. As explained above, the Accused Products implement Cylance's static analysis as part of the classification process. When that occurs "extracted properties" are turned "into a feature vector." On information and belief, there "are thousands of lines of code handling this transformation, but the overall logic is the same: the engine takes an input property and compares

it against a known value or a list of values. One comparison for examples compares the TimeDateStamp field from the File Header of the PE file against a list of 3523 different ranges of timestamps. Depending on what range the timestamp falls into, the engine executes a certain action. That action is really just a sequence of instructions to increment or decrement values of the feature vector. Each action can affect one or more values, and the list of instructions is stored in the model's data."



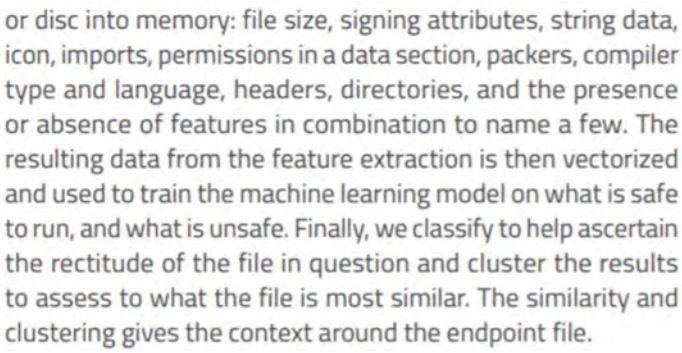
The next major step in the classification process is to turn these extracted properties into a feature vector (AKA: *feature extraction*). As there are thousands of features (7,000 to be precise), we did not bother enumerating all of them. Instead, we focused on the general process that Cylance uses to transform plain file properties into a feature vector.

There are thousands of lines of code handling this transformation, but the overall logic is the same: the engine takes an input property and compares it against a known value or a list of values. One comparison for examples compares the TimeDateStamp field from the File Header of the PE file against a list of 3523 different ranges of timestamps. Depending on what range the timestamp falls into, the engine executes a certain action.
That action is really just a sequence of instructions to increment or decrement values of the feature vector. Each action can affect one or more values, and the list of instructions is stored in the model's data.

In this snippet, we first call *method_28* which will try to search for the extracted *TimeDateStamp* property in 3523 different time-date ranges. Each range has a corresponding action attached to it. If the property is not found in any of the time-date ranges, *method_14* will be called which will trigger an action, designated for instances where this property is not found to be in any of the "known" ranges.

At the end of this very long process, after executing countless actions, we end up with a vector containing 7000 feature values. This feature vector is the extract of the PE file, and it alone will determine the score and classification of the file.

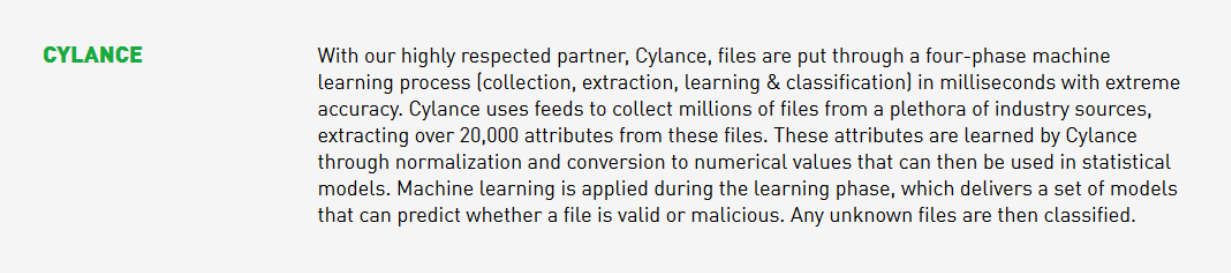
(*See* https://skylightcyber.com/2019/07/18/cylance-i-kill-you/.)



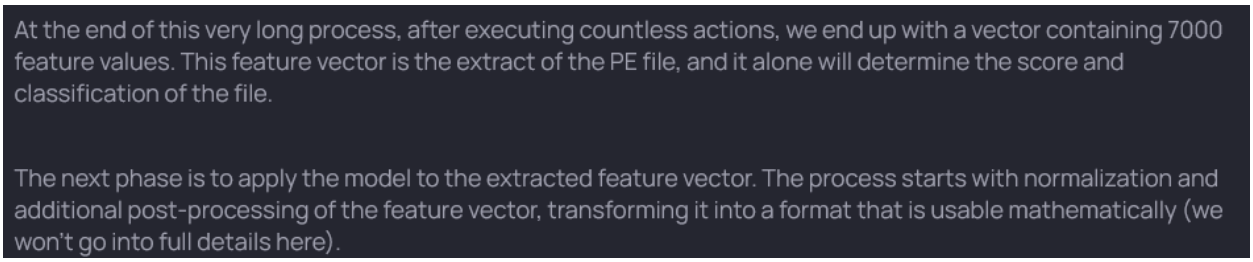
or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination to name a few. The resulting data from the feature extraction is then vectorized and used to train the machine learning model on what is safe to run, and what is unsafe. Finally, we classify to help ascertain the rectitude of the file in question and cluster the results to assess to what the file is most similar. The similarity and clustering gives the context around the endpoint file.

(*See* https://www.blackberry.com/content/dam/cylance/documents/pdf/pdf-feature-focus-protect-malware-control.pdf at page 02.)

140. The Accused Products perform a method that includes *evaluating the feature vector using support vector processing to determine whether the executable file is harmful*. As explained above, the Accused Products use Cylance's static analysis in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification)" and "[m]achine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified." When Forcepoint uses Cylance's static analysis, the "feature vector" is used to "determine the score and classification of the file" by "apply[ing] the model to the extracted feature vector."



(*See* https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf.)



(*See* https://skylightcyber.com/2019/07/18/cylance-i-kill-you/.)

141. Each claim in the '844 Patent recites an independent invention. Neither claim 1, described above, nor any other individual claim is representative of all claims in the '844 Patent.

142. Defendant has been aware of the ’844 Patent since at least the filing of this Complaint. Further, Plaintiffs have marked their products with the ’844 Patent, including on their web site, since at least July 2020.

143. Defendant directly infringes at least claim 1 of the ’844 Patent, either literally or under the doctrine of equivalents, by performing the steps described above. For example, on information and belief, Defendant performs the claimed method in an infringing manner as described above by running the Accused Products to protect their own computer and network operations. On information and belief, Defendant also performs the claimed method in an infringing manner when testing the operation of the Accused Products and corresponding systems. As another example, Defendant performs the claimed method when providing or administering services to third parties, customers, and partners using the Accused Products. Defendant’s partners, customers, and end users of the Accused Products and corresponding systems and services directly infringe at least claim 1 of the ’844 Patent, literally or under the doctrine of equivalents, at least by using the Accused Products and corresponding systems and services, as described above.

144. Defendant’s directly infringing activities in this District, Texas and elsewhere in the United States include at least making and using the inventions in the claims identified above. Defendant makes and uses the claimed inventions in connection with at least the following activities involving the Accused Products (*e.g.*, software and corresponding systems): (1) operating and maintaining data centers that process information; (2) employing the Accused Products in connection with, and to protect, its own computer and network operations and/or the operations of affiliates, (3) using software that includes the Accused Products, (4) testing the operation of the Accused Products and corresponding systems, (5) collecting and gathering

information and data, and (6) providing services to others including installation, maintenance, update, verification, processing, support, and other services.

145. In addition, Defendant directly infringes the above identified claims because it directs and controls the performance of the above-identified claimed inventions by others. As an example, Defendant 1) directs and controls the performance and operation of the Accused Products or components thereof on customers', partners', and end users' devices; and 2) establishes the manner of the operation of the Accused Products. Defendant further directs and controls by means that include contractual agreements such as its End User License Agreements, its continued monitoring, verification and access to the Accused Products, its continued collection of information from the Accused Products, its programming of the software used in the Accused Products, and/or its provision of information, data, and instructions that control the operation of the system (for example, updates to algorithms, malware definitions, and other system and module code). Each of these claimed steps is performed automatically, without the user taking an action or initiating the process. As such, Defendant is liable for all of the steps performed by the Accused Products. *See SiRF Tech., Inc. v. International Trade Commission*, 601 F.3d 1319 (Fed. Cir. 2010).

146. Defendant also actively induces direct infringement of the above identified patent claims by others with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). Defendant's partners, customers, and end users (collectively, "Third Party Users") of the Accused Products and corresponding systems and services directly infringe at least by making and using the inventions in the above-identified claims, for example when they install, operate and use the Accused Products and corresponding systems. Defendant has had knowledge of the asserted patent and of

the specific allegations of infringement with respect to the asserted patent at least as of the filing data of the complaint.

147. Defendant encourages and induces Third Party Users with specific intent to use the Accused Products in a manner that infringes the asserted claims of the ’844 Patent at least by offering, selling, licensing and providing the Accused Products which infringe the above-identified claims of the asserted patent when installed and operated by Third Party Users. Defendant further induces infringement of the above-identified claims by Third Party Users by engaging in activities that include: 1) making its security services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users, and 2) marketing, advertising, promotion, installation, support, and distribution activities involving the Accused Products, including its security software and services in the United States; 3) recommending and selling the Accused Products and providing technical support for the installation, implementation, integration, and ongoing operation of the Accused Products; 4) providing installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Products remain operational for each customer through ongoing technical support. *See, e.g.*, https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.7.0/GUID-EECA15DA-9B8A-4C2F-9C42-D53516ADC19E.html; http://www.websense.com/content/support/library/web/v85/web_help/web_help.pdf; https://www.websense.com/content/support/library/amd-op/v10/install/manager-install.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/brochure_forcepoint_email_security_en_0.pdf; https://www.websense.com/content/support/library/email/v85/email_help/email_help.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf;

https://www.websense.com/content/support/library/casb/admin_guide/casb_admin_guide.pdf. For example, Defendant shares instructions, guides, and manuals, which advertise and instruct Third Party Users on how to use the software in a manner that infringes the above-identified claims of the asserted patent *Id*. Defendant further provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that deploy and use the Accused Products in an infringing manner. *Id*.

148. Further, Defendant induces infringement by Third Party Users of the Accused Products by requiring each user to enter into one or more contractual agreements with Defendant including an End User License Agreement, which obligates each customer to perform certain actions in order to use the Accused Products and to use the Accused Products in a manner that infringes the above-identified claims. In order to receive the benefit of Defendant and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the above-identified claims of the asserted patent.

149. Defendant also contributes to the infringement of the Accused Products by Third Party Users by providing within the United States or importing the Accused Products into the United States, which are for use in practicing, and under normal operation practice, the claimed invention in an infringing manner, constituting a material part of the claimed invention, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. For example, the Accused Products and the example functionality described below have no substantial non-infringing uses but are specifically designed to practice the '844 Patent.

150. The Accused Products have no substantial non-infringing uses because the accused functionality is an integral part of the Accused Products and must be performed for the Accused Products to perform their intended purpose. For example, without compiling and analyzing data about an object running on one or more remote computers, the Accused Products could not timely detect processes (running objects) that pose as threats. *See, e.g.*, https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.7.0/GUID-EECA15DA-9B8A-4C2F-9C42-D53516ADC19E.html; http://www.websense.com/content/support/library/web/v85/web_help/web_help.pdf; https://www.websense.com/content/support/library/amd-op/v10/install/manager-install.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/brochure_forcepoint_email_security_en_0.pdf; https://www.websense.com/content/support/library/email/v85/email_help/email_help.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf; https://www.websense.com/content/support/library/ casb/admin_guide/casb_admin_guide.pdf. These processes are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the asserted claims of the ’844 Patent, that functionality could not be performed.

151. The accused functionality, including the accused threat analysis functionality described below, itself has no substantial non-infringing uses because the components, modules and methods identified are a necessary part of that functionality. For example, without the Cylance PROTECT platform that generates feature vectors from processes that run on the endpoints protected by the Accused Products, the Accused Products could not assess which processes on any

given endpoint constitute a threat. *See, e.g.*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_ en.pdf at 1275. These process are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the asserted claims of the '844 Patent, that functionality could not be performed.

152. In addition, the products, systems, modules, and methods provided by Defendant constitute a material part of the invention—indeed, it provides all the components, modules, and features that perform the claimed methods and systems. For example, the Accused Products and corresponding functionality constitute a material part of the inventions claimed because such analysis is integral to the processes identified above (such as "*generating a feature vector from the plurality of static data points using a classifier trained to classify the plurality of static data points based on a collection of data comprising known malicious executable files, known benign executable files, and known unwanted executable files*"), as required by the asserted claims of the '844 Patent. None of these products are staple goods—they are sophisticated and customized cyber security and malware detection products and systems.

153. Plaintiffs have suffered and continue to suffer damages, including lost profits, as a result of Defendant's infringement of the '844 Patent. Defendant is therefore liable to Plaintiffs under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiffs for Defendant's infringement, but no less than a reasonable royalty.

154. Plaintiffs will continue to suffer irreparable harm unless this Court preliminarily and permanently enjoins Defendant, its agents, employees, representatives, and all others acting in concert with Defendant from infringing the ’844 Patent.

155. Defendant’s infringement of the ’844 Patent is knowing and willful. On information and belief, Defendant had actual knowledge of the ’844 Patent at least by the time Plaintiffs filed this lawsuit and had constructive knowledge of the ’844 Patent from at least the date Plaintiffs marked their products with the ’844 Patent and/or provided notice of the ’844 Patent on their website.

156. On information and belief, the infringing actions of each partner, customer, and/or end-user of the Accused Products are attributable to Forcepoint. For example, on information and belief, Forcepoint directs and controls the activities or actions of its partners in connection with the Accused Products by contractual agreement or otherwise requiring partners to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Forcepoint further directs and controls the operation of devices executing the Accused Products by programming the software which, when executed by a customer or end user, perform the method steps of at least claim 1 of the ’844 Patent.

157. On information and belief, despite Defendant’s knowledge of the Asserted Patents and Plaintiffs’ patented technology, Defendant made the deliberate decision to sell products and services that it knew infringe these patents. Defendant’s continued infringement of the ’844 Patent with knowledge of the ’844 Patent constitutes willful infringement.

158. Despite Defendant’s knowledge of the Asserted Patents and Plaintiffs’ patented technology, Defendant made the deliberate decision to sell products and services that it knew

infringe these patents. Defendant's continued infringement of the '844 Patent with knowledge of the '844 Patent constitutes willful infringement.

159. Plaintiffs' allegations of infringement, indirect infringement, and willful infringement with respect to this patent are further set forth in Plaintiffs' Disclosure of Asserted Claims and Preliminary Infringement Contentions Pursuant to the Court's Standing Order Governing Patent Proceedings served July 12, 2022.

THIRD CAUSE OF ACTION
(INFRINGEMENT OF THE '386 PATENT)

160. Webroot realleges and incorporates by reference the allegations of the preceding paragraphs of this Complaint.

161. Forcepoint has infringed and continues to infringe one or more claims of the '386 patent in violation of 35 U.S.C. § 271 in this judicial district and elsewhere in the United States and will continue to do so unless enjoined by this Court. The Accused Products, including for example products such as Forcepoint's Web Security software that include features such as Forcepoint's Advanced Classification Engine (ACE) and ThreatSeeker Intelligence that, when used for their ordinary and customary purposes, practice each element of at least claim 1 of the '386 Patent, as demonstrated below.

162. For example, claim 1 of the '386 patent recites:

> 1. A method for controlling access to an Internet resource, the method comprising:
>
> transmitting a request for an Internet resource from an Internet-enabled client application from a client-side device of a local area network, the Internet resource residing at a first server;

> receiving the request for the Internet resource at a security appliance of the local area network prior to transmission of the request over the Internet;
>
> determining if a reputation index for the Internet resource is at or above a threshold value established for the local area network, the reputation index generated from a reputation vector for the Internet resource, the reputation vector comprising a plurality of factors for the Internet resource comprising security history, legitimacy, behavior, associations and location, wherein the location factor of the plurality of factors for the reputation vector comprises two or more of: country of domain registration, country of service hosting and country of an internet protocol address block, wherein the legitimacy factor of the plurality of factors for the reputation vector comprises two or more of: age of a domain registration, popularity rank, internet protocol address, number of hosts, top-level domain, and wherein the behavior factor of the plurality of factors for the reputation vector comprises at least one of: plurality of run-time behaviors, script block count, picture count, immediate redirect and response latency; and
>
> transmitting a decision transmission to the Internet-enabled client application of the client-side device, the decision transmission allowing or denying access to the Internet resource.

163. To the extent the preamble is construed to be limiting, the Accused Products perform a method that includes "*controlling access to an Internet resource*." For example, Forcepoint's Web Security, using Forcepoint's ThreatSeeker Intelligence, "provides content-aware web security reputation intelligence that enables customers to manage access to suspicious websites." (*See* Forcepoint, *Master Database URL Categories*, https://www.forcepoint.com/product/feature/master-database-url-categories.)

164. The Accused Products perform a method that includes "*transmitting a request for an Internet resource from an Internet-enabled client application from a client-side device of a local area network, the Internet resource residing at a first server* [*and*] *receiving the request for the Internet resource at a security appliance of the local area network prior to transmission of the request over the Internet*." For example, Forcepoint's Secure Web Gateway includes a Proxy

(SSL) for "in-line inspection of all web traffic [to] ensure[] maximum security efficiency." (*See* Forcepoint, *Forcepoint Web Security: Forcepoint's Cloud and On-Premises Web Security*, https://forcepoint.drift.click/brochure_secure_web_gateway.)

165. The Accused Products perform a method that includes "*determining if a reputation index for the Internet resource is at or above a threshold value established for the local area network, the reputation index generated from a reputation vector for the Internet resource*." For example, Forcepoint's Advanced Classification Engine ("ACE"), which runs analytics on many of Forcepoint's offerings including "Forcepoint Web Security, Forcepoint Email Security, Forcepoint Next Generation Firewall (NGFW), and Forcepoint DLP (Data Loss Prevention)," outputs a composite score for determining whether a resource is malware based on each of ACE's eight defense assessment areas. Forcepoint's literature explains that "ACE defense assessment contributes a risk score and contextual information to the composite scoring algorithms, which then calculates overall risk and consider patterns that may indicate the presence of a threat." (*See* Websense, *Websense ACE (Advanced Classification Engine)* at 7, https://bluekarmasecurity.net/wp-content/uploads/2014/09/Websense-ACE-Advanced-Classification-Engine_whitepaper.pdf.) "An overall composite score is determined and passed back to the submitting security applications (Forcepoint Email Security or Forcepoint Web Security) to take action." (Forcepoint, *Forcepoint Advanced Malware Detection Appliance* 7, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf.) Additionally, "ACE considers more than twenty different characteristics in its URL/IP reputation analysis by including attributes such as traffic volumes, DNS registration details and the autonomous system number (ASN)." (*See*

https://bluekarmasecurity.net/wp-content/uploads/2014/09/Websense-ACE-Advanced-Classification-Engine_whitepaper.pdf at page 6.)

166. The combination of reputation factors analyzed includes *a plurality of factors for the Internet resource comprising security history, legitimacy, behavior, associations, and location*. Regarding *security history*, Forcepoint's ThreatSeeker Intelligence analyzes, for example, a website's reputation for "contain[ing] suspicious content." (*See* https://www.forcepoint.com/product/feature/master-database-url-categories.) Regarding *legitimacy*, Forcepoint ThreatSeeker Intelligence analyzes, for example, a website's reputation for being Newly Registered: "[s]ites whose domain name was registered recently." (*Id.*) Regarding *behavior*, Forcepoint ThreatSeeker Intelligence analyzes, for example, a website's reputation for "camouflage[ing] [its] true nature or … include[ing] elements suggesting latent malicious intent." (*Id.*) Regarding *associations*, Forcepoint ThreatSeeker Intelligence analyzes, for example, a website's reputation for "hosting known and potential exploit code." Regarding *location*, for example, Forcepoint's ACE considers "DNS registration details" and ThreatSeeker Intelligence analyzes a website's reputation for "mask[ing] [its] identity using Dynamic DNS services, often associated with advanced persistent threats (APTs)." (*Id.*)

167. On information and belief, the reputation analysis of *the location factor comprises two or more of: country of domain registration, country of service hosting and country of an internet protocol address block*. For example, "ACE considers more than twenty different characteristics in its URL/IP reputation analysis … including … DNS registration details and the autonomous system number (ASN), thereby going beyond legacy reputation systems..." (*See, e.g.*, https://bluekarmasecurity.net/wp-content/uploads/2014/09/Websense-ACE-Advanced-Classification-Engine_whitepaper.pdf at page 6.) ACE additionally analyzes a website's

reputation for masking its identity using a Dynamic DNS service, which requires knowledge of the website's true identity. (*See* https://www.forcepoint.com/product/feature/master-database-url-categories.) The website's true identity includes where the website's domain is registered. Additionally, the website's full URL address includes the country of the internet protocol, for example .us, .uk, etc.

168. The reputation analysis of *the legitimacy factor … comprises two or more of: age of a domain registration, popularity rank, internet protocol address, number of hosts, or top-level domain.* For example, ThreatSeeker Intelligence analyzes a website's reputation for being Newly Registered: "[s]ites whose domain name was registered recently." (*See* https://www.forcepoint.com/product/feature/master-database-url-categories (navigate to and select "Web Security Reputation").) ThreatSeeker Intelligence also analyzes a website's reputation based on protocol address such as "Private IP Addresses: IP addresses defined in RFC 1918 'Address Allocation for Private Intranets.'" (*Id.* (navigate to and select "Baseline Categories").) Further, as discussed above, ThreatSeeker Intelligence analyzes a website's full URL address, which includes the top-level domain of the internet protocol, for example .com, co.uk, .ru, etc.

169. The reputation analysis of *the behavior factor … comprises at least one of: plurality of run-time behaviors, script block count, picture count, immediate redirect and response latency*. For example, ThreatSeeker Intelligence analyzes a website's reputation for a plurality of run-time behaviors including websites that encrypt using custom encryption methods for the outbound network transmissions and websites that "record all keystrokes, and … send those keystrokes … to an external party." (*See* https://www.forcepoint.com/product/feature/master-database-url-categories (navigate to and select "Forcepoint Security Filtering".) ThreatSeeker Intelligence also analyzes a website's "malicious URL Redirection." (*See* https://bluekarmasecurity.net/wp-

content/uploads/2014/09/Websense-ACE-Advanced-Classification-Engine_whitepaper.pdf at page 10.)

170. Each claim in the ’386 Patent recites an independent invention. Neither claim 1, described above, nor any other individual claim is representative of all claims in the ’386 Patent.

171. Forcepoint has been aware of the ’386 Patent since at least the filing of this Complaint. Further, Plaintiffs have marked their products with the ’386 Patent, including on its web site, since at least July 2020.

172. Forcepoint directly infringes at least claim 1 of the ’386 Patent, literally or under the doctrine of equivalents, by performing the steps described above. For example, on information and belief, the Forcepoint performs a method in an infringing manner as described above by running the Accused Products to protect its own computer and network operations. On information and belief, the Forcepoint performs a method in an infringing manner in testing the operation of the Accused Products and corresponding systems. As another example, Forcepoint performs each of the method steps when providing or administering services to third parties, customers, and partners using the Accused Products.

173. Defendant’s directly infringing activities in this District, Texas and elsewhere in the United States include at least making and using the inventions in the claims identified above. Defendant makes and uses the claimed inventions in connection with at least the following activities involving the Accused Products (*e.g.*, software and corresponding systems): (1) operating and maintaining data centers that process information; (2) employing the Accused Products in connection with, and to protect, its own computer and network operations and/or the operations of affiliates, (3) using software that includes the Accused Products, (4) testing the operation of the Accused Products and corresponding systems, (5) collecting and gathering

information and data, and (6) providing services to others including installation, maintenance, update, verification, processing, support, and other services.

174. In addition, Defendant directly infringes the above identified claims because it directs and controls the performance of the above-identified claimed inventions by others. As an example, Defendant 1) directs and controls the performance and operation of the Accused Products or components thereof on customers', partners', and end users' devices; and 2) establishes the manner of the operation of the Accused Products. Defendant further directs and controls by means that include contractual agreements such as its End User License Agreements, its continued monitoring, verification and access to the Accused Products, its continued collection of information from the Accused Products, its programming of the software used in the Accused Products, and/or its provision of information, data, and instructions that control the operation of the system (for example, updates to algorithms, malware definitions, and other system and module code). Each of these claimed steps is performed automatically, without the user taking an action or initiating the process.

175. Defendant also actively induces direct infringement of the above identified patent claims by others with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). Defendant's partners, customers, and end users (collectively, "Third Party Users") of the Accused Products and corresponding systems and services directly infringe at least by making and using the inventions in the above-identified claims, for example when they install, operate and use the Accused Products and corresponding systems. Defendant has had knowledge of the asserted patent and of the specific allegations of infringement with respect to the asserted patent at least as of the filing data of the complaint.

176. Defendant encourages and induces Third Party Users with specific intent to use the Accused Products in a manner that infringes the asserted claims of the '386 Patent at least by offering, selling, licensing and providing the Accused Products which infringe the above-identified claims of the asserted patent when installed and operated by Third Party Users. Defendant further induces infringement of the above-identified claims by Third Party Users by engaging in activities that include: 1) making its security services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users, and 2) marketing, advertising, promotion, installation, support, and distribution activities involving the Accused Products, including its security software and services in the United States; 3) recommending and selling the Accused Products and providing technical support for the installation, implementation, integration, and ongoing operation of the Accused Products; 4) providing installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Products remain operational for each customer through ongoing technical support. *See*, *e.g.*, http://www.websense.com/content/support/library/web/v85/web_help/web_ help.pdf; http://www.websense.com/content/support/library/web/v85/install/websec_install_full.pdf; https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-secure-web-gateway-en_0_0_0_0_0_0_0_0.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_ 670_ig_a_en-us.pdf; https://www.websense.com/content/support/library/endpoint/v20/installation.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/guide-sase-guide-to-protecting-remote-workers-en_0_0_0_0_0_0_0_0.pdf. For example, Defendant shares instructions, guides, and manuals, which advertise and instruct Third Party Users on how to use

the software in a manner that infringes the above-identified claims of the asserted patent (*Id*.) Defendant further provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that deploy and use the Accused Products in an infringing manner. (*Id*.)

177. Further, Defendant induces infringement by Third Party Users of the Accused Products by requiring each user to enter into one or more contractual agreements with Defendant including an End User License Agreement, which obligates each customer to perform certain actions in order to use the Accused Products and to use the Accused Products in a manner that infringes the above-identified claims. In order to receive the benefit of Defendant and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the above-identified claims of the asserted patent.

178. Defendant also contributes to the infringement of the Accused Products by Third Party Users by providing within the United States or importing the Accused Products into the United States, which are for use in practicing, and under normal operation practice, the claimed invention in an infringing manner, constituting a material part of the claimed invention, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. For example, the Accused Products and the example functionality described below have no substantial non-infringing uses but are specifically designed to practice the '386 Patent.

179. Indeed, as shown below, the Accused Products have no substantial non-infringing uses because the accused functionality is an integral part of the Accused Products and must be performed for the Accused Products to perform their intended purpose. For example, without identifying and assessing reputation of Internet resources, the Accused Products could not timely

identify Internet resources that pose as threats. *See, e.g., infra*; http://www.websense.com/content/support/library/ web/v85/web_help/web_help.pdf; https://www.forcepoint.com/sites/default/files/resources/solution_brief/ solution-brief-secure-web-gateway-en_0_0_0_0_0_0_0_0.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf; https://www.websense.com/content/support/library/endpoint/v20/installation.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/guide-sase-guide-to-protecting-remote-workers-en_0_0_0_0_0_0_0_0.pdf. These processes are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the asserted claims of the '386 Patent, that functionality could not be performed.

180. Additionally, the accused functionality, including the accused threat analysis functionality described below, itself has no substantial non-infringing uses because the components, modules and methods identified are a necessary part of that functionality. For example, without the appliances and systems analyzing Internet resources requested by endpoints protected by the Accused Products, the Accused Products could not assess which processes on any given endpoint constitute a threat. *See, e.g., infra*; http://www.websense.com/content/support/library/ web/v85/web_help/web_help.pdf; https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-secure-web-gateway-en_0_0_0_0_0_0_0_0.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf;

https://www.websense.com/content/support/library/ngfw/v67/mgmt/ngfw_670_pg_a_en-us.pdf; https://www.websense.com/content/support/library/endpoint/v20/installation.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/guide-sase-guide-to-protecting-remote-workers-en_0_0_0_0_0_0_0_0.pdf) . These process are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the asserted claims of the '386 Patent, that functionality could not be performed.

181. In addition, as shown in the detailed analysis below, the products, systems, modules, and methods provided by Defendant constitute a material part of the invention—indeed, it provides all the components, modules, and features that perform the claimed methods and systems. For example, the Accused Products and corresponding functionality constitute a material part of the inventions claimed because such analysis is integral to the processes identified above (such as "*determining if a reputation index for the Internet resource is at or above a threshold value established for the local area network, the reputation index generated from a reputation vector for the Internet resource*"), as required by the asserted claims of the '386 Patent. None of these products are staple goods—they are sophisticated and customized cyber security and malware detection products and systems.

182. Based on current information available to Plaintiffs, Defendant has had notice of the '386 Patent at least as of the filing of the original Complaint. Defendant's infringing actions have continued after the original Complaint was filed. Defendant had knowledge of the '386 Patent, and of the specific conduct that constitutes infringement of the '386 Patent at least based

on the original Complaint, but has continued to engage in the infringing activity, including by selling, making, configuring, and installing the Accused Products and performing the accused functionality, and by engaging in activities that constitute inducing infringement and contributory infringement as described above.

183. The infringing actions of each Third Party User of the Accused Products are attributable to Defendant. For example, Defendant directs and controls the activities or actions of its partners or others in connection with the Accused Products by contractual agreement or otherwise requiring partners or others to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Defendant further directs and controls the operation of devices executing the Accused Products by programming the software which, when executed by a Third Party User, perform the methods and systems of the asserted claims of the ’386 Patent.

184. Plaintiffs have suffered and continue to suffer damages, including lost profits, as a result of Defendant’s infringement of the ’386 Patent. Defendant is therefore liable to Plaintiffs under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiffs for Defendant’s infringement, but no less than a reasonable royalty.

185. Plaintiffs will continue to suffer irreparable harm unless this Court preliminarily and permanently enjoins Defendant, its agents, employees, representatives, and all others acting in concert with Defendant from infringing the ’386 Patent. On information and belief, Plaintiffs have lost potential customers, business opportunities, and goodwill in the community. Plaintiffs will continue to suffer these harms absent an injunction.

186. Defendant’s infringement of the ’386 Patent is knowing and willful. On information and belief, Defendant had actual knowledge of the ’386 Patent at least by the time Plaintiffs filed

this lawsuit and had constructive knowledge of the ’386 Patent from at least the date Plaintiffs marked its products with the ’386 Patent and/or provided notice of the ’386 Patent on its website.

187. On information and belief, despite Defendant’s knowledge of the Asserted Patents and Plaintiffs’ patented technology, Defendant made the deliberate decision to sell products and services that they knew infringe these patents. Defendant’s continued infringement of the ’386 Patent with knowledge of the ’386 Patent constitutes willful infringement.

188. Plaintiffs’ allegations of infringement, indirect infringement, and willful infringement with respect to this patent are further set forth in Plaintiffs’ Disclosure of Asserted Claims and Preliminary Infringement Contentions Pursuant to the Court’s Standing Order Governing Patent Proceedings served July 12, 2022.

FOURTH CAUSE OF ACTION
(INFRINGEMENT OF THE ’721 PATENT)

189. Plaintiffs reallege and incorporate by reference the allegations of the preceding paragraphs of this Complaint.

190. Forcepoint has infringed and continues to infringe one or more claims of the ’721 patent in violation of 35 U.S.C. § 271 in this judicial district and elsewhere in the United States and will continue to do so unless enjoined by this Court. The Accused Products, including for example Forcepoint’s Advanced Malware Detection Systems and systems that incorporate it, such as Next Generation Firewall solutions, when used for their ordinary and customary purposes, practice each element of at least claim 1 the ’721 Patent as demonstrated below.

191. For example, claim 1 of the ’721 patent recites:

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

receiving, at a first threat server, details of a first computer object from a first remote computer, wherein the details of the first computer object include data uniquely identifying the first computer object;

determining, by the first threat server, whether the first computer object has been previously seen by comparing the data uniquely identifying the first computer object to a plurality of data uniquely identifying plural computer objects in a first database associated with the first threat server;

receiving additional information about the first computer object from the first remote computer when the first computer object has not been previously seen;

storing the details of the first computer object and the received additional information about the first computer object in a second database associated with the first threat server when the first computer object has not been previously seen;

providing contents of the second database to at least one database associated with a central server, wherein the contents comprise a signature of the first computer object, behavior information about the first computer object, and information about the first remote computer;

increasing a count associated with a number of times that the first computer object has been seen, and providing the increased count associated with the number of times that the first computer object has been seen to the central server; and

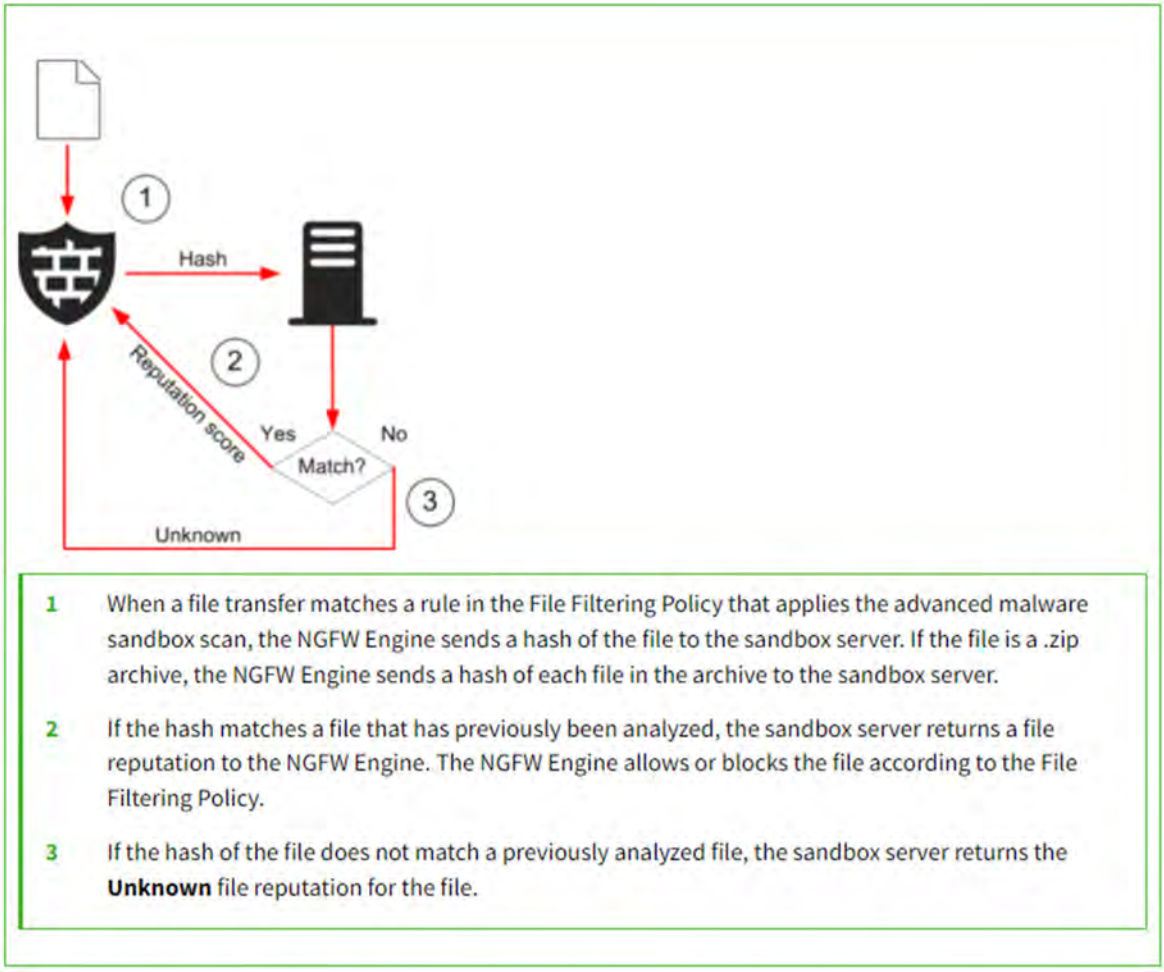
receiving, at a second threat server, at least a portion of the contents of the at least one database associated with the central server, wherein the at least a portion of the contents of the at least one database associated with the central server include a subset of the details of the first computer object stored in the second database.

192. To the extent the preamble is construed as limiting, the Accused Products perform *a method of classifying a computer object as malware*. For example, the Forcepoint Advanced Malware Detection systems and Next Generation Firewall solutions detect "zero-day and other advanced malware."

Forcepoint Advanced Malware Detection (AMD) leverages proven technology to detect zero-day and other advanced malware. Using Deep Content Inspection technology, Forcepoint AMD emulates an entire host, interacting with malware to expose and observe a malicious object's possible actions. These include advanced evasion techniques, O/S or application specific threats, dormant code analysis and even CPU and in-memory activity.

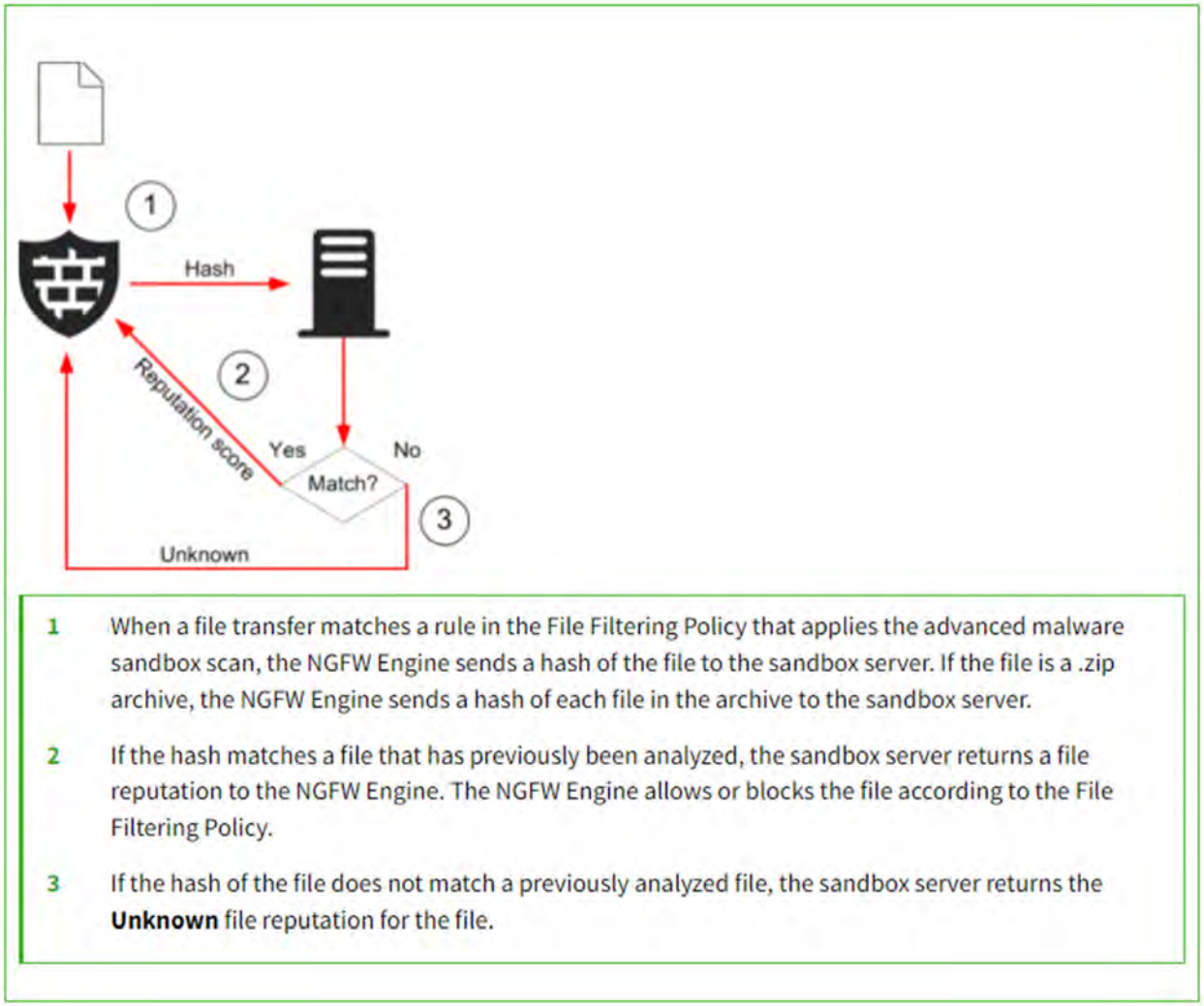
(*See* https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution_brief_forcepoint_advanced_malware_detection_en.pdf; https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.5.0/GUID-DA3B3807-18E1-482B-A5E8-A8BD468E5BB0.html.)

193. The Accused Products perform a method that includes *receiving, at a first threat server, details of a first computer object from a first remote computer, wherein the details of the first computer object include data uniquely identifying the first computer object*. For example, in systems using Forcepoint Advanced Malware Detection and Forcepoint Next Generation Firewall solution, when the Forcepoint File Filtering Policy detects a suspicious file, the Forcepoint NGFW (next generation firewall) Engine sends a hash of the file to a sandbox server; at least the hash uniquely identifies the file object.



(*See* https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.5.0/GUID-EECA15DA-9B8A-4C2F-9C42-D53516ADC19E.html.)

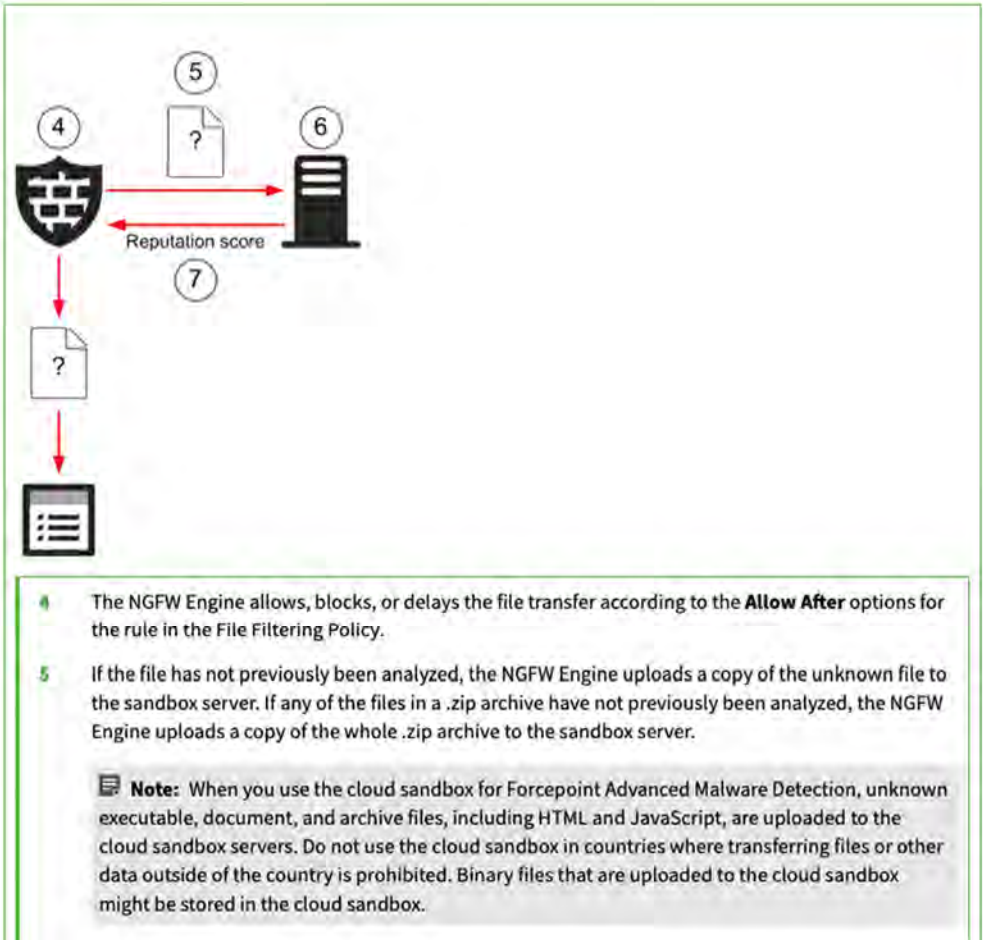
194. The Accused Products perform a method that includes the step of *determining, by the first threat server, whether the first computer object has been previously seen by comparing the data uniquely identifying the first computer object to a plurality of data uniquely identifying plural computer objects in a first database associated with the first threat server*. For example, the sandbox server compares the file hash to a database of hashes of files that have been previously seen to determine whether the hash matches a "previously analyzed file" that has previously been identified.



(*See* https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.5.0/GUID-EECA15DA-9B8A-4C2F-9C42-D53516ADC19E.html.)

195. The Accused Products perform a method that includes *receiving additional information about the first computer object from the first remote computer when the first computer object has not been previously seen* and *storing the details of the first computer object and the received additional information about the first computer object in a second database associated with the first threat server when the first computer object has not been previously seen.* For example, if the suspect file has not been previously analyzed by the sandbox server, the NGFW Engine uploads a copy of the file to the sandbox server. The NFGW Engine also receives

information about whether a file was compressed (.zip) and if so what files/objects were inside the archive, and whether a file object is a "document, [or] archive files, including HTML and JavaScript." Indeed, Forcepoint sends "threat intelligence updates containing the characteristics, behaviors and associated IOCs of every malicious object curated and analyzed within the global service." On information and belief, this information is stored in a database.



(*See* https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.5.0/GUID-EECA15DA-9B8A-4C2F-9C42-D53516ADC19E.html.)

8. If the File Sandbox option was selected:
   a. Forcepoint Advanced Malware Detection for Web updates Forcepoint ThreatSeeker Intelligence with information about the file, the source URL, and the command and control targets.
   b. Forcepoint ThreatSeeker Intelligence updates the Master Database, ACE analytic databases, and other security components, which are then pulled by web protection deployments.
   c. The next time someone tries to browse the site, they and the organization are protected by their Forcepoint Web Security deployment.

(*See* https://www.websense.com/content/support/library/web/v85/web_help/web_help.pdf at page 107.)

196. The Accused Products perform a method that includes *providing contents of the second database to at least one database associated with a central server, wherein the contents comprise a signature of the first computer object, behavior information about the first computer object, and information about the first remote computer*. For example, Forcepoint Advanced Malware Detection and the NGFW send information about the file, the source URL, and the command-and-control targets to Forcepoint ThreatSeeker Intelligence, which updates the Master Database, as well as ACE analytic databases and other security components. Moreover, as discussed above, hashes (signatures) of files are also transmitted. Indeed, Forcepoint sends "threat intelligence updates containing the characteristics, behaviors and associated IOCs of every malicious object curated and analyzed within the global service."

8. If the File Sandbox option was selected:
   a. Forcepoint Advanced Malware Detection for Web updates Forcepoint ThreatSeeker Intelligence with information about the file, the source URL, and the command and control targets.
   b. Forcepoint ThreatSeeker Intelligence updates the Master Database, ACE analytic databases, and other security components, which are then pulled by web protection deployments.
   c. The next time someone tries to browse the site, they and the organization are protected by their Forcepoint Web Security deployment.

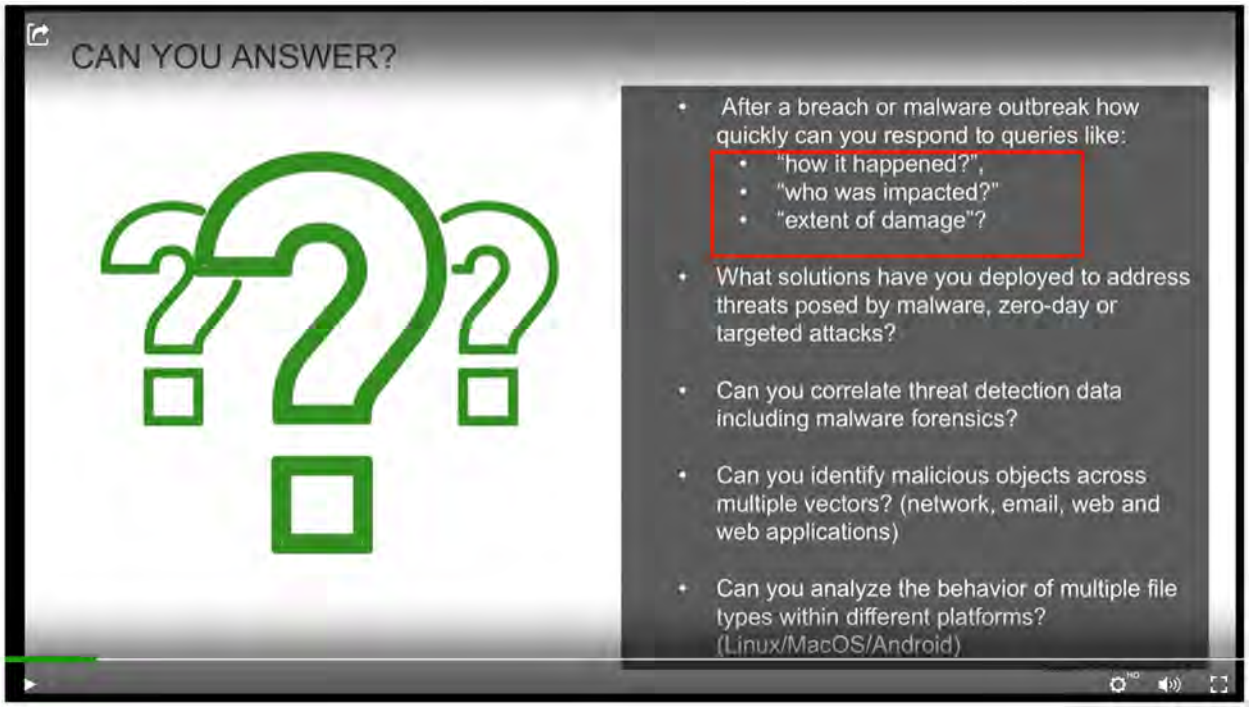
(*See* https://www.websense.com/content/support/library/web/v85/web_help/web_help.pdf at page 107.)

Global Threat Intelligence

Forcepoint sends threat intelligence updates containing the characteristics, behaviors and associated IOCs of every malicious object curated and analyzed within the global service. This allows for faster identification of previously-seen threats, new threats that reuse objects, and streamlines the analysis, detection and response to previously unseen threats.

(*See* https://roi4cio.com/catalog/en/product/forcepoint-advanced-malware-detection.)

197. The Accused Products perform a method that includes *increasing a count associated with a number of times that the first computer object has been seen, and providing the increased count associated with the number of times that the first computer object has been seen to the central server*. On information and belief, Forcepoint Advanced Malware Detection and the NGFW systems keep a count of how many times a malicious file has been encountered. For example, Forcepoint AMD is able to determine how many people have encountered a malicious file across a user group.

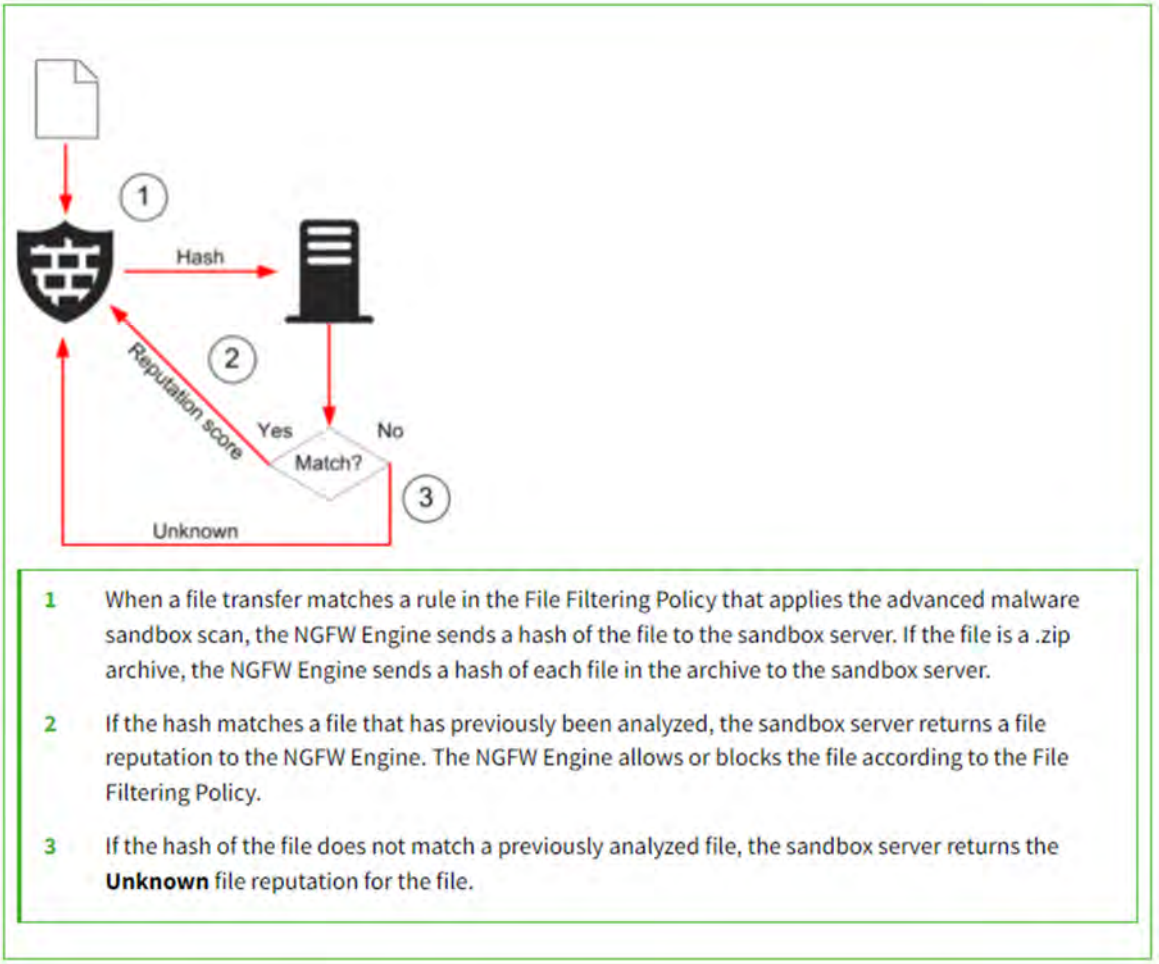


(*See* https://www.forcepoint.com/form/thank-you-your-interest-webcast?form_id=1427&resource=18566&category=webcasts.)

198. In addition, on information and belief, the Accused Products such as Forcepoint Advanced Malware Detection and NGFW solutions log information about network activity and trends are stored in one or more databases. For example, "the Log Database can store trend data to enable presentation reporting on Internet activity trends. When trend reporting is enabled, the ETL database job (see Web protection reporting database jobs, page 452) adds daily trend data to the catalog database, and the trend job runs nightly to store weekly, monthly, and yearly trend information. The Log Database also stores statistical data (like bandwidth and count) for browsers, operating system platforms, and user agent strings to enable application reporting." (*See* https://www.websense.com/content/support/library/web/v85/web_help/.pdf at pages 460-461.)

199. The Accused Products perform a method that includes *receiving, at a second threat server, at least a portion of the contents of the at least one database associated with the central server, wherein the at least a portion of the contents of the at least one database associated with the central server include a subset of the details of the first computer object stored in the second database.* On information and belief, the file's hash and reputation is distributed to all sandbox servers such that the sandboxes can return reputation results. (https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.9.0/GUID-41D85576-7C24-4BA8-8605-710F52790CD8.html ("When the analysis is complete, the sandbox server sends an updated file reputation to the NGFW Engine. The updated file reputation is cached on the NGFW Engine that requested the scan and stored on the sandbox server"); https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.7.0/GUID-EECA15DA-9B8A-4C2F-9C42-D53516ADC19E.html.)

200. This enables, for example, the sandbox server to compare the file hash to a database of hashes of files that have been previously seen to determine whether the hash matches a "previously analyzed file" that has previously been identified. In addition, as explained above, information about the file, uploaded by Forcepoint ThreatSeeker Intelligence to the Master Database, ACE analytics databases, and other security components, are further disseminated to other web protection deployments.



(*See* https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.5.0/GUID-EECA15DA-9B8A-4C2F-9C42-D53516ADC19E.html.)

Limitations of Forcepoint Advanced Malware Detection

There are some limitations when you use Forcepoint Advanced Malware Detection.

Each engine communicates separately with the sandbox service. If different engines detect the same file before an analysis result is stored on the sandbox server, the engines might upload the same file more than once. However, if the hash of the file matches a stored result, the engine does not upload the file again.

(*See* https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.5.0/GUID-222054A1-AAC3-4153-8F8C-47E17903EA70.html.)

8. If the File Sandbox option was selected:
   a. Forcepoint Advanced Malware Detection for Web updates Forcepoint ThreatSeeker Intelligence with information about the file, the source URL, and the command and control targets.
   b. Forcepoint ThreatSeeker Intelligence updates the Master Database, ACE analytic databases, and other security components, which are then pulled by web protection deployments.
   c. The next time someone tries to browse the site, they and the organization are protected by their Forcepoint Web Security deployment.

(*See* https://www.websense.com/content/support/library/web/v85/web_help/web_help.pdf, at page 107.)

Global Threat Intelligence

Your team automatically receives threat intelligence updates containing the malware characteristics, behaviors and associated IOCs of every malicious object curated and analyzed within the global service. This means faster identification of known threats, new threats that reuse objects, and streamlines the analysis, detection and response to previously unknown threats.

(*See* https://www.forcepoint.com/sites/default/files/resources/files/datasheet_advanced_malware_detection_en.pdf.)

201. Each claim in the ’721 Patent recites an independent invention. Neither claim 1, described above, nor any other individual claim is representative of all claims in the ’721 Patent.

202. Forcepoint has been aware of the ’721 Patent since at least the filing of this Complaint. Further, Plaintiffs have marked their products with the ’721 Patent, including on its web site, since at least July 2020.

203. Forcepoint directly infringes at least claim 1 of the ’721 Patent, literally or under the doctrine of equivalents, by performing the steps described above. For example, on information and belief, the Forcepoint performs a method in an infringing manner as described above by running the Accused Products to protect its own computer and network operations. On information and belief, the Forcepoint performs a method in an infringing manner in testing the operation of the Accused Products and corresponding systems. As another example, Forcepoint performs each of the method steps when providing or administering services to third parties, customers, and partners using the Accused Products.

204. Defendant’s directly infringing activities in this District, Texas and elsewhere in the United States include at least making and using the inventions in the claims identified above. Defendant makes and uses the claimed inventions in connection with at least the following activities involving the Accused Products (*e.g.*, software and corresponding systems): (1) operating and maintaining data centers that process information; (2) employing the Accused Products in connection with, and to protect, its own computer and network operations and/or the operations of affiliates, (3) using software that includes the Accused Products, (4) testing the operation of the Accused Products and corresponding systems, (5) collecting and gathering information and data, and (6) providing services to others including installation, maintenance, update, verification, processing, support, and other services.

205. In addition, Defendant directly infringes the above identified claims because it directs and controls the performance of the above-identified claimed inventions by others. As an example, Defendant 1) directs and controls the performance and operation of the Accused Products or components thereof on customers', partners', and end users' devices; and 2) establishes the manner of the operation of the Accused Products. Defendant further directs and controls by means that include contractual agreements such as its End User License Agreements, its continued monitoring, verification and access to the Accused Products, its continued collection of information from the Accused Products, its programming of the software used in the Accused Products, and/or its provision of information, data, and instructions that control the operation of the system (for example, updates to algorithms, malware definitions, and other system and module code). Each of these claimed steps is performed automatically, without the user taking an action or initiating the process.

206. Defendant also actively induces direct infringement of the above identified patent claims by others with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). Defendant's partners, customers, and end users (collectively, "Third Party Users") of the Accused Products and corresponding systems and services directly infringe at least by making and using the inventions in the above-identified claims, for example when they install, operate and use the Accused Products and corresponding systems. Defendant has had knowledge of the asserted patent and of the specific allegations of infringement with respect to the asserted patent at least as of the filing data of the complaint.

207. Defendant encourages and induces Third Party Users with specific intent to use the Accused Products in a manner that infringes the asserted claims of the '721 Patent at least by

offering, selling, licensing and providing the Accused Products which infringe the above-identified claims of the asserted patent when installed and operated by Third Party Users. Defendant further induces infringement of the above-identified claims by Third Party Users by engaging in activities that include: 1) making its security services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users, and 2) marketing, advertising, promotion, installation, support, and distribution activities involving the Accused Products, including its security software and services in the United States; 3) recommending and selling the Accused Products and providing technical support for the installation, implementation, integration, and ongoing operation of the Accused Products; 4) providing installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Products remain operational for each customer through ongoing technical support. *See, e.g.*, https://www.websense.com/content/support/library/amd-op/v10/install/manager-install.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf; https://www.websense.com/content/ support/library/endpoint/v20/installation.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/guide-sase-guide-to-protecting-remote-workers-en_0_0_0_0_0_0_0_0.pdf. For example, Defendant shares instructions, guides, and manuals, which advertise and instruct Third Party Users on how to use the software in a manner that infringes the above-identified claims of the asserted patent *Id.* Defendant further provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that deploy and use the Accused Products in an infringing manner. *Id.*

208. Further, Defendant induces infringement by Third Party Users of the Accused Products by requiring each user to enter into one or more contractual agreements with Defendant including an End User License Agreement, which obligates each customer to perform certain actions in order to use the Accused Products and to use the Accused Products in a manner that infringes the above-identified claims. In order to receive the benefit of Defendant and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the above-identified claims of the asserted patent.

209. Defendant also contributes to the infringement of the Accused Products by Third Party Users by providing within the United States or importing the Accused Products into the United States, which are for use in practicing, and under normal operation practice, the claimed invention in an infringing manner, constituting a material part of the claimed invention, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. For example, the Accused Products and the example functionality described below have no substantial non-infringing uses but are specifically designed to practice the '721 Patent.

210. The Accused Products have no substantial non-infringing uses because the accused functionality, including the database and server architecture and related functionality described herein, is an integral part of the Accused Products and must be performed for the Accused Products to perform their intended purpose. For example, without compiling and correlating threat information about an assessed object from dynamically updated threat databases, the Accused Products could not assess whether to allow those objects to pass through firewalls or subject them to further analysis, triage and final designation as malicious or benign. These processes are continually running when the system is in use and, on information and belief, cannot be removed

or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown above, or without the system and components identified above that practice the '721 patent, that functionality could not be performed.

211. Indeed, the Accused Products have no substantial non-infringing uses because the accused functionality is an integral part of the Accused Products and must be performed for the Accused Products to perform their intended purpose. For example, without compiling and analyzing data about a process running on one or more remote computers, the Accused Products could not timely detect processes (running objects) that pose as threats. *See, e.g., infra*; http://www.websense.com/content/support/library/web/v85/ web_help/web_help.pdf; https://www.websense.com/content/support/library/amd-op/v10/install/manager-install.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/brochure_forcepoint_email_security_en_0.pdf; https://www.websense.com/content/support/library/email/v85/email_help/email_help.pdf; https://www.websense. com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf; https://www.websense.com/content/support/library/casb/ admin_guide/casb_admin_guide.pdf. These processes are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the asserted claims of the '721 Patent, that functionality could not be performed.

212. Additionally, the accused functionality, including the accused threat analysis functionality described below, itself has no substantial non-infringing uses because the

components, modules and methods identified are a necessary part of that functionality. For example, without the appliances that compile data regarding processes run on endpoints protected by the Accused Products, the Accused Products could not assess which processes on any given endpoint constitute a threat. *See, e.g.*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 1275. These process are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the asserted claims of the '721 Patent, that functionality could not be performed. Similarly, without leveraging information from multiple dynamically updated threat databases, the Accused Products could not propagate the latest intelligence about objects to their firewalls, and therefore could not enable their firewalls to make accurate decisions about the malicious or benign nature of assessed objects. These processes are continually running when the system is in use and, on information and belief, cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown above, or without the system and components identified above that practice the '721 patent, that functionality could not be performed.

213. In addition, the products, systems, modules, and methods provided by Defendant constitute a material part of the invention—indeed, they provide all the components, modules, and features that perform the claimed methods and systems. For example, the Accused Products and accused functionalities (including database and server architecture) constitute a material part of the inventions claimed at least because such functionality is integral to the processes identified

above (such as "receiving, at a first threat server, details of a first computer object . . ."; "determining, by the first threat server, whether the first computer object has been previously seen . . ."; "receiving additional information about the first computer object . . ."; "storing the details of the first computer object . . . "; "providing contents of the second database . . ."; "increasing a count associated with a number of times that the first computer object has been seen . . ."; "receiving, at a second threat server, at least a portion of the contents of the at least one database associated with the central server . . . ."), as recited in the claims of the '721 patent. None of these products are staple goods—they are sophisticated and customized cyber security and malware detection products, methods, and systems.

214. Plaintiffs have suffered and continue to suffer damages, including lost profits, as a result of Defendant's infringement of the '721 Patent. Defendant is therefore liable to Plaintiffs under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiffs for Defendant's infringement, but no less than a reasonable royalty.

215. Plaintiffs will continue to suffer irreparable harm unless this Court preliminarily and permanently enjoins Defendant, its agents, employees, representatives, and all others acting in concert with Defendant from infringing the '721 Patent. On information and belief, Plaintiffs have lost potential customers, business opportunities, and goodwill in the community. Plaintiffs will continue to suffer these harms absent an injunction.

216. Defendant's infringement of the '721 Patent is knowing and willful. Defendant acquired actual knowledge of the '721 Patent when Plaintiffs filed this lawsuit and had constructive knowledge of the '721 Patent from at least the date Plaintiffs marked its products with the '721 Patent and/or provided notice of the '721 Patent on its website.

217. On information and belief, despite Defendant's knowledge of the Asserted Patents and Plaintiffs' patented technology, Defendant made the deliberate decision to sell products and services that they knew infringe these patents. Defendant's continued infringement of the '721 Patent with knowledge of the '721 Patent constitutes willful infringement.

218. Plaintiffs' allegations of infringement, indirect infringement, and willful infringement with respect to this patent are further set forth in Plaintiffs' Disclosure of Asserted Claims and Preliminary Infringement Contentions Pursuant to the Court's Standing Order Governing Patent Proceedings served July 12, 2022.

FIFTH CAUSE OF ACTION
(INFRINGEMENT OF THE '928 Patent)

219. Plaintiffs reallege and incorporate by reference the allegations of the preceding paragraphs of this Complaint.

220. Forcepoint has infringed and continues to infringe one or more claims of the '928 Patent in violation of 35 U.S.C. § 271 in this judicial district and elsewhere in the United States and will continue to do so unless enjoined by this Court. The Accused Products, for example Forcepoint's Remote Browser Isolation and related software and services, such as Ericom Shield Zero Trust and Cyberinc's Isla Smart Isolation feature incorporated therein ("Remote Browser" or "RBI") that, when used for their ordinary and customary purposes, practice each element of at least claim 1 of the '928 Patent as demonstrated below.

221. For example, claim 1 of the '928 Patent recites:

1. A content analysis and malware prevention method comprising:

intercepting, by a protection agent, a request from a web browser;

determining, by the protection agent, whether the request is associated with known malicious content;

when the request is associated with known malicious content, blocking, by the protection agent, the request; and

when the request is not associated with known malicious content:

sending, by the protection agent, the request to one or more web servers

receiving, at the protection agent on a remote computer, web content from the one or more web servers, wherein web content comprises data for assembling a web page, and wherein the web content is received in response to the request;

identifying, by the protection agent, a malware threat within the web content;

in response to identifying the malware threat, modifying, by the protection agent, the web content, wherein modifying the web content comprises modifying the web content at protocol level to remove the malware threat from the web content;

assembling, by the protection agent, a modified version of the web page, wherein the modified web page is assembled using the modified web content, and wherein the modified version of the web page does not comprise the malware threat; and

providing, by the protection agent, the modified version of the web page to the web browser application on the remote computer for rendering and display.

222. To the extent the preamble is construed to be limiting, the Accused Products perform *a content analysis and malware prevention method*. For example, Forcepoint's RBI "gives us two modes necessary to deal with all the ultra-risky sites as well as the unknown or potentially risky ones": "Secure Streaming "and "Secure Rendering." RBI "automatically adjust[s] between the two rendering modes based on potential risk or verified trust of the page and associated content." The Accused Products thereby "provide[] malware protection against ransomware" as well as "zero-day threats." Indeed, "[i]f there is any malware on the site, this process strips it from what is passed to the user, allowing them a safe browsing experience even on a malicious site."

That's where Remote Browser Isolation (RBI) comes in. RBI enables organizations to take a Zero Trust approach to web access. With RBI, web traffic is automatically redirected through an isolated browser in a secure container. There, potentially risky web content can be rendered, with the results sent to the user as a safe stream of pixels, providing a familiar, native browser experience. If there is any malware on the site, this process strips it from what is passed to the user, allowing them a safe browsing experience even on a malicious site. This is the best kind of protection: the security tool itself is almost invisible to the user but delivers the highest possible security efficacy.

(*See* https://www.forcepoint.com/blog/insights/remote-browser-isolation-higher-level-web-security at page 01.)

Smart Isolation

The hard part about web security is knowing which types of pages are more like a high contact sport and which are less risky, after all, that is what the hackers are doing every day; working to make dangerous sites look just like normal, safe ones.

> Forcepoint created Smart Isolation, to automatically adjust between the two rendering modes based on potential risk

RBI gives us two modes necessary to deal with all the ultra-risky sites as well as the unknown or potentially risky ones but choosing when to use which mode for each combination of web category and user group can cause anxiety in its own right. This is why Forcepoint created Smart Isolation, to automatically adjust between the two rendering modes based on potential risk or verified trust of the page and associated content.

(*See* https://www.sourcesecurity.com/news/forcepoint-launches-industry-smart-remote-browser-co-1642591598-ga-npr.1642592193.html?ref=nav at page 02.)

Table of Feature

FEATURE	VALUE
Isolation Modes	
Secure Streaming	Secure Streaming provides the highest assurance web security technology available
Secure Rendering	Secure Rendering enables a seamless native browsing experience
Advanced RBI	
Smart Isolation	Smart Isolation intelligently and automatically optimizes RBI delivery to balance performance and security by switching between the two isolation modes based on a given destination's risk
Smart Redirection	Smart Redirection automatically determines which links should remain in an RBI session and which should be redirected to Forcepoint SWG for policy enforcement
Privacy	
Data Sovereignty	Data sovereignty controls give customers control over the duration PII is retained, with the ability to wipe PII data for users on request.
File Download Security	
Zero Trust Content Disarm & Reconstruction (CDR)	Zero Trust CDR automatically sanitizes file downloads by stripping active content and rebuilding files safely from scratch using only benign elements and maintains the original file format
Antivirus Scanning	AV scanning engines scan downloaded files for malware signatures to block malicious files
Export to PDF	Convert downloaded files to PDF to neutralize executable code for file types not supported by Forcepoint Zero Trust CDR

(*See* https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-remote-browser-isolation-en_0_0_0_0_0_0_0_0_0_0_0_0_0_0.pdf.)

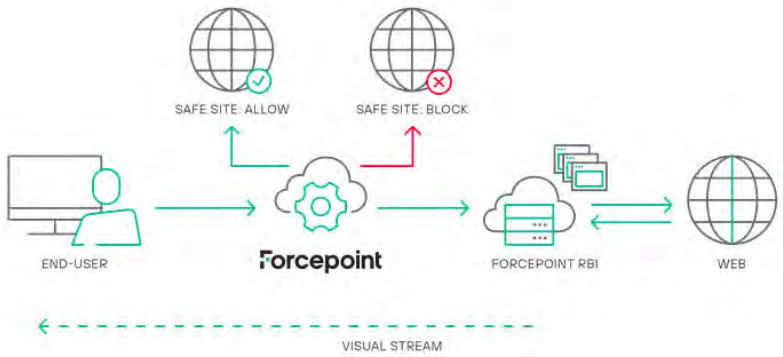
Automatically Balance Security and Performance

Forcepoint RBI makes it easy to provide an end-use experience so seamless you won't even notice a difference to native performance, while also enforcing the highest assurance web security technology whenever there is potential risk. RBI provides a Zero Trust approach to web browsing by neutralizing malware through remote isolation without relying on detection. This not only provides malware protection against ransomware, but zero-day threats as well. Forcepoint RBI also utilizes Zero Trust Content Disarm and Reconstruction (CDR) to provide automatic file sanitization for files downloaded during an RBI session.

(*See* https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-remote-browser-isolation-en_0_0_0_0_0_0_0_0_0_0_0_0_0_0.pdf at page 01.)

223. The Accused Products perform a method that includes *intercepting, by a protection agent, a request from a web browser*. For example, "[w]ith RBI, web traffic is automatically redirected through an isolated browser in a secure container." (*See*

https://www.forcepoint.com/blog/insights/remote-browser-isolation-higher-level-web-security at page 01.)

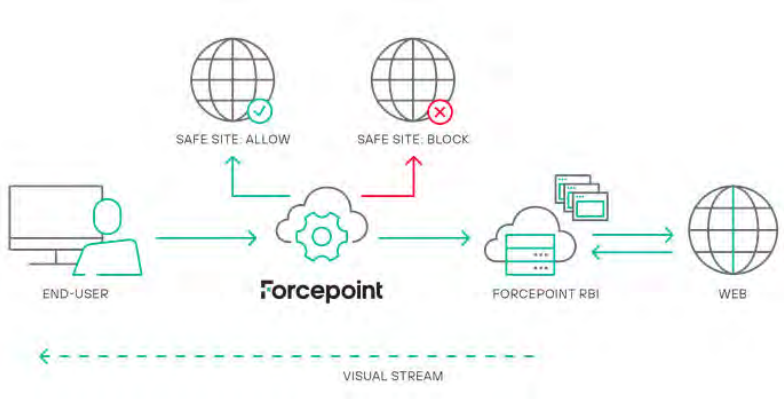


(*See* https://www.forcepoint.com/cyberinc.)

224. "Download interception prevents drive-by downloads that push malware onto endpoints." (*See* https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-remote-browser-isolation-en_0_0_0_0_0_0_0_0_0_0_0_0_0_0.pdf at page 02.)

225. The Accused Products perform a method that includes *determining, by the protection agent, whether the request is associated with known malicious content* and *when the request is associated with known malicious content, blocking, by the protection agent, the request*. As one example, the Accused Products utilize risk-based isolation (*e.g.*, using Threat Intelligence) to determine if websites should be trusted, isolated, blocked, etc. As another example, the Accused Products are "commonly used when dealing with websites that aren't known to be either 'good' (and can typically be safely allowed) or 'bad' (and are usually blocked), allowing such uncategorized sites to be used without fear." RBI can "detect[] a potential security threat" such as a "website built by hackers with malicious intent" and "block[] access" to protect "your organization and you from malware infections and other harm." On information and belief, each request is analyzed to determine "good" (allow) and "bad: (block). Indeed, Forcepoint "analyzes millions of websites every hour, putting them into more than 90 categories in over 50 languages

and inspecting them for malicious content." If "you've been denied access to a website because of Forcepoint, your organization is using our technology to apply its Internet use policy."



(*See* https://www.forcepoint.com/cyberinc.)

Selective Isolation Enables a Seamless, Progressive Rollout

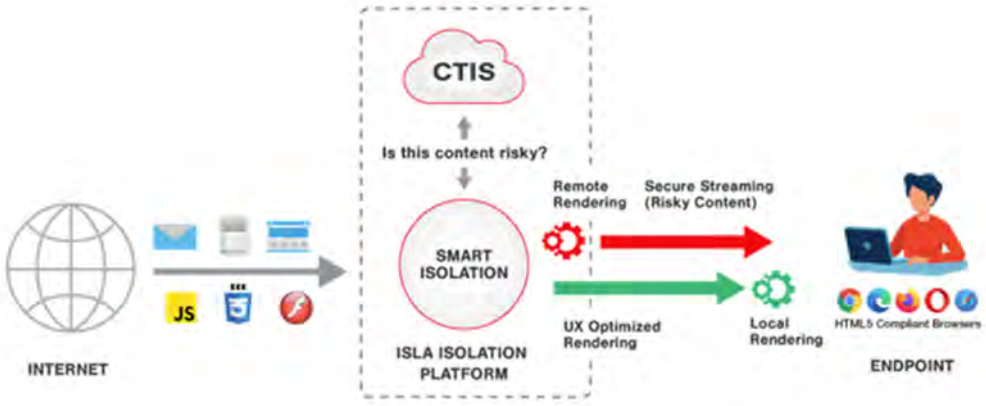
Isla 6 includes new Selective Isolation capabilities which allow for a fast, simple deployment by offering easy administrative control over what traffic needs to be isolated. Traffic segmentation can be based on the category of traffic, risk levels and aligned with the organization's risk appetite and policies. For example, users attempting to visit an online gambling or other suspicious site can be blocked or redirected to the site through an isolated browser to ensure nothing malicious can be downloaded or sent to the device.

(*See* https://www.businesswire.com/news/home/20210218005375/en/Cyberinc-Introduces-Isla-Isolation-Platform-6-With-Smart-Isolation-Setting-a-New-Standard-in-Security-and-a-Seamless-User-Experience.)

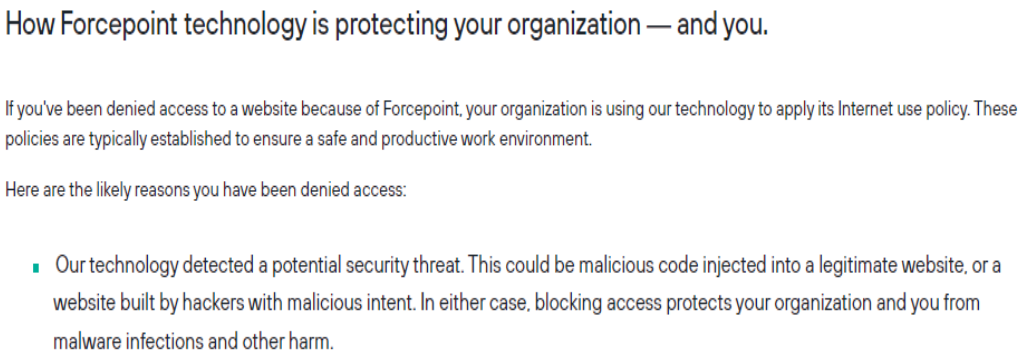


(*See* https://www.forcepoint.com/form/thank-you-your-interest-webcast?form_id=1427&resource=36739&category=webcasts.)

Remote browser isolation, or RBI for short, has dramatically changed the face of web security. It's most often used in secure web gateways, such as our own Forcepoint **Cloud Security Gateway**, to enable people to visit websites safely, even ones that have been compromised by malicious code or content. RBI is commonly used when dealing with websites that aren't known to be either "good" (and can typically be safely allowed) or "bad" (and are usually blocked), allowing such uncategorized sites to be used without fear. But that's just the beginning of what it can do.



(*See* https://www.forcepoint.com/blog/insights/forcepoint-acquires-cyberinc-rbi at page 02.)



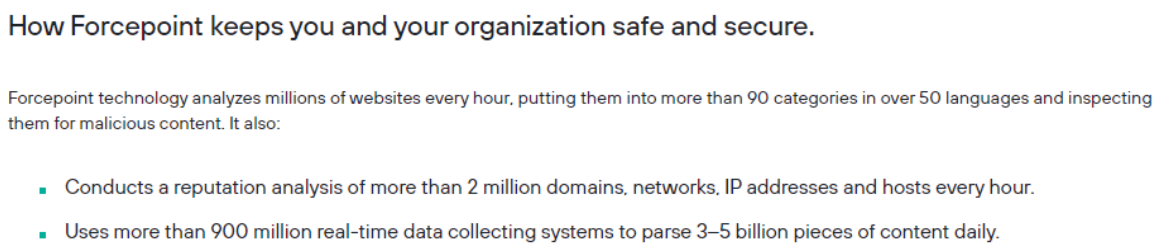
How Forcepoint technology is protecting your organization — and you.

If you've been denied access to a website because of Forcepoint, your organization is using our technology to apply its Internet use policy. These policies are typically established to ensure a safe and productive work environment.

Here are the likely reasons you have been denied access:

- Our technology detected a potential security threat. This could be malicious code injected into a legitimate website, or a website built by hackers with malicious intent. In either case, blocking access protects your organization and you from malware infections and other harm.

(*See* https://www.forcepoint.com/company/blocked-by-forcepoint at page 01.)



How Forcepoint keeps you and your organization safe and secure.

Forcepoint technology analyzes millions of websites every hour, putting them into more than 90 categories in over 50 languages and inspecting them for malicious content. It also:

- Conducts a reputation analysis of more than 2 million domains, networks, IP addresses and hosts every hour.
- Uses more than 900 million real-time data collecting systems to parse 3–5 billion pieces of content daily.

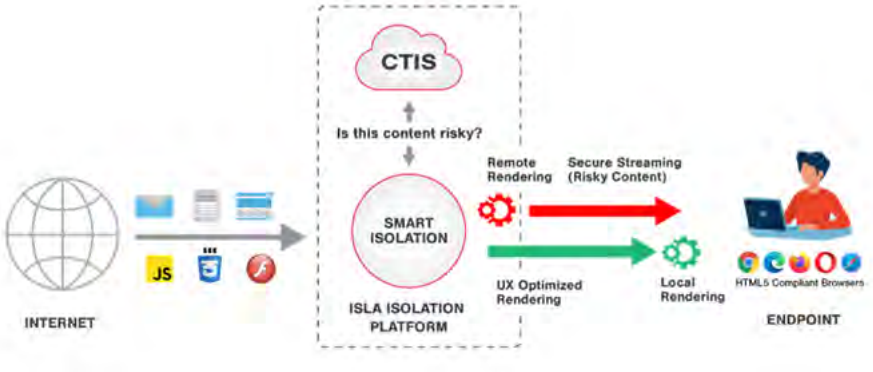
(*See* https://www.forcepoint.com/company/blocked-by-forcepoint at page 01.)

226. The Accused Products perform a method that includes *when the request is not associated with known malicious content: sending, by the protection agent, the request to one or more web servers* and *receiving, at the protection agent on a remote computer, web content from the one or more web servers, wherein web content comprises data for assembling a web page, and wherein the web content is received in response to the request*. As one example "Smart Isolation fetches…all pages remotely." (https://staging.cyberinc.com/wp-content/uploads/2021/03/isla-smart-isolation-datasheet-1.pdf.) As explained above, "[w]ith RBI, web traffic is automatically redirected through an isolated browser in a secure container. There, potentially risky web content can be rendered with the results sent to the user as a safe stream of pixels, providing a familiar, native browser experience." (*See* https://www.forcepoint.com/blog/insights/remote-browser-isolation-higher-level-web-security at page 01.)

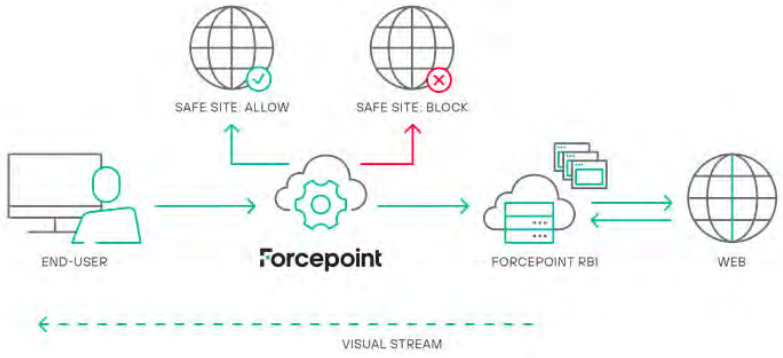
227. In addition, Forcepoint's RBI "adapts the browsing experience with dynamic risk assessment, powered by Cyberinc Threat Intelligence Service, to remotely fetch and execute web

pages and safely render them according to risk levels." RBI "adapts web rendering according to the risk levels of the page or web element with two complementary approaches to rendering" including "Secure Streaming model" that "renders elements remotely and securely streams harmless pixels to the endpoint to offer the strongest possible security" and "UX Optimized model" that "intelligently renders harmful pages and web elements remotely while rendering the less harmful pages and elements locally to balance native user experience and security." (*See* https://www.businesswire.com/news/home/20210218005375/en/Cyberinc-Introduces-Isla-Isolation-Platform-6-With-Smart-Isolation-Setting-a-New-Standard-in-Security-and-a-Seamless-User-Experience.)

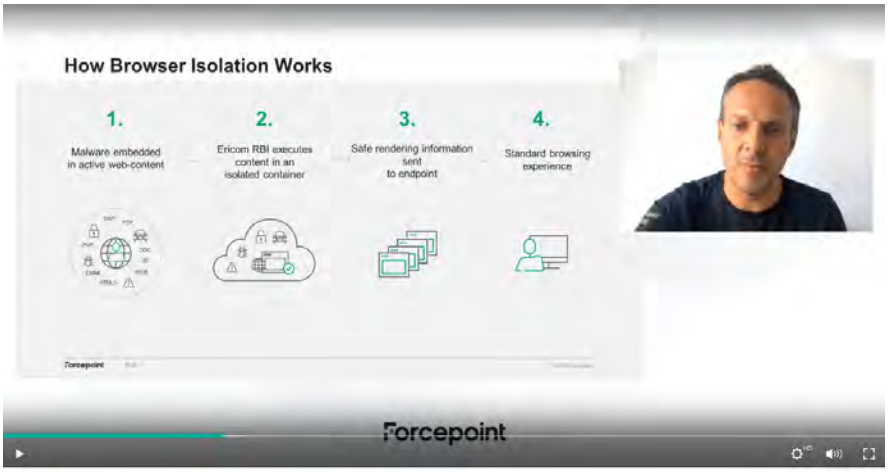
Today, we announced that Forcepoint has acquired Cyberinc, a true innovator in the remote browser isolation industry. Their unique Isla Smart Isolation technology is context-aware, enabling the isolation to be dynamically adapted according to the risk associated with each page—and even elements within the page. When combined with the granular control they give administrators, Isla enables businesses and government agencies to minimize risk from ransomware, malware, and other malicious code while preserving browsing performance and user productivity.



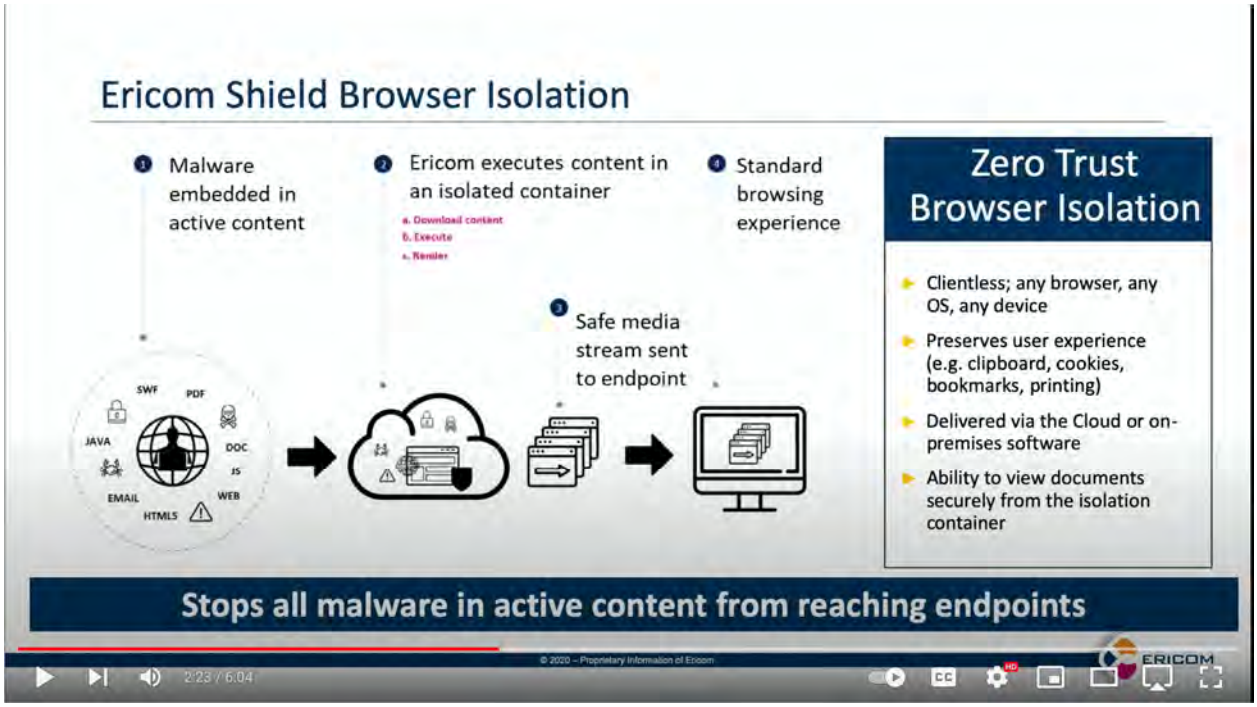
(*See* https://www.forcepoint.com/blog/insights/forcepoint-acquires-cyberinc-rbi.)

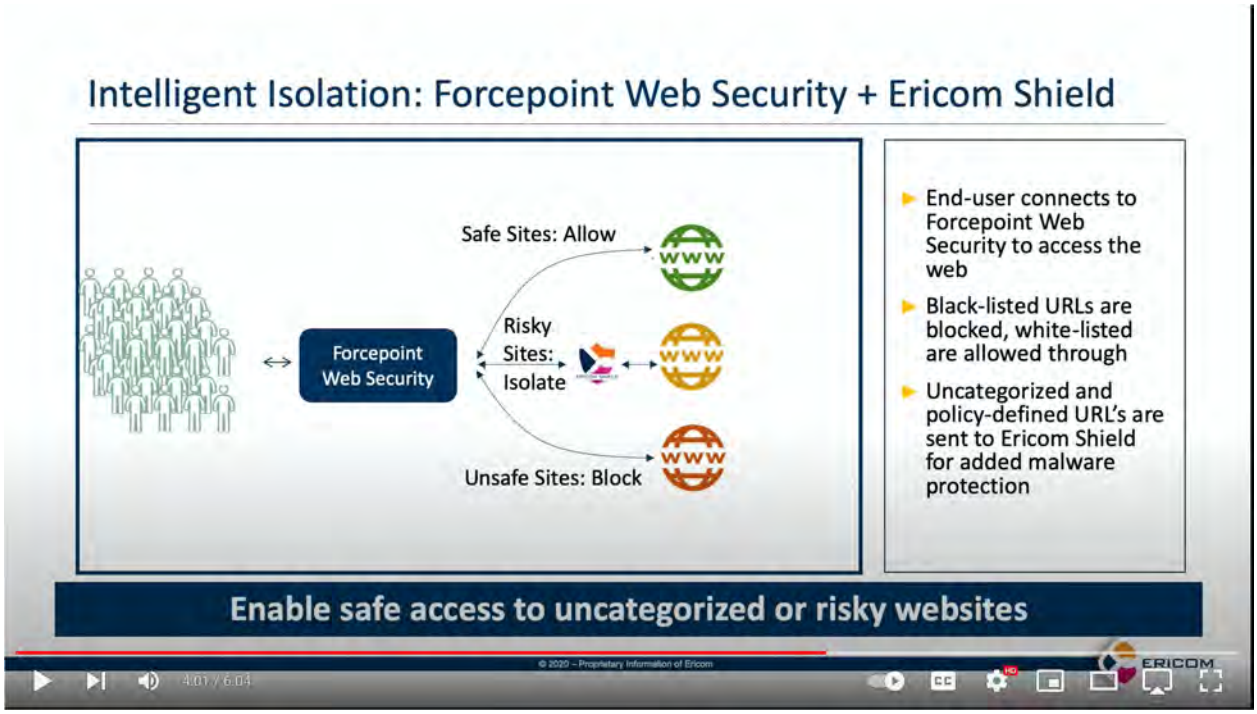


(*See* https://www.forcepoint.com/cyberinc.)



(*See* https://www.forcepoint.com/form/thank-you-your-interest-webcast?form_id=1427&resource=36739&category=webcasts.)

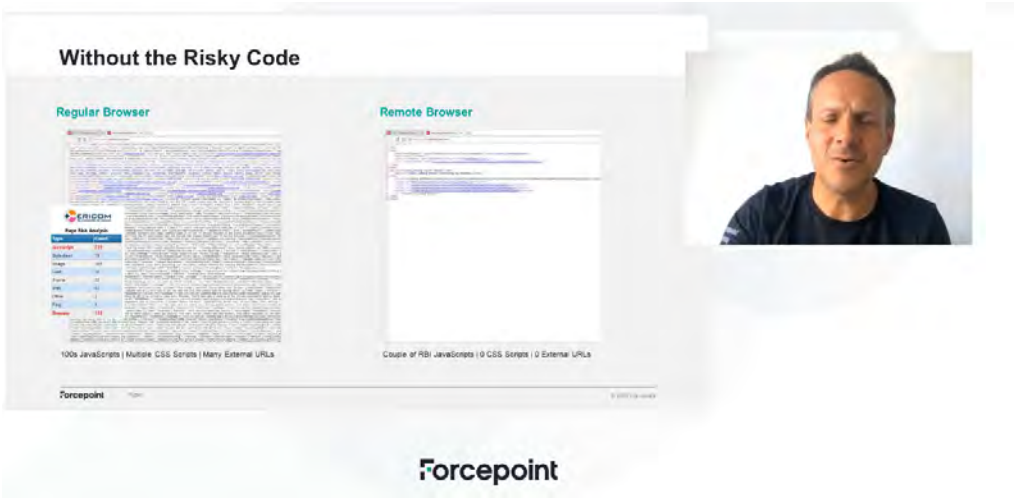




(*See* https://www.youtube.com/watch?v=1R0J2LPT1R8.)

228. The Accused Products perform a method that includes *when the request is not associated with known malicious content: . . . identifying, by the protection agent, a malware threat*

within the web content and *in response to identifying the malware threat, modifying, by the protection agent, the web content, wherein modifying the web content comprises modifying the web content at protocol level to remove the malware threat from the web content.* As explained above, RBI detects "a potential security threat" such as "malicious code injected into a legitimate website." (*See* https://www.forcepoint.com/company/blocked-by-forcepoint at page 01.) The Accused Products also include "Smart Isolation technology," which is "context-aware, enabling the isolation to be dynamically adapted according to the risk associated with each page—and even elements within the page." For example, RBI "adapts web rendering according to the risk levels of the page or web element with two complementary approaches to rendering" including "Secure Streaming model" that "renders elements remotely and securely streams harmless pixels to the endpoint to offer the strongest possible security" and "UX Optimized model" that "intelligently renders harmful pages and web elements remotely while rendering the less harmful pages and elements locally to balance native user experience and security." (*See* https://www.businesswire.com/news/home/20210218005375/en/Cyberinc-Introduces-Isla-Isolation-Platform-6-With-Smart-Isolation-Setting-a-New-Standard-in-Security-and-a-Seamless-User-Experience.) The Accused Products modify the web content, such as by removing "Javascripts," "CSS scripts," and "URLs."



(*See* https://www.forcepoint.com/form/thank-you-your-interest-webcast?form_id=1427&resource=36739&category=webcasts.)

229. Indeed, "[i]f there is any malware on the site, this process strips it from what is passed to the user, allowing them a safe browsing experience even on a malicious site."

> That's where Remote Browser Isolation (RBI) comes in. RBI enables organizations to take a Zero Trust approach to web access. With RBI, web traffic is automatically redirected through an isolated browser in a secure container. There, potentially risky web content can be rendered, with the results sent to the user as a safe stream of pixels, providing a familiar, native browser experience. If there is any malware on the site, this process strips it from what is passed to the user, allowing them a safe browsing experience even on a malicious site. This is the best kind of protection: the security tool itself is almost invisible to the user but delivers the highest possible security efficacy.

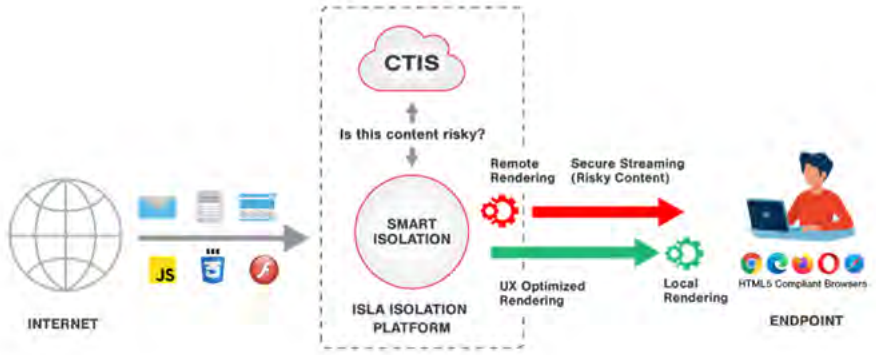
(*See* https://www.forcepoint.com/blog/insights/remote-browser-isolation-higher-level-web-security at page 01.)

230. The Accused Products perform a method that includes *when the request is not associated with known malicious content: . . . assembling, by the protection agent, a modified version of the web page, wherein the modified web page is assembled using the modified web content, and wherein the modified version of the web page does not comprise the malware threat* and *providing, by the protection agent, the modified version of the web page to the web browser*

application on the remote computer for rendering and display. As explained above, "[w]ith RBI, web traffic is automatically redirected through an isolated browser in a secure container. There, potentially risky web content can be rendered with the results sent to the user as a safe stream of pixels, providing a familiar, native browser experience." (*See* https://www.forcepoint.com/blog/insights/remote-browser-isolation-higher-level-web-security at page 01.)

231. In addition, Forcepoint's RBI "adapts the browsing experience with dynamic risk assessment, powered by Cyberinc Threat Intelligence Service, to remotely fetch and execute web pages and safely render them according to risk levels." RBI "adapts web rendering according to the risk levels of the page or web element with two complementary approaches to rendering" including "Secure Streaming model" that "renders elements remotely and securely streams harmless pixels to the endpoint to offer the strongest possible security" and "UX Optimized model" that "intelligently renders harmful pages and web elements remotely while rendering the less harmful pages and elements locally to balance native user experience and security." (*See* https://www.businesswire.com/news/home/20210218005375/en/Cyberinc-Introduces-Isla-Isolation-Platform-6-With-Smart-Isolation-Setting-a-New-Standard-in-Security-and-a-Seamless-User-Experience.)

Today, we announced that Forcepoint has acquired Cyberinc, a true innovator in the remote browser isolation industry. Their unique Isla Smart Isolation technology is context-aware, enabling the isolation to be dynamically adapted according to the risk associated with each page—and even elements within the page. When combined with the granular control they give administrators, Isla enables businesses and government agencies to minimize risk from ransomware, malware, and other malicious code while preserving browsing performance and user productivity.



(*See* https://www.forcepoint.com/blog/insights/forcepoint-acquires-cyberinc-rbi.)

232. Moreover, "[i]f there is any malware on the site, this process strips it from what is passed to the user, allowing them a safe browsing experience even on a malicious site." "Safe rendering information representing the website is sent from [the RBI] to the endpoint browser, providing a fully interactive user experience" that "ensures no active web content—including malware—every reaches the endpoint."

That's where Remote Browser Isolation (RBI) comes in. RBI enables organizations to take a Zero Trust approach to web access. With RBI, web traffic is automatically redirected through an isolated browser in a secure container. There, potentially risky web content can be rendered, with the results sent to the user as a safe stream of pixels, providing a familiar, native browser experience. If there is any malware on the site, this process strips it from what is passed to the user, allowing them a safe browsing experience even on a malicious site. This is the best kind of protection: the security tool itself is almost invisible to the user but delivers the highest possible security efficacy.

(*See* https://www.forcepoint.com/blog/insights/remote-browser-isolation-higher-level-web-security at page 01.)

> With the integration of RBI, a user who connects to a Forcepoint gateway and engages with risky uncategorized sites, social media, or embedded URLs in e-mails is forwarded to Ericom Shield for isolation. Shield renders all website content in a virtual remote browser. Safe rendering information representing the website is sent from Shield to the endpoint browser, providing a fully interactive user experience. Since this approach ensures that no active web content – including malware — ever reaches the endpoint, Shield stops 100% of the malware present on any isolated page. In addition, to prevent phishing, sites launched from URLs in emails are rendered in read-only mode so credentials cannot be entered.

(*See* https://blog.ericom.com/Browser-Isolation-as-a-Key-Part-of-a-Blueprint-for-SASE-Success/ at page 02.)

233. Each claim in the ’928 Patent recites an independent invention. Neither claim 1, described above, nor any other individual claim is representative of all claims in the ’928 Patent.

234. Forcepoint has been aware of the ’928 Patent since at least the filing of this Complaint. Further, Plaintiffs have marked their products with the ’928 Patent, including on its web site, since at least July 2020.

235. Forcepoint directly infringes at least claim 1 of the ’928 Patent, literally or under the doctrine of equivalents, by performing the steps described above. For example, on information and belief, the Forcepoint performs a method in an infringing manner as described above by running the Accused Products to protect its own computer and network operations. On information and belief, the Forcepoint performs a method in an infringing manner when testing the operation of the Accused Products and corresponding systems. As another example, Forcepoint performs the claimed method when providing or administering services to third parties, customers, and partners using the Accused Products.

236. Defendant’s directly infringing activities in this District, Texas and elsewhere in the United States include at least making and using the inventions in the claims identified above.

Defendant makes and uses the claimed inventions in connection with at least the following activities involving the Accused Products (*e.g.*, software and corresponding systems): (1) operating and maintaining data centers that process information; (2) employing the Accused Products in connection with, and to protect, its own computer and network operations and/or the operations of affiliates, (3) using software that includes the Accused Products, (4) testing the operation of the Accused Products and corresponding systems, (5) collecting and gathering information and data, and (6) providing services to others including installation, maintenance, update, verification, processing, support, and other services.

237. In addition, Defendant directly infringes the above identified claims because it directs and controls the performance of the above-identified claimed inventions by others. As an example, Defendant 1) directs and controls the performance and operation of the Accused Products or components thereof on customers', partners', and end users' devices; and 2) establishes the manner of the operation of the Accused Products. Defendant further directs and controls by means that include contractual agreements such as its End User License Agreements, its continued monitoring, verification and access to the Accused Products, its continued collection of information from the Accused Products, its programming of the software used in the Accused Products, and/or its provision of information, data, and instructions that control the operation of the system (for example, updates to algorithms, malware definitions, and other system and module code). Each of these claimed steps is performed automatically, without the user taking an action or initiating the process.

238. Defendant also actively induces direct infringement of the above identified patent claims by others with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). Defendant's

partners, customers, and end users (collectively, "Third Party Users") of the Accused Products and corresponding systems and services directly infringe at least by making and using the inventions in the above-identified claims, for example when they install, operate and use the Accused Products and corresponding systems. Defendant has had knowledge of the asserted patent and of the specific allegations of infringement with respect to the asserted patent at least as of the filing data of the complaint.

239. Defendant encourages and induces Third Party Users with specific intent to use the Accused Products in a manner that infringes the asserted claims of the '928 Patent at least by offering, selling, licensing and providing the Accused Products which infringe the above-identified claims of the asserted patent when installed and operated by Third Party Users. Defendant further induces infringement of the above-identified claims by Third Party Users by engaging in activities that include: 1) making its security services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users, and 2) marketing, advertising, promotion, installation, support, and distribution activities involving the Accused Products, including its security software and services in the United States; 3) recommending and selling the Accused Products and providing technical support for the installation, implementation, integration, and ongoing operation of the Accused Products; 4) providing installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Products remain operational for each customer through ongoing technical support. *See, e.g.*, https://www.websense.com/content/support/library/web/hosted/admin_guide/rbi_support.aspx; https://www.forcepoint.com/sites/ default/files/resources/solution_brief/solution-brief-secure-web-gateway-en_0_0_0_0_0_0_0_0.pdf; https://www.websense.com/

content/support/library/endpoint/v20/installation.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/guide-sase-guide-to-protecting-remote-workers-en_0_0_0_0_0_0_0_0.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/brochure-zero-trust-cdr-en_0_0_0_0_0_0.pdf; https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-remote-browser-isolation-en_0_0_0_0_0_0_0_0_0_0_0_0_0_0.pdf. For example, Defendant shares instructions, guides, and manuals, which advertise and instruct Third Party Users on how to use the software in a manner that infringes the above-identified claims of the asserted patent *Id*. Defendant further provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that deploy and use the Accused Products in an infringing manner. *Id*.

240. Further, Defendant induces infringement by Third Party Users of the Accused Products by requiring each user to enter into one or more contractual agreements with Defendant including an End User License Agreement, which obligates each customer to perform certain actions in order to use the Accused Products and to use the Accused Products in a manner that infringes the above-identified claims. In order to receive the benefit of Defendant and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the above-identified claims of the asserted patent.

241. Defendant also contributes to the infringement of the Accused Products by Third Party Users by providing within the United States or importing the Accused Products into the

United States, which are for use in practicing, and under normal operation practice, the claimed invention in an infringing manner, constituting a material part of the claimed invention, and not a staple article or commodity of commerce suitable for substantial non-infringing uses. For example, the Accused Products and the example functionality described below have no substantial non-infringing uses but are specifically designed to practice the '928 Patent.

242. The Accused Products have no substantial non-infringing uses because the accused functionality is an integral part of the Accused Products and must be performed for the Accused Products to perform their intended purpose. For example, without compiling and analyzing data about a process running on one or more remote computers, the Accused Products could not timely detect processes (running objects) that pose as threats. *See, e.g.*, https://www.websense.com/content/support/library/web/hosted/ admin_guide/rbi_support.aspx; https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-secure-web-gateway-en_0_0_0_0_0_0_0_0.pdf; https://www.websense.com/content/support/library/endpoint/v20/installation.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/guide-sase-guide-to-protecting-remote-workers-en_0_0_0_0_0_0_0_0.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/brochure-zero-trust-cdr-en_0_0_0_0_0_0.pdf; https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-remote-browser-isolation-en_0_0_0_0_0_0_0_ 0_0_0_0_0_0_0.pdf. These processes are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and

components identified below that practice the asserted claims of the ’928 Patent, that functionality could not be performed.

243. The accused functionality, including the accused threat analysis functionality described below, itself has no substantial non-infringing uses because the components, modules and methods identified are a necessary part of that functionality. For example, without the appliances and systems analyzing Internet resources requested by endpoints protected by the Accused Products, the Accused Products could not assess which processes on any given endpoint constitute a threat. *See, e.g.*, https://www.websense.com/content/support/library/web/hosted/admin_guide/rbi_support.aspx; https://www.forcepoint.com/sites/ default/files/resources/solution_brief/solution-brief-secure-web-gateway-en_0_0_0_0_0_0_0_0.pdf; https://www.websense.com/content/support/library/endpoint/v20/installation.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/guide-sase-guide-to-protecting-remote-workers-en_0_0_0_0_0_0_0_0.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/ brochure-zero-trust-cdr-en_0_0_0_0_0_0.pdf; https://www.forcepoint.com/sites/default/files/resources/solution_brief/solution-brief-remote-browser-isolation-en_0_0_0_0_0_0_0_0_0_0_0_0_0_0.pdf. These process are continually running when the system is in use and cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the asserted claims of the ’928 Patent, that functionality could not be performed.

244. In addition, the products, systems, modules, and methods provided by Defendant constitute a material part of the invention—indeed, it provides all the components, modules, and features that perform the claimed methods and systems. For example, the Accused Products and corresponding functionality constitute a material part of the inventions claimed because such analysis is integral to the processes identified above (such as "*determining, by the protection agent, whether the request is associated with known malicious content*"), as required by the asserted claims of the '928 Patent. None of these products are staple goods—they are sophisticated and customized cyber security and malware detection products and systems.

245. Plaintiffs have suffered and continue to suffer damages, including lost profits, as a result of Defendant's infringement of the '928 Patent. Defendant is therefore liable to Plaintiffs under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiffs for Defendant's infringement, but no less than a reasonable royalty.

246. Plaintiffs will continue to suffer irreparable harm unless this Court preliminarily and permanently enjoins Defendant, its agents, employees, representatives, and all others acting in concert with Defendant from infringing the '928 Patent. On information and belief, Plaintiffs have lost potential customers, business opportunities, and goodwill in the community. Plaintiffs will continue to suffer these harms absent an injunction.

247. Defendant's infringement of the '928 Patent is knowing and willful. On information and belief, Defendant had actual knowledge of the '928 Patent at least by the time Plaintiffs filed this lawsuit and had constructive knowledge of the '928 Patent from at least the date Plaintiffs marked its products with the '928 Patent and/or provided notice of the '928 Patent on its website.

248. On information and belief, despite Defendant's knowledge of the Asserted Patents and Plaintiffs' patented technology, Defendant made the deliberate decision to sell products and

services that they knew infringe these patents. Defendant's continued infringement of the '928 Patent with knowledge of the '928 Patent constitutes willful infringement.

249. Plaintiffs' allegations of infringement, indirect infringement, and willful infringement with respect to this patent are further set forth in Plaintiffs' Disclosure of Asserted Claims and Preliminary Infringement Contentions Pursuant to the Court's Standing Order Governing Patent Proceedings served July 12, 2022.

SIXTH CAUSE OF ACTION
(INFRINGEMENT OF THE '869 PATENT)

250. Plaintiffs reallege and incorporate by reference the allegations of the preceding paragraphs of this Amended Complaint.

251. Forcepoint has infringed and continues to infringe one or more claims of the '869 Patent in violation of 35 U.S.C. § 271 in this judicial District and elsewhere in the United States and will continue to do so unless enjoined by this Court. The Accused Products, including features of the Forcepoint Advanced Malware Detection Appliance (as well as any other products that include the features described below, including, but not limited to, Forcepoint's Cloud Access Security Broker ("CASB"), Next-Gen Firewall ("NGFW"), Web Security, and Email Security products), when used for their ordinary and customary purposes, practice each element of at least claim 46 of the '869 Patent, as demonstrated below.

252. Claim 46 of the '869 Patent recites:

> 46. A system comprising:
>
> at least one memory;
>
> at least one processor configured to perform operation of:
>
> identifying static data points that may be indicative of either a harmful or benign executable file; and

associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files; and

at least one processor configured to perform operation of:

identifying an executable file to be evaluated;

extracting a plurality of static data points from the executable file;

generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.

253. The Accused Products include each component of the system of claim 46 of the '869 Patent. To the extent the preamble is construed to be limiting, the Accused Products include *a system*, as further explained below. For example, the Forcepoint Advanced Malware Detection Appliance "is an on-premises, automated malware analysis framework developed for organizations needing to add detection and prevention against stealthy and advanced threats to their existing Forcepoint Web and Email Security solutions" including processing "files through seven distinct static analytic agents and a dual-sandboxing process." (Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4* at 2, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf).

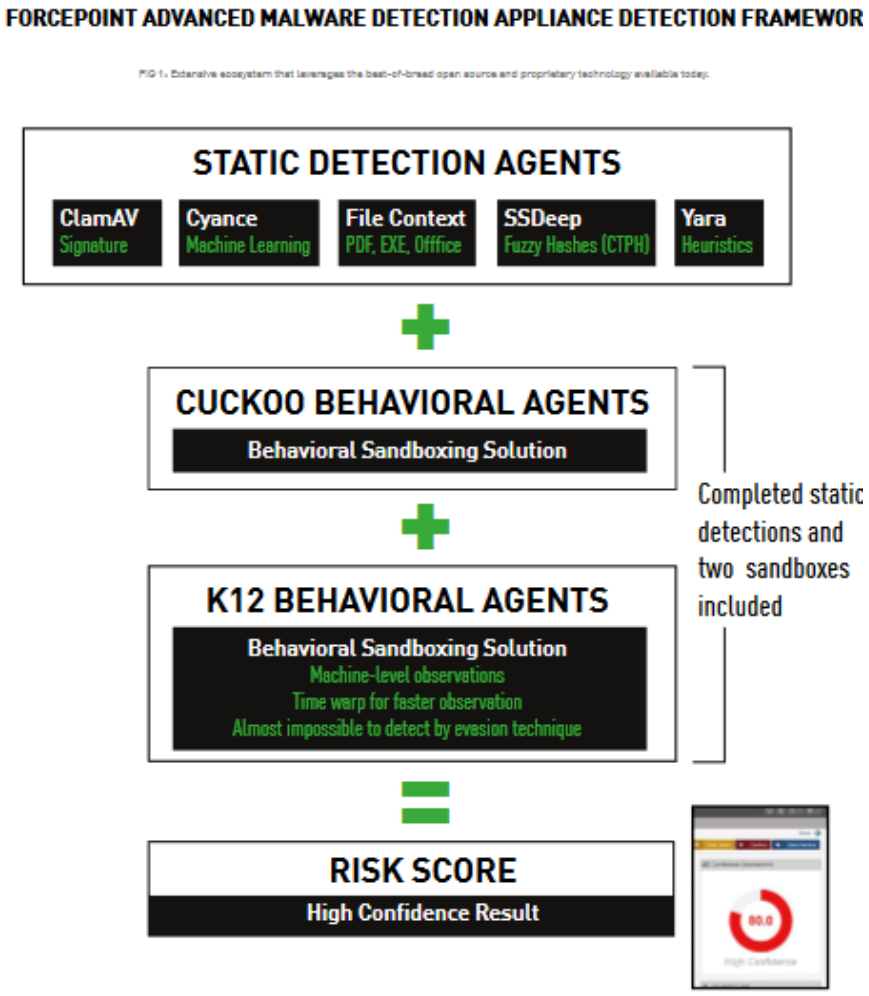
For years, Security and Risk (SR) professionals made major investments in signature-based defenses of email, network and endpoint security solutions. The methodology of these solutions has proven itself ineffective against today's evasive malware being developed by highly sophisticated and well-funded adversaries. As a response, SR professionals are turning to Automated Malware Analysis (AMA) technologies in order to arm themselves against zero day and Advanced Persistent Threats (APTs) attacking their organizations. AMA tools automate the unique skill set of malware analysis traditionally done only by highly qualified manual practitioners. Due to the shortage of this expertise, manual processes have been replaced with automation that performs a combination of static and behavioral analysis to detect and prevent the entry of known malware and brand-new exploits.

Forcepoint Advanced Malware Detection Appliance 3.4

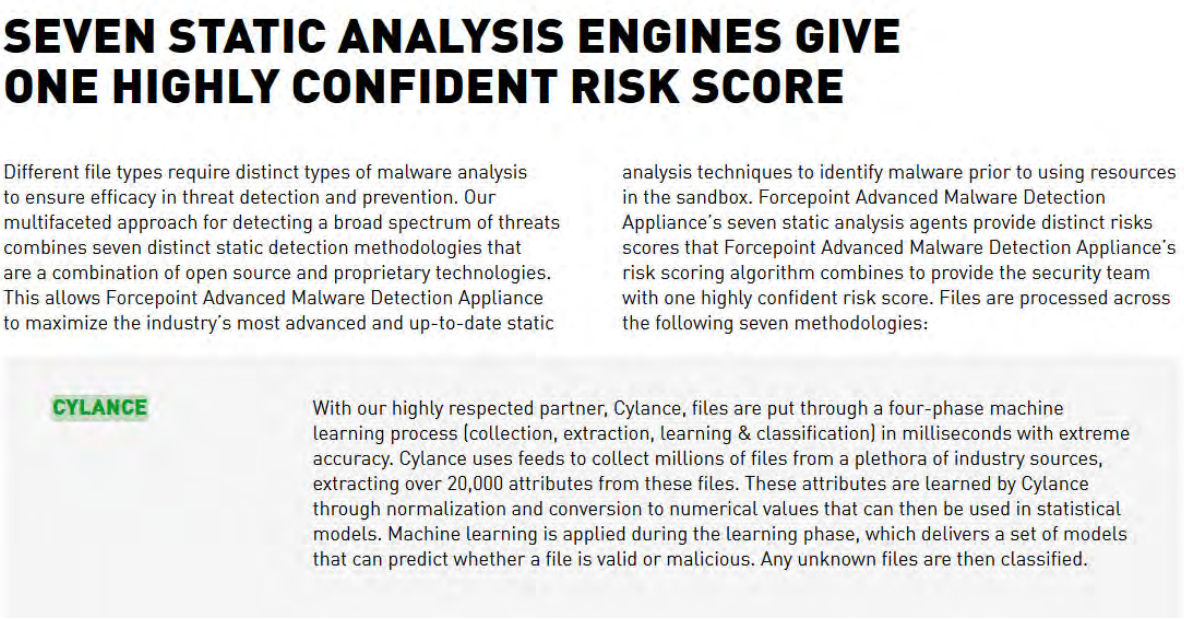
Forcepoint Advanced Malware Detection Appliance is an on-premises, automated malware analysis framework developed for organizations needing to add detection and prevention against stealthy and advanced threats to their existing Forcepoint Web and Email Security solutions. Forcepoint Advanced Malware Detection Appliance framework's unmatched efficacy processes files through seven distinct static analytic agents and a dual-sandboxing process. Its ecosystem analyses malware behavior with a combination of best-of-breed open source and Forcepoint proprietary static and dynamic technologies. Unique to the market is the defense-grade anti-evasion technology within Forcepoint's proprietary ThINK sandbox, stopping malware typically capable of circumventing commercially available sandboxes.

(*Id.*)

254. The Accused Products perform static data analysis, including by integrating with Forcepoint's partner Cylance's systems, in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy." (*Id.* at 4). This analysis includes the use of "models" that are "deliver[ed]" by Cylance servers to the Advanced Malware Detection appliance and then used to "predict whether a file is valid or malicious." (*See Id.*).


FORCEPOINT ADVANCED MALWARE DETECTION APPLIANCE DETECTION FRAMEWOR
STATIC DETECTION AGENTS
ClamAV
Signature
Cyance
Machine Learning
File Context
PDF, EXE, Office
SSDeep
Fuzzy Hashes (CTPH)
Yara
Heuristics
CUCKOO BEHAVIORAL AGENTS
Behavioral Sandboxing Solution
K12 BEHAVIORAL AGENTS
Behavioral Sandboxing Solution
Machine-level observations
Time warp for faster observation
Almost impossible to detect by evasion technique
Completed static detections and two sandboxes included
RISK SCORE
High Confidence Result

(*Id.* at 3)



SEVEN STATIC ANALYSIS ENGINES GIVE ONE HIGHLY CONFIDENT RISK SCORE

Different file types require distinct types of malware analysis to ensure efficacy in threat detection and prevention. Our multifaceted approach for detecting a broad spectrum of threats combines seven distinct static detection methodologies that are a combination of open source and proprietary technologies. This allows Forcepoint Advanced Malware Detection Appliance to maximize the industry's most advanced and up-to-date static analysis techniques to identify malware prior to using resources in the sandbox. Forcepoint Advanced Malware Detection Appliance's seven static analysis agents provide distinct risks scores that Forcepoint Advanced Malware Detection Appliance's risk scoring algorithm combines to provide the security team with one highly confident risk score. Files are processed across the following seven methodologies:

CYLANCE

With our highly respected partner, Cylance, files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy. Cylance uses feeds to collect millions of files from a plethora of industry sources, extracting over 20,000 attributes from these files. These attributes are learned by Cylance through normalization and conversion to numerical values that can then be used in statistical models. Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified.

(*Id*. at 4)

255. The Accused Products include "*at least one memory* and "*at least one processor*" such as the memory and processor(s) in one or more servers, appliances, and/or endpoint devices. For example, the Forcepoint Advanced Malware Detection Appliance may be provided via "Controller Node Small," "Controller Node Large," and "Behavioral Nodes" models with 64 gigabytes, 128 gigabytes, and 32 gigabytes of memory, respectively.

Technical Specifications

	CONTROLLER NODE SMALL	CONTROLLER NODE LARGE	BEHAVIORAL NODES
Model Name	M5000C	M10000C	M5000BA M5000BB
Files Per Day	<300,000	<750,000	N/A
Form Factor	1U	1U	1U
Hardware Platform	Dell PowerEdge R430	Dell PowerEdge R430	Dell PowerEdge R430
Memory	64 GB	128 GB	32 GB
Processor	(2) Intel E5-2650 v3	(2) Intel E5-2650 v3	(2) Intel E5-2650 v3
On Board NIC	4 Port LOM	4 Port LOM	4 Port LOM
Hard Drives	(4) 1.2 TB 10K	(4) 1.2 TB 10K	(4) 300GB 15K
Raid Setting	RAID 10	RAID 10	RAID 10
iDRAC License	Enterprise	Enterprise	Enterprise
Hardware Support	ProSupport: 7x24 HW/SW w/Keep Your Drive	ProSupport: 7x24 HW/SW w/Keep Your Drive	ProSupport: 7x24 HW/SW w/Keep Your Drive

(*Id*. at 10)

256. As another example, the Accused Products implement Cylance's static analysis in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification)." "Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified." Upon information and belief, these classifications, among others, are stored in memory.

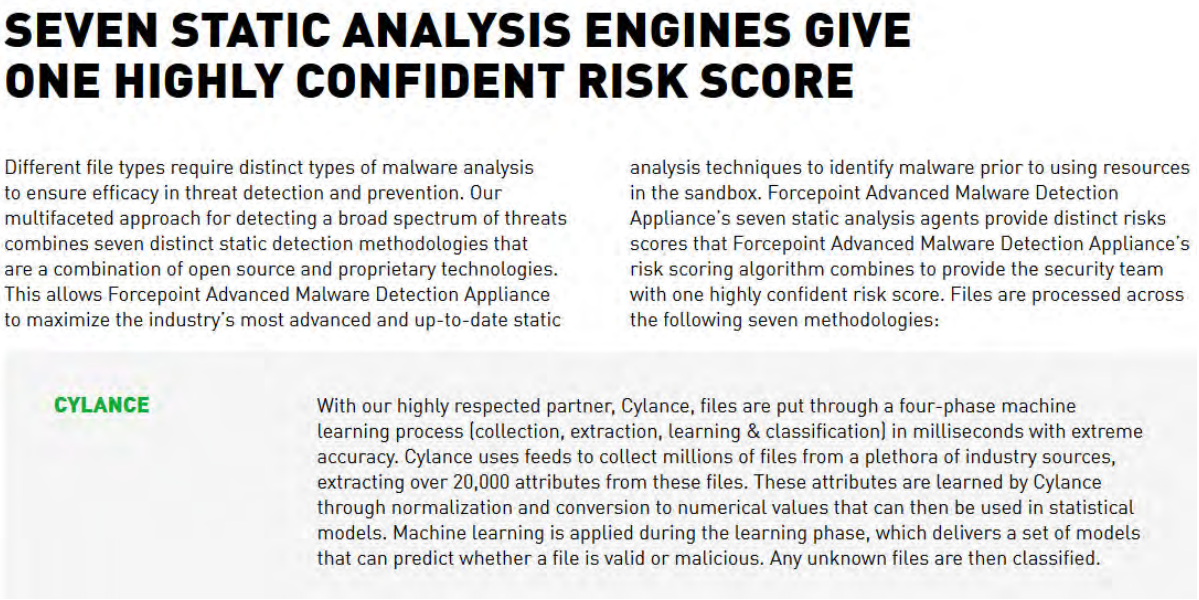
CYLANCE

With our highly respected partner, Cylance, files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy. Cylance uses feeds to collect millions of files from a plethora of industry sources, extracting over 20,000 attributes from these files. These attributes are learned by Cylance through normalization and conversion to numerical values that can then be used in statistical models. Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified.

(*See id*. at 4)

257. The Accused Products include at least on processor *to perform operation of* "*identifying static data points that may be indicative of either a harmful or benign executable file*."

For example, one or more servers, appliances, and/or endpoint devices each have one or more processors. As example, and as shown above, the Forcepoint Advanced Malware Detection Appliance is provided via "Controller Node Small," "Controller Node Large," and "Behavioral Nodes" models. (*Id*.). Each model includes an Intel E5-2650 v3 processor. (*Id.* at 10). The Forcepoint Advanced Malware Detection Appliance "framework's unmatched efficacy processes files through seven distinct analytic agents and a dual-sandboxing process. Its ecosystem analyses malware behavior with a combination of best-of-breed open source and Forcepoint static and dynamic technologies." (*Id*.). "After performing static analysis of suspicious files, AMD can be called upon to perform the deep behavioral analysis necessary to identify zero-day threats and other modern malware." (Forcepoint, *Forcepoint Advanced Malware Detection at 1*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf). As another example, the Accused Products use "seven distinct static detection methodologies" including, upon information and belief, computers with processors that perform Cylance's static analysis, in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy."



SEVEN STATIC ANALYSIS ENGINES GIVE ONE HIGHLY CONFIDENT RISK SCORE

Different file types require distinct types of malware analysis to ensure efficacy in threat detection and prevention. Our multifaceted approach for detecting a broad spectrum of threats combines seven distinct static detection methodologies that are a combination of open source and proprietary technologies. This allows Forcepoint Advanced Malware Detection Appliance to maximize the industry's most advanced and up-to-date static analysis techniques to identify malware prior to using resources in the sandbox. Forcepoint Advanced Malware Detection Appliance's seven static analysis agents provide distinct risks scores that Forcepoint Advanced Malware Detection Appliance's risk scoring algorithm combines to provide the security team with one highly confident risk score. Files are processed across the following seven methodologies:

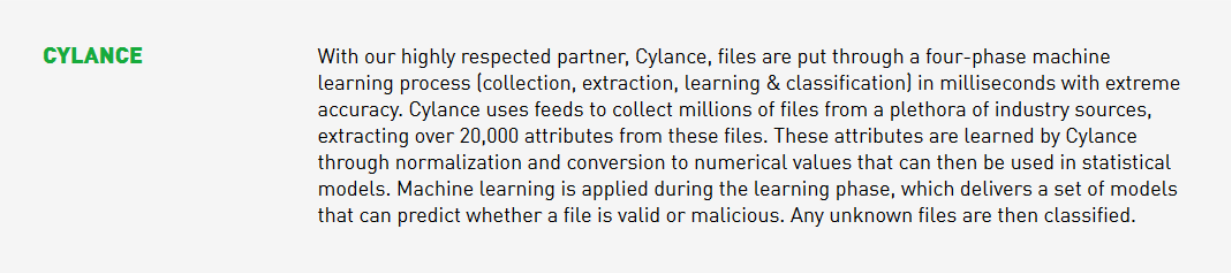
CYLANCE

With our highly respected partner, Cylance, files are put through a four-phase machine learning process [collection, extraction, learning & classification] in milliseconds with extreme accuracy. Cylance uses feeds to collect millions of files from a plethora of industry sources, extracting over 20,000 attributes from these files. These attributes are learned by Cylance through normalization and conversion to numerical values that can then be used in statistical models. Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified.

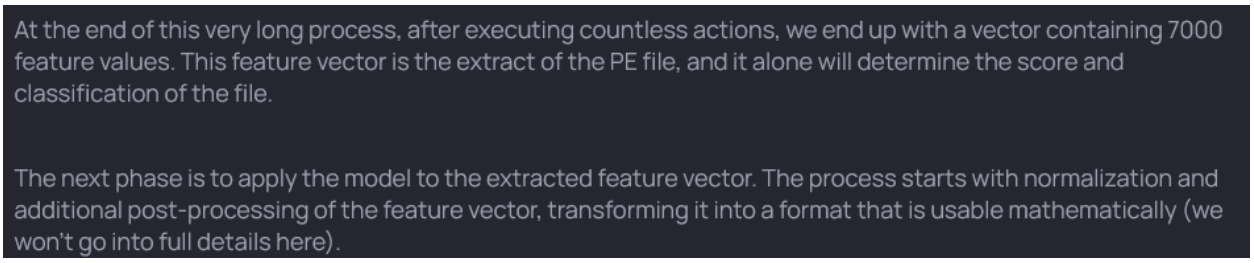
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4* at 4, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf).

258. The Accused Products include "*at least one processor configured to perform operation of . . . associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files*."

259. The Accused Products use Cylance's static analysis in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification)" and "[m]achine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified." (*Id*. (emphasis added)). When Forcepoint uses Cylance's static analysis, the "feature vector" is used to "determine the score and classification of the file" by "apply[ing] the model to the extracted feature vector." (*See* Skylight Cyber, Cylance, I Kill You!, https://skylightcyber.com/2019/07/18/cylance-i-kill-you/.).

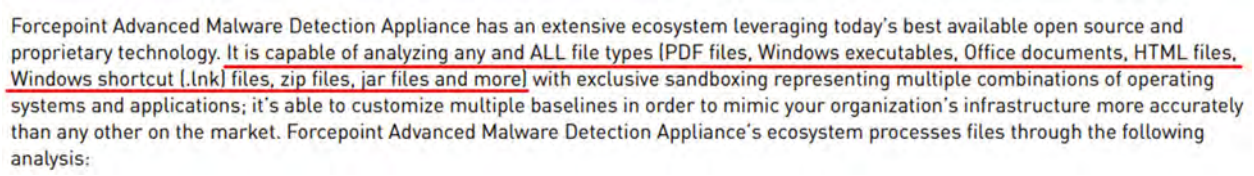


(*See* Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4* at 4, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf)



(*See* Skylight Cyber, *Cylance, I Kill You!*, https://skylightcyber.com/2019/07/18/cylance-i-kill-you/.)

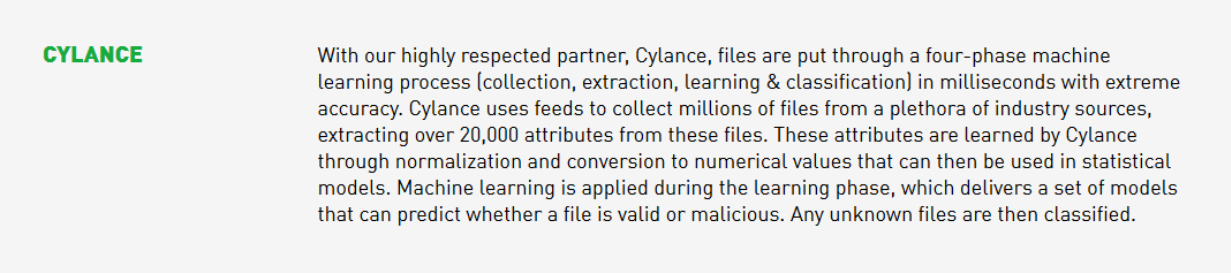
260. The Accused Products include "*at least one processor configured to perform operation of: identifying an executable file to be evaluated*." For example, the Accused Products are "capable of analyzing any and ALL file types (PDF files, Windows executables, Office documents, HTML files, Windows shortcut (.lnk) files, zip files, jar files and more)."



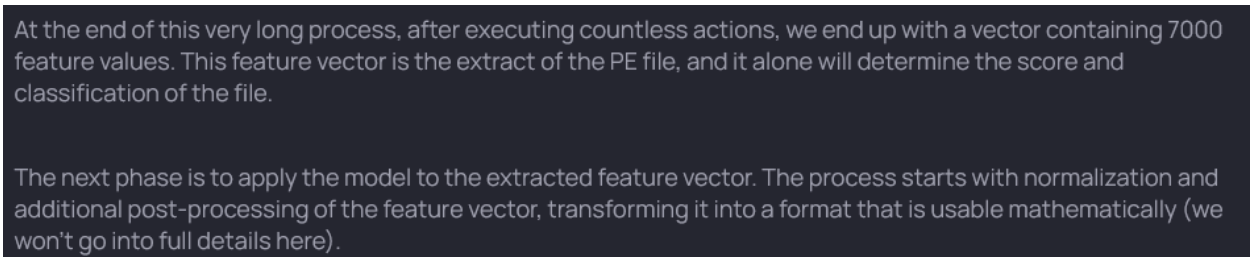
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4* at 2, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf)

261. The Accused Products use Cylance's static analysis to select an executable file to "put through a four-phase machine learning process (collection, extraction, learning & classification)" and "[m]achine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified." (*See id*. at 4).

262. When Forcepoint uses Cylance's static analysis, the "feature vector" is used to "determine the score and classification of the file" by "apply[ing] the model to the extracted feature vector." (*See* Skylight Cyber, *Cylance, I Kill You!*, https://skylightcyber.com/2019/07/18/cylance-i-kill-you/.).

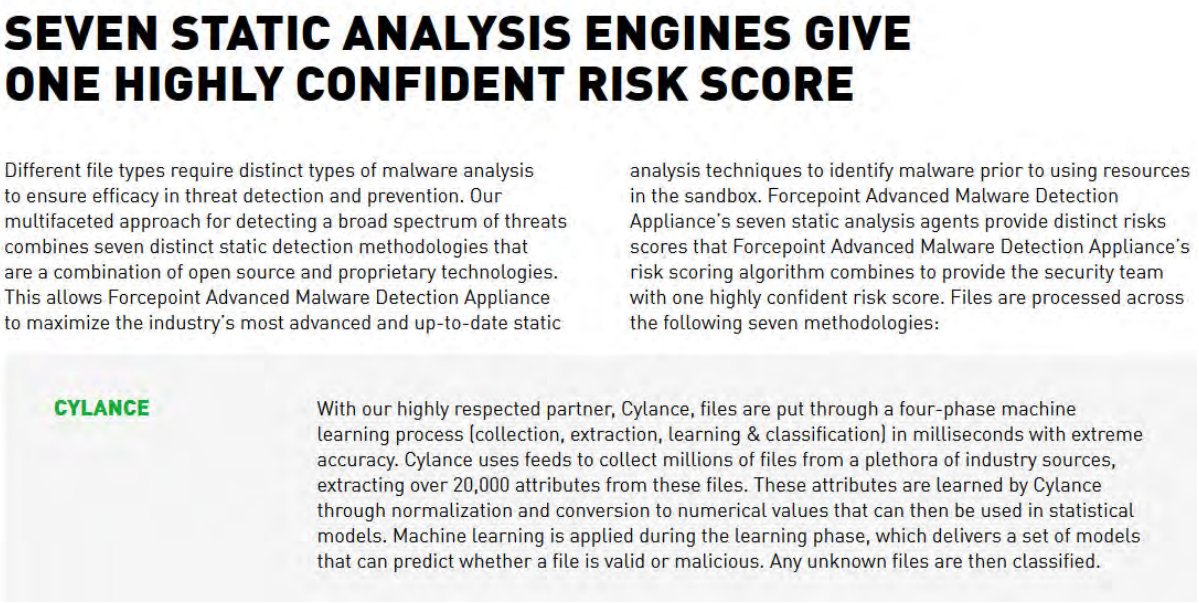


(*See* Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4* at 4, *available at* https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf)



(*See* Skylight Cyber, *Cylance, I Kill You!*, https://skylightcyber.com/2019/07/18/cylance-i-kill-you/.)

263. The Accused Products include "*at least one processor configured to perform operation of . . . extracting a plurality of static data points from the executable file.*" For example, the Accused Products use "seven distinct static detection methodologies" including Cylance's static analysis, in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification) in milliseconds with extreme accuracy." (Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4* at 4, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf).



SEVEN STATIC ANALYSIS ENGINES GIVE ONE HIGHLY CONFIDENT RISK SCORE

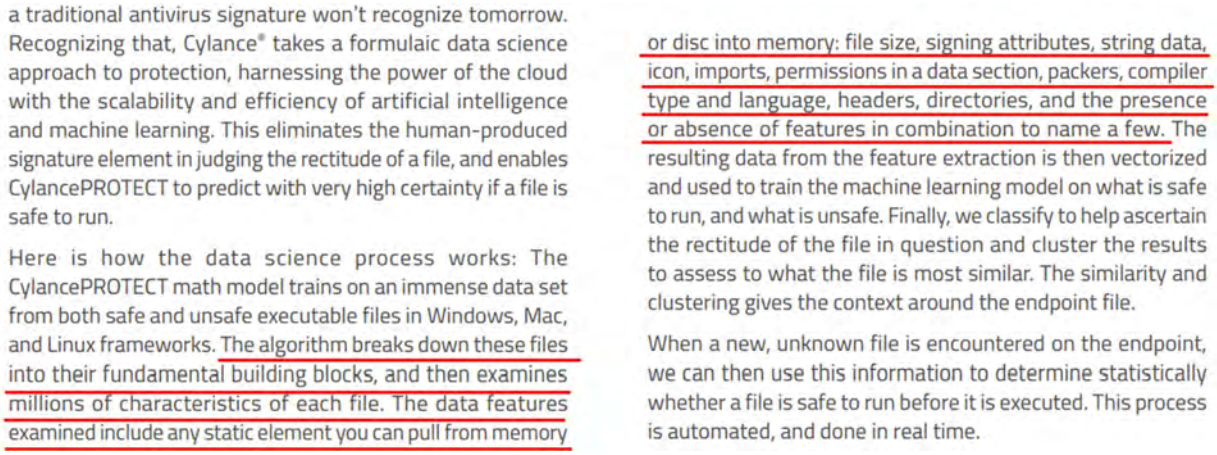
Different file types require distinct types of malware analysis to ensure efficacy in threat detection and prevention. Our multifaceted approach for detecting a broad spectrum of threats combines seven distinct static detection methodologies that are a combination of open source and proprietary technologies. This allows Forcepoint Advanced Malware Detection Appliance to maximize the industry's most advanced and up-to-date static analysis techniques to identify malware prior to using resources in the sandbox. Forcepoint Advanced Malware Detection Appliance's seven static analysis agents provide distinct risks scores that Forcepoint Advanced Malware Detection Appliance's risk scoring algorithm combines to provide the security team with one highly confident risk score. Files are processed across the following seven methodologies:

CYLANCE

With our highly respected partner, Cylance, files are put through a four-phase machine learning process [collection, extraction, learning & classification] in milliseconds with extreme accuracy. Cylance uses feeds to collect millions of files from a plethora of industry sources, extracting over 20,000 attributes from these files. These attributes are learned by Cylance through normalization and conversion to numerical values that can then be used in statistical models. Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified.

264. The Accused Products implement Cylance's static analysis—*e.g.* through the use of a set of models delivered to the appliance. That analysis extracts and examines data features that include "any static element you can pull from memory or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination." (Blackberry, *CylancePROTECT: Malware Execution Control* at 2,

blackberry.com/us/en/products/unified-endpoint-security/cylance-is-now-blackberry). Further, "[w]hen a new, unknown file is encountered on the endpoint, we can then use this information to determine statistically whether a file is safe to run before it is executed." (*Id.*).



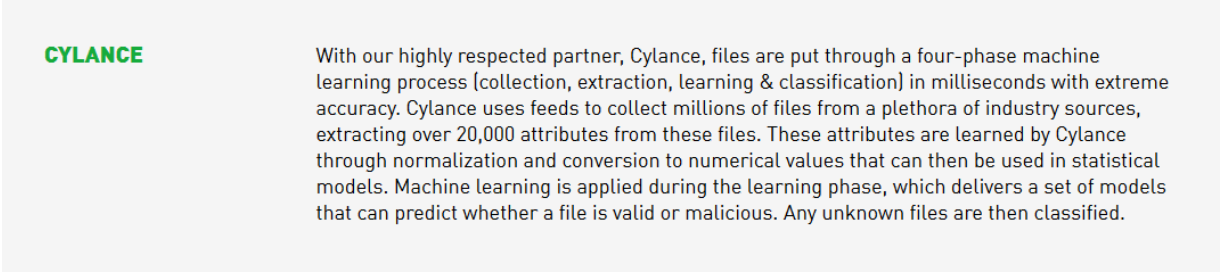
a traditional antivirus signature won't recognize tomorrow. Recognizing that, Cylance® takes a formulaic data science approach to protection, harnessing the power of the cloud with the scalability and efficiency of artificial intelligence and machine learning. This eliminates the human-produced signature element in judging the rectitude of a file, and enables CylancePROTECT to predict with very high certainty if a file is safe to run.

Here is how the data science process works: The CylancePROTECT math model trains on an immense data set from both safe and unsafe executable files in Windows, Mac, and Linux frameworks. The algorithm breaks down these files into their fundamental building blocks, and then examines millions of characteristics of each file. The data features examined include any static element you can pull from memory or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination to name a few. The resulting data from the feature extraction is then vectorized and used to train the machine learning model on what is safe to run, and what is unsafe. Finally, we classify to help ascertain the rectitude of the file in question and cluster the results to assess to what the file is most similar. The similarity and clustering gives the context around the endpoint file.

When a new, unknown file is encountered on the endpoint, we can then use this information to determine statistically whether a file is safe to run before it is executed. This process is automated, and done in real time.

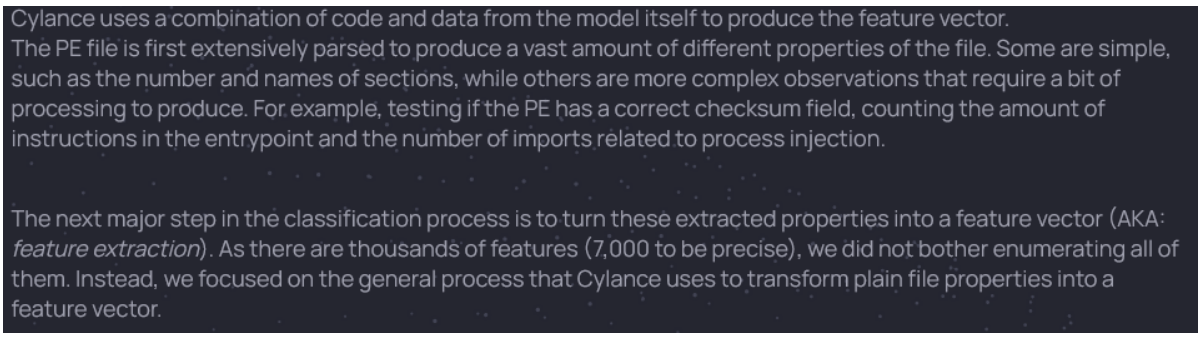
265. (*See id.*; *see also* Blackberry, *Cylance is Blackberry Cybersecurity*, blackberry.com/us/en/products/unified-endpoint-security/cylance-is-now-blackberry ("Cylance is now part of Blackberry Cybersecurity").

266. The Accused Products include "*at least one processor configured to perform operation of . . . generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range*." For example, as explained above, the Accused Products implement Cylance's static analysis in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification)." "Machine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified."



(*See* Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4* at 4, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf).

267. Additionally, the resulting data "from the feature extraction is . . . vectorized and used to train the machine learning." On information and belief, Cylance's static analysis similarly generates a feature vector from the static data points during the classification process. (*See* Blackberry, *CylancePROTECT, Malware Execution Control* at 2, https://www.blackberry.com/content/dam/cylance/documents/pdf/pdf-feature-focus-protect-malware-control.pdf). For example, the Accused Products, implementing the Cylance static analysis, uses "any static element you can pull from memory or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination." During the classification process, the Accused Products generate a "feature vector" using "extracted properties" from the file:



(*See* Skylight Cyber, *Cylance, I Kill You!*, https://skylightcyber.com/2019/07/18/cylance-i-kill-you/)

268. As explained above, the Accused Products implement Cylance's static analysis (*e.g.*, through the use of a set of models provided to the appliance), which uses a "math model [that] trains on an immense data set from both safe and unsafe executable files in Windows, Mac, and Linux frameworks. The algorithm breaks down these files into their fundamental building blocks, and then examines millions of characteristics of each file."

a traditional antivirus signature won't recognize tomorrow. Recognizing that, Cylance® takes a formulaic data science approach to protection, harnessing the power of the cloud with the scalability and efficiency of artificial intelligence and machine learning. This eliminates the human-produced signature element in judging the rectitude of a file, and enables CylancePROTECT to predict with very high certainty if a file is safe to run.

Here is how the data science process works: The CylancePROTECT math model trains on an immense data set from both safe and unsafe executable files in Windows, Mac, and Linux frameworks. The algorithm breaks down these files into their fundamental building blocks, and then examines millions of characteristics of each file. The data features examined include any static element you can pull from memory or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination to name a few. The resulting data from the feature extraction is then vectorized and used to train the machine learning model on what is safe to run, and what is unsafe. Finally, we classify to help ascertain the rectitude of the file in question and cluster the results to assess to what the file is most similar. The similarity and clustering gives the context around the endpoint file.

When a new, unknown file is encountered on the endpoint, we can then use this information to determine statistically whether a file is safe to run before it is executed. This process is automated, and done in real time.

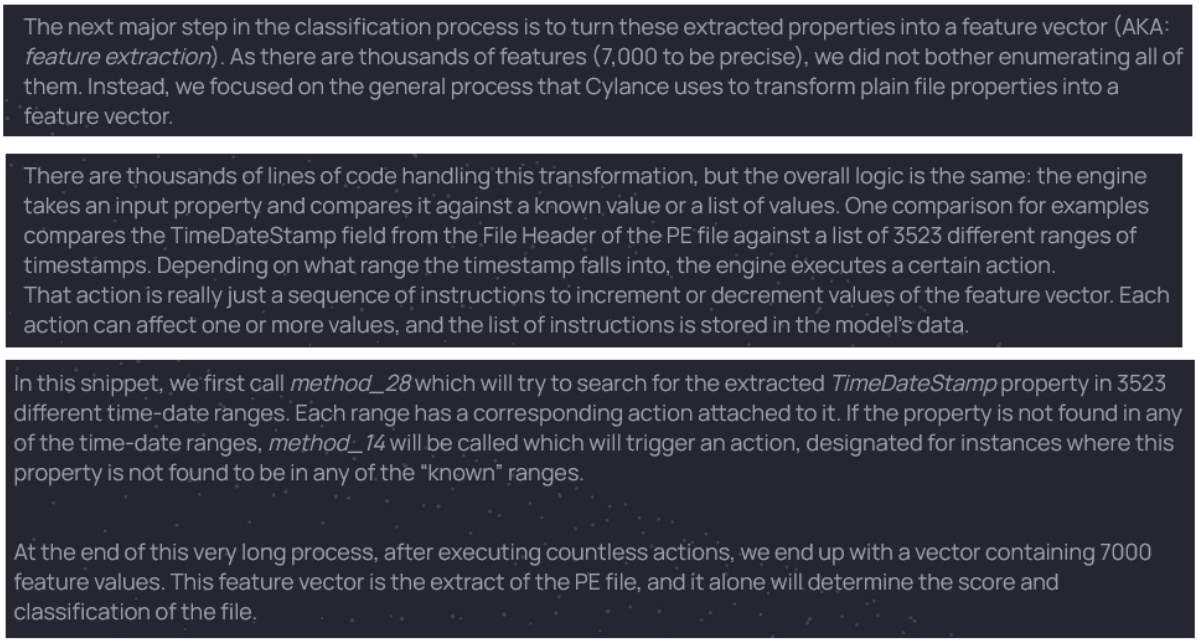
(*See* Blackberry, *CylancePROTECT, Malware Execution Control* at 2, https://www.blackberry.com/content/dam/cylance/documents/pdf/pdf-feature-focus-protect-malware-control.pdf.)

By applying the same approach to classification of files as malicious or benign, we get clear and substantial benefits:

- Prediction by design — a well-trained model should have the ability to identify a malicious file it has never seen and has no prior knowledge of.
- Infrequent updates — a model is trained once and can last years without updates.
- Lower resource consumption — AI vendors claim that the nature of their technology leads to lower CPU, memory and disk consumption.

(*See* Skylight Cyber, *Cylance, I Kill You!*, https://skylightcyber.com/2019/07/18/cylance-i-kill-you/)

269. The Accused Products implement Cylance's static analysis as part of the classification process. When that occurs "extracted properties" are turned "into a feature vector." On information and belief, there "are thousands of lines of code handling this transformation, but the overall logic is the same: the engine takes an input property and compares it against a known value or a list of values. One comparison for examples compares the TimeDateStamp field from the File Header of the PE file against a list of 3523 different ranges of timestamps. Depending on what range the timestamp falls into, the engine executes a certain action. That action is really just a sequence of instructions to increment or decrement values of the feature vector. Each action can affect one or more values, and the list of instructions is stored in the model's data."

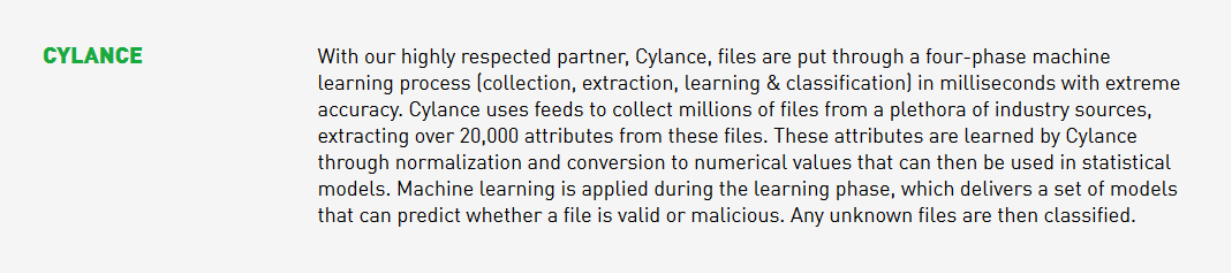


(*See* Skylight Cyber, *Cylance, I Kill You!*, https://skylightcyber.com/2019/07/18/cylance-i-kill-you/)

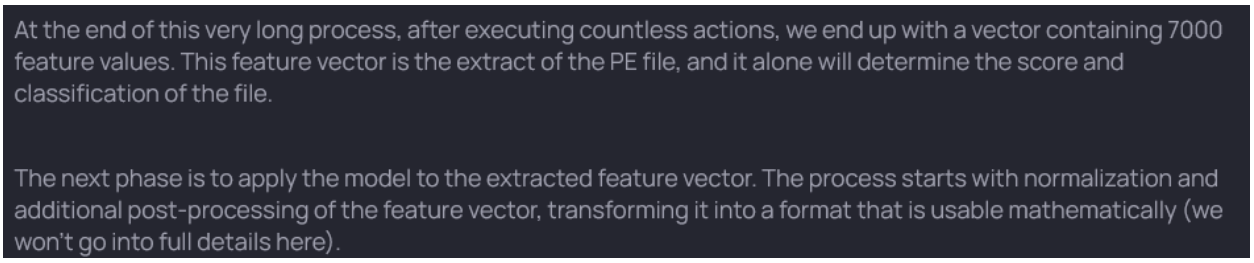
> or disc into memory: file size, signing attributes, string data, icon, imports, permissions in a data section, packers, compiler type and language, headers, directories, and the presence or absence of features in combination to name a few. The resulting data from the feature extraction is then vectorized and used to train the machine learning model on what is safe to run, and what is unsafe. Finally, we classify to help ascertain the rectitude of the file in question and cluster the results to assess to what the file is most similar. The similarity and clustering gives the context around the endpoint file.

(*See* Blackberry, *CylancePROTECT Malware Execution Control* at 2, https://www.blackberry.com/content/dam/cylance/documents/pdf/pdf-feature-focus-protect-malware-control.pdf)

270. The Accused Products include "*at least one processor configured to perform operation of . . . evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files*." For example, as explained above, the Accused Products use Cylance's static analysis in which "files are put through a four-phase machine learning process (collection, extraction, learning & classification)" and "[m]achine learning is applied during the learning phase, which delivers a set of models that can predict whether a file is valid or malicious. Any unknown files are then classified." (*See* Forcepoint, *Forcepoint: Threat Protection Appliance 3.4* at 4, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf.). When Forcepoint uses Cylance's static analysis, the "feature vector" is used to "determine the score and classification of the file" by "apply[ing] the model to the extracted feature vector."



(*See* Forcepoint, *Forcepoint: Threat Protection Appliance 3.4* at 4, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf.)



(*See* Skylight Cyber, *Cylance, I Kill You!*, https://skylightcyber.com/2019/07/18/cylance-i-kill-you/)

271. Each claim in the ’869 Patent recites an independent invention. Neither claim 46, described above, nor any other individual claim is representative of all claims in the ’869 Patent.

272. Forcepoint has been aware of the ’869 Patent since at least the filing of this Amended Complaint.

273. Defendant directly infringes at least claim 46 of the ’869 Patent, either literally or under the doctrine of equivalents, by practicing the elements described above. For example, on information and belief, Defendant makes and uses the claimed system in an infringing manner as described above by running this software and system to protect its own computer and network operations. On information and belief, Defendant also makes and uses the claimed system in an infringing manner when testing the operation of the Accused Products and corresponding systems.

As another example, Defendant makes and uses the claimed system when providing or administering services to third parties, customers, and partners using the Accused Products.

274. Defendant's partners, customers, and end users of its Accused Products and corresponding systems and services directly infringe at least claim 46 of the '869 Patent, literally or under the doctrine of equivalents, at least by using the Accused Products and corresponding systems and services, as described above.

275. Defendant has actively induced and is actively inducing infringement of at least claim 46 of the '869 patent with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). For example, Defendant encourages and induces customers with specific intent to use Forcepoint's security software in a manner that infringes claim 46 of the '869 patent at least by offering and providing software and/or hardware that embodies the system that infringes claim 46 when installed and operated by the customer, and by engaging in activities relating to selling, marketing, advertising, promotion, installation, support, and distribution of the Accused Products, including the activities described below.

276. Defendant encourages, instructs, directs, and/or requires third parties – including its certified partners and/or customers – to practice the claims using the software, services, and systems in infringing ways, as described above.

277. Defendant further encourages and induces its customers to infringe claim 46 of the '869 patent: 1) by making its security services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users, and 2) through activities relating to marketing, advertising, promotion, installation, support, and distribution of the Accused Products, including its Forcepoint

security software, and services in the United States. *See, e.g.*, https://www.websense.com/content/support/library/amd-op/v10/install/manager-install.pdf; https://www.websense.com/content/support/library/ngfw/v67/install/ngfw_670_ig_a_en-us.pdf; https://www.websense.com/content/support/library/endpoint/v20/installation.pdf; https://www.forcepoint.com/sites/default/files/resources/brochures/guide-sase-guide-to-protecting-remote-workers-en_0_0_0_0_0_0_0_0.pdf.

278. For example, on information and belief, Defendant shares instructions, guides, and manuals, which advertise and instruct third parties on how to use the software as described above, including at least customers and partners. *Id.* On further information and belief, Defendant also provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that deploy and use the Accused Products in an infringing manner. *Id.*

279. Defendant and/or its partners recommend and sell the Accused Products and provide technical support for the installation, implementation, integration, and ongoing operation of the Accused Products for each individual customer. On information and belief, each customer enters into a contractual relationship with Defendant and/or one of its partners, which obligates each customer to perform certain actions in order to use the Accused Products. Further, in order to receive the benefit of Defendant and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the '869 Patent.

280. Further, as the entity that provides installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Product remains operational for each customer through ongoing technical support, on information and belief, Defendant and/or its

partners affirmatively aid and abet each customer's use of the Accused Products in a manner that infringes the claimed system of, and infringes, the '869 Patent.

281. Defendant also contributes to the infringement of the its partners, customers, and end-users of the Accused Products by providing within the United States or importing the Accused Products into the United States, which are for use in practicing, and under normal operation practice, systems claimed in the Asserted Patents, constituting a material part of the inventions claimed, and not a staple article or commodity of commerce suitable for substantial non-infringing uses.

282. Indeed, as shown above, the Accused Products have no substantial non-infringing uses because the accused functionality, including the machine learning static analysis and related functionality, are integral parts of the Accused Products and must be performed for the Accused Products to perform their intended purpose. These processes are continually running when the system is in use and, on information and belief, cannot be removed or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown above, or without the system and components identified above that practice the '869 patent, that functionality could not be performed.

283. Additionally, the accused functionality, including the machine learning static analysis related functionality described below, itself has no substantial non-infringing uses because the components, modules and methods identified above are a necessary part of that functionality. For example, without the Accused Products' machine learning functionality, the Accused Products could not deploy their machine learning model to determine whether or not an object is malicious or benign based on the object's static features. These process are continually running when the

system is in use and, on information and belief, cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the '869 Patent, that functionality could not be performed.

284. In addition, as shown in the detailed analysis above, the products, systems, modules, and methods provided by Defendant constitute a material part of the invention—indeed, they provide all the components, modules, and features that perform the claimed methods and systems. For example, the Accused Products and accused functionalities (including the machine learning static analysis) constitute a material part of the inventions claimed because such analysis is integral to the processes identified above as recited in the claims of the '869 Patent. None of these products are staple goods—they are sophisticated and customized cyber security and malware detection products, methods, and systems.

285. On information and belief, the infringing actions of each partner, customer, and/or end-user of the Accused Products are attributable to Defendant. For example, on information and belief, Defendant directs and controls the activities or actions of its partners or others in connection with the Accused Products by contractual agreement or otherwise requiring partners or others to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Defendant further directs and controls the operation of devices executing the Accused Products by programming the software which, when executed by a customer or end user, perform the claimed method of at least claim 46 of the '869 Patent.

286. Plaintiffs have suffered and continue to suffer damages, including lost profits, as a result of Defendant's infringement of the '869 Patent. Defendant is therefore liable to Plaintiffs

under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiffs for Defendant's infringement, but no less than a reasonable royalty.

287. Plaintiffs will continue to suffer irreparable harm unless this Court preliminarily and permanently enjoins Defendant, its agents, employees, representatives, and all others acting in concert with Defendant from infringing the '869 Patent.

288. Defendant's infringement of the '869 Patent is knowing and willful. Defendant acquired knowledge of the '869 Patent and of the specific conduct that constitutes infringement of the '869 patent at least based on this Amended Complaint. Defendant continues to engage in infringing activities after the filing of this amended complaint, including by selling, making, configuring, and installing the Accused Products and performing the accused functionality, and by engaging in activities that constitute inducing infringement and contributory infringement as described above.

289. On information and belief, despite Defendant's knowledge of the Asserted Patents, Defendant made the deliberate decision to sell products and services that it knew infringe these patents. Defendant's continued infringement of the '869 Patent with knowledge of the '869 Patent constitutes willful infringement.

SEVENTH CAUSE OF ACTION
(INFRINGEMENT OF THE '243 PATENT)

290. Plaintiffs reallege and incorporate by reference the allegations of the preceding paragraphs of this Amended Complaint.

291. Forcepoint has infringed and continues to infringe one or more claims of the '243 Patent in violation of 35 U.S.C. § 271 in this judicial District and elsewhere in the United States and will continue to do so unless enjoined by this Court. The Accused Products, including features of the Forcepoint's Advanced Malware Detection system (as well as any other products that

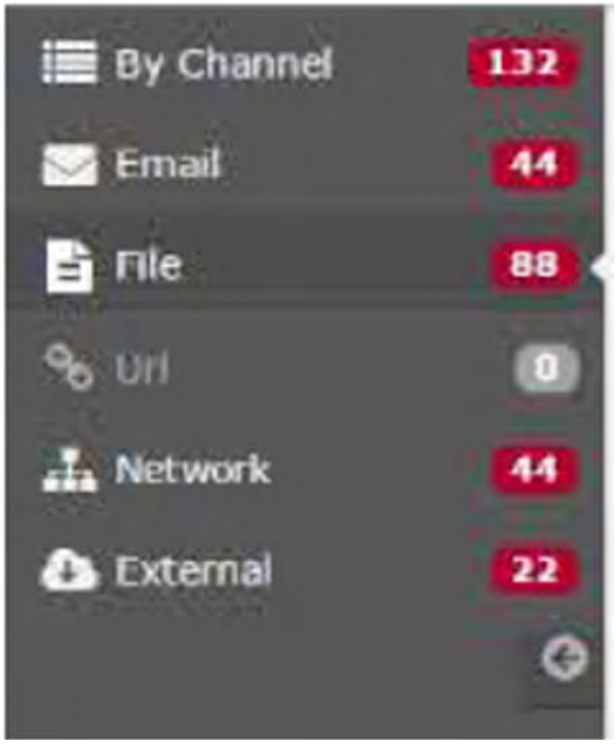
include the features described below), when used for their ordinary and customary purposes, practice each element of at least claim 1 of the '243 Patent, as demonstrated below.

292. Claim 1 of the '243 Patent recites:

> 1. A method for identifying an origin of activity on a computer that is indicative of pestware comprising:
>
> monitoring, using a kernel-mode driver, the computer for activity that is indicative of pestware, wherein the monitoring includes monitoring API calls and storing a history of at least a portion of the API calls in an activity log;
>
> analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;
>
> identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;
>
> accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and
>
> reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware.

293. The Accused Products include "*a method for identifying an origin of activity on a computer that is indicative of pestware*." For example, Forcepoint Advanced Malware Detection (AMD) provides leading malware detection capabilities. The sandbox used by AMD is based on an architecture that emulates and analyzes the activity of a host, including the CPU, system memory, all input/output devices, operating system, and hardware environment. Forcepoint Advanced Malware Detection Appliance integrates with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one platform. An "Events section" speeds incident response with a dynamic and

interactive screen, providing information by threat channel category such as Email, File, URL, Network or External. Forcepoint Advanced Malware Detection Appliance's link analysis visualization feature provides relationships between events, sender/receiver, files and other information.



(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)

Forcepoint Advanced Malware Detection (AMD) leverages proven technology to detect zero-day and other advanced malware. Using Deep Content Inspection technology, Forcepoint AMD emulates an entire host, interacting with malware to expose and observe a malicious object's possible actions. These include advanced evasion techniques, O/S or application specific threats, dormant code analysis and even CPU and in-memory activity.

(Forcepoint Advanced Malware Detection Appliance 3.4, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 01)

Forcepoint Advanced Malware Detection (AMD) leverages proven Lastline technology to detect zero-day and other advanced malware. Using Lastline's Deep Content Inspection technology, Forcepoint AMD emulates an entire host, interacting with malware to expose and observe a malicious object's possible actions. These include advanced evasion techniques, O/S or application specific threats, dormant code analysis and even CPU and in-memory activity.

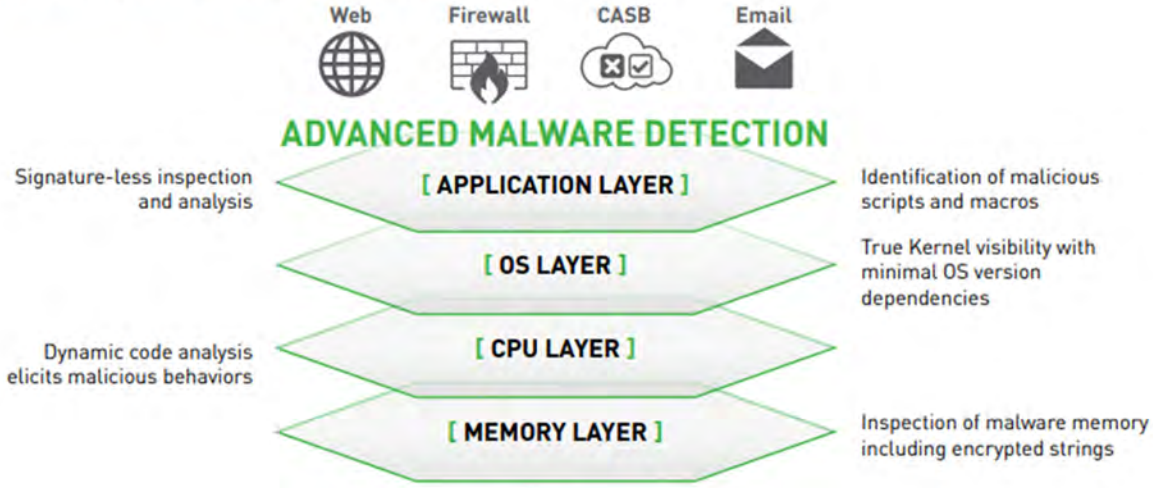
In a recent study of Breach Detection Systems (BDS) by NSS Labs, Lastline was named the most effective advanced malware detection system and was the only product to achieve **100% detection** with **zero false positives.**

(Forcepoint, *FORCEPOINT Advanced Malware Detection*, https://www.guardsense.com/datasheets/datasheet_forcepoint_advanced_malware_detection_lastline_en_0.pdf at 1)

The industry's best malware detection engine

Forcepoint chose Lastline as a partner for Advanced Malware Detection because of their leading malware detection capabilities (as demonstrated in the NSS Labs study). The sandbox is based on a unique architecture that emulates and analyzes the activity of an entire host, including the CPU, system memory and all input/output devices. Often missed by other security technologies, Lastline's Deep Content Inspection provides visibility into the behavior of malicious code by emulating a complete operating system and hardware environment. Emulation eliminates the clues that malware often uses to evade detection in more traditional, virtualized sandboxes.

THE DEEP CONTENT INSPECTION DIFFERENCE



(Forcepoint, *FORCEPOINT Advanced Malware Detection*, https://www.guardsense.com/datasheets/datasheet_forcepoint_advanced_malware_detection_lastline_en_0.pdf at 2)



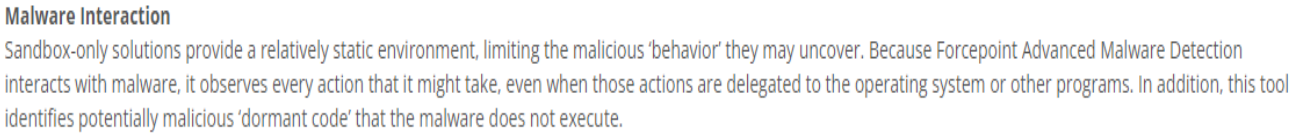
Integrated with Forcepoint defenses across all key threat vectors

AMD is available as a fully integrated option for Forcepoint CASB, NGFW, Web Security and Email security. In this integration, Forcepoint's core solutions first assess the broader context of an internet transaction for potential indicators of compromise. After performing static analysis of suspicious files, AMD can be called upon to perform the deep behavioral analysis necessary to identify zero-day threats and other modern malware.

Available as a cloud service or on-premises solution (for more cautious or otherwise cloud-adverse organizations), Forcepoint AMD is the perfect complement to your Forcepoint CASB, NGFW, Web Security or Email security solution. It provides unparalleled threat detection, as well as consistent threat forensic information, to optimize incident response teams.

Forcepoint AMD will give you all the information you need—regardless of the threat vector—while 'zero-false positives' means you'll spend your valuable time working against true threats. Regardless of your size or industry, Forcepoint provides the comprehensive security solutions you need to challenge today's fast evolving, highly evasive threats.

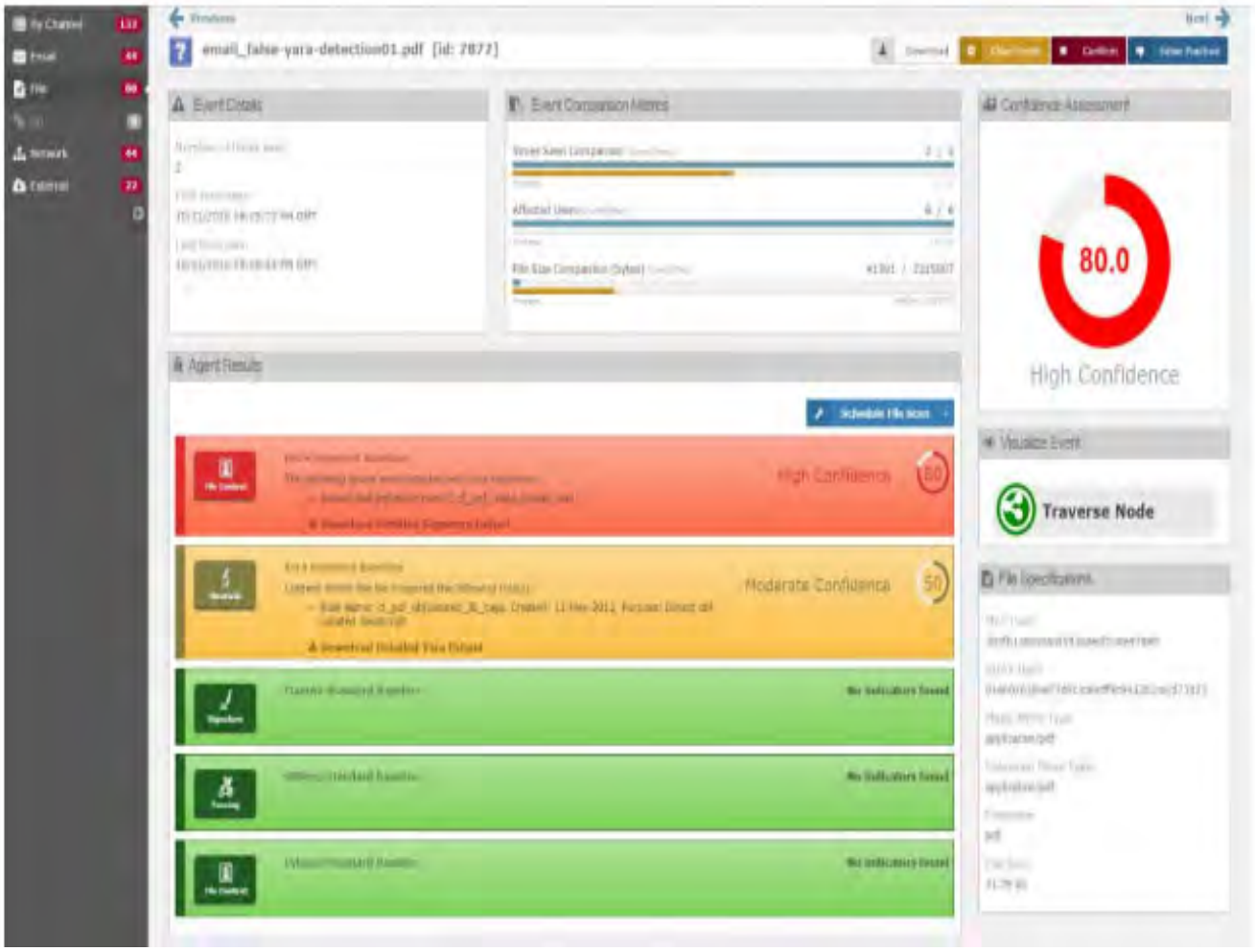
(Forcepoint, *FORCEPOINT Advanced Malware Detection*, https://www.guardsense.com/datasheets/datasheet_forcepoint_advanced_malware_detection_lastline_en_0.pdf at 2)



Malware Interaction

Sandbox-only solutions provide a relatively static environment, limiting the malicious 'behavior' they may uncover. Because Forcepoint Advanced Malware Detection interacts with malware, it observes every action that it might take, even when those actions are delegated to the operating system or other programs. In addition, this tool identifies potentially malicious 'dormant code' that the malware does not execute.

(Forcepoint, *Forcepoint Advanced Malware Detection*, https://www.guardsense.com/AMD.asp at 2)



(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)

Integration with Forcepoint Web and Email Security Ensures a Consolidated Threat View

Forcepoint Advanced Malware Detection Appliance integrates seamlessly with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. In turn, this reduces risk and boosts efficacy of existing security investments. In addition to integration with TRITON Architecture, Forcepoint Advanced Malware Detection Appliance provides security practitioners with the ability to highly customize their malware analysis as the company grows and malware evolves. This is via Forcepoint Advanced Malware Detection Appliance's standalone threat management console.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 8)

EVENTS SECTION	This section speeds incident response with a dynamic and interactive screen, providing information by threat channel category such as Email, File, URL, Network or External. Analysts are capable of dissecting the threat landscape to have a more thorough understanding of the attack vector.
TOOLS & ADMINISTRATION FEATURES	Multiple tools and administration features provide flexibility for customizing the ecosystem to meet unique needs. Role based access and privilege information controls are extremely flexible. Agent and file type management and system health monitoring are easily performed through use of the console.
FORENSIC TOOLS	Investigative analytics tools address the security professional's need to quickly respond to incidents and launch forensic investigations. Forcepoint Advanced Malware Detection Appliance's link analysis visualization feature instantly unearths relationships between events, sender/receiver, files and other information in order to find an association where the naked eye cannot. Typically a daunting manual process, with one click, the analyst can now see relationships across multiple variables as one picture, enabling them to instantly assess what happened, when, where and how, and determine why.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 9)

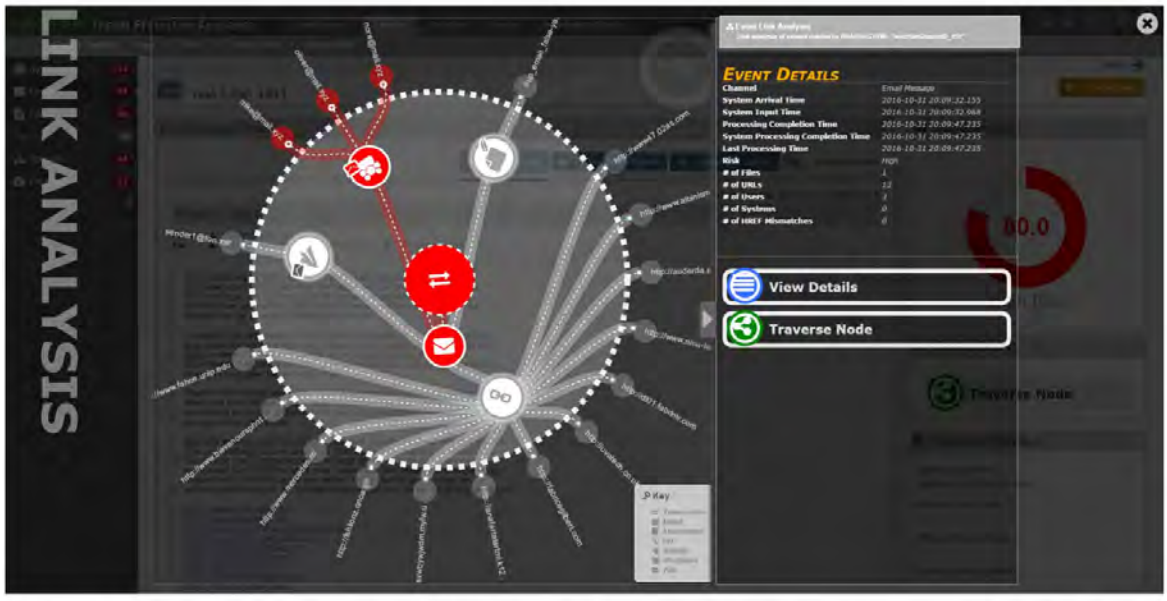


FIG 3: Forensic Tools provide advanced visualizations for teams to quickly and easily examine complex threat data and bring forward hidden relationships between events, sender, files and other information.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 10)

294. The Accused Products include "*monitoring, using a kernel-mode driver, the computer for activity that is indicative of pestware, wherein the monitoring includes monitoring API calls and storing a history of at least a portion of the API calls in an activity log*." For example, Forcepoint Advanced Malware Detection Appliance utilizes Cuckoo's malware analysis system with Raytheon's defense-grade ThiNK sandboxing technologies which add dual behavioral analysis to the static analysis processes as the sandboxing process begins with sending the file to Cuckoo's malware analysis system which is able to trace API calls. Following Cuckoo's analysis, the file is sent to ThINK, Raytheon's proprietary sandbox, which is a custom hypervisor to contain and isolate malicious files.

Malware Interaction

Sandbox-only solutions provide a relatively static environment, limiting the malicious 'behavior' they may uncover. Because Forcepoint Advanced Malware Detection interacts with malware, it observes every action that it might take, even when those actions are delegated to the operating system or other programs. In addition, this tool identifies potentially malicious 'dormant code' that the malware does not execute.

(Forcepoint, *Forcepoint Advanced Malware Detection (AMD)*, https://www.guardsense.com/AMD.asp at 02)

DUAL SANDBOXING IDENTIFIES THE MOST ADVANCED ADVERSARIES IN THE WORLD:

Forcepoint Advanced Malware Detection Appliance leverages best-of-breed open source technology Cuckoo with Raytheon's defense-grade ThiNK, proprietary sandboxing technology. Adding dual behavioral analysis to the above mentioned seven step static analysis process makes Forcepoint Advanced Malware Detection Appliance one of the most robust automated malware analysis solutions in the world. In addition, it lowers the cost and complexity of managing 2 distinct sandboxes or having to manually integrate multiple sandboxes into one system*, maximizing the ability to catch malicious code.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 1)

CUCKOO OPEN SOURCE SANDBOX

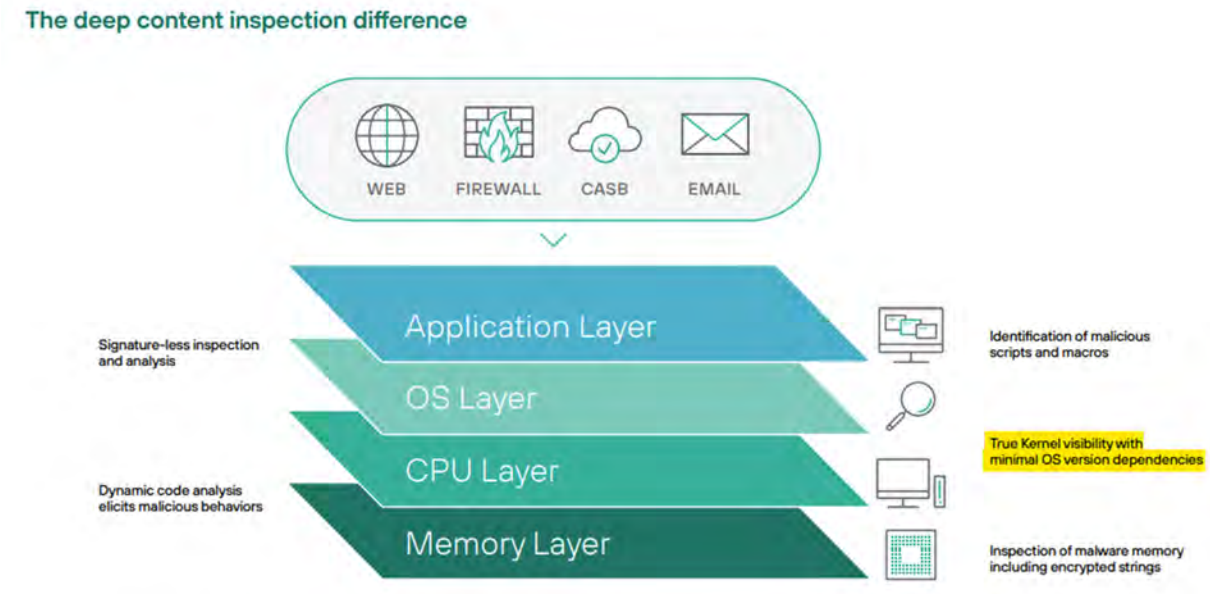
The sandboxing process begins with Forcepoint Advanced Malware Detection Appliance sending the file to Cuckoo's malware analysis system. Cuckoo can analyze the behavior of a wide array of malicious files (executables, document exploits, Java applets), as well as malicious websites, in Windows, OS X, Linux, and Android virtualized environments. Cuckoo is able to trace API calls and general behavior of the file, dump and analyze encrypted network traffic and perform advanced memory analysis of the infected virtualized system with integrated support for Volatility**. Followed by Cuckoo's analysis the file is sent to ThINK, Raytheon's proprietary sandbox.

ThINK PROPRIETARY SANDBOX

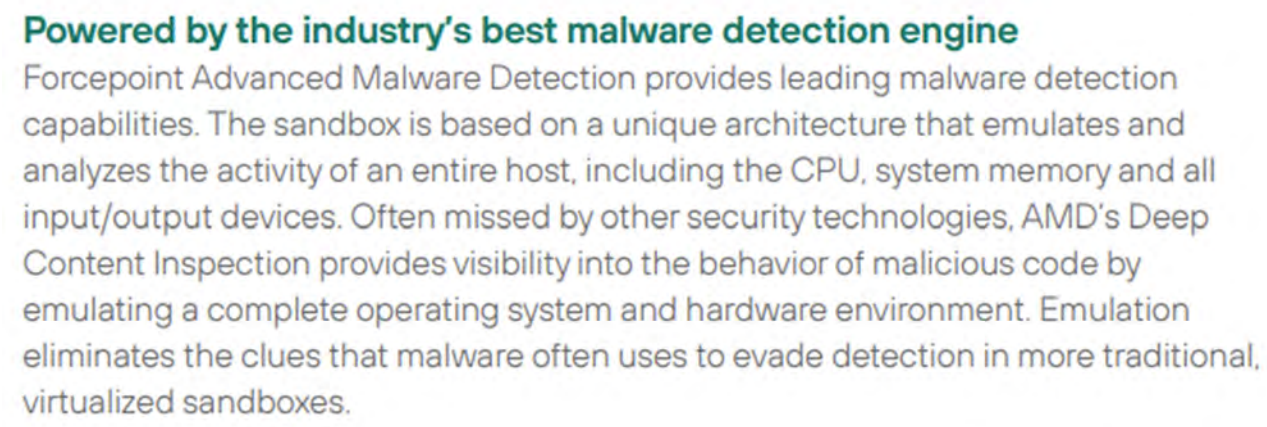
ThINK behavioral analysis is one of the most comprehensive sandboxes that exists in the world today. ThINK is a custom hypervisor that provides a fully integrated system-level debugger and an integrated malware sandbox. The sandbox executes files within a virtualized environment to contain and isolate malicious files before they can infect production systems. ThINK does not require any custom software on the guest OS. This helps avoid altering guest performance and prevents malware from detecting virtualization.

ThINK monitors all incoming and outgoing guest machine network traffic and flags hardware and software exceptions that are likely to indicate an attempted exploit and uses hardware breakpoints to monitor file, registry, process, thread creation and destruction from outside of the guest. ThINK also implements heap-spray analytics to highlight entropy changes to the process heap that are indicative of a heap-spray attack, a common component of browser and Acrobat Reader exploits.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 6)

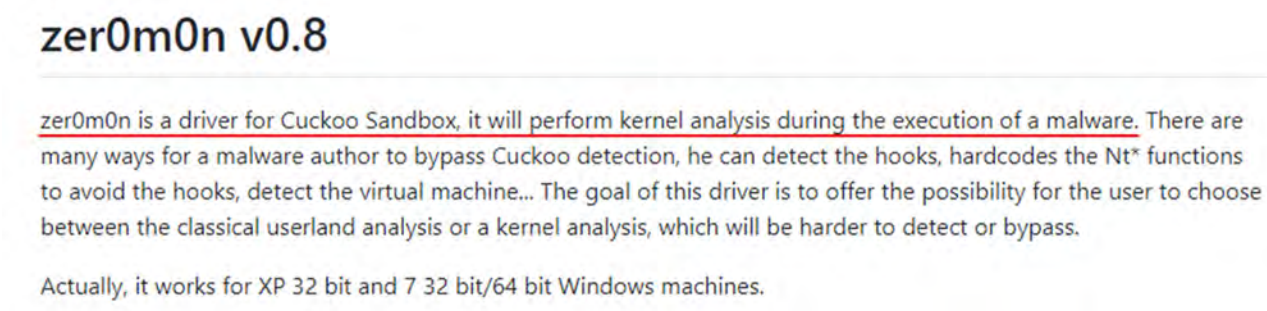


(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 2)



(Forcepoint, *Forcepoint Advanced Malware Detection*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 1)

295. The Cuckoo Sandbox utilizes a kernel level driver that "logs kernel activity during a malware execution in hooking the SSDT." The driver also logs function calls and parameters, along with taking "advantage of the callbacks provided by the windows API to log registry operations and kernel components loading (drivers)."



(https://github.com/conix-security/zer0m0n)

How it works

The driver logs kernel activity during a malware execution in hooking the SSDT, overwriting the pointers to new functions in order to log the calls and the parameters. It also takes advantage of the callbacks provided by the windows API to log registry operations and kernel components loading (drivers).

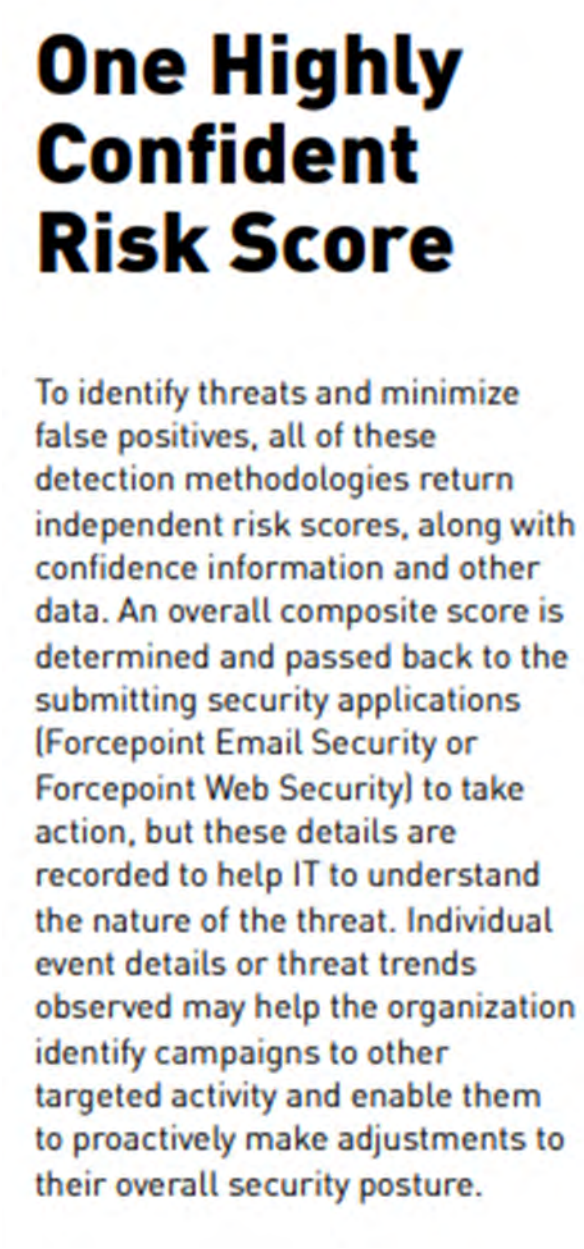
When submitting a new file to cuckoo and choosing the kernel analysis, Cuckoo (analyzer.py script) will start the process in a suspended state and send the PID to be monitored to the driver using an IOCTL. The driver will add it to its list of monitored processes and Cuckoo will then resume the execution of the malware.

For each activity traced by the driver (through the SSDT / callbacks) a log is produced and sent to a userland process ("logs_dispatcher.exe") using filter communication port. This one will be in charge of parsing the logs and send them to Cuckoo (analyzer.py) in bson format.

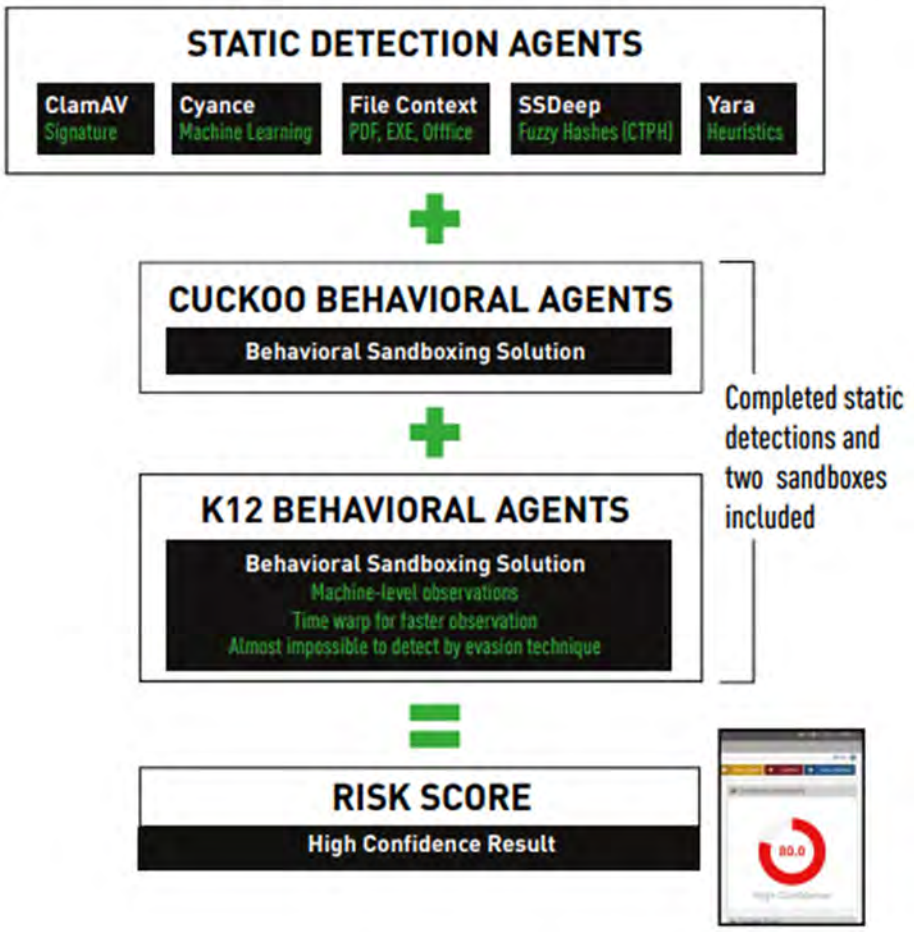
Code injections techniques are monitored by the driver: if a monitored process performs a known code-injection technique into another process, this process will be added to the monitored processes list.

(https://github.com/conix-security/zer0m0n); (*See also* https://github.com/conix-security/zer0m0n/blob/master/README.md)

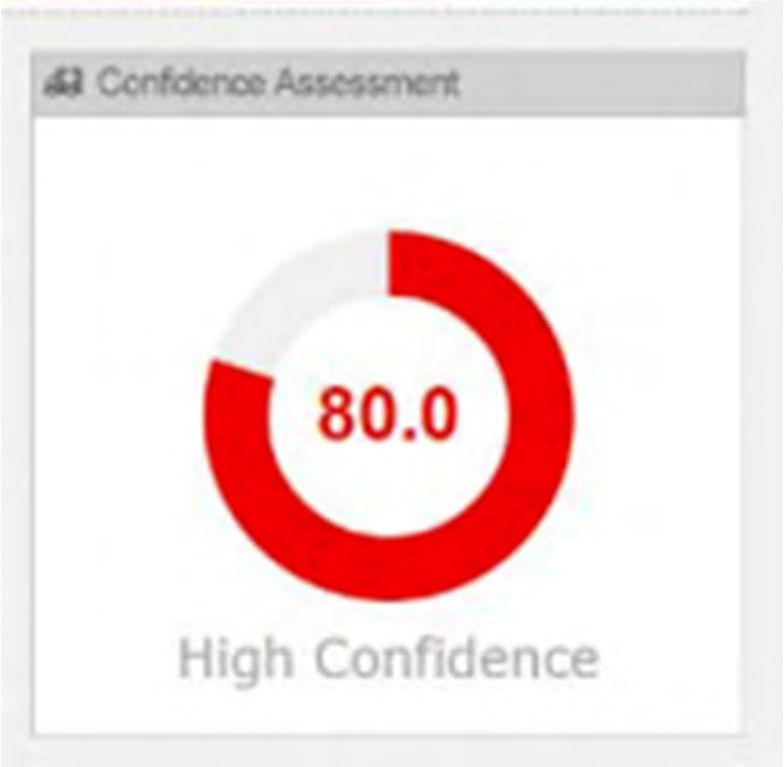
296. The Accused Products include "*analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware*." For example, Forcepoint Advanced Malware Detection utilizes various detection methodologies, including behavioral sandboxing, that return independent risk scores, along with confidence information and an overall composite score (which is associated with a level of risk) is determined, and passed back to the submitting security applications to take action.



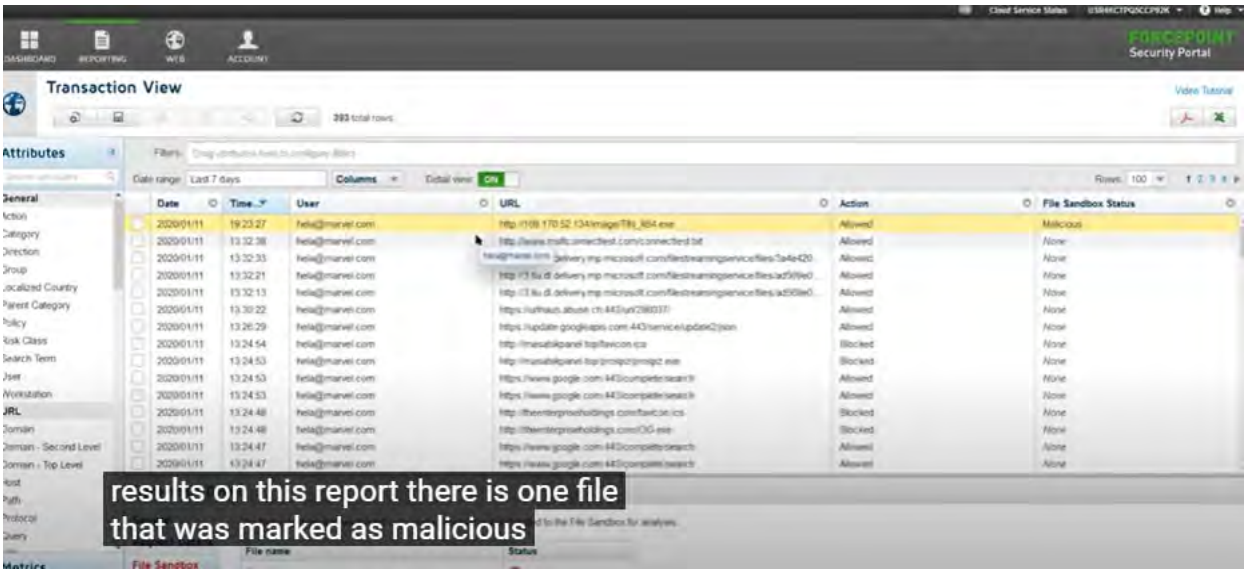
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)



(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 3)



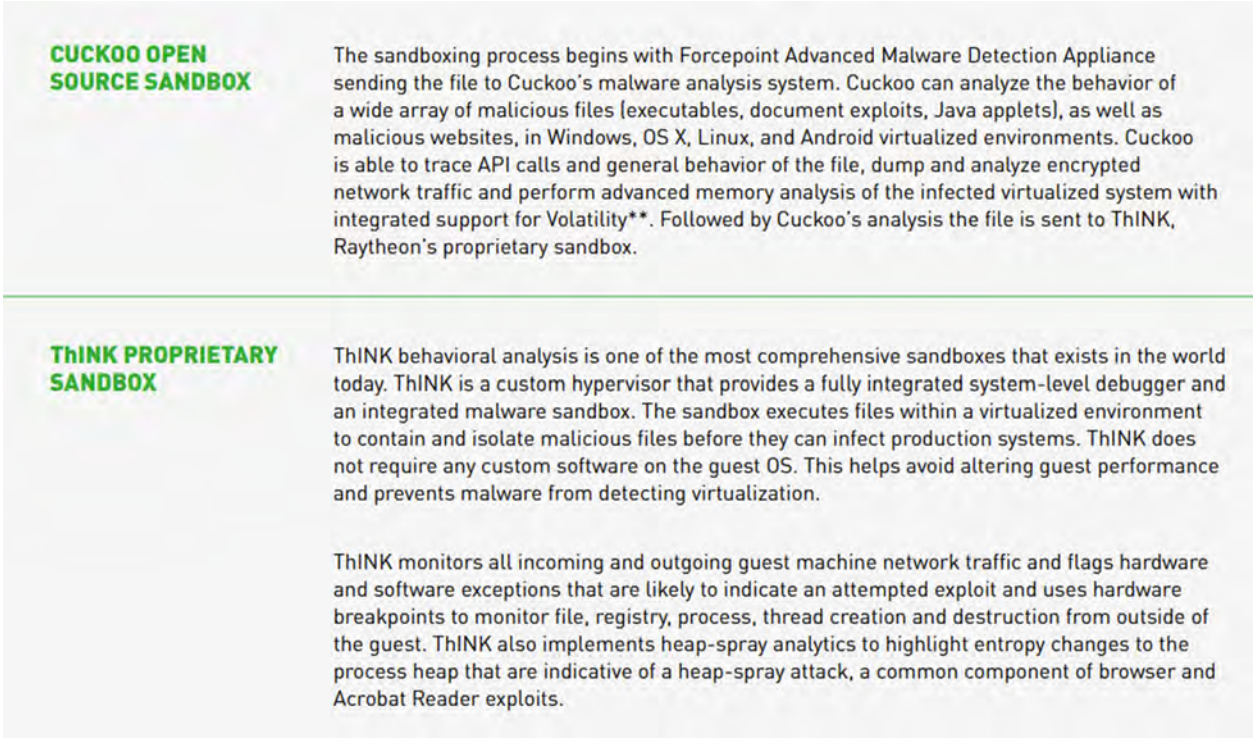
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)



(https://www.youtube.com/watch?v=hwW7NppQnCc (Time Stamp: 1:45))

297. The Accused Products include "*identifying, based upon the activity, an object residing on the computer that is a suspected pestware object*." For example, the Accused Products

utilize a Cuckoo sandbox that performed "advanced memory analysis of [an] infected virtualized system with integrated support for Volatility**. Followed by Cuckoo's analysis the file is sent to ThiNK." The ThiNK proprietary sandbox "executes files within a virtualized environment to contain and isolate malicious files before they can infect production systems."



CUCKOO OPEN SOURCE SANDBOX

The sandboxing process begins with Forcepoint Advanced Malware Detection Appliance sending the file to Cuckoo's malware analysis system. Cuckoo can analyze the behavior of a wide array of malicious files (executables, document exploits, Java applets), as well as malicious websites, in Windows, OS X, Linux, and Android virtualized environments. Cuckoo is able to trace API calls and general behavior of the file, dump and analyze encrypted network traffic and perform advanced memory analysis of the infected virtualized system with integrated support for Volatility**. Followed by Cuckoo's analysis the file is sent to ThINK, Raytheon's proprietary sandbox.

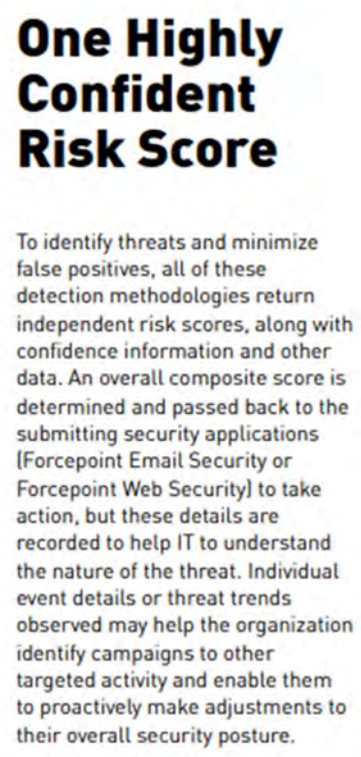
ThINK PROPRIETARY SANDBOX

ThINK behavioral analysis is one of the most comprehensive sandboxes that exists in the world today. ThINK is a custom hypervisor that provides a fully integrated system-level debugger and an integrated malware sandbox. The sandbox executes files within a virtualized environment to contain and isolate malicious files before they can infect production systems. ThINK does not require any custom software on the guest OS. This helps avoid altering guest performance and prevents malware from detecting virtualization.

ThINK monitors all incoming and outgoing guest machine network traffic and flags hardware and software exceptions that are likely to indicate an attempted exploit and uses hardware breakpoints to monitor file, registry, process, thread creation and destruction from outside of the guest. ThINK also implements heap-spray analytics to highlight entropy changes to the process heap that are indicative of a heap-spray attack, a common component of browser and Acrobat Reader exploits.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 6)

298. Further, as noted above, various detection methodologies return independent risk scores, along with confidence information and an overall composite score is determined and passed back to the submitting security applications to take action.



One Highly Confident Risk Score

To identify threats and minimize false positives, all of these detection methodologies return independent risk scores, along with confidence information and other data. An overall composite score is determined and passed back to the submitting security applications (Forcepoint Email Security or Forcepoint Web Security) to take action, but these details are recorded to help IT to understand the nature of the threat. Individual event details or threat trends observed may help the organization identify campaigns to other targeted activity and enable them to proactively make adjustments to their overall security posture.

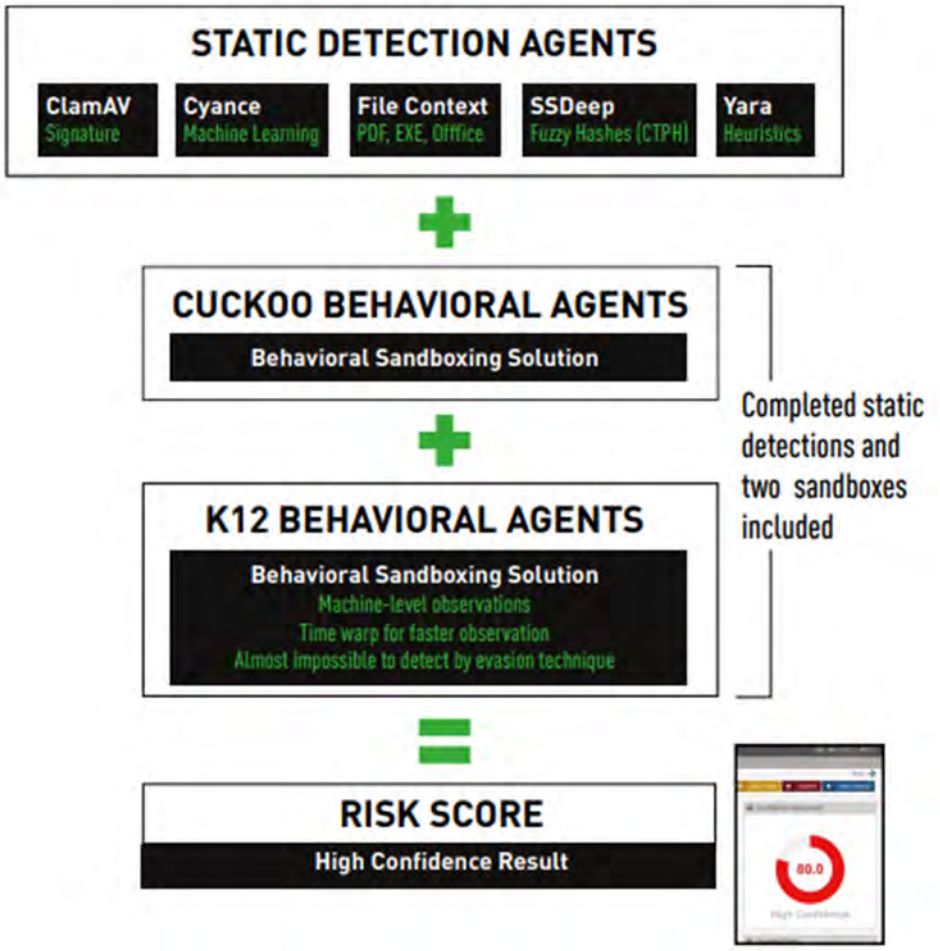
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)



Integration with Forcepoint Web and Email Security Ensures a Consolidated Threat View

Forcepoint Advanced Malware Detection Appliance integrates seamlessly with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. In turn, this reduces risk and boosts efficacy of existing security investments. In addition to integration with TRITON Architecture, Forcepoint Advanced Malware Detection Appliance provides security practitioners with the ability to highly customize their malware analysis as the company grows and malware evolves. This is via Forcepoint Advanced Malware Detection Appliance's standalone threat management console.

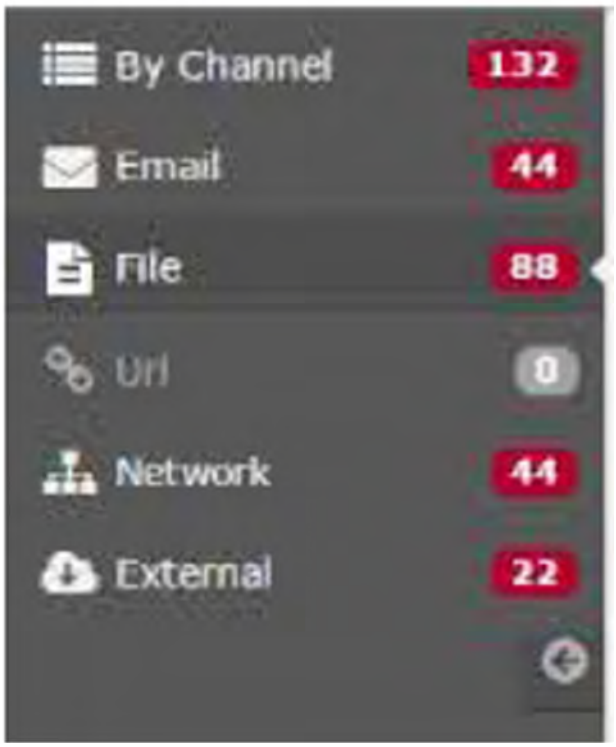
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 3)



(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 3)

299. The Accused Products include "*accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from.*" For example, the Accused Products "can analyze the behavior of a wide array of malicious files (executables, document exploits, Java applets), as well as malicious websites, in Windows, OS X, Linux, and Android virtualized environments."

300. Forcepoint Advanced Malware Detection Appliance also integrates with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. An "Events section" includes a dynamic and interactive screen, providing information by threat channel category such as Email, File, URL, Network or External.

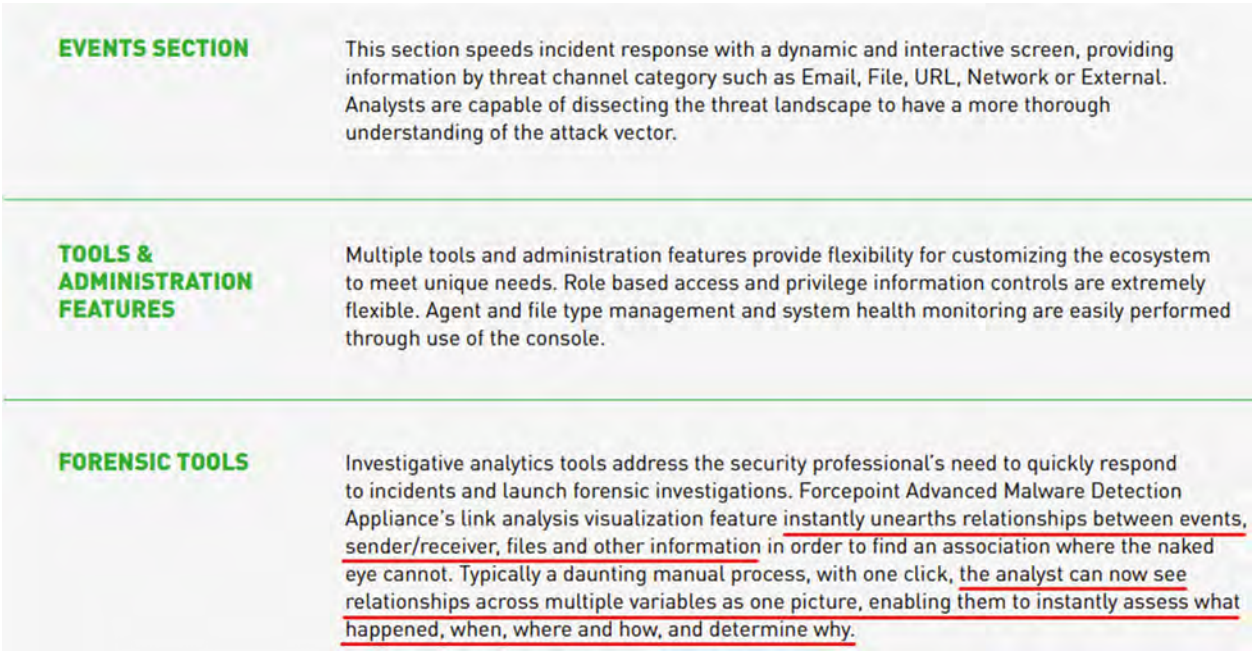


(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)

Integration with Forcepoint Web and Email Security Ensures a Consolidated Threat View

Forcepoint Advanced Malware Detection Appliance integrates seamlessly with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. In turn, this reduces risk and boosts efficacy of existing security investments. In addition to integration with TRITON Architecture, Forcepoint Advanced Malware Detection Appliance provides security practitioners with the ability to highly customize their malware analysis as the company grows and malware evolves. This is via Forcepoint Advanced Malware Detection Appliance's standalone threat management console.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 8)



EVENTS SECTION	This section speeds incident response with a dynamic and interactive screen, providing information by threat channel category such as Email, File, URL, Network or External. Analysts are capable of dissecting the threat landscape to have a more thorough understanding of the attack vector.
TOOLS & ADMINISTRATION FEATURES	Multiple tools and administration features provide flexibility for customizing the ecosystem to meet unique needs. Role based access and privilege information controls are extremely flexible. Agent and file type management and system health monitoring are easily performed through use of the console.
FORENSIC TOOLS	Investigative analytics tools address the security professional's need to quickly respond to incidents and launch forensic investigations. Forcepoint Advanced Malware Detection Appliance's link analysis visualization feature instantly unearths relationships between events, sender/receiver, files and other information in order to find an association where the naked eye cannot. Typically a daunting manual process, with one click, the analyst can now see relationships across multiple variables as one picture, enabling them to instantly assess what happened, when, where and how, and determine why.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 9)

301. The Accused Products include "*reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware*." For example, Forcepoint's Advanced Malware Detection engine communicates identified pestware object and other analysis results to an incident response team, which further analyzes the pestware object and determines the source of the pestware object, potentially related pestware, and various other potential threats.

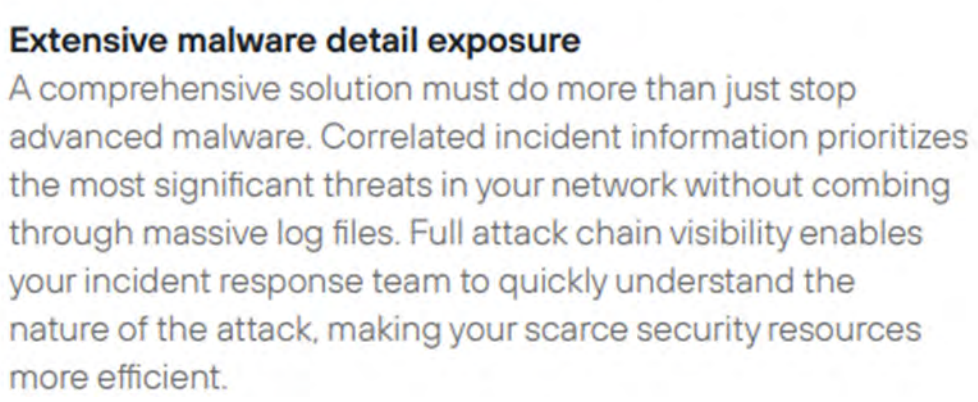
Outcome

- Increase protection for your organization with proven security effectiveness.
- Enable improved situational awareness for incident response teams.
- Achieve better and faster ROI with integrations.

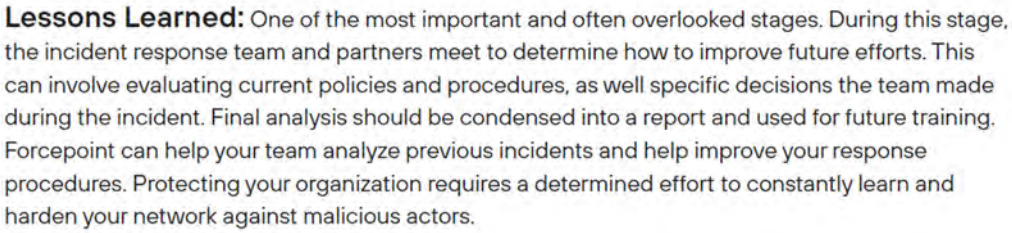
Extensive malware detail exposure

A comprehensive solution must do more than just stop advanced malware. Correlated incident information prioritizes the most significant threats in your network without combing through massive log files. Full attack chain visibility enables your incident response team to quickly understand the nature of the attack, making your scarce security resources more efficient.

Available as a cloud service for high availability, scalability, low maintenance and other SaaS benefits, on-premises for cloud-adverse organizations, or even deployed as an air gapped solution with Forcepoint NGFW for physically isolated network requirements. Forcepoint AMD is the perfect complement to your Forcepoint CASB, NGFW, Web Security or Email Security solution. It provides unparalleled threat detection, as well as consistent threat forensic information, to optimize incident response teams.



(Forcepoint, *Forcepoint Advanced Malware Detection*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 2)



(*See* Forcepoint, *What is Incident Response?*, https://www.forcepoint.com/cyber-edu/incident-response)

302. Each claim in the ’243 Patent recites an independent invention. Neither claim 1, described above, nor any other individual claim is representative of all claims in the ’243 Patent.

303. Forcepoint has been aware of the ’243 Patent since at least the date it received a copy of this Amended Complaint.

304. Defendant directly infringes at least claim 1 of the ’243 Patent, either literally or under the doctrine of equivalents, by practicing the elements described above. For example, on information and belief, Defendant makes and uses the claimed system in an infringing manner as described above by running this software and system to protect its own computer and network operations. On information and belief, Defendant also makes and uses the claimed system in an

infringing manner when testing the operation of the Accused Products and corresponding systems. As another example, Defendant makes and uses the claimed system when providing or administering services to third parties, customers, and partners using the Accused Products.

305. Defendant's partners, customers, and end users of its Accused Products and corresponding systems and services directly infringe at least claim 1 of the '243 Patent, literally or under the doctrine of equivalents, at least by using the Accused Products and corresponding systems and services, as described above.

306. Defendant has actively induced and is actively inducing infringement of at least claim 1 of the '243 Patent with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). For example, Defendant encourages and induces customers with specific intent to use Forcepoint's security software in a manner that infringes claim 1 of the '243 Patent at least by offering and providing software and/or hardware that embodies the system that infringes claim 1 when installed and operated by the customer, and by engaging in activities relating to selling, marketing, advertising, promotion, installation, support, and distribution of the Accused Products, including the activities described below.

307. Defendant encourages, instructs, directs, and/or requires third parties – including its certified partners and/or customers – to practice the claims using the software, services, and systems in infringing ways, as described above.

308. Defendant further encourages and induces its customers to infringe claim 1 of the '243 Patent: 1) by making its security services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users, and 2) through activities relating to marketing, advertising,

promotion, installation, support, and distribution of the Accused Products, including its Forcepoint security software, and services in the United States. *See, e.g.*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf; https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf; https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.7.0/GUID-41D85576-7C24-4BA8-8605-710F52790CD8.html.

309. For example, on information and belief, Defendant shares instructions, guides, and manuals, which advertise and instruct third parties on how to use the software as described above, including at least customers and partners. *Id.* On further information and belief, Defendant also provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that deploy and use the Accused Products in an infringing manner. *Id.*

310. Defendant and/or its partners recommend and sell the Accused Products and provide technical support for the installation, implementation, integration, and ongoing operation of the Accused Products for each individual customer. On information and belief, each customer enters into a contractual relationship with Defendant and/or one of its partners, which obligates each customer to perform certain actions in order to use the Accused Products. Further, in order to receive the benefit of Defendant and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the '243 Patent.

311. Further, as the entity that provides installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Product remains operational for each customer through ongoing technical support, on information and belief, Defendant and/or its partners affirmatively aid and abet each customer's use of the Accused Products in a manner that infringes the claimed system of, and infringes, the '243 Patent.

312. Defendant also contributes to the infringement of its partners, customers, and end-users of the Accused Products by providing within the United States or importing the Accused Products into the United States, which are for use in practicing, and under normal operation practice, systems claimed in the Asserted Patents, constituting a material part of the inventions claimed, and not a staple article or commodity of commerce suitable for substantial non-infringing uses.

313. Indeed, as shown above, the Accused Products have no substantial non-infringing uses because the accused functionality, including malware detection, sandboxing, and related functionality, are integral parts of the Accused Products and must be performed for the Accused Products to perform their intended purpose. These processes are continually running when the system is in use and, on information and belief, cannot be removed or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown above, or without the system and components identified above that practice the '243 Patent, that functionality could not be performed.

314. Additionally, the accused functionality, including the malware detection and sandboxing described below, itself has no substantial non-infringing uses because the components, modules and methods identified above are a necessary part of that functionality. For example,

without the Accused Products' kernel level driver for monitoring API calls, the Accused Products could not deploy sandbox features to detect malware in a computer. These processes are continually running when the system is in use and, on information and belief, cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the '243 Patent, that functionality could not be performed.

315. In addition, as shown in the detailed analysis above, the products, systems, modules, and methods provided by Defendant constitute a material part of the invention—indeed, they provide all the components, modules, and features that perform the claimed methods and systems. For example, the Accused Products and accused functionalities (including the machine learning static analysis) constitute a material part of the inventions claimed because such analysis is integral to the processes identified above as recited in the claims of the '243 Patent. None of these products are staple goods—they are sophisticated and customized cyber security products, methods, and systems.

316. On information and belief, the infringing actions of each partner, customer, and/or end-user of the Accused Products are attributable to Defendant. For example, on information and belief, Defendant directs and controls the activities or actions of its partners or others in connection with the Accused Products by contractual agreement or otherwise requiring partners or others to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Defendant further directs and controls the operation of devices executing the Accused Products by programming the software which, when executed by a customer or end user, perform the claimed method of at least claim 1 of the '243 Patent.

317. Plaintiffs have suffered and continue to suffer damages, including lost profits, as a result of Defendant's infringement of the '243 Patent. Defendant is therefore liable to Plaintiffs under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiffs for Defendant's infringement, but no less than a reasonable royalty.

318. Plaintiffs will continue to suffer irreparable harm unless this Court preliminarily and permanently enjoins Defendant, its agents, employees, representatives, and all others acting in concert with Defendant from infringing the '243 Patent.

319. Defendant's infringement of the '243 Patent is knowing and willful. Defendant acquired knowledge of the '243 Patent and of the specific conduct that constitutes infringement of the '243 Patent at least when it received a copy of this Amended Complaint. Defendant continues to engage in infringing activities after the filing of this amended complaint, including by selling, making, configuring, and installing the Accused Products and performing the accused functionality, and by engaging in activities that constitute inducing infringement and contributory infringement as described above.

320. On information and belief, despite Defendant's knowledge of the Asserted Patents, Defendant made the deliberate decision to sell products and services that it knew infringe these patents. Defendant's continued infringement of the '243 Patent with knowledge of the '243 Patent constitutes willful infringement.

EIGHTH CAUSE OF ACTION
(INFRINGEMENT OF THE '932 PATENT)

321. Plaintiffs reallege and incorporate by reference the allegations of the preceding paragraphs of this Amended Complaint.

322. Forcepoint has infringed and continues to infringe one or more claims of the '932 Patent in violation of 35 U.S.C. § 271 in this judicial District and elsewhere in the United States

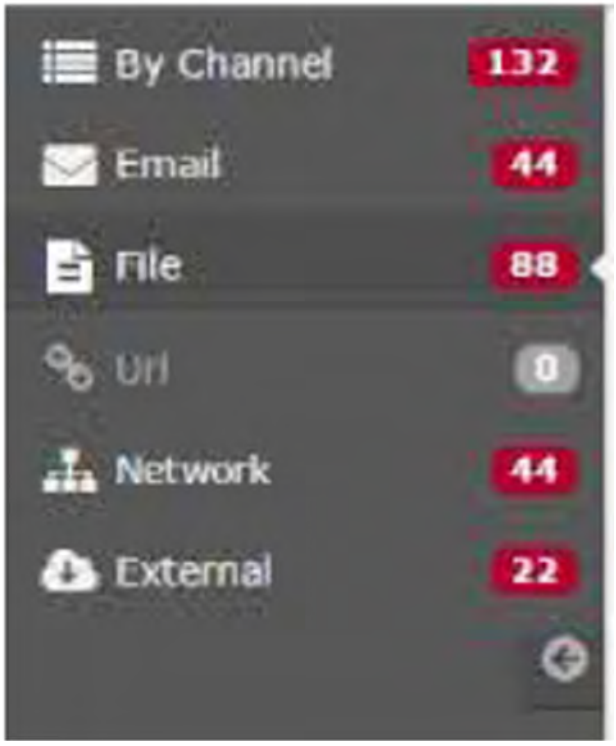
and will continue to do so unless enjoined by this Court. The Accused Products, including features of the Forcepoint's Advanced Malware Detection system (as well as any other products that include the features described below), when used for their ordinary and customary purposes, practice each element of at least claim 1 of the '932 Patent, as demonstrated below.

323. Claim 1 of the '932 Patent recites:

> 1. A method for identifying an origin of activity on a computer that is indicative of pestware comprising:
>
> monitoring, using a kernel-mode driver, API call activity on the computer;
>
> storing information related to the API call activity in a log;
>
> analyzing, heuristically, the API call activity to determine whether one or more weighted factors associated with the API call activity exceeds a threshold;
>
> identifying, based upon the API call activity, a suspected pestware object on the computer;
>
> identifying, in response to the identifying the suspected pestware object, a reference to an identity of an externally networked source of the suspected pestware object; and
>
> reporting the identity of the externally networked source to an externally networked pestware research entity, wherein
>
> the identity of the externally networked source is selected from a group consisting of an I.P. address, a URL, an email client and a program.

324. The Accused Products include "*a method for identifying an origin of activity on a computer that is indicative of pestware*." For example, Forcepoint Advanced Malware Detection (AMD) provides malware detection capabilities. The sandbox used by AMD is based on a unique architecture that emulates and analyzes the activity of a host, including the CPU, system memory, all input/output devices, a complete operating system, and hardware environment. Forcepoint Advanced Malware Detection Appliance integrates with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk

assessments into one platform. An "Events section" facilitates incident response with a user interface, providing information by threat channel category such as Email, File, URL, Network or External.



(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)

Forcepoint Advanced Malware Detection (AMD) leverages proven technology to detect zero-day and other advanced malware. Using Deep Content Inspection technology, Forcepoint AMD emulates an entire host, interacting with malware to expose and observe a malicious object's possible actions. These include advanced evasion techniques, O/S or application specific threats, dormant code analysis and even CPU and in-memory activity.

(Forcepoint Advanced Malware Detection Appliance 3.4, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 01)

Forcepoint Advanced Malware Detection (AMD) leverages proven Lastline technology to detect zero-day and other advanced malware. Using Lastline's Deep Content Inspection technology, Forcepoint AMD emulates an entire host, interacting with malware to expose and observe a malicious object's possible actions. These include advanced evasion techniques, O/S or application specific threats, dormant code analysis and even CPU and in-memory activity.

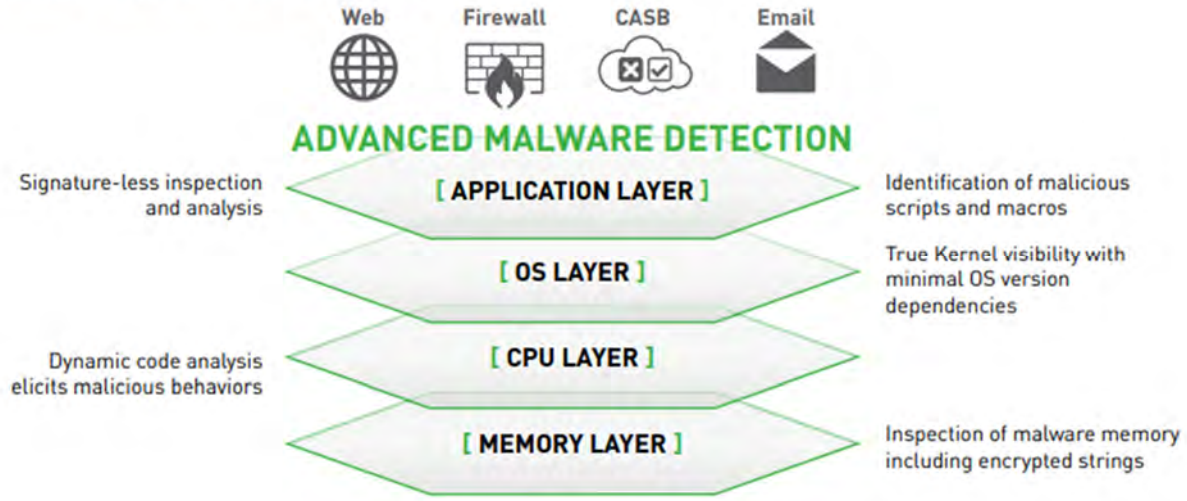
In a recent study of Breach Detection Systems (BDS) by NSS Labs, Lastline was named the most effective advanced malware detection system and was the only product to achieve **100% detection** with **zero false positives.**

(Forcepoint, *FORCEPOINT Advanced Malware Detection*, https://www.guardsense.com/datasheets/datasheet_forcepoint_advanced_malware_detection_lastline_en_0.pdf at 1)

The industry's best malware detection engine

Forcepoint chose Lastline as a partner for Advanced Malware Detection because of their leading malware detection capabilities (as demonstrated in the NSS Labs study). The sandbox is based on a unique architecture that emulates and analyzes the activity of an entire host, including the CPU, system memory and all input/output devices. Often missed by other security technologies, Lastline's Deep Content Inspection provides visibility into the behavior of malicious code by emulating a complete operating system and hardware environment. Emulation eliminates the clues that malware often uses to evade detection in more traditional, virtualized sandboxes.

THE DEEP CONTENT INSPECTION DIFFERENCE



(Forcepoint, *FORCEPOINT Advanced Malware Detection*, https://www.guardsense.com/datasheets/datasheet_forcepoint_advanced_malware_detection_lastline_en_0.pdf at 2)

Integrated with Forcepoint defenses across all key threat vectors

AMD is available as a fully integrated option for Forcepoint CASB, NGFW, Web Security and Email security. In this integration, Forcepoint's core solutions first assess the broader context of an internet transaction for potential indicators of compromise. After performing static analysis of suspicious files, AMD can be called upon to perform the deep behavioral analysis necessary to identify zero-day threats and other modern malware.

Available as a cloud service or on-premises solution (for more cautious or otherwise cloud-adverse organizations), Forcepoint AMD is the perfect complement to your Forcepoint CASB, NGFW, Web Security or Email security solution. It provides unparalleled threat detection, as well as consistent threat forensic information, to optimize incident response teams.

Forcepoint AMD will give you all the information you need—regardless of the threat vector—while 'zero-false positives' means you'll spend your valuable time working against true threats. Regardless of your size or industry, Forcepoint provides the comprehensive security solutions you need to challenge today's fast evolving, highly evasive threats.

(Forcepoint, *FORCEPOINT Advanced Malware Detection*, https://www.guardsense.com/datasheets/datasheet_forcepoint_advanced_malware_detection_lastline_en_0.pdf at 2)

Malware Interaction

Sandbox-only solutions provide a relatively static environment, limiting the malicious 'behavior' they may uncover. Because Forcepoint Advanced Malware Detection interacts with malware, it observes every action that it might take, even when those actions are delegated to the operating system or other programs. In addition, this tool identifies potentially malicious 'dormant code' that the malware does not execute.

(Forcepoint, *Forcepoint Advanced Malware Detection*, https://www.guardsense.com/AMD.asp at 2)

Integration with Forcepoint Web and Email Security Ensures a Consolidated Threat View

Forcepoint Advanced Malware Detection Appliance integrates seamlessly with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. In turn, this reduces risk and boosts efficacy of existing security investments. In addition to integration with TRITON Architecture, Forcepoint Advanced Malware Detection Appliance provides security practitioners with the ability to highly customize their malware analysis as the company grows and malware evolves. This is via Forcepoint Advanced Malware Detection Appliance's standalone threat management console.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 8)

EVENTS SECTION	This section speeds incident response with a dynamic and interactive screen, providing information by threat channel category such as Email, File, URL, Network or External. Analysts are capable of dissecting the threat landscape to have a more thorough understanding of the attack vector.
TOOLS & ADMINISTRATION FEATURES	Multiple tools and administration features provide flexibility for customizing the ecosystem to meet unique needs. Role based access and privilege information controls are extremely flexible. Agent and file type management and system health monitoring are easily performed through use of the console.
FORENSIC TOOLS	Investigative analytics tools address the security professional's need to quickly respond to incidents and launch forensic investigations. Forcepoint Advanced Malware Detection Appliance's link analysis visualization feature instantly unearths relationships between events, sender/receiver, files and other information in order to find an association where the naked eye cannot. Typically a daunting manual process, with one click, the analyst can now see relationships across multiple variables as one picture, enabling them to instantly assess what happened, when, where and how, and determine why.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 9)

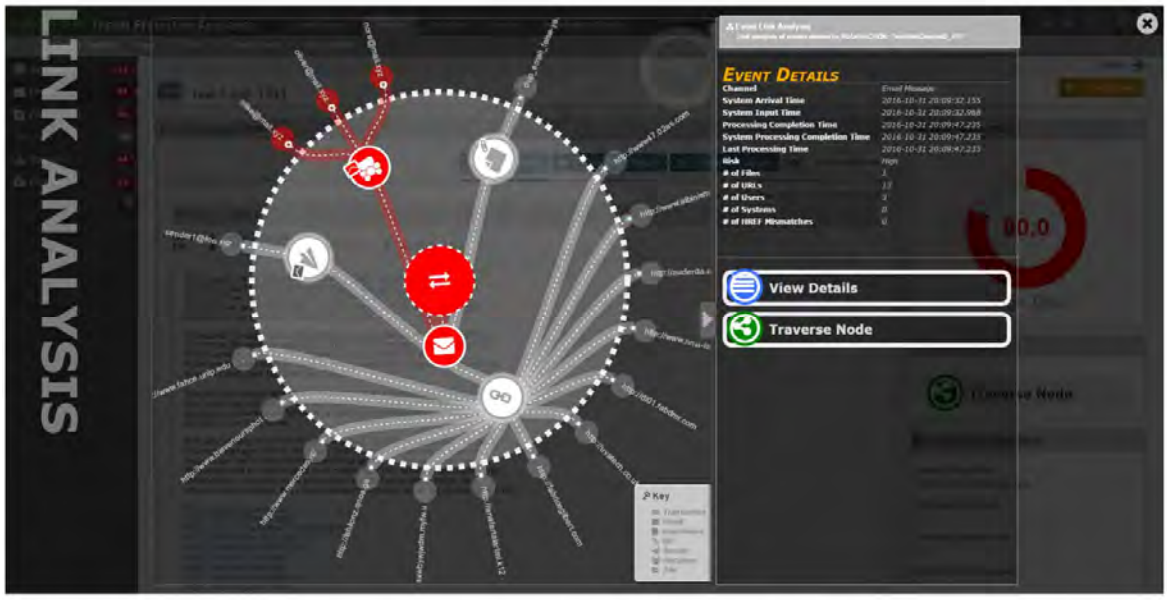


FIG 3: Forensic Tools provide advanced visualizations for teams to quickly and easily examine complex threat data and bring forward hidden relationships between events, sender, files and other information.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 10)

325. The Accused Products include "*monitoring, using a kernel-mode driver, API call activity on the computer*." For example, Forcepoint Advanced Malware Detection Appliance utilizes Cuckoo's malware analysis system with Raytheon's defense-grade ThiNK, sandboxing technologies which adds dual behavioral analysis to the static analysis processes as the sandboxing process begins with sending the file to Cuckoo's malware analysis system which can analyze the behavior of a wide array of malicious files and is able to trace API calls, followed by Cuckoo's analysis, the file is sent to ThINK, Raytheon's proprietary sandbox, which is a custom hypervisor to contain and isolate malicious files before they can infect production systems.

Malware Interaction

Sandbox-only solutions provide a relatively static environment, limiting the malicious 'behavior' they may uncover. Because Forcepoint Advanced Malware Detection interacts with malware, it observes every action that it might take, even when those actions are delegated to the operating system or other programs. In addition, this tool identifies potentially malicious 'dormant code' that the malware does not execute.

(Forcepoint, *Forcepoint Advanced Malware Detection (AMD)*, https://www.guardsense.com/AMD.asp at 02)

DUAL SANDBOXING IDENTIFIES THE MOST ADVANCED ADVERSARIES IN THE WORLD:

Forcepoint Advanced Malware Detection Appliance leverages best-of-breed open source technology Cuckoo with Raytheon's defense-grade ThiNK, proprietary sandboxing technology. Adding dual behavioral analysis to the above mentioned seven step static analysis process makes Forcepoint Advanced Malware Detection Appliance one of the most robust automated malware analysis solutions in the world. In addition, it lowers the cost and complexity of managing 2 distinct sandboxes or having to manually integrate multiple sandboxes into one system*, maximizing the ability to catch malicious code.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 1)

CUCKOO OPEN SOURCE SANDBOX

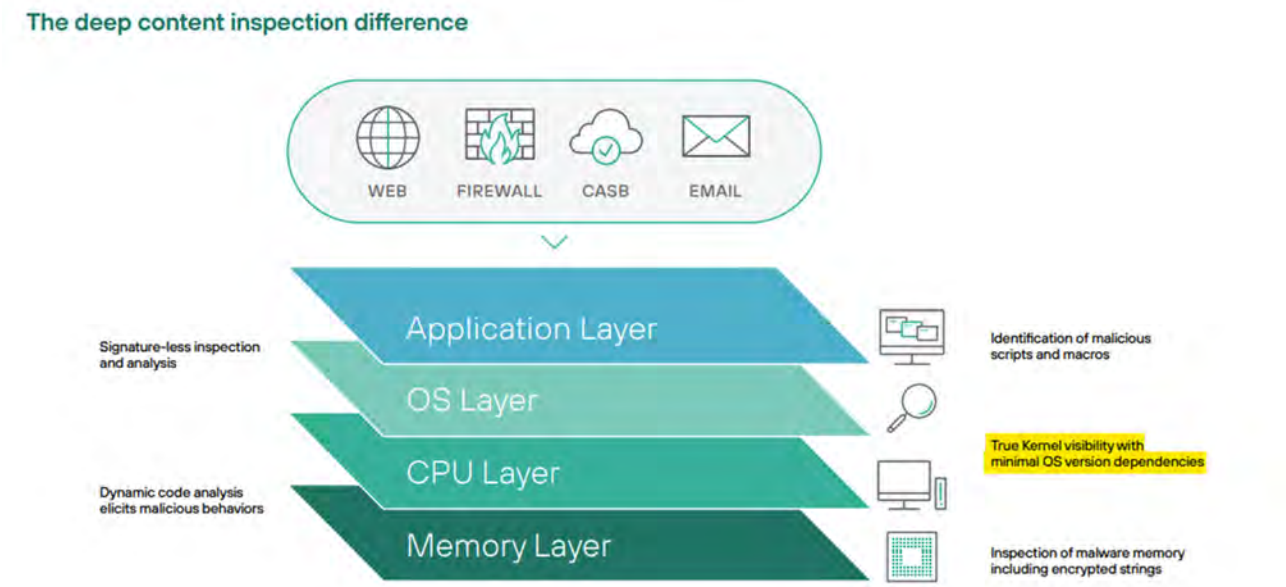
The sandboxing process begins with Forcepoint Advanced Malware Detection Appliance sending the file to Cuckoo's malware analysis system. Cuckoo can analyze the behavior of a wide array of malicious files (executables, document exploits, Java applets), as well as malicious websites, in Windows, OS X, Linux, and Android virtualized environments. Cuckoo is able to trace API calls and general behavior of the file, dump and analyze encrypted network traffic and perform advanced memory analysis of the infected virtualized system with integrated support for Volatility**. Followed by Cuckoo's analysis the file is sent to ThINK, Raytheon's proprietary sandbox.

ThINK PROPRIETARY SANDBOX

ThINK behavioral analysis is one of the most comprehensive sandboxes that exists in the world today. ThINK is a custom hypervisor that provides a fully integrated system-level debugger and an integrated malware sandbox. The sandbox executes files within a virtualized environment to contain and isolate malicious files before they can infect production systems. ThINK does not require any custom software on the guest OS. This helps avoid altering guest performance and prevents malware from detecting virtualization.

ThINK monitors all incoming and outgoing guest machine network traffic and flags hardware and software exceptions that are likely to indicate an attempted exploit and uses hardware breakpoints to monitor file, registry, process, thread creation and destruction from outside of the guest. ThINK also implements heap-spray analytics to highlight entropy changes to the process heap that are indicative of a heap-spray attack, a common component of browser and Acrobat Reader exploits.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 6)



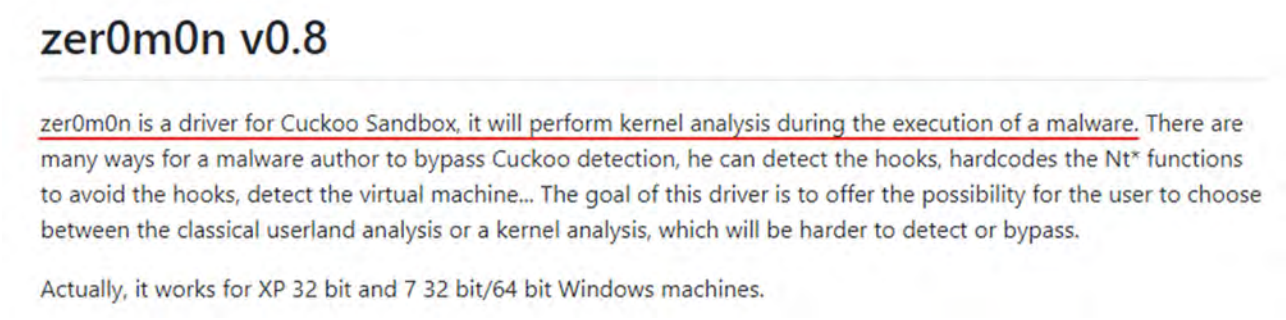
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 2)

Powered by the industry's best malware detection engine

Forcepoint Advanced Malware Detection provides leading malware detection capabilities. The sandbox is based on a unique architecture that emulates and analyzes the activity of an entire host, including the CPU, system memory and all input/output devices. Often missed by other security technologies, AMD's Deep Content Inspection provides visibility into the behavior of malicious code by emulating a complete operating system and hardware environment. Emulation eliminates the clues that malware often uses to evade detection in more traditional, virtualized sandboxes.

(Forcepoint, *Forcepoint Advanced Malware Detection*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 1)

326. With the use of the Cuckoo sandbox, the Accused Products use a kernel-mode driver to monitor various activities on the computer, including API calls.

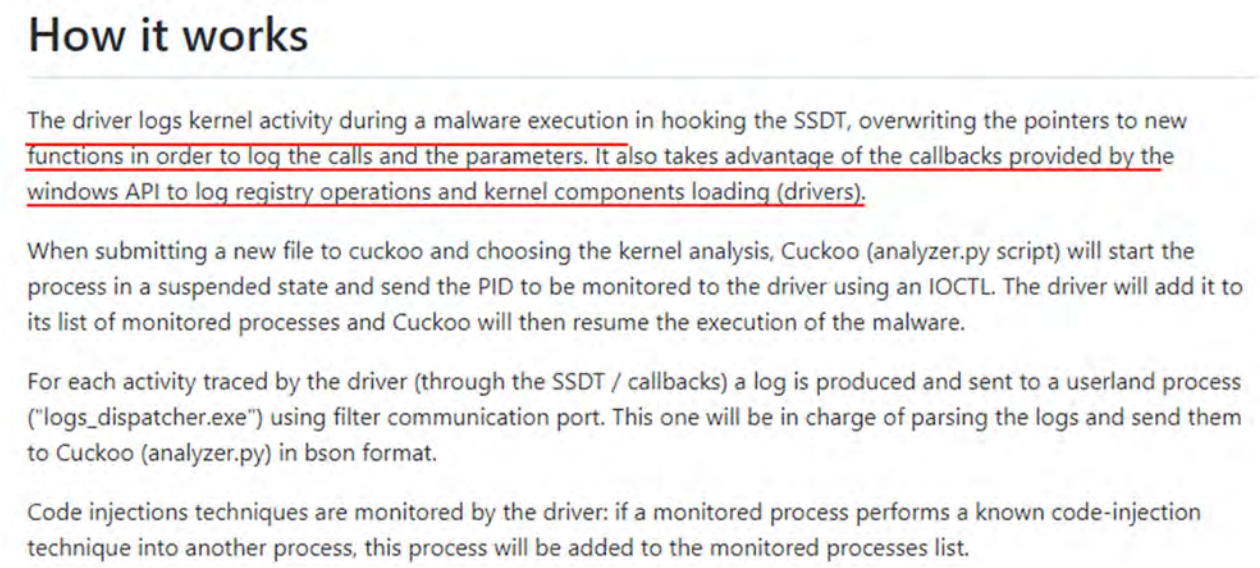


zer0m0n v0.8

zer0m0n is a driver for Cuckoo Sandbox, it will perform kernel analysis during the execution of a malware. There are many ways for a malware author to bypass Cuckoo detection, he can detect the hooks, hardcodes the Nt* functions to avoid the hooks, detect the virtual machine... The goal of this driver is to offer the possibility for the user to choose between the classical userland analysis or a kernel analysis, which will be harder to detect or bypass.

Actually, it works for XP 32 bit and 7 32 bit/64 bit Windows machines.

(https://github.com/conix-security/zer0m0n)



How it works

The driver logs kernel activity during a malware execution in hooking the SSDT, overwriting the pointers to new functions in order to log the calls and the parameters. It also takes advantage of the callbacks provided by the windows API to log registry operations and kernel components loading (drivers).

When submitting a new file to cuckoo and choosing the kernel analysis, Cuckoo (analyzer.py script) will start the process in a suspended state and send the PID to be monitored to the driver using an IOCTL. The driver will add it to its list of monitored processes and Cuckoo will then resume the execution of the malware.

For each activity traced by the driver (through the SSDT / callbacks) a log is produced and sent to a userland process ("logs_dispatcher.exe") using filter communication port. This one will be in charge of parsing the logs and send them to Cuckoo (analyzer.py) in bson format.

Code injections techniques are monitored by the driver: if a monitored process performs a known code-injection technique into another process, this process will be added to the monitored processes list.

(https://github.com/conix-security/zer0m0n); (*See also* https://github.com/conix-security/zer0m0n/blob/master/README.md)

327. The Accused Products include "*storing information related to the API call activity in a log*." For example, the Forcepoint Advanced Malware Detection system monitors API call activity, among other events and threats, and stores the corresponding data in a log. For instance, the Cuckoo sandbox includes a driver that "logs kernel activity during a malware execution in hooking the SSDT, overwriting the pointers to new functions in order to log the calls and parameters. It also takes advantage of the callbacks provided by the windows API to log registry operations and kernel components loading (drivers)."

zer0m0n v0.8

zer0m0n is a driver for Cuckoo Sandbox, it will perform kernel analysis during the execution of a malware. There are many ways for a malware author to bypass Cuckoo detection, he can detect the hooks, hardcodes the Nt* functions to avoid the hooks, detect the virtual machine... The goal of this driver is to offer the possibility for the user to choose between the classical userland analysis or a kernel analysis, which will be harder to detect or bypass.

Actually, it works for XP 32 bit and 7 32 bit/64 bit Windows machines.

(https://github.com/conix-security/zer0m0n)

How it works

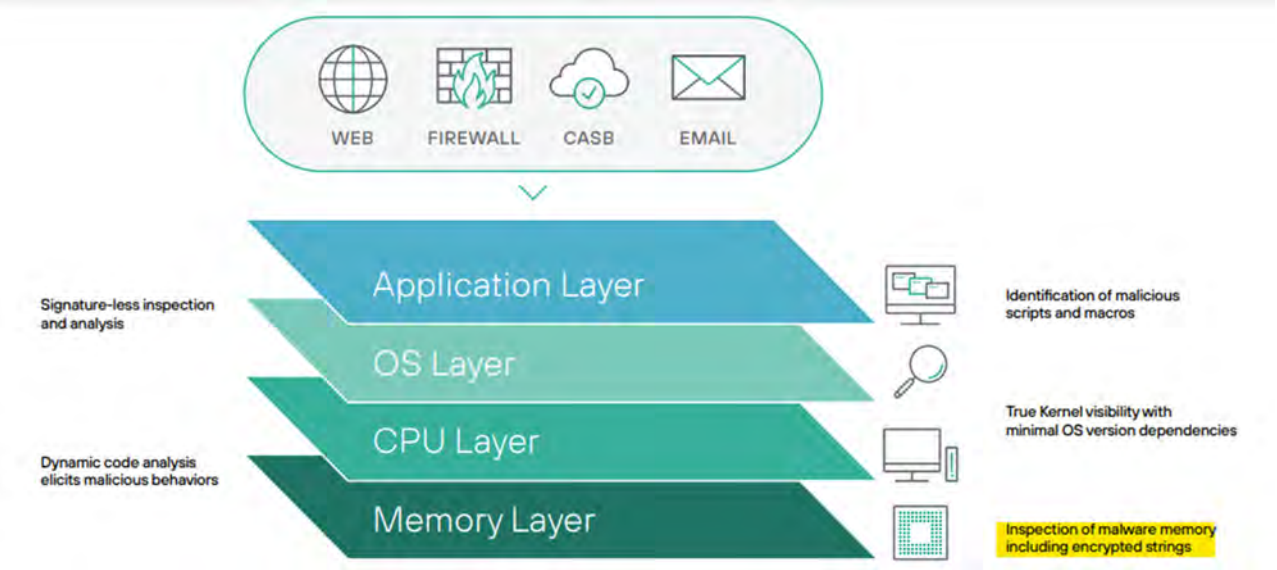
The driver logs kernel activity during a malware execution in hooking the SSDT, overwriting the pointers to new functions in order to log the calls and the parameters. It also takes advantage of the callbacks provided by the windows API to log registry operations and kernel components loading (drivers).

When submitting a new file to cuckoo and choosing the kernel analysis, Cuckoo (analyzer.py script) will start the process in a suspended state and send the PID to be monitored to the driver using an IOCTL. The driver will add it to its list of monitored processes and Cuckoo will then resume the execution of the malware.

For each activity traced by the driver (through the SSDT / callbacks) a log is produced and sent to a userland process ("logs_dispatcher.exe") using filter communication port. This one will be in charge of parsing the logs and send them to Cuckoo (analyzer.py) in bson format.

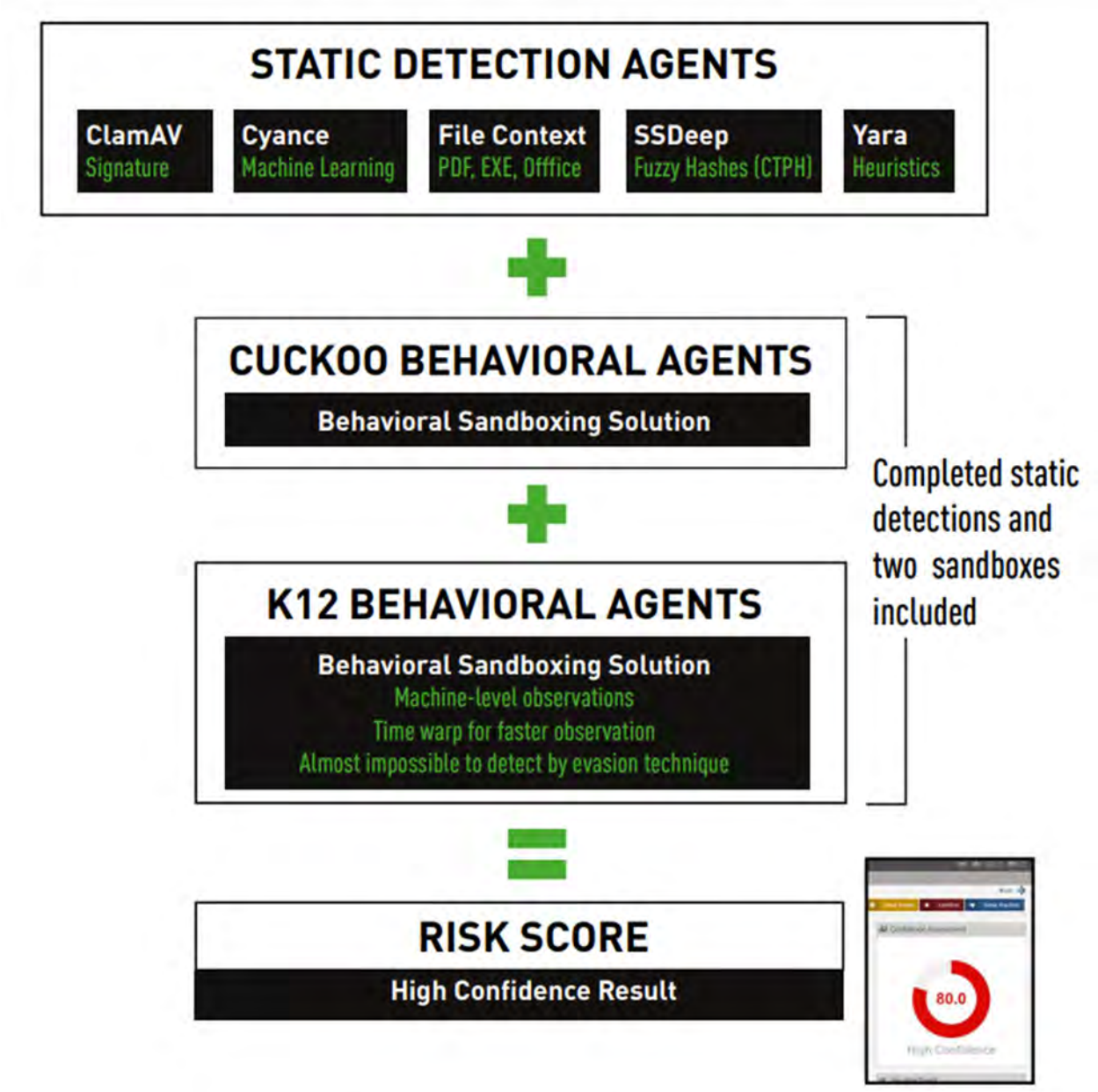
Code injections techniques are monitored by the driver: if a monitored process performs a known code-injection technique into another process, this process will be added to the monitored processes list.

(https://github.com/conix-security/zer0m0n; *see also* https://github.com/conix-security/zer0m0n/blob/master/README.md)



(Forcepoint, *Forcepoint Advanced Malware Detection*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 1)

328. The Accused Products include "*analyzing, heuristically, the API call activity to determine whether one or more weighted factors associated with the API call activity exceeds a threshold*." For example, Forcepoint Advanced Malware Detection utilizes Cuckoo's malware analysis system which can analyze the behavior of a wide array of malicious files and is able to trace API calls. Further, to identify threats, various detection methodologies return independent risk scores, along with confidence information and an overall composite score is determined and passed back to the submitting security applications to take action.



(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 3)

CUCKOO OPEN SOURCE SANDBOX

The sandboxing process begins with Forcepoint Advanced Malware Detection Appliance sending the file to Cuckoo's malware analysis system. Cuckoo can analyze the behavior of a wide array of malicious files (executables, document exploits, Java applets), as well as malicious websites, in Windows, OS X, Linux, and Android virtualized environments. Cuckoo is able to trace API calls and general behavior of the file, dump and analyze encrypted network traffic and perform advanced memory analysis of the infected virtualized system with integrated support for Volatility**. Followed by Cuckoo's analysis the file is sent to ThINK, Raytheon's proprietary sandbox.

ThINK PROPRIETARY SANDBOX

ThINK behavioral analysis is one of the most comprehensive sandboxes that exists in the world today. ThINK is a custom hypervisor that provides a fully integrated system-level debugger and an integrated malware sandbox. The sandbox executes files within a virtualized environment to contain and isolate malicious files before they can infect production systems. ThINK does not require any custom software on the guest OS. This helps avoid altering guest performance and prevents malware from detecting virtualization.

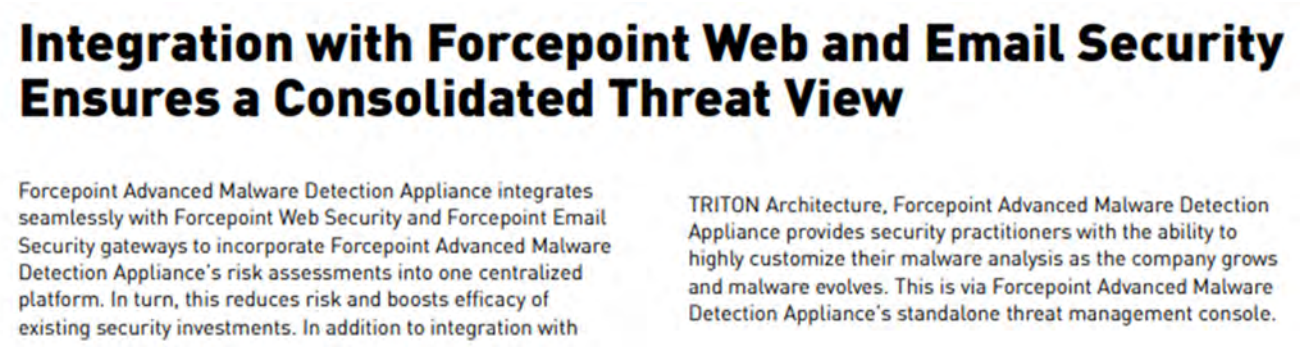
ThINK monitors all incoming and outgoing guest machine network traffic and flags hardware and software exceptions that are likely to indicate an attempted exploit and uses hardware breakpoints to monitor file, registry, process, thread creation and destruction from outside of the guest. ThINK also implements heap-spray analytics to highlight entropy changes to the process heap that are indicative of a heap-spray attack, a common component of browser and Acrobat Reader exploits.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 6)

One Highly Confident Risk Score

To identify threats and minimize false positives, all of these detection methodologies return independent risk scores, along with confidence information and other data. An overall composite score is determined and passed back to the submitting security applications (Forcepoint Email Security or Forcepoint Web Security) to take action, but these details are recorded to help IT to understand the nature of the threat. Individual

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)

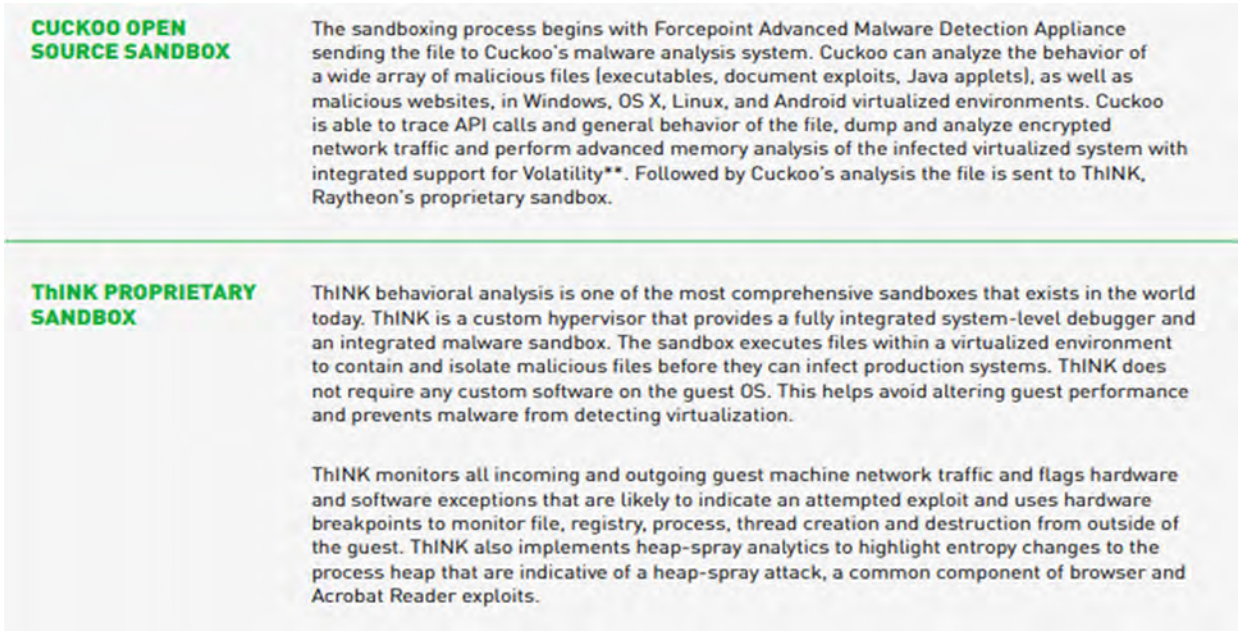


Integration with Forcepoint Web and Email Security Ensures a Consolidated Threat View

Forcepoint Advanced Malware Detection Appliance integrates seamlessly with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. In turn, this reduces risk and boosts efficacy of existing security investments. In addition to integration with TRITON Architecture, Forcepoint Advanced Malware Detection Appliance provides security practitioners with the ability to highly customize their malware analysis as the company grows and malware evolves. This is via Forcepoint Advanced Malware Detection Appliance's standalone threat management console.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 8)

329. The Accused Products include "*identifying, based upon the API call activity, a suspected pestware object on the computer*." For example, the Accused Products utilize a Cuckoo sandbox that performed "advanced memory analysis of [an] infected virtualized system with integrated support for Volatility**. Followed by Cuckoo's analysis the file is sent to ThiNK." The ThiNK proprietary sandbox that "executes files within a virtualized environment to contain and isolate malicious files before they can infect production systems."



CUCKOO OPEN SOURCE SANDBOX	The sandboxing process begins with Forcepoint Advanced Malware Detection Appliance sending the file to Cuckoo's malware analysis system. Cuckoo can analyze the behavior of a wide array of malicious files (executables, document exploits, Java applets), as well as malicious websites, in Windows, OS X, Linux, and Android virtualized environments. Cuckoo is able to trace API calls and general behavior of the file, dump and analyze encrypted network traffic and perform advanced memory analysis of the infected virtualized system with integrated support for Volatility**. Followed by Cuckoo's analysis the file is sent to ThINK, Raytheon's proprietary sandbox.
ThINK PROPRIETARY SANDBOX	ThINK behavioral analysis is one of the most comprehensive sandboxes that exists in the world today. ThINK is a custom hypervisor that provides a fully integrated system-level debugger and an integrated malware sandbox. The sandbox executes files within a virtualized environment to contain and isolate malicious files before they can infect production systems. ThINK does not require any custom software on the guest OS. This helps avoid altering guest performance and prevents malware from detecting virtualization. ThINK monitors all incoming and outgoing guest machine network traffic and flags hardware and software exceptions that are likely to indicate an attempted exploit and uses hardware breakpoints to monitor file, registry, process, thread creation and destruction from outside of the guest. ThINK also implements heap-spray analytics to highlight entropy changes to the process heap that are indicative of a heap-spray attack, a common component of browser and Acrobat Reader exploits.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 6)

One Highly Confident Risk Score

To identify threats and minimize false positives, all of these detection methodologies return independent risk scores, along with confidence information and other data. An overall composite score is determined and passed back to the submitting security applications (Forcepoint Email Security or Forcepoint Web Security) to take action, but these details are recorded to help IT to understand the nature of the threat. Individual

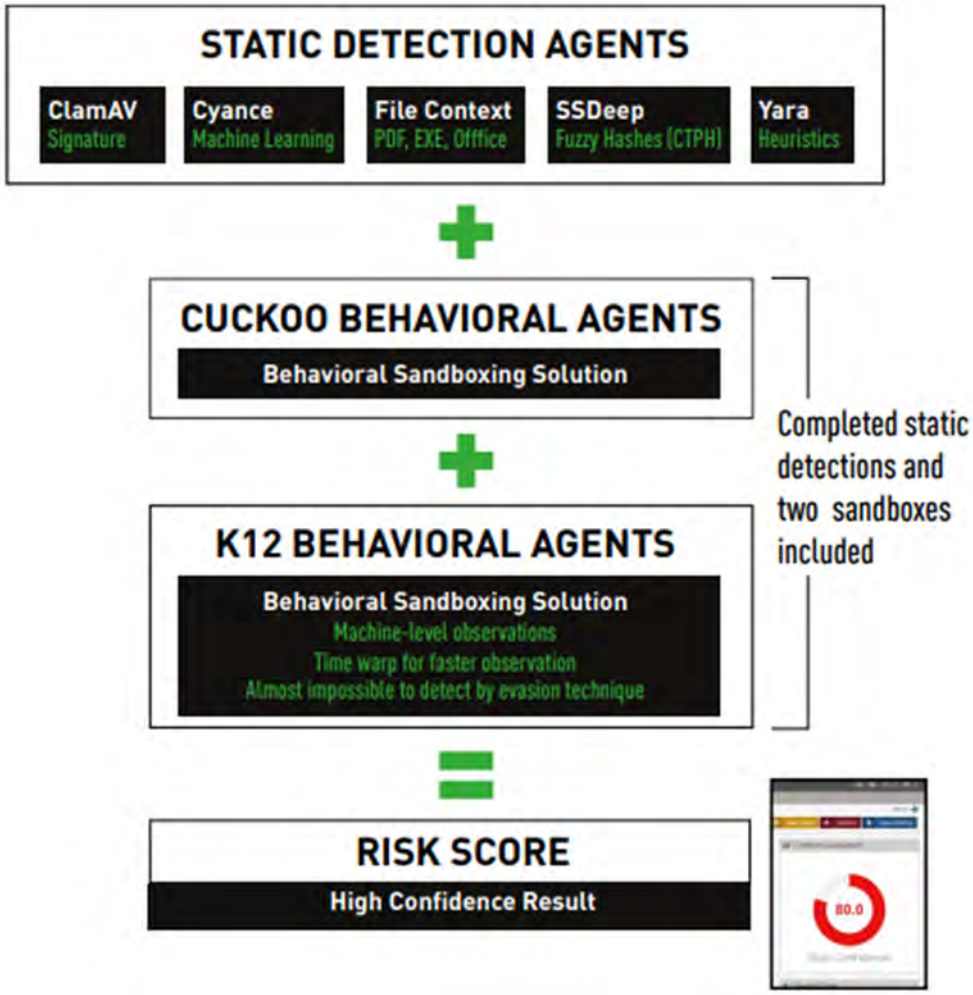
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)

Integration with Forcepoint Web and Email Security Ensures a Consolidated Threat View

Forcepoint Advanced Malware Detection Appliance integrates seamlessly with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. In turn, this reduces risk and boosts efficacy of existing security investments. In addition to integration with

TRITON Architecture, Forcepoint Advanced Malware Detection Appliance provides security practitioners with the ability to highly customize their malware analysis as the company grows and malware evolves. This is via Forcepoint Advanced Malware Detection Appliance's standalone threat management console.

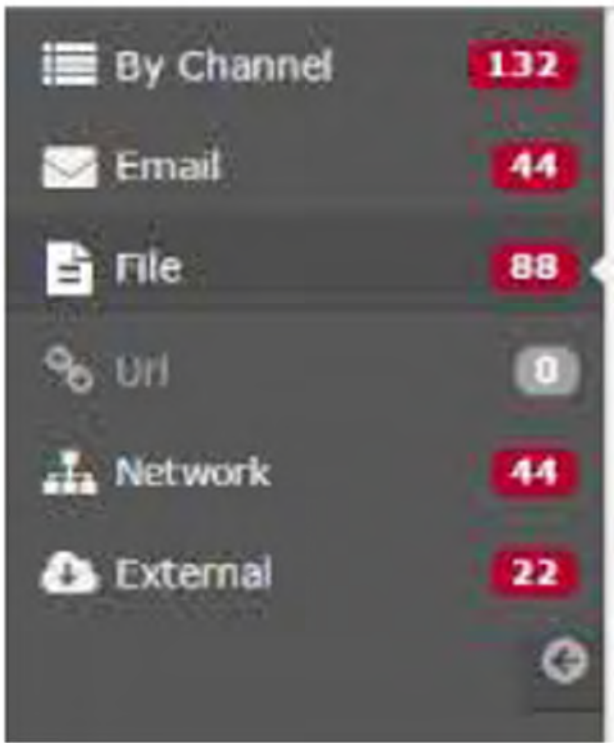
(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 8)



(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 3)

330. The Accused Products include "*identifying, in response to the identifying the suspected pestware object, a reference to an identity of an externally networked source of the suspected pestware object*." For example, the Accused Products "can analyze the behavior of a wide array of malicious files (executables, document exploits, Java applets), as well as malicious websites, in Windows, OS X, Linux, and Android virtualized environments."

331. Forcepoint Advanced Malware Detection Appliance also integrates with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. An "Events section" includes a dynamic and interactive screen, providing information by threat channel category such as Email, File, URL, Network or External.

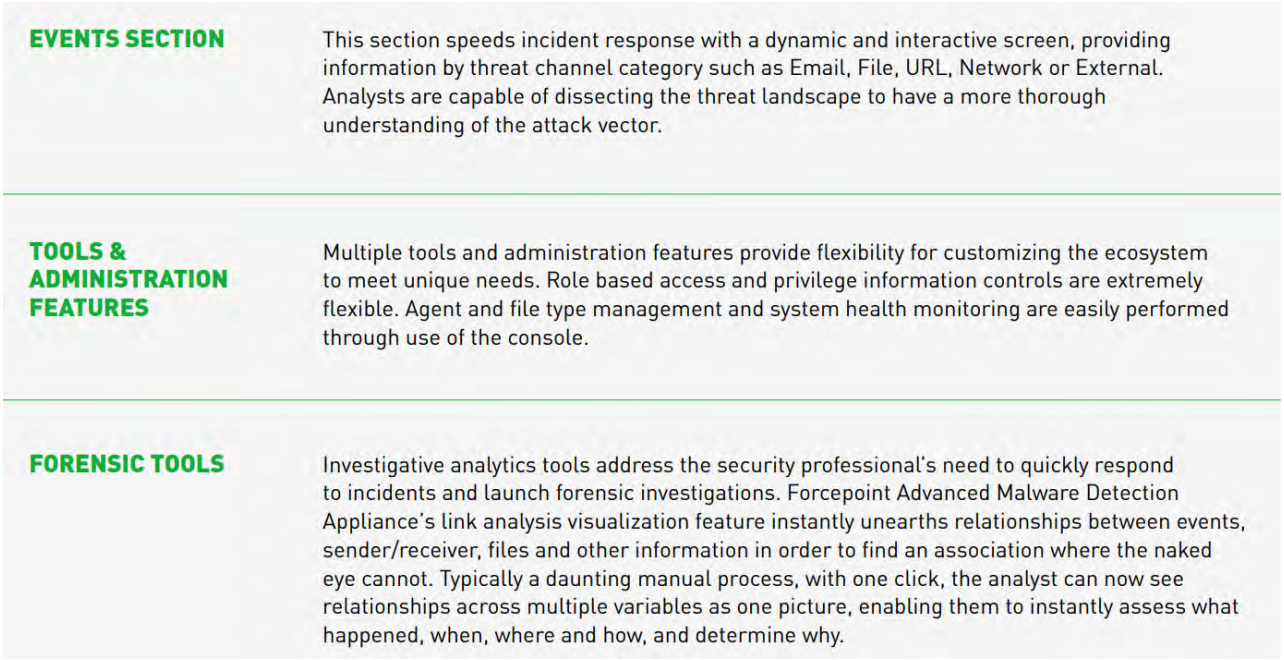


(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)

Integration with Forcepoint Web and Email Security Ensures a Consolidated Threat View

Forcepoint Advanced Malware Detection Appliance integrates seamlessly with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. In turn, this reduces risk and boosts efficacy of existing security investments. In addition to integration with TRITON Architecture, Forcepoint Advanced Malware Detection Appliance provides security practitioners with the ability to highly customize their malware analysis as the company grows and malware evolves. This is via Forcepoint Advanced Malware Detection Appliance's standalone threat management console.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 8)



EVENTS SECTION	This section speeds incident response with a dynamic and interactive screen, providing information by threat channel category such as Email, File, URL, Network or External. Analysts are capable of dissecting the threat landscape to have a more thorough understanding of the attack vector.
TOOLS & ADMINISTRATION FEATURES	Multiple tools and administration features provide flexibility for customizing the ecosystem to meet unique needs. Role based access and privilege information controls are extremely flexible. Agent and file type management and system health monitoring are easily performed through use of the console.
FORENSIC TOOLS	Investigative analytics tools address the security professional's need to quickly respond to incidents and launch forensic investigations. Forcepoint Advanced Malware Detection Appliance's link analysis visualization feature instantly unearths relationships between events, sender/receiver, files and other information in order to find an association where the naked eye cannot. Typically a daunting manual process, with one click, the analyst can now see relationships across multiple variables as one picture, enabling them to instantly assess what happened, when, where and how, and determine why.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 9)

332. The Accused Products include "*reporting the identity of the externally networked source to an externally networked pestware research entity*." For example, Forcepoint's Advanced Malware Detection engine communicates identified pestware object and other analysis results to an incident response team, which further analyzes the pestware object and results in order to determine the source of the pestware object, potentially related pestware, and various other potential threats.

Outcome

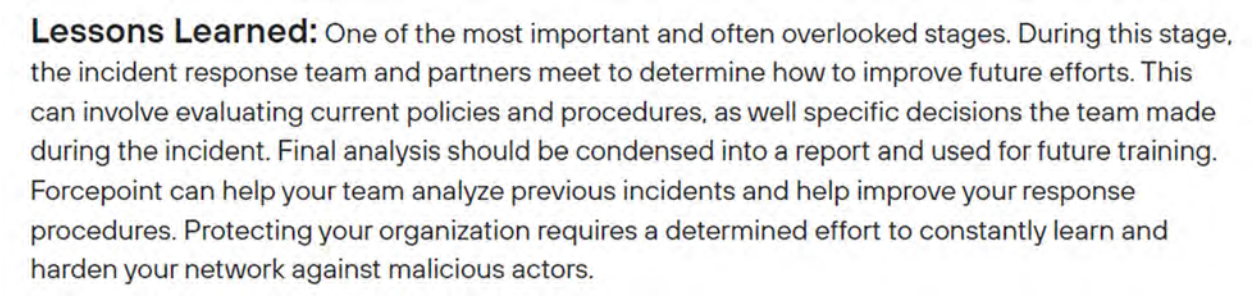
- Increase protection for your organization with proven security effectiveness.
- Enable improved situational awareness for incident response teams.
- Achieve better and faster ROI with integrations.

Available as a cloud service for high availability, scalability, low maintenance and other SaaS benefits, on-premises for cloud-adverse organizations, or even deployed as an air gapped solution with Forcepoint NGFW for physically isolated network requirements. Forcepoint AMD is the perfect complement to your Forcepoint CASB, NGFW, Web Security or Email Security solution. It provides unparalleled threat detection, as well as consistent threat forensic information, to optimize incident response teams.

Extensive malware detail exposure

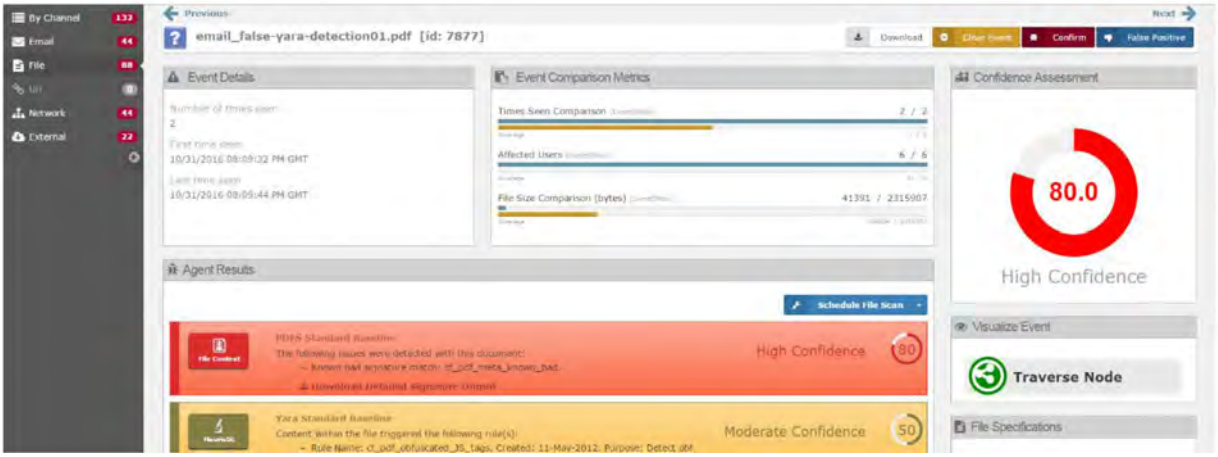
A comprehensive solution must do more than just stop advanced malware. Correlated incident information prioritizes the most significant threats in your network without combing through massive log files. Full attack chain visibility enables your incident response team to quickly understand the nature of the attack, making your scarce security resources more efficient.

(Forcepoint, *Forcepoint Advanced Malware Detection*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf at 1-2)


Lessons Learned: One of the most important and often overlooked stages. During this stage, the incident response team and partners meet to determine how to improve future efforts. This can involve evaluating current policies and procedures, as well specific decisions the team made during the incident. Final analysis should be condensed into a report and used for future training. Forcepoint can help your team analyze previous incidents and help improve your response procedures. Protecting your organization requires a determined effort to constantly learn and harden your network against malicious actors.

(*See* Forcepoint, *What is Incident Response?*, https://www.forcepoint.com/cyber-edu/incident-response)

333. The Accused Products include "*wherein the identity of the externally networked source is selected from a group consisting of an I.P. address, a URL, an email client and a program*." For example, Forcepoint Advanced Malware Detection Appliance integrates with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into a centralized platform. An "Events section" provides information by threat channel category such as Email, File, URL, Network or External. Forcepoint's Advanced Malware Detection includes an interface that may identify URLs associated with the identified pestware. The analysis also allows IT administrators to respond accordingly to the source of the pestware, such as by "[c]hoos[ing] to ignore certain file types or URLs."





(Forcepoint, *Forcepoint Advanced Malware Detection (AMD)*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 7)

Integration with Forcepoint Web and Email Security Ensures a Consolidated Threat View

Forcepoint Advanced Malware Detection Appliance integrates seamlessly with Forcepoint Web Security and Forcepoint Email Security gateways to incorporate Forcepoint Advanced Malware Detection Appliance's risk assessments into one centralized platform. In turn, this reduces risk and boosts efficacy of existing security investments. In addition to integration with TRITON Architecture, Forcepoint Advanced Malware Detection Appliance provides security practitioners with the ability to highly customize their malware analysis as the company grows and malware evolves. This is via Forcepoint Advanced Malware Detection Appliance's standalone threat management console.

(Forcepoint, *Forcepoint Advanced Malware Detection (AMD)*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 8)

EVENTS SECTION	This section speeds incident response with a dynamic and interactive screen, providing information by threat channel category such as Email, File, URL, Network or External. Analysts are capable of dissecting the threat landscape to have a more thorough understanding of the attack vector.
TOOLS & ADMINISTRATION FEATURES	Multiple tools and administration features provide flexibility for customizing the ecosystem to meet unique needs. Role based access and privilege information controls are extremely flexible. Agent and file type management and system health monitoring are easily performed through use of the console.
FORENSIC TOOLS	Investigative analytics tools address the security professional's need to quickly respond to incidents and launch forensic investigations. Forcepoint Advanced Malware Detection Appliance's link analysis visualization feature instantly unearths relationships between events, sender/receiver, files and other information in order to find an association where the naked eye cannot. Typically a daunting manual process, with one click, the analyst can now see relationships across multiple variables as one picture, enabling them to instantly assess what happened, when, where and how, and determine why.

(Forcepoint, *Forcepoint Advanced Malware Detection Appliance 3.4*, https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf at 9)

334. Each claim in the ’932 Patent recites an independent invention. Neither claim 1, described above, nor any other individual claim is representative of all claims in the ’932 Patent.

335. Forcepoint has been aware of the ’932 Patent since at least the date it received a copy of this Amended Complaint.

336. Defendant directly infringes at least claim 1 of the ’932 Patent, either literally or under the doctrine of equivalents, by practicing the elements described above. For example, on information and belief, Defendant makes and uses the claimed system in an infringing manner as described above by running this software and system to protect its own computer and network operations. On information and belief, Defendant also makes and uses the claimed system in an infringing manner when testing the operation of the Accused Products and corresponding systems.

As another example, Defendant makes and uses the claimed system when providing or administering services to third parties, customers, and partners using the Accused Products.

337. Defendant's partners, customers, and end users of its Accused Products and corresponding systems and services directly infringe at least claim 1 of the '932 Patent, literally or under the doctrine of equivalents, at least by using the Accused Products and corresponding systems and services, as described above.

338. Defendant has actively induced and is actively inducing infringement of at least claim 1 of the '932 Patent with specific intent to induce infringement, and/or willful blindness to the possibility that its acts induce infringement, in violation of 35 U.S.C. § 271(b). For example, Defendant encourages and induces customers with specific intent to use Forcepoint's security software in a manner that infringes claim 1 of the '932 Patent at least by offering and providing software and/or hardware that embodies the system that infringes claim 1 when installed and operated by the customer, and by engaging in activities relating to selling, marketing, advertising, promotion, installation, support, and distribution of the Accused Products, including the activities described below.

339. Defendant encourages, instructs, directs, and/or requires third parties – including its certified partners and/or customers – to practice the claims using the software, services, and systems in infringing ways, as described above.

340. Defendant further encourages and induces its customers to infringe claim 1 of the '932 Patent: 1) by making its security services available on its website, providing applications that allow users to access those services, widely advertising those services, and providing technical support and instructions to users, and 2) through activities relating to marketing, advertising, promotion, installation, support, and distribution of the Accused Products, including its Forcepoint

security software, and services in the United States. *See, e.g.*, https://www.forcepoint.com/sites/default/files/resources/datasheets/datasheet_forcepoint_advanced_malware_detection.pdf; https://www.forcepoint.com/sites/default/files/resources/files/brochure_forcepoint_advanced_malware_detection_appliance_en.pdf; https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.7.0/GUID-41D85576-7C24-4BA8-8605-710F52790CD8.html.

341. For example, on information and belief, Defendant shares instructions, guides, and manuals, which advertise and instruct third parties on how to use the software as described above, including at least customers and partners. *Id.* On further information and belief, Defendant also provides customer service and technical support to purchasers of the Accused Products and corresponding systems and services, which directs and encourages customers to perform certain actions that deploy and use the Accused Products in an infringing manner. *Id.*

342. Defendant and/or its partners recommend and sell the Accused Products and provide technical support for the installation, implementation, integration, and ongoing operation of the Accused Products for each individual customer. On information and belief, each customer enters into a contractual relationship with Defendant and/or one of its partners, which obligates each customer to perform certain actions in order to use the Accused Products. Further, in order to receive the benefit of Defendant and/or its partner's continued technical support and their specialized knowledge and guidance of the operability of the Accused Products, each customer must continue to use the Accused Products in a way that infringes the '932 Patent.

343. Further, as the entity that provides installation, implementation, and integration of the Accused Products in addition to ensuring the Accused Product remains operational for each

customer through ongoing technical support, on information and belief, Defendant and/or its partners affirmatively aid and abet each customer's use of the Accused Products in a manner that infringes the claimed system of, and infringes, the '932 Patent.

344. Defendant also contributes to the infringement of its partners, customers, and end-users of the Accused Products by providing within the United States or importing the Accused Products into the United States, which are for use in practicing, and under normal operation practice, systems claimed in the Asserted Patents, constituting a material part of the inventions claimed, and not a staple article or commodity of commerce suitable for substantial non-infringing uses.

345. Indeed, as shown above, the Accused Products have no substantial non-infringing uses because the accused functionality, including malware detection, sandboxing, and related functionality, are integral parts of the Accused Products and must be performed for the Accused Products to perform their intended purpose. These processes are continually running when the system is in use and, on information and belief, cannot be removed or disabled (or, if they could, the system would no longer suitably function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown above, or without the system and components identified above that practice the '932 Patent, that functionality could not be performed.

346. Additionally, the accused functionality, including the malware detection and sandboxing described below, itself has no substantial non-infringing uses because the components, modules and methods identified above are a necessary part of that functionality. For example, without the Accused Products' kernel level driver for monitoring API calls, the Accused Products could not deploy sandbox features to detect malware in a computer. These processes are

continually running when the system is in use and, on information and belief, cannot be removed or disabled (or, if they could, the system would no longer function for its intended purpose). Moreover, for the same reasons, without performing each of the steps as described and shown below, or without the system and components identified below that practice the ’932 Patent, that functionality could not be performed.

347. In addition, as shown in the detailed analysis above, the products, systems, modules, and methods provided by Defendant constitute a material part of the invention—indeed, they provide all the components, modules, and features that perform the claimed methods and systems. For example, the Accused Products and accused functionalities (including the machine learning static analysis) constitute a material part of the inventions claimed because such analysis is integral to the processes identified above as recited in the claims of the ’932 Patent. None of these products are staple goods—they are sophisticated and customized cyber security products, methods, and systems.

348. On information and belief, the infringing actions of each partner, customer, and/or end-user of the Accused Products are attributable to Defendant. For example, on information and belief, Defendant directs and controls the activities or actions of its partners or others in connection with the Accused Products by contractual agreement or otherwise requiring partners or others to provide information and instructions to customers who acquire the Accused Products which, when followed, results in infringement. Defendant further directs and controls the operation of devices executing the Accused Products by programming the software which, when executed by a customer or end user, perform the claimed method of at least claim 1 of the ’932 Patent.

349. Plaintiffs have suffered and continue to suffer damages, including lost profits, as a result of Defendant’s infringement of the ’932 Patent. Defendant is therefore liable to Plaintiffs

under 35 U.S.C. § 284 for damages in an amount that adequately compensates Plaintiffs for Defendant's infringement, but no less than a reasonable royalty.

350. Plaintiffs will continue to suffer irreparable harm unless this Court preliminarily and permanently enjoins Defendant, its agents, employees, representatives, and all others acting in concert with Defendant from infringing the '932 Patent.

351. Defendant's infringement of the '932 Patent is knowing and willful. Defendant acquired knowledge of the '932 Patent and of the specific conduct that constitutes infringement of the '932 Patent at least when it received a copy of this Amended Complaint. Defendant continues to engage in infringing activities after the filing of this amended complaint, including by selling, making, configuring, and installing the Accused Products and performing the accused functionality, and by engaging in activities that constitute inducing infringement and contributory infringement as described above.

352. On information and belief, despite Defendant's knowledge of the Asserted Patents, Defendant made the deliberate decision to sell products and services that it knew infringe these patents. Defendant's continued infringement of the '932 Patent with knowledge of the '932 Patent constitutes willful infringement.

PRAYER FOR RELIEF

WHEREFORE, Plaintiffs respectfully request the following relief:

a) That this Court adjudge and decree that Defendant has been, and is currently, infringing each of the Asserted Patents;

b) That this Court award damages to Plaintiffs to compensate them for Defendant's past infringement of the Asserted Patents, through the date of trial in this action;

c) That this Court award pre- and post-judgment interest on such damages to Plaintiffs;

d) That this Court order an accounting of damages incurred by Plaintiffs from six years prior to the date this lawsuit was filed through the entry of a final, non-appealable judgment;

e) That this Court determine that this patent infringement case is exceptional and award Plaintiffs their costs and attorneys' fees incurred in this action;

f) That this Court award increased damages under 35 U.S.C. § 284;

g) That this Court preliminarily and permanently enjoin Defendant from infringing any of the Asserted Patents;

h) That this Court order Defendant to:

(i) recall and collect from all persons and entities that have purchased any and all products found to infringe any of the Asserted Patents that were made, offered for sale, sold, or otherwise distributed in the United States by Defendant or anyone acting on their behalf;

(ii) destroy or deliver all such infringing products to Plaintiffs;

(iii) revoke all licenses to all such infringing products;

(iv) disable all web pages offering or advertising all such infringing products;

(v) destroy all other marketing materials relating to all such infringing products;

(vi) disable all applications providing access to all such infringing software; and

(vii) destroy all infringing software that exists on hosted systems,

i) That this Court, if it declines to enjoin Defendant from infringing any of the Asserted Patents, award damages for future infringement in lieu of an injunction; and

j) That this Court award such other relief as the Court deems just and proper.

DEMAND FOR JURY TRIAL

Plaintiffs respectfully requests a trial by jury on all issues triable thereby.

DATED: October 28, 2022

By:*/s/ Jeffrey D. Mills*
Jeffrey D. Mills
Texas Bar No. 24034203
KING & SPALDING LLP
500 West Second St.
Suite 1800
Austin, Texas 78701
Telephone: (512) 457-2027
Facsimile: (512) 457-2100
jmills@kslaw.com

Mark D. Siegmund
STECKLER WAYNE CHERRY & LOVE, PLLC
8416 Old McGregor Rd.
Waco, Texas 76712
Telephone: (254) 651-3690
Facsimile: (254) 651-3689
mark@swclaw.com

Steve Sprinkle
Texas Bar No. 00794962
SPRINKLE IP LAW GROUP, P.C.
1301 W. 25th Street, Suite 408
Austin, Texas 78705
Telephone: 512-637-9220
ssprinkle@sprinklelaw.com

Christopher C. Campbell (DC Bar No. 444262)
Patrick M. Lafferty *(pro hac vice)*
KING & SPALDING LLP
1700 Pennsylvania Avenue, NW
Suite 200

Washington, DC 20006
Telephone: (202) 626-5578
Facsimile: (202) 626-3737
ccampbell@kslaw.com
plafferty@kslaw.com

Britton F. Davis *(pro hac vice)*
Brian Eutermoser (*pro hac vice)*
KING & SPALDING LLP
1401 Lawrence Street
Suite 1900
Denver, CO 80202
Telephone: (720) 535-2300
Facsimile: (720) 535-2400
bfdavis@kslaw.com
beutermoser@kslaw.com

Attorneys for Plaintiffs Open Text, Inc. and Webroot, Inc.

CERTIFICATE OF SERVICE

A true and correct copy of the foregoing instrument was served or delivered electronically via the U.S. District Court ECF filing system to all counsel of record on this 28th day of October, 2022.

/s/ Jeffrey D. Mills
Jeffrey D. Mills

Exhibit 1



D 8164050

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

October 07, 2021

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *8,726,389*
ISSUE DATE: *May 13, 2014*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

R GLOVER
Certifying Officer

US008726389B2

(12) **United States Patent**
Morris et al.

(10) **Patent No.: US 8,726,389 B2**
(45) **Date of Patent: May 13, 2014**

(54) **METHODS AND APPARATUS FOR DEALING WITH MALWARE**

(75) Inventors: **Melvyn Morris**, Turnditch (GB); **Paul Stubbs**, Wyboston (GB); **Markus Hartwig**, Milton Keynes (GB); **Darren Harter**, Hucclecote (GB)

(73) Assignee: **Prevx Limited** (GB)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **13/543,866**

(22) Filed: **Jul. 8, 2012**

(65) **Prior Publication Data**

US 2012/0278895 A1 Nov. 1, 2012

Related U.S. Application Data

(63) Continuation of application No. 11/477,807, filed on Jun. 30, 2006.

(30) **Foreign Application Priority Data**

Jun. 30, 2005 (GB) 0513375.6

(51) **Int. Cl.**

G06F 11/00	(2006.01)
G06F 7/04	(2006.01)
G06F 12/14	(2006.01)
G06F 17/30	(2006.01)

(52) **U.S. Cl.**

USPC **726/24**; 726/22; 726/23; 726/25; 726/26

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,338,141	B1	1/2002	Wells
6,772,346	B1	8/2004	Chess
6,944,772	B2	9/2005	Dozortsev
7,013,483	B2	3/2006	Cohen et al.
7,093,239	B1	8/2006	van der Made
7,603,374	B2 *	10/2009	Cameron et al. 1/1

(Continued)

FOREIGN PATENT DOCUMENTS

EP	1315066	5/2003
EP	1280040	3/2004

(Continued)

OTHER PUBLICATIONS

U.S. Appl. No. 60/786,156, filed Apr. 5, 2006, Chiriac.
English Language Abstract of JP 3-233629 published Oct. 17, 1991.
English Language Abstract and Translation of JP 6-110718 published Apr. 22, 1994.

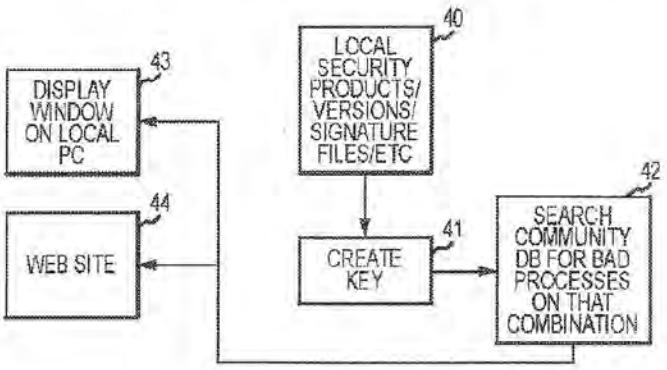
(Continued)

Primary Examiner — Cordelia Zecher
Assistant Examiner — Trang Doan
(74) *Attorney, Agent, or Firm* — Sheridan Ross P.C.

(57) **ABSTRACT**

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

30 Claims, 3 Drawing Sheets



Copy provided by USPTO from the PIRS Image Database on 10-05-2021

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,793,338	B1	9/2010	Beddoe et al.	
2001/0052014	A1	12/2001	Sheymov et al.	
2002/0087734	A1*	7/2002	Marshall et al.	709/310
2002/0099952	A1	7/2002	Lambert et al.	
2002/0194490	A1	12/2002	Halperin et al.	
2003/0023857	A1	1/2003	Hinchlife	
2003/0084323	A1	5/2003	Gales	
2003/0101381	A1	5/2003	Mateev	
2003/0131256	A1	7/2003	Ackroyd	
2003/0135791	A1	7/2003	Natvig	
2003/0177394	A1*	9/2003	Dozortsev	713/201
2003/0195861	A1	10/2003	McClure et al.	
2004/0006704	A1*	1/2004	Dahlstrom et al.	713/200
2004/0039921	A1	2/2004	Chuang	
2004/0068618	A1	4/2004	Hooker	
2004/0073810	A1	4/2004	Racine	
2004/0083408	A1	4/2004	Spiegel	
2004/0153644	A1	8/2004	McCorkendale	
2004/0255165	A1	12/2004	Szor	
2005/0021994	A1*	1/2005	Barton et al.	713/200
2005/0022014	A1	1/2005	Shipman	
2005/0027686	A1	2/2005	Shipp	
2005/0086500	A1	4/2005	Albomoz	
2005/0210035	A1*	9/2005	Kester et al.	707/10
2006/0085857	A1*	4/2006	Omote et al.	726/24
2007/0016953	A1	1/2007	Morris et al.	
2008/0209562	A1	8/2008	Szor	
2008/0320595	A1	12/2008	van der Made	
2009/0271867	A1	10/2009	Zhang	
2012/0278891	A1	11/2012	Morris et al.	

FOREIGN PATENT DOCUMENTS

EP	1536341	6/2005
EP	1549012	6/2005
JP	3-233629	10/1991
JP	6-110718	4/1994
JP	9-504395	4/1997
JP	2003-196112	7/2003
WO	WO 95/12162	5/1995
WO	WO 96/30829	10/1996
WO	WO 99/15966	4/1999
WO	WO 02/33525	4/2002
WO	WO 03/021402	3/2003
WO	WO 2004/097602	11/2004

OTHER PUBLICATIONS

English Language Abstract of JP 2003-196112 published Jul. 11, 2003.
Related U.S. Appl. No. 11/694,261.
Zenkin "Fighting Against the invisible Enemy—Methods for detecting an unknown virus," Computers & Security, Elsevier Science Publishers, Amsterdam, NL, vol. 20, No. 4, Jul. 31, 2001, pp. 316-321, XP004254268.
International Search Report issued in PCT/GB2006/002439 issued Jan. 15, 2007.
English Translation of Official Action for China Patent Application No. 201110036121.8, dated Sep. 24, 2012, 8 pages.
Official Action with English translation for China Patent Application No. 201110036121.8, dated Oct. 10, 2011, 14 pages.
Japanese Office Action issued in JP 2008-518975 on Sep. 13, 2011.
English Language Translation of Japanese Office Action issued in JP 2008-518975 on Sep. 13, 2011.
Official Action for U.S. Appl. No. 11/477,807, mailed Jan. 13, 2010, 8 pages.
Official Action for U.S. Appl. No. 11/477,807, mailed May 4, 2010, 13 pages.
Official Action for U.S. Appl. No. 11/477,807, mailed Sep. 28, 2010, 11 pages.
Official Action for U.S. Appl. No. 11/477,807, mailed Feb. 15, 2011, 13 pages.
Official Action for U.S. Appl. No. 11/477,807, mailed Jul. 25, 2011, 13 pages.
Notice of Allowance for U.S. Appl. No. 11/477,807, mailed Mar. 21, 2012, 9 pages.
Notice of Allowance for U.S. Appl. No. 11/477,807, mailed Dec. 13, 2012, 7 pages.
Official Action for China Patent Application No. 201110036124.1, dated Apr. 17, 2013, 7 pages.
Official Action for U.S. Appl. No. 13/543,865, mailed Apr. 15, 2013, 9 pages.
Official Action for European Patent Application No. 06755686.0 dated Jul. 31, 2013, 5 pages.
Extended European Search Report for European Patent Application No. 13167434.3 dated Aug. 1, 2013, 6 pages.
Extended European Search Report for European Patent Application No. 13167436.8 dated Aug. 30, 2013, 5 pages.
Official Action for U.S. Appl. No. 13/543,865, mailed Aug. 6, 2013, 11 pages.
Official Action with English Translation for China Patent Application No. 20110036124.1, dated Oct. 23, 2013, 8 pages.

* cited by examiner

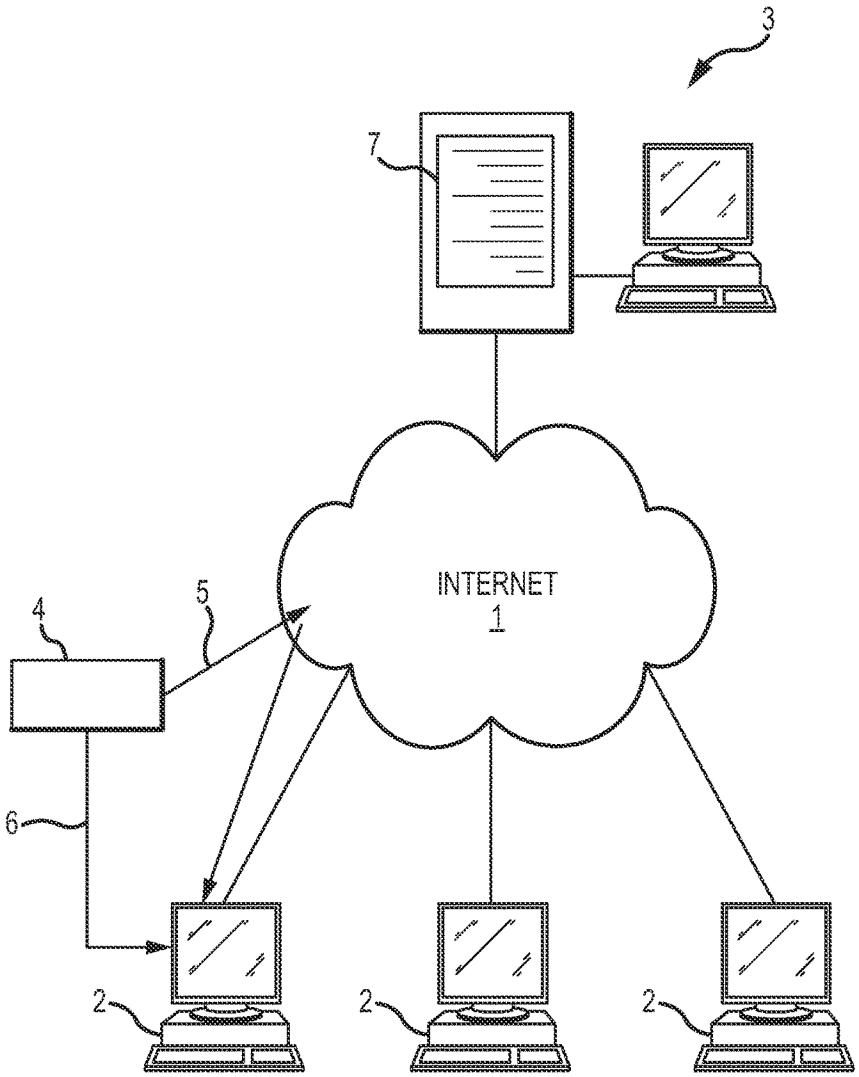


FIG.1

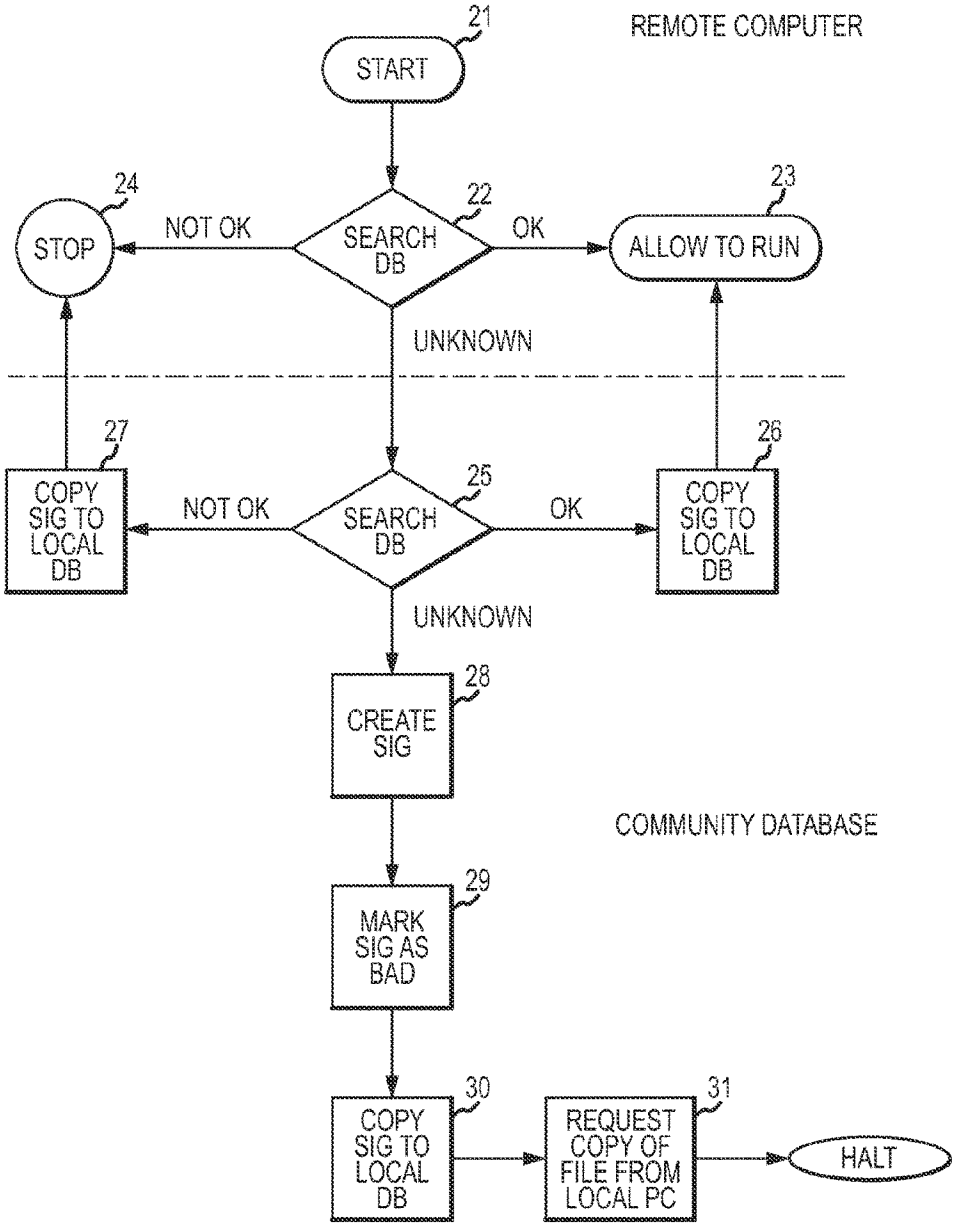

21
REMOTE COMPUTER
START
24
STOP
NOT OK
22
SEARCH DB
OK
23
ALLOW TO RUN
UNKNOWN
27
COPY SIG TO LOCAL DB
NOT OK
25
SEARCH DB
OK
26
COPY SIG TO LOCAL DB
UNKNOWN
28
CREATE SIG
COMMUNITY DATABASE
29
MARK SIG AS BAD
30
COPY SIG TO LOCAL DB
31
REQUEST COPY OF FILE FROM LOCAL PC
HALT

FIG.2

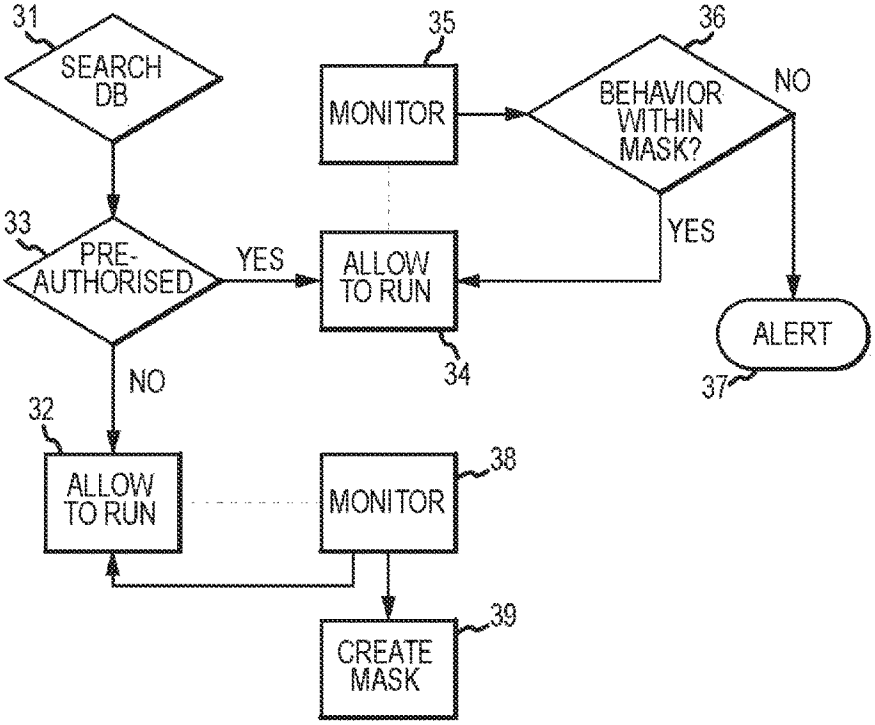

31
SEARCH DB
35
MONITOR
36
BEHAVIOR WITHIN MASK?
NO
YES
33
PRE-AUTHORISED
YES
ALLOW TO RUN
34
ALERT
37
NO
32
ALLOW TO RUN
MONITOR
38
39
CREATE MASK

FIG.3

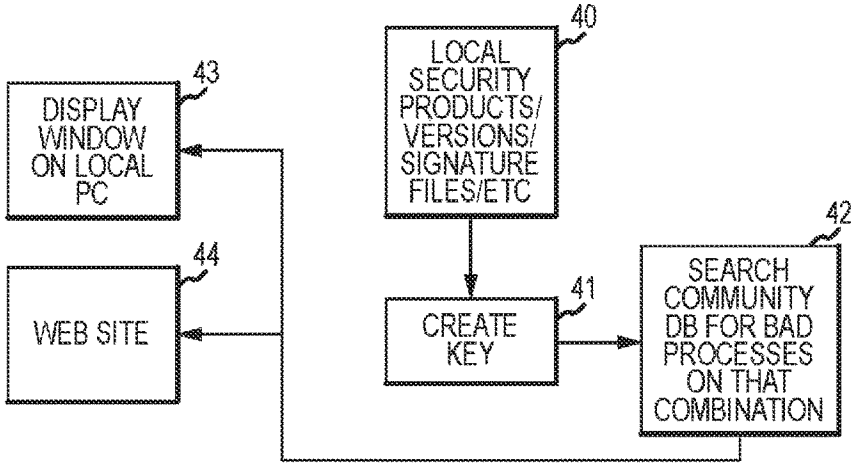

40
LOCAL SECURITY PRODUCTS/ VERSIONS/ SIGNATURE FILES/ETC
43
DISPLAY WINDOW ON LOCAL PC
42
44
WEB SITE
41
CREATE KEY
SEARCH COMMUNITY DB FOR BAD PROCESSES ON THAT COMBINATION

FIG.4

METHODS AND APPARATUS FOR DEALING WITH MALWARE

This application is a continuation of U.S. application Ser. No. 11/477,807 filed Jun. 30, 2006, which claims priority to United Kingdom Application No. 0513375.6, filed Jun. 30, 2005. The entirety of all of the above-listed Applications are incorporated herein by reference.

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

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

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

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

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

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

U.S. Pat. No. 6,944,772, U.S. Pat. No. 6,772,346, EP-A-1549012 and EP-A-1280040 all disclose "community-based" anti-malware systems in which a plurality of "local" computers all connect via a network (which may be a LAN or the Internet, for example) to a central computer. On encountering a file that is not already known to them, the local computers send a request to the central computer for authorisation to run the file. If the file is recognised at the central computer, then the central computer can send permission for the local computer to run the file if the file is known to be safe or send a "deny" command if the file is known to be malicious. However, in each of these prior art proposals, if the file is not known at the central computer, then the whole file is sent to the central computer where it can be analysed to determine whether it should be regarded as safe or malware. Such analysis is typically carried out manually or "semi-manually" by subjecting the file to detailed analysis, for example by emulation or interpretation, which can still take days given the human involvement that is typically required. There is therefore still a considerable period of time before a new file is classified as safe or as malware. In the case of these prior art systems, the request for authorisation to run the file that is sent by a local computer to the central computer may comprise sending a checksum or "signature" or "key" that uniquely represents the file.

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

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

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

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

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

Thus, the prior art systems either rely on deep analysis of a new object in order to determine whether or not the object is malicious, which introduces delay and therefore risk to users during the period that the file is analysed and new anti-malware signatures distributed, or limited analysis of the opera-

tion of the particular object or its method of transmission to a computer is carried out to decide a likelihood of the object being malicious.

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

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

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

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

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

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

Preferably, the comparing identifies relationships between the object and other objects. In an example, this can be used immediately and automatically to mark a child object as bad (or good) if the or a parent or other related object is bad (or good). Thus, if at least one other object to which said object is related is classed as malware, then the method may comprise classifying said object as malware. Said other objects include the object or similar objects stored on at least some of the remote computers. Said other objects may include other objects that are parent objects or child objects or otherwise process-related objects to said object.

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

The key preferably has at least one component that represents executable instructions contained within or constituted by the object. This important preferred feature allows a comparison to be made at the base computer of only the executable instructions of the object. This means for example that differently named objects that basically have the same executable instructions, which is often an indicator that the objects are malware, can nevertheless be regarded as the "same" object for this purpose. As another example, a new version of a program may be released which has minor changes compared to a previous version already known to the base computer and which in substance, at least in respect of the executable instructions, can be regarded as being the same as the previous version. In that case, the minor differences can be ignored and the objects regarded as being the same. Not only is this useful in distinguishing between malware and for example revised versions of previous software, it also keeps down the data transmission and storage requirements because the base computer can inform the remote computers that an apparently new object is for this purpose the same as a previously known object, thus avoiding having the remote computers send full details about the object or the object itself to the base computer.

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

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

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

The method may comprise initially classifying an object as not malware, generating a mask for said object that defines acceptable behaviour for the object, and comprising monitoring operation of the object on at least one of the remote computers and reclassifying the object as malware if the actual monitored behaviour extends beyond that permitted by the mask. This provides an efficient and effective way of monitoring the behaviour of an object that has been classified or regarded as good and allows the object to be reclassified quickly as malware if the behaviour of the object warrants it.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

classifying a first computer object as malware or as safe;

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

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

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

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

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

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

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

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

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

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

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

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

proceeding with the installation;

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

storing said data at the local computer.

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

Preferably, the locally stored data is referred to during the installation to ensure that all objects created or used during

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

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

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

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

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

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

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

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

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

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

FIG. **4** is a flowchart showing schematically an information obtaining stage.

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

In one preferred embodiment, the base computer **3** holds a database **7** with which the remote computers **2** can interact when the remote computers **2** run an object **4** to determine whether the object **4** is safe or unsafe. The community database **7** is populated, over time, with information relating to each object run on all of the connected remote computers **2**. As will be discussed further below, data representative of each object **4** preferably takes the form of a so-called signature or key relating to the object and its effects. As will also be discussed further below, the database **7** may further include a mask for the object **4** that sets out the parameters of the object's performance and operation.

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

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

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

When the community database **7** is searched for the signature of the object that was not previously known at the remote computer **2** concerned, then if the signature is found and indicates that that object is safe, then a copy of the signature or at least a message that the object is safe is sent to the local

database of the remote computer **2** concerned at step **26** to populate the local database. In this way, the remote computer **2** has this information immediately to hand the next time the object **4** is encountered. A separate message is also passed back to the remote computer **2** to allow the object to run in the current instance.

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

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

If the user at the remote computer **2** chooses to run a process that is considered unsafe because it is too new, then that process may be monitored by the remote computer **2** and/or community database **7** and, if no ill effect occurs or is exhibited after a period of time of n days for example, it may then be considered to be safe. Alternatively, the community database **7** may keep a log of each instance of the process which is found by the many remote computers **2** forming part of the network and after a particular number of instances have been recorded, possibly with another particular number of instances or the process being allowed to run and running safely, the signature in the community database **7** may then be marked as safe rather than unsafe. Many other variations of monitoring safety may be done within this concept.

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

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

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

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

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

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

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

This allows a core checksum to be generated by viewing only the executable elements of the checksum and making a comparison between two executables that share common executable code.

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

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

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

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

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

ered safe to have its status reconsidered. Alternatively, the score based on identities may be considered along with scores for other objects or scores created by other processes on this object to be considered in combinations.

The second of the said processes utilises an object's relationship to other objects that act upon it or upon which it acts. For example, analysis can be made of which object created this object, which objects this object created, which objects created a registry key to run or configure this object, which objects were configured by or had registry keys created by this object, etc. In this regard an object is considered to have a relationship based on the event performed by it upon another object, or upon it by another object. This simple 1-to-1 relationship chain provides a complex series of correlation points, allowing ancestral relationships to be considered for any object and its predecessors by event or its issue (i.e.

child and sub-child processes) by event. This allows a score to be developed that describes its relationships and associations with known, unknown, known safe or known bad objects or a combination thereof. Objects that have specific relationships, volumes of relationships or mixes of relationships to one type or another may be judged safe or unsafe accordingly. Alternatively, the relationship-based score may be considered along with other scores to arrive at a determination of safe or unsafe. This data can also be used to deduce a number of factors about objects related directly or via other objects and their behaviours. For example it is possible to deduce how one object's behaviour can be influenced or changed by its association or linkage to another. Consider for example notepad.exe as supplied by Microsoft with the Windows series of operating systems. It has a limited range of functionality and would not be expected therefore to perform a wide variety of events, such as transmitting data to another computer or running other programs etc. However, the behaviour of notepad.exe could be modified by injecting new code into it, such as via dynamic link library injection (DLL injection). In this case notepad.exe would now have new capabilities derived by the code injection or linkage to another object. Using the data that defines the relationships between objects it is possible to deduce that the new behaviours of a program can be attributed to the association with another object. If that new behaviour is malevolent, then it is possible to mark either or all processes as unsafe as appropriate.

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

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

The fourth said process considers the behaviour of an object. This allows a score to be developed based on the types of events performed by an object or events performed on it by itself or other objects. The centralised system of the community database **7** allows for an unlimited number of event types and can consider the object performing the event or the object having the event performed upon it, or both. Some event types also relate to external information other than objects, for example a program performing an event to connect with an Internet Chat Relay site, or a program modifying a non-executable file such as the Windows hosts file. The behavioural events of an object, be they as "actor" (i.e. the object doing something to another object) or as "victim" (i.e. the object has something done to it by another object) of any event can be considered in many ways, such as in combination, in sequence, in volume, in presence or absence, or in any combination thereof. The behavioural events in the preferred embodiment may have been provided by a remote computer **2** or from other external sources. The process can consider these in isolation or in combination. Furthermore it is a feature of the preferred embodiment that the behavioural events can be considered in combination with the status of other objects upon which the object acts or that act upon the object. For example, creating a program may have a different score if the program being created is safe, unsafe, new, unknown or suspicious. Similarly, a program that is created by a known bad program will likely have a different score attributed to its creation event depending on the status of the object creating it.

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

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

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

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

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

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

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

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

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

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

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

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

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

Thus, the community database **7** can be used to make early diagnosis, or simply to take precautionary measures, and thus stop potentially fast propagating worms and other malware very, very early in their life cycle. Given that it is possible to create a worm that can infect every computer connected to the Internet within a matter of a few minutes, this feature is highly desirable.

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

The signature of an object may comprise or be associated with a mask which can be built up with use of that object and which indicates the particular types of behaviour to be expected from the object. If an object is allowed to run on a remote computer **2**, even if the initial signature search **22** indicates that the object is safe, then operation of that object may be monitored within the parameters of the mask. The

mask might indicate for example, the expected behaviour of the object; any external requests or Internet connections that that object might legitimately have to make or call upon the remote computer **2** to make, including details of any ports or interfaces that might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, that may be expected to be interrogated by that object; and so on. Thus, the mask can give an overall picture of the expected "normal" behaviour of that object.

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

This is shown schematically in FIG. **3**. FIG. **3** also shows the concept that any object can be pre-authorised by, for example a trusted partner, such as a major software company, a verification authority, a Government department, and so on. Pre-authorisation enables a supplier of a new object, which has not been released before, to get pre-authorisation for that object, and optionally includes the provision by that supplier of a mask detailing the expected and allowable behaviour of that object.

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

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

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

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

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

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

In a simple example, if for example the database knows that a computer in the past has version A of anti-virus product B with downloaded signature update C, and also has a firewall D with particular settings E, and perhaps anti-spyware software F with signature updates G, but that a particular malware process P was not detected by this combination of programs at that point in time, then this information can be provided to a computer **2** that is known as having that combination of security programs/settings, and can be used to indicate that that computer **2** is vulnerable in the short term to attack by that particular malware process. This information can be presented to the user either by displaying a window **43** on the

screen display of the computer **2** or by directing the user to a particular website which explains the position in more detail. The user might be informed for example that their particular combination of security products, etc., exposes their computer to a risk of being infected by the Sobig virus as that virus is not detectable by their computer. The user might be offered specific advice (e.g. to update a particular anti-virus program with a particular signature file) or software to download and install to remove the risk.

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

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

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

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

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

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

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

f) An ability to watch clones or identical twins of objects (e.g. the same process running on other systems in the community) to compare the pattern of their issue and to make decisions as to whether or not to treat any particular process as malware.

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

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

Reference is now made to "Defeating Polymorphism: Beyond Emulation" by Adrian Stepan of Microsoft Corporation published in "Virus Bulletin Conference, October 2005" and also to U.S. 60/789,156 filed on 5 Apr. 2006, the entire content of which are hereby incorporated by reference. In that paper and patent application, there are disclosed methods of decrypting files to allow the files to be analysed to determine whether or not the file actually is malware. In U.S. 60/789,156 in particular, there is disclosed a method of unpacking/decrypting an executable computer file using a host computer, the method comprising: partitioning the executable computer file into plural basic blocks of code; translating at least some of the basic blocks of code into translated basic blocks of code that can be executed by the host computer; linking at least some of the translated basic blocks of code in memory of the host computer; and, executing at least some of the translated basic blocks of code on the host computer so as to enable the executable computer file to be unpacked or decrypted, whereupon the unpacked or decrypted executable computer file can be analyzed to determine whether the executable computer file is or should be classed as malware. There is also disclosed in U.S. 60/789,156 a method of unpacking/decrypting an executable computer file, the method comprising: partitioning the executable computer file into plural basic blocks of code; creating at least a read page of cache memory for at least some of the basic blocks, the read page of cache memory storing a read cached real address corresponding to a read cached virtual memory address for the respective basic block,

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

The techniques disclosed in these documents can be used in the present context when it is desired to analyse a file in detail. More generally however, the techniques disclosed in these papers, and particularly the enhanced techniques disclosed in U.S. 60/789,156, can be used to provide information about the activity of a file when it is run on a computer because the techniques disclosed in these papers emulate the running of the file and therefore allow the file's activity to be interpreted.

A further situation arises when users wish to install software while offline. In a preferred embodiment, when a user attempts to install new software while offline, agent software running on the user's computer **2** prompts the user for authorisation to allow the installation process to proceed such that the execution of the installation can itself be "authorised" by the user. This authorisation by the user is treated as a "Genesis" event and will be so termed hereinafter. However there are some processes commonly used in installation of software that communicate with other existing programs on the installation machine, e.g. Microsoft's MSIEXEC.

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

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

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

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

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

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

The invention claimed is:

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

at a base computer, receiving data about a computer object from a first remote computer on which the computer object or similar computer objects are stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured or runs on the first remote computer, said information including at least an identity of an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed;

at the base computer, receiving data about the computer object from a second remote computer on which the computer object or similar computer objects are stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured, or runs on the second remote computer, said information including at least an identity of an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed;

storing, at the base computer, said data received from the first and second remote computers;

correlating, by the base computer, at least a portion of the data about the computer object received from the first remote computer to at least a portion of the data about the computer object received from the second remote computer;

comparing, by the base computer, the correlated data about the computer object received from the first and second remote computers to other objects or entities to identify relationships between the correlated data and the other objects or entities; and

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

2. A method according to claim **1**, wherein the relationship is between objects related directly, or between objects related via other objects.

3. A method according to claim **1**, wherein where an object and other objects are found to have a relationship, monitoring

those related objects as a homogenous entity and identifying a common object of the homogenous entity as a malware object.

4. A method according to claim **1**, comprising deducing an object to be malware based on the behaviour of a related object.

5. A method according to claim **1**, wherein if at least one other object to which said computer object is related is classed as malware, then classifying said computer object as malware.

6. A method according to claim **1**, wherein said other objects include the object or similar objects stored on at least one of the first and second remote computers.

7. A method according to claim **1**, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said computer object.

8. A method according to claim **1**, wherein the data about the computer object that is sent from the respective remote computer to the base computer and that is used in the comparison includes one or more of:

executable instructions contained within or constituted by the computer object;

the size of the computer object;

the current name of the computer object;

the physical and folder location of the computer object on disk;

the original name of the computer object;

the creation and modification dates of the computer object;

vendor, product and version and any other information stored within the computer object; and

the computer object header or header held by the respective remote computer.

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

10. A method according to claim **9**, wherein the key has at least one component that represents executable instructions contained within or constituted by the computer object.

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

12. A method according to claim **11**, wherein said data about said computer object includes at least one of:

the current name of the computer object;

the physical and folder location of the computer object on disk;

the original name of the computer object;

the creation and modification dates of the computer object;

vendor, product and version and any other information stored within the computer object;

the computer object header or header held by the respective remote computer; and,

events initiated by or involving the computer object when the computer object is created, configured or runs on the respective remote computers.

13. A method according to claim **9**, wherein the key has at least one component that represents the physical size of the computer object.

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

15. An apparatus for classifying a computer object as malware, the apparatus comprising:

a base computer constructed and arranged to receive data about a computer object from a first remote computer on which the computer object or similar computer objects are stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured, or runs on the first remote computer, said information including at least an identity of an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed;

the base computer being constructed and arranged to receive data about the computer object from a second remote computer on which the computer object or similar computer objects re stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured, or runs on the second remote computer, said information including at least an identity of an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed;

the base computer being constructed and arranged to correlate at least a portion of the data about the computer object received from the first remote computer to at least a portion of the data about the computer object received from the second remote computer;

the base computer being constructed and arranged to compare the data about the computer object received from the first and second remote computers to other objects or entities to identify relationships between the correlated data and the other objects or entities; and

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

16. Tha apparatus according to claim **15**, wherein the base computer is constructed and arranged so as to identify the relationship between objects related directly, or between objects related via other objects.

17. The apparatus according to claim **15**, wherein the base computer is constructed and arranged so as, where an object and other objects are found to have a relationship, to monitor those related objects as a homogenous entity and identify a common object of the homogenous entity as a malware object.

18. The apparatus according to claim **15**, wherein the base computer is constructed and arranged so as to deduce an object to be malware based on the behaviour of a related object.

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

20. The apparatus according to claim **15**, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least one of the first and second remote computers.

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

22. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as to compare the data where the data includes one or more of:

executable instructions contained within or constituted by the computer object;

the size of the computer object;

the current name of the computer object;

the physical and folder location of the computer object on disk;

the original name of the computer object;

the creation and modification dates of the computer object;

vendor, product, and version and any other information stored within the computer object; and

the computer object header or header held by the respective remote computer.

23. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on said respective remote computer.

24. The apparatus according to claim 23, wherein the key has at least one component that represents executable instructions contained within or constituted by the computer object.

25. The apparatus according to claim 23, wherein the key has at least one component that represents data about said computer object.

26. Apparatus according to claim 25, wherein said data about said object includes at least one of:

the current name of the computer object;

the physical and folder location of the computer object on disk;

the original name of the comptuer object;

the creation and modification dates of the computer object;

vendor, product, and version and any other information stored within the computer object;

the computer object header or header held by the respective remote computer; and

events initiated by or involving the computer object when the computer object is created, configured, or runs on the respective remote computers.

27. The apparatus according to claim 23, wherein the key has at least one component that represents the physical size of the computer object.

28. The apparatus according to claim 15, wherein the base computer is constructed and arranged to generate a mask for a computer object that is initially classed not as malware, said mask defining acceptable behaviour for the computer object, and the base computer is constructed and arranged to monitor an operation of the computer object on at least one of the first and second remote computers and to reclassify the computer object as malware if the actual monitored behaviour extends beyond that permitted by the mask.

29. A computer program recorded on non-transitory computer readable medium comprising program instructions for causing a computer to perform the method of claim 1.

30. A method according to claim 1, wherein the data received about the computer object from the first remote computer is different than the data received about the computer object from the second remote computer.

* * * * *

Exhibit 2

U 8170010

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

October 22, 2021

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *10,599,844*
ISSUE DATE: *March 24, 2020*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

M. TARVER
Certifying Officer



US010599844B2

(12) **United States Patent**
Schmidtler et al.

(10) **Patent No.: US 10,599,844 B2**
(45) **Date of Patent: Mar. 24, 2020**

(54) **AUTOMATIC THREAT DETECTION OF EXECUTABLE FILES BASED ON STATIC DATA ANALYSIS**

(71) Applicant: **Webroot Inc.**, Broomfield, CO (US)

(72) Inventors: **Mauritius Schmidtler**, Escondido, CA (US); **Gaurav Dalal**, San Jose, CA (US); **Reza Yoosoofmiya**, San Diego, CA (US)

(73) Assignee: **Webroot, Inc.**, Broomfield, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 95 days.

(21) Appl. No.: **14/709,875**

(22) Filed: **May 12, 2015**

(65) **Prior Publication Data**

US 2016/0335435 A1 Nov. 17, 2016

(51) **Int. Cl.**

G06F 21/56	(2013.01)
G06N 5/04	(2006.01)
G06N 20/00	(2019.01)
G06F 8/53	(2018.01)
G06F 11/30	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 21/565*** (2013.01); ***G06F 21/562*** (2013.01); ***G06N 20/00*** (2019.01); *G06F 8/53* (2013.01); *G06F 2221/033* (2013.01)

(58) **Field of Classification Search**

CPC .. G06F 21/565; G06F 99/005; G06F 221/033; G06F 21/562; G06F 2221/033; G06F 8/53; G06N 20/00

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,510,836	B1 *	8/2013	Nachenberg	H04L 63/145 706/12
8,719,924	B1 *	5/2014	Williamson	G06F 21/568 713/188
9,197,663	B1 *	11/2015	Gilbert	G06F 3/04817
9,465,940	B1 *	10/2016	Wojnowicz	G06F 21/565
2007/0240220	A1 *	10/2007	Tuvell	G06F 21/56 726/24

(Continued)

OTHER PUBLICATIONS

Shabtai, A et al. "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey", Information Security Technical Report, Elsevier Advanced Technology. Amsterdam, NL, vol. 14, No. 1, Feb. 1, 2009, pp. 16-29, XP026144093, ISSN: 1363-4127.*

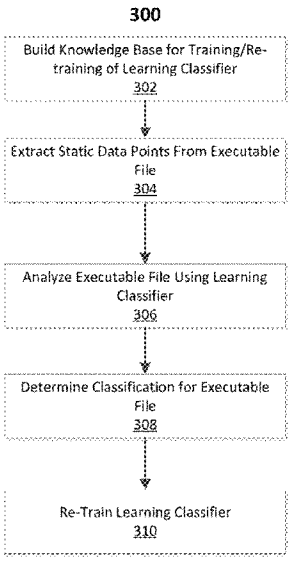
(Continued)

Primary Examiner — Michael Simitoski
(74) *Attorney, Agent, or Firm* — Sprinkle IP Law Group

(57) **ABSTRACT**

Aspects of the present disclosure relate to threat detection of executable files. A plurality of static data points may be extracted from an executable file without decrypting or unpacking the executable file. The executable file may then be analyzed without decrypting or unpacking the executable file. Analysis of the executable file may comprise applying a classifier to the plurality of extracted static data points. The classifier may be trained from data comprising known malicious executable files, known benign executable files and known unwanted executable files. Based upon analysis of the executable file, a determination can be made as to whether the executable file is harmful.

20 Claims, 4 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

2009/0013405	A1*	1/2009	Schipka	G06F 21/562 726/22
2009/0274376	A1*	11/2009	Selvaraj	G06K 9/00442 382/224
2009/0300765	A1*	12/2009	Moskovitch	G06F 21/562 726/24
2010/0082642	A1*	4/2010	Forman	G06F 16/35 707/749
2010/0192222	A1*	7/2010	Stokes	G06F 21/563 726/22
2012/0079596	A1*	3/2012	Thomas	G06F 21/55 726/24
2012/0227105	A1*	9/2012	Friedrichs	G06F 21/564 726/22
2013/0291111	A1*	10/2013	Zhou	G06F 21/566 726/23
2014/0165203	A1*	6/2014	Friedrichs	G06F 21/56 726/24
2014/0310517	A1*	10/2014	Christodorescu	H04L 41/142 713/160
2016/0021174	A1*	1/2016	De Los Santos Vilchez	H04L 43/10 709/201
2016/0156646	A1*	6/2016	Hsueh	G06F 21/563 726/1
2017/0262633	A1*	9/2017	Miserendino	G06F 21/566
2018/0365573	A1*	12/2018	Sultana	G06N 5/04

OTHER PUBLICATIONS

Bost, Raphael, et al. "Machine learning classification over encrypted data." NDSS. vol. 4324. 2015. (Year: 2015).*

Choi, Junho, et al. "Efficient malicious code detection using n-gram analysis and SVM." 2011 14th International Conference on Network-Based Information Systems. IEEE, 2011. (Year: 2011).*

Devi, Dhruwajita, and Sukumar Nandi. "Detection of packed malware." Proceedings of the First International Conference on Security of Internet of Things. ACM, 2012. (Year: 2012).*

Dube, Thomas, et al. "Malware target recognition via static heuristics." Computers & Security 31.1 (2012): 137-147. (Year: 2012).*

Merkel, Ronny, et al. "Statistical detection of malicious pe-executables for fast offline analysis." IFIP International Conference on Communications and Multimedia Security. Springer, Berlin, Heidelberg, 2010. (Year: 2010).*

Singla, Sanjam, et al. "A Novel Approach to Malware Detection using Static Classification." International Journal of Computer Science and Information Security 13.3 (2015): 1. (Year: 2015).*

Tabish, S. Momina, M. Zubair Shafiq, and Muddassar Farooq. "Malware detection using statistical analysis of byte-level file content." Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics. ACM, 2009. (Year: 2009).*

Treadwell, Scott, and Mian Zhou. "A heuristic approach for detection of obfuscated malware." 2009 IEEE International Conference on Intelligence and Security Informatics. IEEE, 2009. (Year: 2009).*

International Patent Application No. PCT/US2016/032090, International Search Report and Written Opinion dated Jul. 29, 2016, 13 pages.

Perdisci et al. (2008) Pattern Recognition Letters 29:1941-1946 "Classification of packed executables for accurate computer virus detection".

Perdisci et al. (2008) IEEE: Annual Computer Security Applications Conference 301-310 "McBoost: Boosting Scalability in Malware Collection and Analysis Using Statistical Classification of Executables".

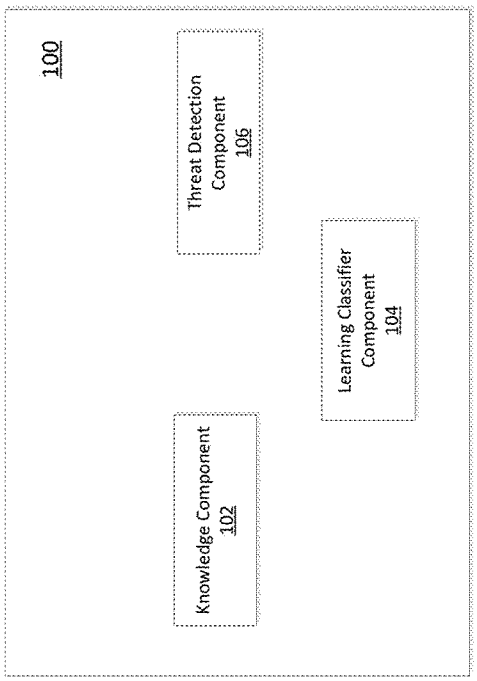
Khan, Hassan et al., "Determining Malicious Executable Distinguishing Attributes and Low-Complexity Detection", J. Comput Virol, 7.2 (2011), pp. 95-105.

Kolter, J. Zico et al., "Learning to Detect and Classift Malicious Executables in the Wild", Journal of Machine Learning Research 7 (2006), pp. 2721-2744.

International Patent Application No. PCT/US2016/032090, International Preliminary Report on Patentability dated Nov. 14, 2017, 7 pages.

* cited by examiner

FIG. 1


100
Threat Detection Component 106
Learning Classifier Component 104
Knowledge Component 102

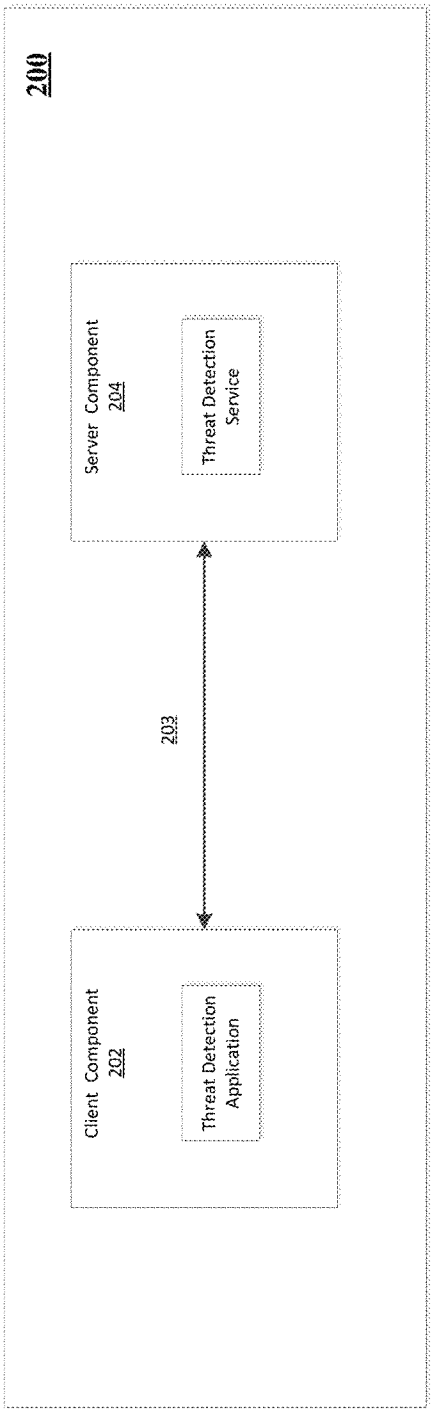
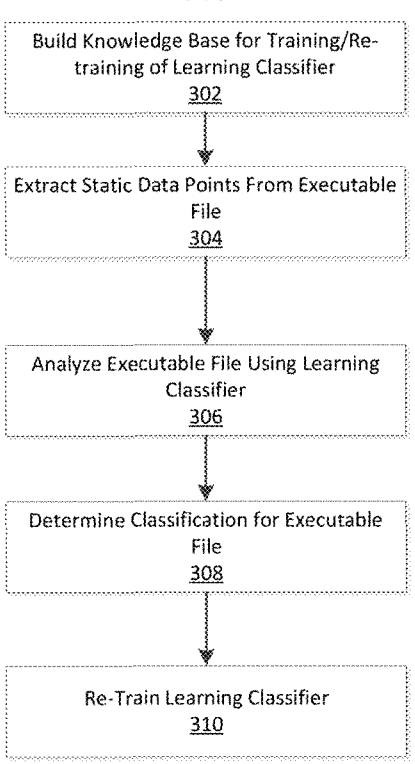

200
Server Component
204
Threat Detection
Service
203
Client Component
202
Threat Detection
Application

FIG. 2

FIG. 3


300
Build Knowledge Base for Training/Re-training of Learning Classifier
302
Extract Static Data Points From Executable File
304
Analyze Executable File Using Learning Classifier
306
Determine Classification for Executable File
308
Re-Train Learning Classifier
310

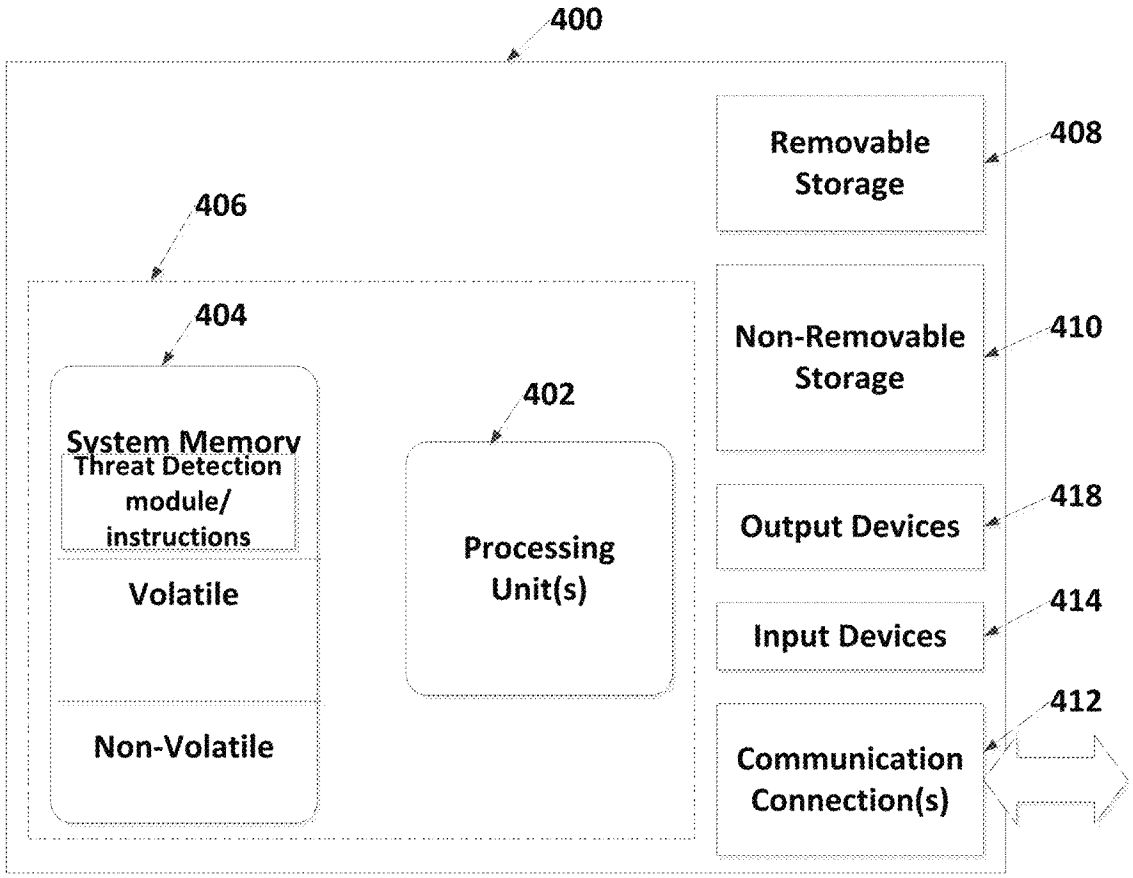

400
406
404
System Memory
Threat Detection module/ instructions
Volatile
Non-Volatile
402
Processing Unit(s)
Removable Storage
408
Non-Removable Storage
410
Output Devices
418
Input Devices
414
Communication Connection(s)
412

FIG. 4

AUTOMATIC THREAT DETECTION OF EXECUTABLE FILES BASED ON STATIC DATA ANALYSIS

BACKGROUND

Everyday new executable files are created and distributed across networks. A large portion of these distributed executable files are unknown. For instance, it is not known if such distributed executable files are malicious or not. Given the high volume of new unknown files distributed on a daily basis, it is important to determine threats contained in the set of new unknown files instantaneously and accurately. It is with respect to this general environment that aspects of the present technology disclosed herein have been contemplated.

SUMMARY

Aspects of the present disclosure relate to threat detection of executable files. A plurality of static data points are extracted from an executable file without decrypting or unpacking the executable file. The executable file may then be analyzed without decrypting or unpacking the executable file. Analyzing of the executable file comprises applying a classifier to the plurality of static data points extracted from the executable file. The classifier is trained from data comprising known malicious executable files, known benign executable files and potentially unwanted executable files. Based upon the analysis of the executable file, a determination is made as to whether the executable file is harmful. In some examples, execution of the executable file is prevented when a determined probability value that the executable file is harmful exceeds a threshold value.

This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

Non-limiting and non-exhaustive examples are described with reference to the following figures. As a note, the same number represents the same element or same type of element in all drawings.

FIG. **1** illustrates an exemplary system **100** showing interaction of components for implementation threat detection as described herein.

FIG. **2** illustrates an exemplary distributed system **200** showing interaction of components for implementation of exemplary threat detection as described herein.

FIG. **3** illustrates an exemplary method **300** for implementation of threat detection systems and methods described herein.

FIG. **4** illustrates one example of a suitable operating environment **400** in which one or more of the present examples may be implemented.

DETAILED DESCRIPTION

Non-limiting examples of the present disclosure relate to the threat detection of executable files. The examples disclosed herein may also be employed to detect zero day threats from unknown executable files. However, one skilled in the art will recognize that the present disclosure is not

limited to detection of executable files that may present zero day threats and can be applicable to any unknown file that is attempting to execute on a system (e.g., processing device). The present disclosure is able to detect whether an executable file is harmful or benign before the executable file is actually executed on a processing device. In examples, machine learning processing applies a classifier to evaluate an executable file based on static points data collected from the executable file. The classifier is trained from a collection of data comprising known malicious files, potentially unwanted files and benign files. The classifier is designed and trained such that it can handle encrypted and/or compressed files without decrypting and/or decompressing the files.

Approaches to detecting threats typically focus on finding malicious code blocks within a file and analyzing the behavior of the file. Such approaches are expensive and time-consuming operations that require decrypting encrypted files, dissembling code, analyzing behavior of malware, among other things. Additionally, behavioral detection requires the execution of the potentially malicious code, thereby presenting an opportunity for the malicious code to harm the computing system is executing on. The present disclosure achieves high and accurate classification rates for potentially malicious executables without the need to employ time consuming processing steps like decryption, unpacking or executing unknown files while maintaining a controlled and safe environment. The determination of unknown files is achieved by evaluating static data points extracted from an executable file using a trained classification system that is able to identify potential threats without analyzing executable behavior of a file. The present disclosure also provides for the creation of a training set that adaptively learns what static data points may indicate the existence of malicious code using training data that contains a statistically significant number of both encrypted and non-encrypted examples of malicious or unwanted executable files, among other information. By collecting a large set of training examples as they appear as publicly available and distributed over network connection (e.g., "in the wild"), the present disclosure ensures that its adaptive learning processing is robust enough to comprise sufficient representation of features (and distributions) from files that may be encrypted, not-encrypted, compressed, and uncompressed, among other examples.

A number of technical advantages are achieved based on the present disclosure including, but not limited to: enhanced security protection including automatic detection of threats, reduction or minimization of error rates in identification and marking of suspicious behavior or files (e.g., cut down on the number of false positives), ability to adapt over time to continuously and quickly detect new threats or potentially unwanted files/applications, improved efficiency in detection of malicious files, and improved usability and interaction for users by eliminating the need to continuously check for security threats, among other benefits that will be apparent to one of skill in the art.

FIG. **1** illustrates an exemplary system **100** showing an interaction of components for implementation threat detection as described herein. Exemplary system **100** may be a combination of interdependent components that interact to form an integrated whole for execution of threat detection and/or prevention operations. Components of the systems may be hardware components or software implemented on and/or executed by hardware components of the systems. In examples, system **100** may include any of hardware components (e.g., used to execute/run operating system (OS)),

and software components (e.g., applications, application programming interfaces, modules, virtual machines, runtime libraries, etc.) running on hardware. In one example, an exemplary system **100** may provide an environment for software components to run, obey constraints set for operating, and/or makes use of resources or facilities of the system **100**, where components may be software (e.g., application, program, module, etc.) running on one or more processing devices. For instance, threat detection operations (e.g., application, instructions, modules, etc.) may be run on a processing device such as a computer, a client device (e.g., mobile processing device, laptop, smartphone/phone, tablet, etc.) and/or any other electronic devices, where the components of the system may be executed on the processing device. In other examples, the components of systems disclosed herein may be spread across multiple devices. For instance, files to be evaluated may be present on a client device and information may be processed or accessed from other devices in a network such as, for example, one or more server devices that may be used to perform threat detection processing and/or evaluation a file before execution of the file by the client device.

As one example, system **100** comprises a knowledge component **102**, a learning classifier component **104**, and a threat determination component **106**, each having one or more additional components. The scale of systems such as system **100** may vary and include more or less components than those described in FIG. **1**. In alternative examples of system **100**, interfacing between components of the system **100** may occur remotely, for example where threat detection processing is implemented on a first device (e.g., server) that remotely monitors and controls process flow for threat detection and prevention of a second processing device (e.g., client).

As an example, threat detection may detection exploits that are executable files. However, one skilled in the art will recognize that the descriptions herein referring to executable files are just an example. Threat detection examples described herein can relate to any computer file. In one example, an executable file may be a portable executable (PE) file that is a file format for executables, object code, dynamic link library files (DLLs), and font library files, among other examples. However, one skilled in the art will recognize that executable files are not limited to PE files and can be any file or program that executes a task according to an encoded instruction. Knowledge component **102** described herein may collect data for use in building, training and/or re-training a learning classifier to evaluate executable files. The knowledge component **102** is one or more storages, memories, and/or modules that continuously collects and manages data that may be used to detect threats in files such as executable files. In one example, the knowledge component **102** maintains a robust collection of data comprising known malicious executable files, known benign executable files, and potentially unwanted executable files. As an example, Malicious executable files may be any type of code, data, objects, instructions, etc., that may cause harm or alter an intended function of a system and/or any system resources (e.g., memory, processes, peripherals, etc.) and/or applications running on a device such as an operating system (OS). A benign executable file is a file that upon execution, will not cause harm/damage or alter an intended system function. In other examples, a benign executable may cause harm/damage or alter an intended system; however, the potential harm/damage or alteration may be acceptable to the owner of the system or device. In examples, potentially unwanted executable files are files that may be installed on system **100** that may not be malicious, but a user of the device may not want such a file to execute and/or a file that is executed/installed unknowingly to a user. In one example, classification of executable files as malicious, benign or potentially unwanted are done by research and development support associated with developments and programming of a threat detection application/module. However, one skilled in the art will recognize that identification and classification of files into the above identified categories may be done by monitoring or evaluating a plurality of resources including but not limited to: network data, executable file libraries and information on previously known malicious executable files as well as benign executable files and potentially unwanted executable files, users/customers of threat detection/computer security software, network flow observed from use of threat detection processing and products, business associations (e.g., other existing threat detection services or partners), third-party feeds, and updates from threat detection performed using learning classifier of present disclosure, among other examples.

To classify executable files into one of the above identified categories, the knowledge component **102** collects static data on a large variety of executable files (e.g., PE files). Examples of different types of executable files collected and evaluated include but are not limited to: bit files (e.g., 32/64 bit files), operating system files (e.g., Windows, Apple, Linux, Unix, etc.), custom built files (e.g., internal tool files), corrupted files, partial downloaded files, packed files, encrypted files, obfuscated files, third party driver files, manually manipulated binary files, unicode files, infected files, and/or memory snapshots, among other examples. The data collected and maintained by the knowledge component **102** yields a knowledgebase that may be used to periodically train a classifier, e.g. learning classifier component **104** utilized by system **100**. The learning classifier component may be used to classify an executable file as one of the following classifications: malicious, benign or potentially unwanted. In some examples, classification may span two or more two or more of those classification categories. For example, an executable file may be benign in the sense that it is not harmful to system **100** but may also be classified as potentially unwanted as it might be installed without explicit user consent, for example. Data maintained by the knowledge component **102** may be continuously updated system **100** or a service that updates system **100** with new exploits to add to the training sample. For example, a research team may be employed to continuously collect new examples of harmful executables, benign executables, and potentially unwanted executables, as many unknown executable files are generated on a daily basis over the Internet. Executable files may be evaluated by the research team such as by applying applications or processing to evaluate executable files including data associated with the file and/or actions associated with the file (e.g., how it is installed and what a file does upon execution). Continuous update of the data maintained by the knowledge component **102** in conjunction with on-going re-learning/re-training of the learning classifier component **104** ensures that system **100** is up-to-date on current threats. Knowledge of the most current threats improves the generalization capability of system **100** to new unknown threats. By incorporating malicious files, benign files and potentially unwanted files, the present disclosure greatly improves a knowledge base that may be used to in training a learning classifier thereby resulting in more accurate classifications as compared with other knowledge bases that are based on only malicious and/or benign files.

In examples, the collected data on the executable files is analyzed to identify static data points that may indicate one of a malicious file, a benign file or a potentially unwanted file. For instance, the knowledge component **102** may employ one or more programming operations to identify static data points for collection, and to associate the static data points with one of the categories of files (e.g., malicious, benign or potentially unwanted). Programming operations utilized by the knowledge component **102** include operations to collect file data (e.g., executable files or data points from executable files), parse the file data, and store extracted data points. In at least one example, the knowledge component **102** comprises one or more components to manage file data. For example, the knowledge component **102** may comprise one or more storages such as databases, and one or more additional components (e.g., processors executing programs, applications, application programming interfaces (APIs), etc.).

Further, the knowledge component **102** may be configured to continuously collect data, generate a more robust collection of file data to improve classification of file data and train a classifier used to detect whether an executable file is harmful, benign, or potentially unwanted. The identification of static data points to be collected and analyzed for executable files may be continuously updated as more information becomes available to the knowledge component **102**. A static data point may be a point of reference used to evaluate an executable file. As examples, static data points include, but are not limited to: header information, section information, import and export information, certificate information, resource information, string and flag information, legal information, comments, and/or program information (e.g., APIs and DLLs), among other examples. Static data points may be organized into categories that identify a type of static data point. Categories of static data points comprise, but are not limited to: numeric values, nominal values string sequences, byte sequences, and/or Boolean values, among other examples. Any number of static data points may be collected and analyzed for an executable file. In general, collecting and analyzing a greater number of static data points results in more accurate classification of an executable file. For example, eighty static data points may be identified and collected (or attempted to be collected) from an executable file during analysis of an executable file. While a specific number is static data points are provided herein, one of skill in the art will appreciate the more or fewer static data points may be collected without departing from the scope of this disclosure. As an example, the following table, Table 1.1, identifies some of the static data points identified for analysis of an executable file, where the static data points are organized by category:

TABLE 1.1

Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heapr	file OS	special build	Code Characteristics Anomaly
size of stackr	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

Examples of the present disclosure need not distinguish between encrypted files and non-encrypted files. The static data points may be extracted from files regardless of whether or not they are encrypted and/or compressed. Given that the training set contains a statistically significant number of encrypted and non-encrypted examples, as well as compressed and decompressed files, a learning classifier (e.g., learning classifier component **104**) may be trained to identify features in the extracted data to identify malicious files. Since a very large majority of the files "in the wild" use few

tools to encrypt the files (e.g., encryption algorithm, packers, etc.), the distribution of the data found across large number of files is preserved after encryption, although the actual content of the data is transformed into something different. By collecting a large set of training examples as they appear "in the wild", a sufficient representation of features and distributions associated with the represented features exist in a training set that are both encrypted and not-encrypted. The present disclosure provides at least one benefit over other threat detection applications/programs by utilizing a very large and diverse training sample, thereby enabling a more intelligent and adaptive learning classifier that is able to achieve high detection rates for threats and low false positive rates, among other benefits.

In addition to collecting and managing information to train a learning classifier, the knowledge component **102** is used to evaluate an executable file and extract/collect static data points for evaluation of the file by the learning classifier before the file is executed. In one example, the system automatically identifies a new file (e.g., unknown file) for evaluation. As examples, identification of an executable file for evaluation may occur in examples such as: when a download of a file is requested, while a download is being performed, evaluation of streaming data, when a new file is detected as attempting to execute (e.g., potentially unwanted file), and before an unknown file or file containing executable code that was not previously checked attempts to execute, among other examples. In another example, a user of a device with which components of system **100** are operating may identify a file to be evaluated. The knowledge component **102** collects as many static data points as it can to evaluate the executable file using a learning classifier built by the learning classifier component **102**. In an exemplary extraction, the knowledge component **102** extracts each of the static data points identified in Table 1.1. The learning classifier component **102** intelligently builds a learning classifier to evaluate a file based on the extracted static data points of the file and the data managed by the knowledge component **102**.

The learning classifier component **104** is a component used to evaluate an executable file using the information provided by the knowledge component **102**. The learning classifier component **104** interfaces with the knowledge component **102** and a threat detection component **106** for the system **100** to evaluate an executable file as a possible threat. As an example, the learning classifier **104** applies programming operations or machine learning processing to evaluate static data points extracted from a file to be analyzed. In doing so, the learning classifier component **104** builds a learning classifier based on information from the knowledge component **102** including the extracted static data points of a file and the information used to train/re-train the learning classifier (e.g., static data on the robust collection of executable files including variety of file types of malicious executable files, benign executable files, and potentially unwanted executable files).

As an example, the learning classifier can adaptively set features of a learning classifier based on the static data points extracted for a file. For example, ranges of data can be identified for static data points that may enable the learning classifier to controllably select (e.g., turn on/off) features for evaluation by learning classifier. In one instance where a static data point extracted for evaluation is a file size (e.g., numeric value), the learning classifier may turn on/off features for evaluation based on whether a the file size of the file is within a certain range. In another example, the learning classifier may detect that the file does not contain "legal information" in the file. Legal information may be any identifying information indicating data that conveys rights to one or more parties including: timestamp data, licensing information, copyright information, indication of intellectual property protection, etc. In an example where the learning classifier detects the presence of legal information does not exist in the executable file, this may trigger the learning classifier to adaptively check for additional features that might be indicative of a threat or malicious file as well as turn off other features related to an evaluation of the "legal information." The learning classifier component **104** utilizes programming operations or machine-learning processing to train/re-train the learning classifier to uniquely evaluate any file/file type. One of skill in the art will appreciate different types of processing operations may be employed without departing from the spirit of this disclosure. For example, the learning classifier component **104** may utilize an artificial neural network (ANN), a decision tree, association rules, inductive logic, a support vector machine, clustering analysis, and Bayesian networks, among other examples.

The learning classifier component **104** processes and encodes collected/extracted data from a file to make the static data points suitable for processing operations. The processing and encoding executed by the learning classifier component **104** may vary depending on the identified categories and/or type (e.g., numeric values, nominal values, string/byte sequences, Boolean values, etc.) of static data points collected by the knowledge component **102**. As an example, string sequence data as well as byte sequence data may be parsed and processed as n-grams and/or n-gram word prediction (e.g., word-grams). For instance, for a given string all unigrams, bigrams and so forth up to a given length n are generated and the counts of the individual n-grams are determined. The resulting counts of the unique n-grams string and/or byte sequences are then used as input to a generated feature vector. In one example, strings are processed directly according to a bag of word model. A bag of word model is a simplifying representation used in natural language processing and information retrieval. In another example, numeric value static data points may be binned appropriately ensuring a good balance between information loss through data binning and available statistics. Nominal values as well Boolean values may be encoded using true/false flags. All of the different encoded data may be combined into one or more final feature vectors, for example after a complex vector coding/processing is performed (e.g., L2 norm).

As an application example of processing performed by the learning classifier component **104**, suppose extraction of static data points from a file identifies the following four (4) data points:

File size: 4982784
PEHeader Anomaly: False
Section name: .TEXT
Legal copyright: Copyright (C) Webroot Inc. 1997

Processing these 4 data points into a feature vector may require binning the file size, labeling the PE Header Anomaly, build n-grams for the section name and building word-gram for legal copyright. Encapsulation of special characters that sometimes appear in text (e.g., section names) may be desirable. In an example, strings of data may be transformed first into hex code representation and then build n-grams or word-grams. In this particular example, the section name is transformed into n-grams while the legal copyright text is transformed into word-grams. In evaluation using the classifier, the features related to the extracted data

points may be weighted by the particular data point being evaluated or by category, for example. For instance, the training of the learning classifier may indicate that legal information (e.g., "legal copyright") provides a better indication that a file may be malicious than that of a section name. Features being evaluated (and their distributions) differ for benign files as compared to malicious files (e.g., harmful executables such as malware). For example, the data field "Legal Copyright" for benign files tends to have meaningful words, whereas in a malicious file, this data field is often left empty or it is filled with random characters. Commercial software files tend to have a valid certificate, whereas malware in general do not have valid certificates. Similarly, each data field for a static data point evaluated provides further indication about the maliciousness of the file. The combined information from all these data fields coupled with machine learning enables accurate prediction determination as to whether a file is malicious, benign or potentially unwanted. As an example, the resulting feature vector in sparse form is shown below:

secname_002e00740065:0.204124145232
secname_0065:0.0944911182523
secname_0074006500780074:0.353553390593
secname_006500780074:0.408248290464
secname_00740065:0.144337567297
secname_002e007400650078:0.353553390593
secname_00650078:0.144337567297
secname_002e0074:0.144337567297
secname_002e:0.0944911182523
secname_00780074:0.433012701892
secname_0078:0.0944911182523
secname_007400650078:0.204124145232
secname_0074:0.472455591262
legalcopyright_0043006f00700079007200690067006-80074:0.4472135955
legalcopyright_002800430029:0.4472135955
legalcopyright_0043006f00720070002e:0.4472135955
legalcopyright_0031003900390035:0.4472135955
legalcopyright_004d0069006300720006f0073006f00-660074:0.4472135955
filesize_9965:1.0
PEHeaderAnomaly_False:1.0

In examples, the learning classifier component **104** provides the one or more feature vectors as input to a support vector machine (SVM). The SVM may then perform data analysis and pattern recognition on the one or more feature vectors. In one example the SVM, may be a linear SVM. Given a set of training examples provided by the knowledge component **102**, an SVM may build a probabilistic model that indicates whether or not a file may be malicious. In another example, a hybrid approach may be elected that combines two or more individual linear SVM classifiers into a final classifier using ensemble methods. Specifically, the evaluated static data points may be subdivided into a set of families (e.g., sections, certificate, header data, and bytes sequences are used as different families). For each family, a linear SVM may be trained. The resulting classification scores generated by each linear SVM may then be combined into a final classification score using a decision tree. As an example, the decision tree may be trained using two-class logistic gradient boosting. In yet another example of feature classification evaluation, a classification may be subdivided into a three class classification problem defined by malicious files, potentially unwanted files/applications and benign files. The resulting three class problem is solved using multi-class classification (e.g., Directed Acyclic Graph (DAG) SVM) or, in case of the hybrid approach based on feature families, using a decision tree based on three-class logistic gradient boosting.

The threat detection component **106** is a component of system **100** that evaluates results of the processing performed by the learning classifier component **104** to make a determination as to whether the executable file is malicious, benign or a potentially unwanted file/application. As an example, a probabilistic value may be a final score determined from evaluating all static data points (or feature distributions) individually and determining an aggregate score that may be used for the evaluation of an executable file. In another example, correlation between static data points may be determined by the learning classification component **104** and a final score may be generated based on the correlation between static data points evaluated.

In one example, a classification as to whether a file is malicious or not may be based on comparison with a predetermined threshold value. For instance, a final score (e.g., probability value/evaluation) is determined based on feature vector processing performed by the learning classifier component **104** and compared with a probability threshold value for determining whether an executable file is malicious. As an example, a threshold value(s) may be set based on predetermined false positive range data where ranges may be set for one or more of the malicious executable files, the benign executable files and the potentially unwanted executable files. The threshold values for each of the different types may be the same or different. For instance, a threshold value may be set that indicates a confidence score in classifying an executable file as a malicious executable file. Ranges may be determined indicating how confident the system **100** is in predicting that a file is malicious. That may provide an indication of whether further evaluation of an executable file should occur. However, one skilled in the art will recognize that threshold determinations can be set in any way that be used to determine whether an executable file is malicious or not. Examples of further evaluation include but are not limited to: additional processing using a learning classifier, identification of an executable file as potentially malicious where services associated with the system **100** may follow-up, quarantining a file, and moving the file to a secure environment for execution (e.g., sandbox) among other examples. In other examples, retraining of the classifier may occur based on confidence scores. In examples, when a probability value exceeds or threshold value (or alternatively is less than a threshold value), it may be determined that an executable file may be identified as malicious. One skilled in the art will recognize that determination for predictive classification of executable files is not limited to threshold determinations. For example, any type of analytical, statistical or graphical analysis may be performed on data to classify an executable file/unknown executable file. As identified above, the threat detection component **106** may interface with the learning classifier component **104** to make a final determination as to how to classify an executable file as well as interface with the knowledge component **102** for training/retraining associated with the learning classifier.

FIG. **2** illustrates an exemplary distributed system **200** showing interaction of components for implementation of exemplary threat detection system as described herein. Where FIG. **1** illustrates an example system **100** having components (hardware or software) operating on a client, FIG. **2** illustrates an exemplary distributed system **200** comprising a client component **202** connected with at least one server component **204** via communication line **203**. Communication line **203** represents the ability of the client component **202** to communicate with the server component **204**, for example, to send or receive information over a

network connection. That is, client component **202** and server component **204** are connectable over a network connection (e.g., a connection Internet via, for example, a wireless connection, a mobile connection, a hotspot, a broadband connection, a dial-up, a digital subscriber line, a satellite connection, an integrated services digital network, etc.).

The client component **202** may be any hardware (e.g., processing device) or software (e.g., application/service or remote connection running on a processing device) that accesses a service made available by the server component **204**. The server component **204** may be any hardware or software (e.g., application or service running on a processing device) capable of communicating with the client component **202** for execution of threat detection processing (e.g., threat detection application/service). Threat detection processing may be used to evaluate a file before execution of the file as described in FIG. **1**. Threat detection application or services may be present on at least one of the client component **202** and the server component **204**. In other examples, applications or components (e.g., hardware or software) may be present on both the client component **202** and the server component **204** to enable processing by threat detection application/services when a network connection to the server cannot be established. In one example of system **200**, client component **202** (or server component **204**) may comprise one or more components for threat detection as described in system **100** including a knowledge component **102**, a learning classifier component **104** and/or a threat detection component **106**, as described in the description of FIG. **1**. In other examples, the client component **202** may transmit data to/from the server component **204** to enable threat detections services over distributed network **200**, for example as represented by communication line **203**. In one example, threat detection applications/services operating on the client component **202** may receive updates from the server component **204**. For instance, updates may be received by the client component **202** for re-training of a learning classifier used to evaluate an executable file on the client component **202**.

FIG. **3** illustrates an exemplary method **300** for performing threat detection. As an example, method **300** may be executed by an exemplary system such as system **100** of FIG. **1** and system **200** of FIG. **2**. In other examples, method **300** may be executed on a device comprising at least one processor configured to store and execute operations, programs or instructions. However, method **300** is not limited to such examples. Method **300** may be performed by any application or service that may include implementation of threat detection processing as described herein. The method **300** may be implemented using software, hardware or a combination of software and hardware.

Method **300** begins at operation **302** where a knowledge base is built for training/retraining of a learning classifier used to detect threats in executable files. Operation **302** builds the knowledge base from data collected and evaluated related to known malicious executable files, known benign executable files and potentially unwanted executable files as described in the description of FIG. **1** (e.g., knowledge component **102**). The knowledge base may be used to automatically train/re-train one or more learning classifiers used to evaluate threats in executable files based on the known malicious executable files, known benign executable files and the potentially unwanted executable files. In examples, the knowledge base may be maintained on at least one of a client component and a server component, and the knowledge base may be continuously updated with information from the resources described in FIG. **1** including update information based on threat detection processing evaluation performed on unknown executable files.

When an executable file is identified for evaluation, flow proceeds to operation **304** where a one or more static data points are extracted from an executable file. In operation **304**, an executable file is analyzed using machine learning processing as described with respect to the knowledge component **102** of FIG. **1** to collect/extract static data points from an executable file for evaluation. In examples, operation **304** occurs without decrypting or unpacking the executable file. However, in other examples, machine learning processing performed has the capability to evaluate static data points from extracted/unpacked content. In one example, machine learning processing is used to extract static data points from encrypted and/or compress versions of one or more files. In another example, machine learning processing is used to extract static data points from decrypted and/or unpacked versions of one or more files. In any example, extraction of static data points from different files (e.g. executable files) can be used to enhance training of a learning classifier, providing better results for extraction of static data points and classification of files.

In examples, operation **304** further comprises classifying extracted data according to a type of data extracted. For instance, the actions performed at operation **304** may be used to classify the plurality of static data points extracted into categories (categorical values) comprising numeric values, nominal values, string or byte sequences, and Boolean values, for example, as described with respect to FIG. **1**. In examples, data of an executable file (e.g., binary file data) may be parsed and compared against data maintained by a threat detection application/service as described in the present disclosure to determine static data points for extraction/collection.

In operation **306**, the executable file is analyzed for threats. As an example, operation **306** analyzes an executable without decrypting or unpacking the executable file. However, in other examples, the machine learning processing being performed has the capability to evaluate static data points from extracted/unpacked content. Operation **306** comprises applying a learning classifier (e.g., the learning classifier generated during performance of operation **302**) to the plurality of static data points extracted from the file. As discussed, the learning classifier may be built from data comprising known malicious executable files, known benign executable files and known unwanted executable files, for example. In one example, operation **306** comprises generating at least one feature vector from the plurality of static data points extracted using the learning classifier trained by the knowledge base. In order to generate the feature vector for the learning classifier, data may be parsed and encoded for machine learning processing.

In one example, generation of the feature vector may comprise selectively setting features of the learning classifier based on the one or more of static data points extracted. Features of the generated feature vector may be weighted based on classified categories identified by the knowledge base (as described above) and the plurality of static data points extracted from the file. As an example, one or more features of the feature vector may be selectively turned on or off based on evaluation of whether a value of a static data point is within a predetermined range. However, one skilled in the art will recognize that the learning classifier can uniquely generate a feature vector for analysis of the executable file based on any data used to train/ re-train the learning classifier. In examples, operation **306** further comprises

evaluating the feature vector using linear or nonlinear support vector processing to determine a classification for the executable file, for example whether the executable file is harmful, benign, or unwanted.

Flow proceeds to operation **308**, where a determination is made as to a classification of the executable file. For example, operation **308** makes a final determination as to whether the executable file is harmful (e.g., malicious, malware) or not based on results of the analysis of the executable file (e.g., using machine learning processing by a learning classifier). In one example, results of the analysis of an executable file may be data obtained from learning classifier, such as, for example an SVM, processing data. In one example, operation **308** further comprises preventing execution of the executable file when a probability value that the executable file is harmful exceeds a threshold value. The probability value for the executable file may be determined based on applying the learning classifier to the executable file. As an example, the threshold value may be set based on predetermined false positive range data for identifying a malicious or harmful executable file. False positive range data may be determined from the analysis/evaluation of the known malicious executable files, known benign files and potentially unwanted executable files/applications, of the knowledge base. However, as acknowledged above, determining a classification of an executable file may be based on any type of analytical, statistical or graphical analysis, or machine learning processing. In one example, ranges can be based on evaluation of data during operation of the threat detection service as well as analysis related to unknown files, for example analytics performed to evaluate unknown files.

At any point in time, operation **310** may occur where a learning classifier used for threat detection processing is re-trained. Continuously re-training of the learning classifier may ensure that the threat detection application/service is up to date and able to accurately detect new threats. As identified above, re-training may occur through results of threat detection processing including updated information added to the knowledge base. In one example, training of a learning classifier can be based on evaluation of data during operation of the threat detection service as well as analysis related to unknown files, for example analytics performed to evaluate unknown files.

FIG. **4** and the additional discussion in the present specification are intended to provide a brief general description of a suitable computing environment in which the present invention and/or portions thereof may be implemented. Although not required, the embodiments described herein may be implemented as computer-executable instructions, such as by program modules, being executed by a computer, such as a client workstation or a server. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. Moreover, it should be appreciated that the invention and/or portions thereof may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

FIG. **4** illustrates one example of a suitable operating environment **400** in which one or more of the present embodiments may be implemented. This is only one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality. Other well-known computing systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics such as smart phones, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

In its most basic configuration, operating environment **400** typically includes at least one processing unit **402** and memory **404**. Depending on the exact configuration and type of computing device, memory **404** (storing, among other things, executable evaluation module(s), e.g., malware detection applications, APIs, programs etc. and/or other components or instructions to implement or perform the system and methods disclosed herein, etc.) may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.), or some combination of the two. This most basic configuration is illustrated in FIG. **4** by dashed line **406**. Further, environment **400** may also include storage devices (removable, **408**, and/or non-removable, **410**) including, but not limited to, magnetic or optical disks or tape. Similarly, environment **400** may also have input device(s) **414** such as keyboard, mouse, pen, voice input, etc. and/or output device(s) **416** such as a display, speakers, printer, etc. Also included in the environment may be one or more communication connections, **412**, such as LAN, WAN, point to point, etc.

Operating environment **400** typically includes at least some form of computer readable media. Computer readable media can be any available media that can be accessed by processing unit **402** or other devices comprising the operating environment. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other non-transitory medium which can be used to store the desired information. Computer storage media does not include communication media.

Communication media embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

The operating environment **400** may be a single computer operating in a networked environment using logical connec-

tions to one or more remote computers. The remote computer may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above as well as others not so mentioned. The logical connections may include any method supported by available communications media. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

The different aspects described herein may be employed using software, hardware, or a combination of software and hardware to implement and perform the systems and methods disclosed herein. Although specific devices have been recited throughout the disclosure as performing specific functions, one of skill in the art will appreciate that these devices are provided for illustrative purposes, and other devices may be employed to perform the functionality disclosed herein without departing from the scope of the disclosure.

As stated above, a number of program modules and data files may be stored in the system memory **404**. While executing on the processing unit **402**, program modules **408** (e.g., applications, Input/Output (I/O) management, and other utilities) may perform processes including, but not limited to, one or more of the stages of the operational methods described herein such as method **300** illustrated in FIG. **3**, for example.

Furthermore, examples of the invention may be practiced in an electrical circuit comprising discrete electronic elements, packaged or integrated electronic chips containing logic gates, a circuit utilizing a microprocessor, or on a single chip containing electronic elements or microprocessors. For example, examples of the invention may be practiced via a system-on-a-chip (SOC) where each or many of the components illustrated in FIG. **4** may be integrated onto a single integrated circuit. Such an SOC device may include one or more processing units, graphics units, communications units, system virtualization units and various application functionality all of which are integrated (or "burned") onto the chip substrate as a single integrated circuit. When operating via an SOC, the functionality described herein may be operated via application-specific logic integrated with other components of the operating environment **400** on the single integrated circuit (chip). Examples of the present disclosure may also be practiced using other technologies capable of performing logical operations such as, for example, AND, OR, and NOT, including but not limited to mechanical, optical, fluidic, and quantum technologies. In addition, examples of the invention may be practiced within a general purpose computer or in any other circuits or systems.

This disclosure described some aspects of the present technology with reference to the accompanying drawings, in which only some of the possible embodiments were shown. Other aspects may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein. Rather, these aspects were provided so that this disclosure was thorough and complete and fully conveyed the scope of the possible embodiments to those skilled in the art.

Although specific aspects were described herein, the scope of the technology is not limited to those specific embodiments. One skilled in the art will recognize other embodiments or improvements that are within the scope and spirit of the present technology. Therefore, the specific structure, acts, or media are disclosed only as illustrative embodiments. The scope of the technology is defined by the following claims and any equivalents therein.

What is claimed is:

1. A computer-implemented method comprising:

extracting a plurality of static data points from an executable file without decrypting or unpacking the executable file, wherein the plurality of static data points represent predefined character strings in the executable file;

generating a feature vector from the plurality of static data points using a classifier trained to classify the plurality of static data points based on a collection of data comprising known malicious executable files, known benign executable files, and known unwanted executable files, wherein the collection of data comprises at least a portion of the plurality of static data points, and wherein one or more features of the feature vector are selectively turned on or off based on whether a value of one or more static data points from the plurality of extracted static data points is within a predetermined range; and

evaluating the feature vector using support vector processing to determine whether the executable file is harmful.

2. The computer-implemented method according to claim **1**, wherein the extracting the plurality of static data points further comprises classifying the extracted static data points according to a type of data extracted from the executable file, and encoding the extracted static data points for the classifier based on the classification.

3. The computer-implemented method according to claim **2**, wherein generating the at least one feature vector comprises selectively setting features of the classifier based on the plurality of extracted static data points.

4. The computer-implemented method according to claim **2**, wherein the classifying the at least one static data point comprises classifying the plurality of extracted static data points into categories comprising numeric values, nominal values, string or byte sequences, and Boolean values.

5. The computer-implemented method according to claim **4**, wherein analyzing the executable file further comprises generating at least one feature vector from the plurality of extracted static data points, wherein features of the generated feature vector are weighted based at least upon the classified categories and the plurality of extracted static data points.

6. The computer-implemented method according to claim **1**, wherein determining whether the executable file is harmful further comprises preventing execution of the executable file when a determined probability value that the executable file is harmful exceeds a threshold value.

7. The computer-implemented method according to claim **6**, wherein the threshold value is set based on predetermined false positive range data.

8. The computer-implemented method according to claim **1**, wherein the plurality of static data points is extracted using a machine learning technique.

9. The computer-implemented method according to claim **1**, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

10. A computer-readable storage device containing instructions, that when executed on at least one processor, causing the processor to execute a process comprising:

extracting a plurality of static data points from an executable file without decrypting or unpacking the execut-

able file, wherein the plurality of static data points represent predefined character strings in the executable file;

generating a feature vector from the plurality of static data points using a classifier trained to classify the plurality of static data points based on a collection of data comprising known malicious executable files, known benign executable files and known unwanted executable files, wherein the collection of data comprises at least a portion of the plurality of static data points, and wherein one or more features of the feature vector are selectively turned on or off based on whether one or more values of one or more static data points from the plurality of extracted static data points is within a predetermined range; and

evaluating the feature vector using support vector processing to determine whether the executable file is harmful.

11. The computer-readable storage device according to claim **10**, wherein the extracting of the plurality of static data points executed by the processor further comprises classifying the extracted static data points according to a type of data extracted from the executable file, and encoding the extracted static data points for the classifier based on the classifying.

12. The computer-readable storage device according to claim **11**, wherein generating the at least one feature vector comprises selectively setting features of the classifier based on the plurality of extracted static data points.

13. The computer-readable storage device according to claim **11**, wherein the classifying the at least one static data point comprises classifying the plurality of extracted static data points into categories comprising numeric values, nominal values, string or byte sequences, and Boolean values.

14. The computer-readable storage device according to claim **13**, wherein analyzing the executable file further comprises generating a feature vector from the plurality of extracted static data points, wherein features of the generated feature vector are weighted based at least upon the classified categories and the plurality of extracted static data points.

15. The computer-readable storage device according to claim **10**, wherein determining whether the executable file is harmful further comprises preventing execution of the executable file when a determined probability value that the executable file is harmful exceeds a threshold value.

16. The computer-readable storage device according to claim **15**, wherein the threshold value is set based on predetermined false positive range data.

17. A system comprising:

at least one memory; and

at least one processor operatively connected with the memory and configured to perform operation of:

extracting a plurality of predefined character strings from an executable file without decrypting or unpacking the executable file;

generating a feature vector from the plurality of predefined character strings using a classifier trained to classify the plurality of predefined character strings based on a collection of data comprising known malicious executable files, known benign executable files and known unwanted executable files, wherein the collection of data comprises at least a portion of one or more of the plurality of predefined character strings, and wherein one or more features of the feature vector are selectively turned on or off based on whether a value of one or more predefined character strings from the plurality of predefined character strings is within a predetermined range; and

evaluating the feature vector using support vector processing to determine whether the executable file is harmful.

18. The system according to claim **17**, wherein the determining further comprises preventing execution of the executable file when a determined probability value that the executable file is harmful exceeds a threshold value, and wherein the threshold value is set based on predetermined false positive range data.

19. The system according to claim **17**, wherein the extracting the plurality of predefined character strings further comprises classifying the extracted plurality of predefined character strings according to a type of data extracted from the executable file, and encoding the extracted plurality of predefined character strings for the classifier based on the classification.

20. The system according to claim **17**, wherein determining whether the executable file is harmful further comprises preventing execution of the executable file when a determined probability value that the executable file is harmful exceeds a threshold value.

* * * * *

Exhibit 3



THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

October 07, 2021

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *8,438,386*

ISSUE DATE: *May 07, 2013*

By Authority of the

Under Secretary of Commerce for Intellectual Property

and Director of the United States Patent and Trademark Office

R GLOVER

Certifying Officer

US008438386B2

(12) **United States Patent**
Hegli et al.

(10) **Patent No.: US 8,438,386 B2**
(45) **Date of Patent: May 7, 2013**

(54) **SYSTEM AND METHOD FOR DEVELOPING A RISK PROFILE FOR AN INTERNET SERVICE**

(75) Inventors: **Ron Hegli**, San Diego, CA (US); **Hal Lonas**, Carlsbad, CA (US); **Christopher K. Harris**, San Diego, CA (US)

(73) Assignee: **Webroot Inc.**, Broomfield, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 319 days.

(21) Appl. No.: **12/709,504**

(22) Filed: **Feb. 21, 2010**

(65) **Prior Publication Data**

US 2010/0269168 A1 Oct. 21, 2010

Related U.S. Application Data

(60) Provisional application No. 61/171,264, filed on Apr. 21, 2009, provisional application No. 61/241,389, filed on Sep. 10, 2009.

(51) **Int. Cl.**

H04L 29/06	(2006.01)
G06F 17/30	(2006.01)
G06F 17/00	(2006.01)
G06F 7/04	(2006.01)
G06F 15/167	(2006.01)
G06F 15/16	(2006.01)
G06F 15/173	(2006.01)

(52) **U.S. Cl.**

USPC **713/166**; 713/153; 713/165; 726/4; 726/11; 726/12; 726/29; 709/216; 709/217; 709/224; 709/225

(58) **Field of Classification Search** 726/11, 726/4, 12, 29; 713/153, 165, 166; 709/216, 709/217, 224, 225

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,606,659	B1	8/2003	Hegli et al.
6,973,488	B1	12/2005	Yavatkar et al.
7,287,279	B2	10/2007	Bertman et al.
7,383,299	B1	6/2008	Hailpern et al.
7,406,456	B2	7/2008	Roy et al.
7,467,206	B2	12/2008	Moore et al.
7,483,982	B2	1/2009	Hegli et al.

(Continued)

OTHER PUBLICATIONS

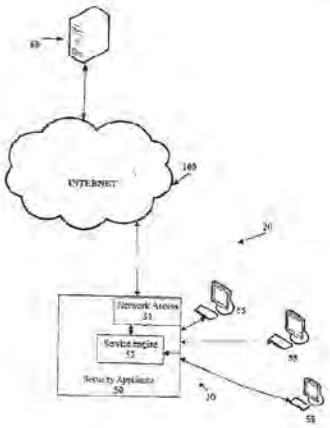
PCT International Search Report for PCT Application No. PCT/US2010/025702, Mailed Sep. 17, 2010.

Primary Examiner — Aravind Moorthy
(74) *Attorney, Agent, or Firm* — Sheridan Ross P.C.

(57) **ABSTRACT**

A method and system for controlling access to an Internet resource is disclosed herein. When a request for an Internet resource, such as a Web site, is transmitted by an end-user of a LAN, a security appliance for the LAN analyzes a reputation index for the Internet resource before transmitting the request over the Internet. The reputation index is based on a reputation vector which includes a plurality of factors for the Internet resource such as country of domain registration, country of service hosting, country of an internet protocol address block, age of a domain registration, popularity rank, internet protocol address, number of hosts, to-level domain, a plurality of run-time behaviors, JavaScript block count, picture count, immediate redirect and response latency. If the reputation index for the Internet resource is at or above a threshold value established for the LAN, then access to the Internet resource is permitted. If the reputation index for the Internet resource is below a threshold value established for the LAN, then access to the Internet resource is denied.

26 Claims, 11 Drawing Sheets



Copy provided by USPTO from the PIRS Image Database on 10-05-2021

U.S. PATENT DOCUMENTS

7,493,403	B2	2/2009	Shull et al.	
7,591,016	B2	9/2009	Horne	
7,721,333	B2	5/2010	Horne	
2004/0122926	A1	6/2004	Moore et al.	
2005/0204050	A1	9/2005	Turley et al.	
2006/0143703	A1	6/2006	Hopen et al.	
2006/0253578	A1	11/2006	Dixon et al.	
2007/0078675	A1	4/2007	Kaplan	
2007/0130350	A1	6/2007	Alperovitch et al.	
2008/0015925	A1	1/2008	Sundaresan	
2008/0052758	A1	2/2008	Byrnes	
2008/0082662	A1 *	4/2008	Dandliker et al.	709/225
2008/0097936	A1	4/2008	Schmidtler et al.	
2009/0006569	A1 *	1/2009	Morss et al.	709/206
2009/0063248	A1	3/2009	Chong et al.	
2009/0064337	A1	3/2009	Chien	
2009/0178125	A1	7/2009	Barber et al.	
2010/0184505	A1 *	7/2010	Bryson et al.	463/25

* cited by examiner

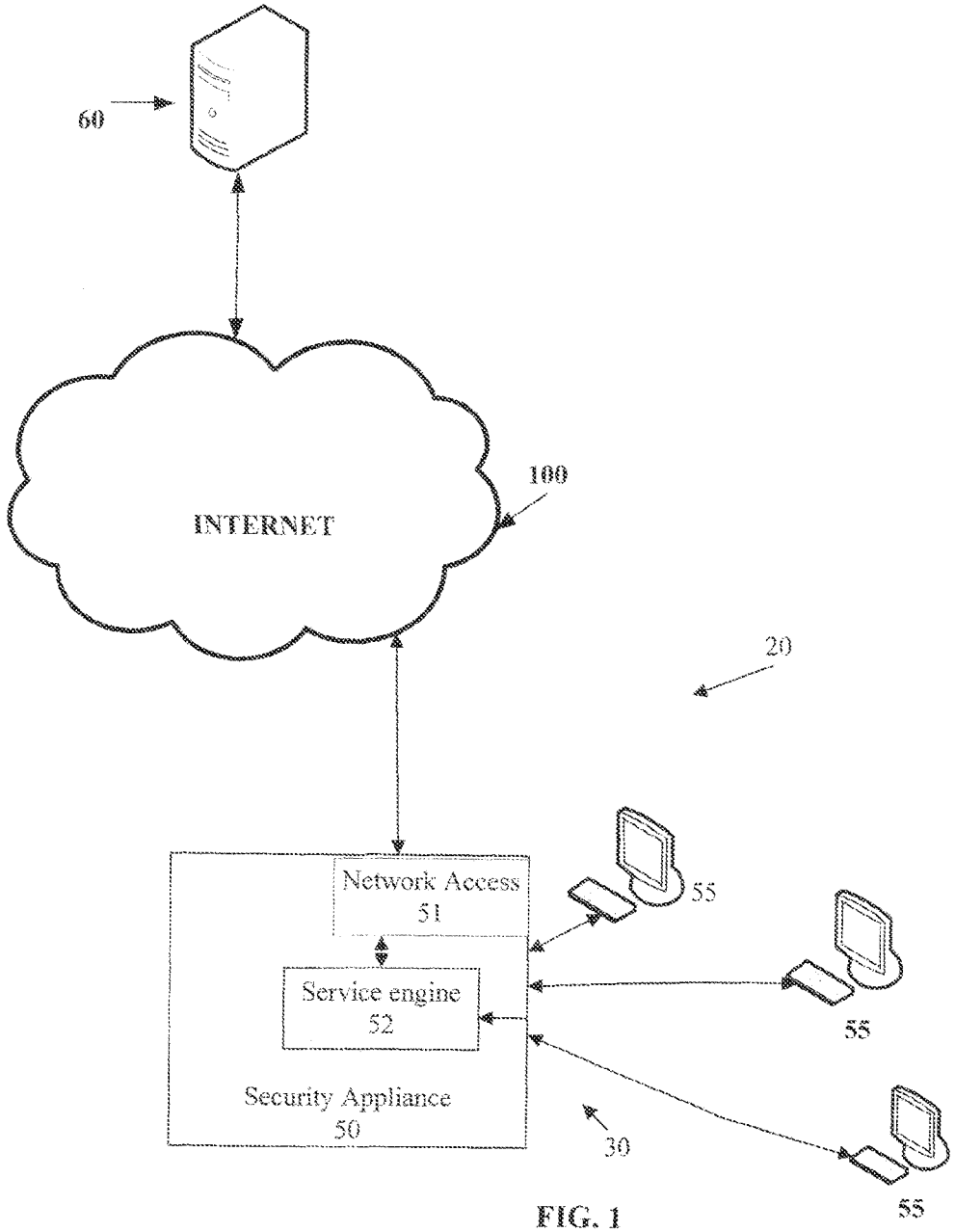

60
INTERNET
100
20
Network Access
51
Service engine
52
Security Appliance
50
30
55
55
55

FIG. 1

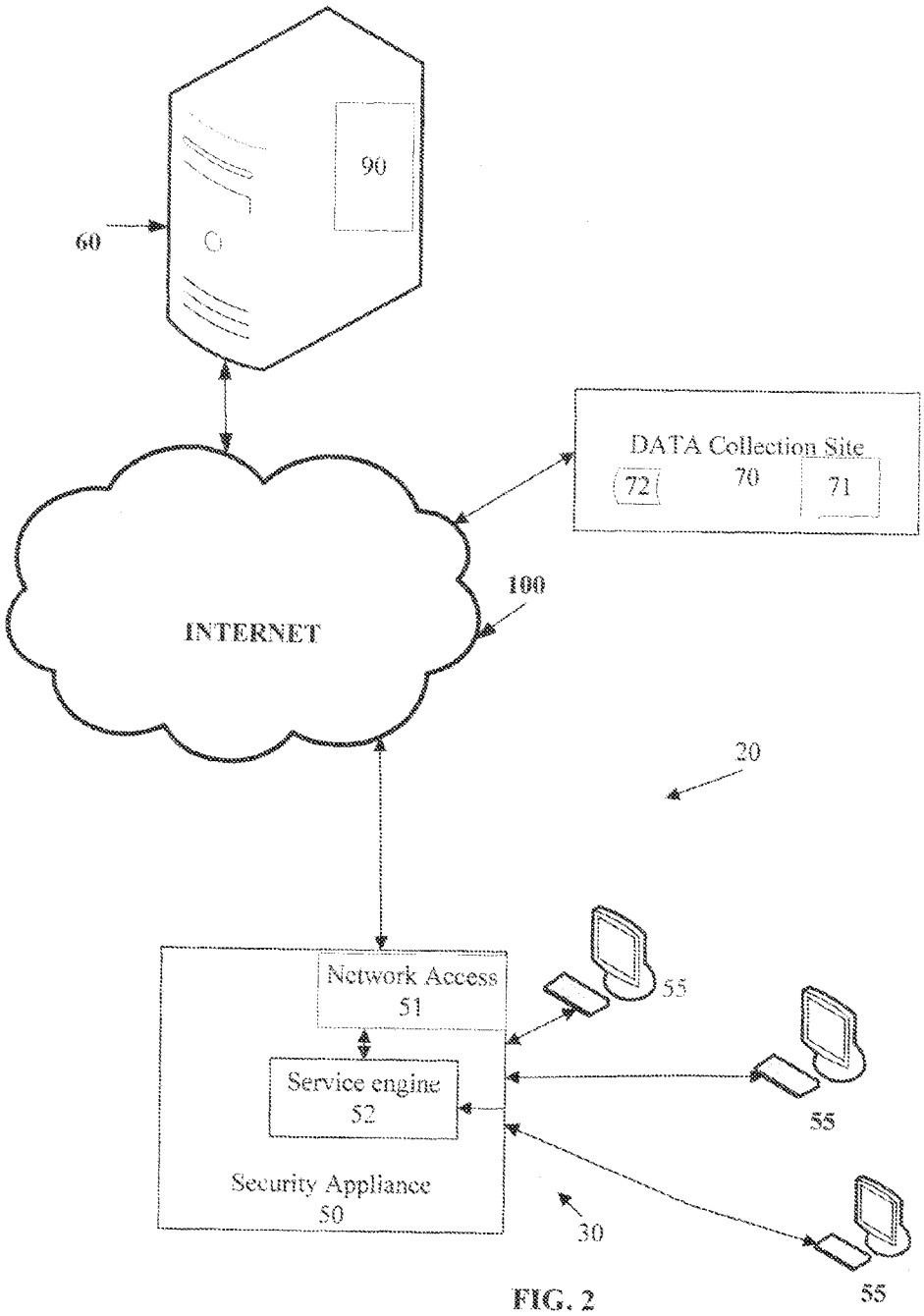

90
60
DATA Collection Site
72
70
71
INTERNET
100
20
Network Access
51
55
Service engine
52
55
Security Appliance
50
30
55

FIG. 2

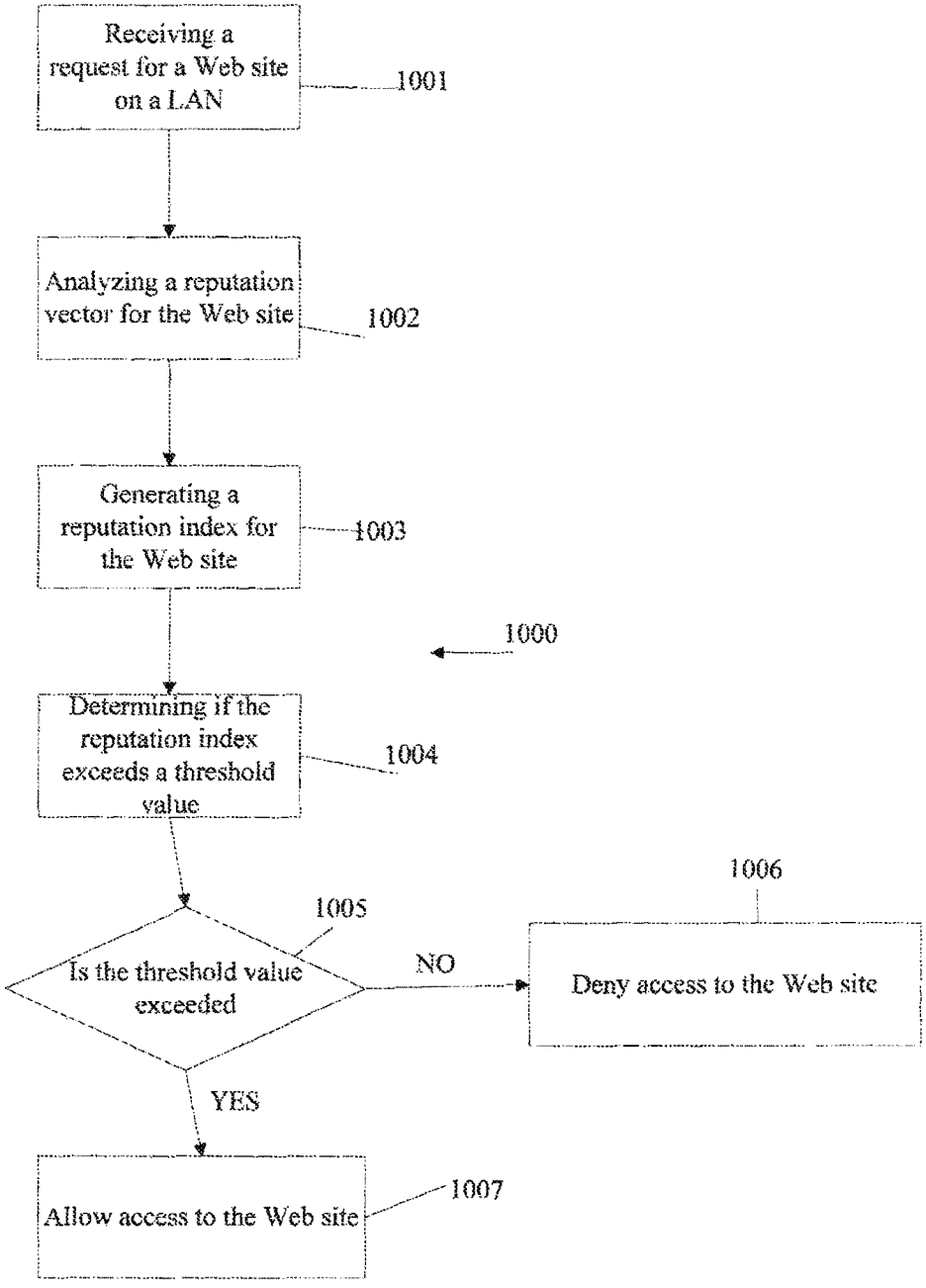

Receiving a request for a Web site on a LAN
1001
Analyzing a reputation vector for the Web site
1002
Generating a reputation index for the Web site
1003
1000
Determining if the reputation index exceeds a threshold value
1004
1005
Is the threshold value exceeded
NO
1006
Deny access to the Web site
YES
Allow access to the Web site
1007

FIG. 3

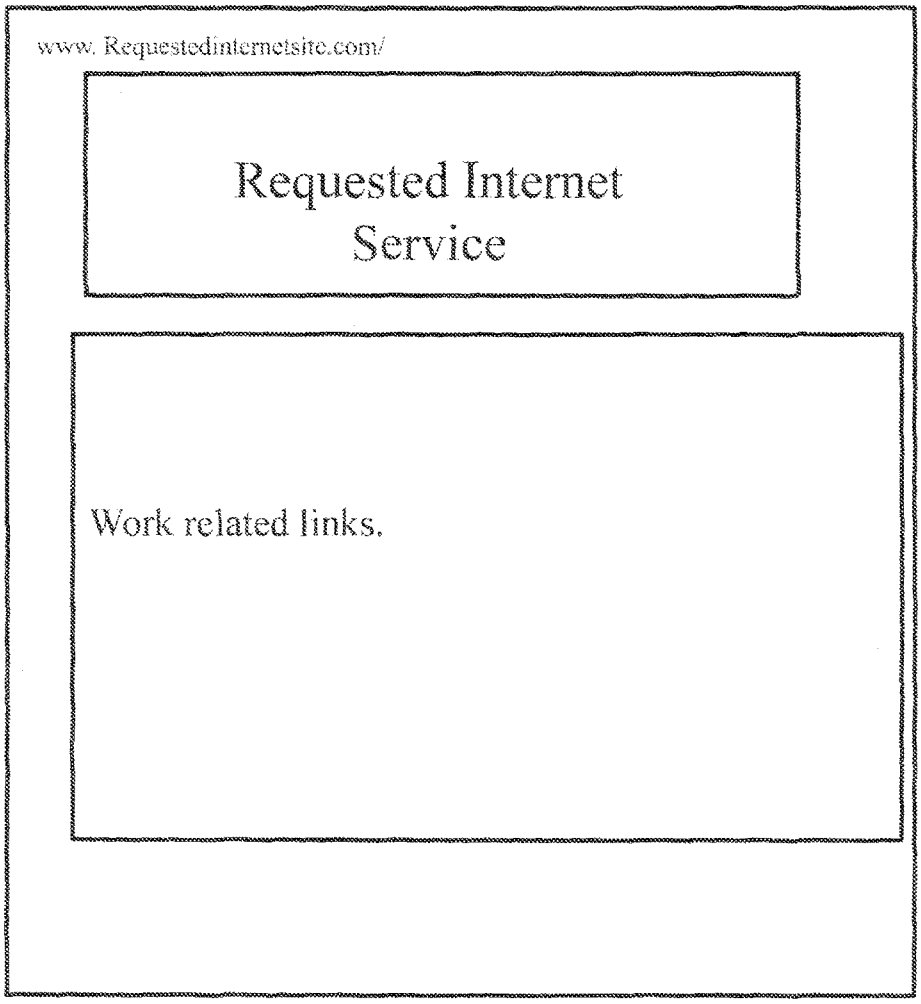

www. Requestedinternetsite.com/
Requested Internet Service
Work related links.
400

FIG. 4

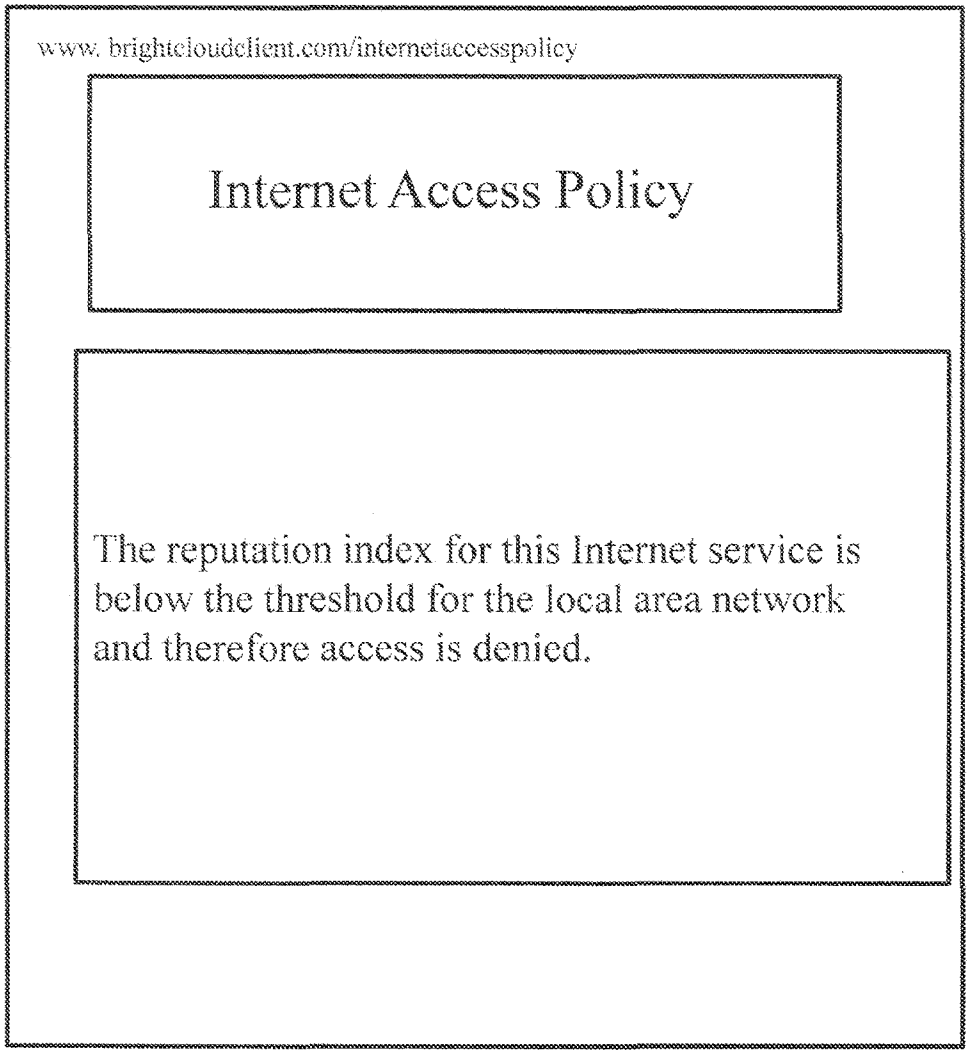
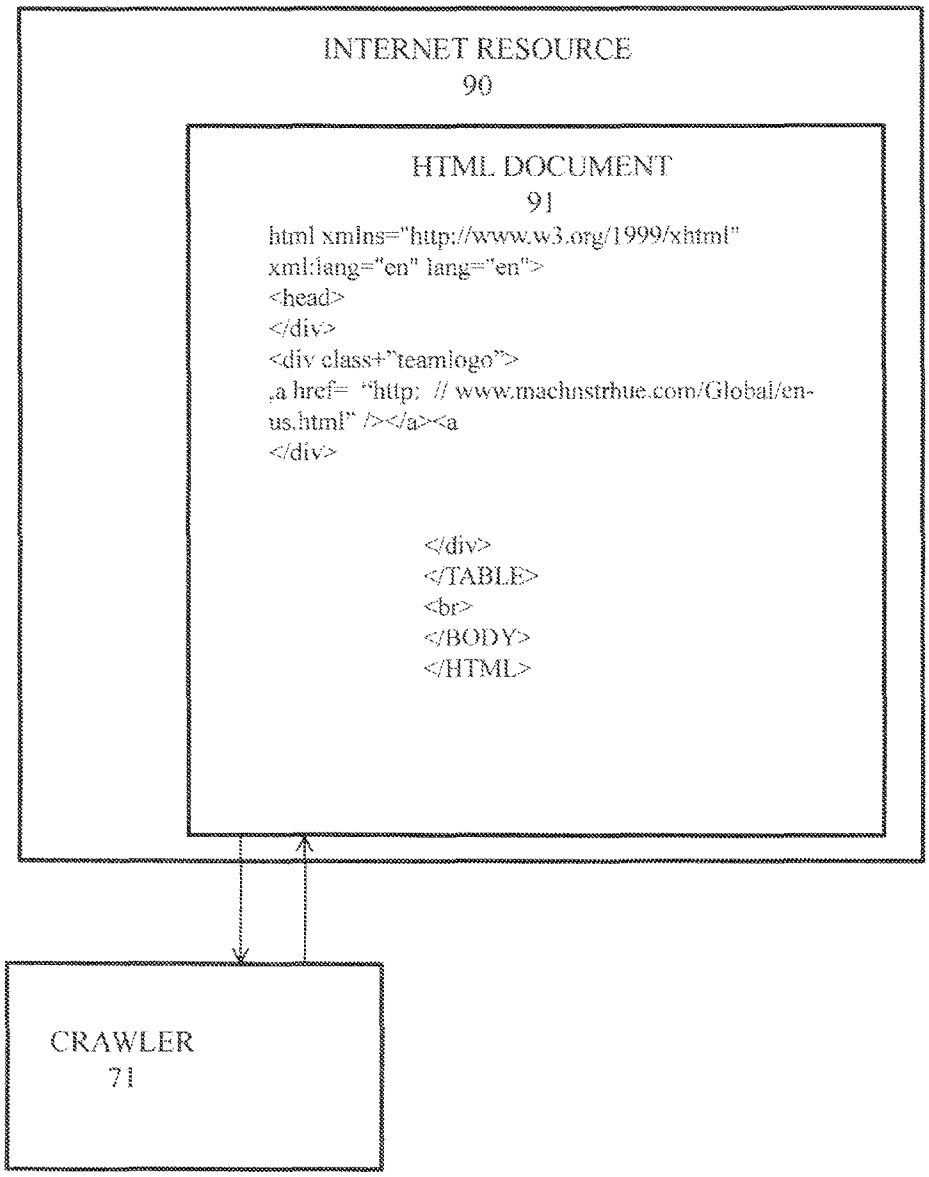

www. brightcloudclient.com/internetaccesspolicy
Internet Access Policy
The reputation index for this Internet service is below the threshold for the local area network and therefore access is denied.
500

FIG. 5


INTERNET RESOURCE
90
HTML DOCUMENT
91
html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en" lang="en">
<head>

<div class+"teamlogo">
,a href= "http: // www.machnstrhue.com/Global/en-
us.html" /><a

</TABLE>

</BODY>
</HTML>
CRAWLER
71

FIG. 6

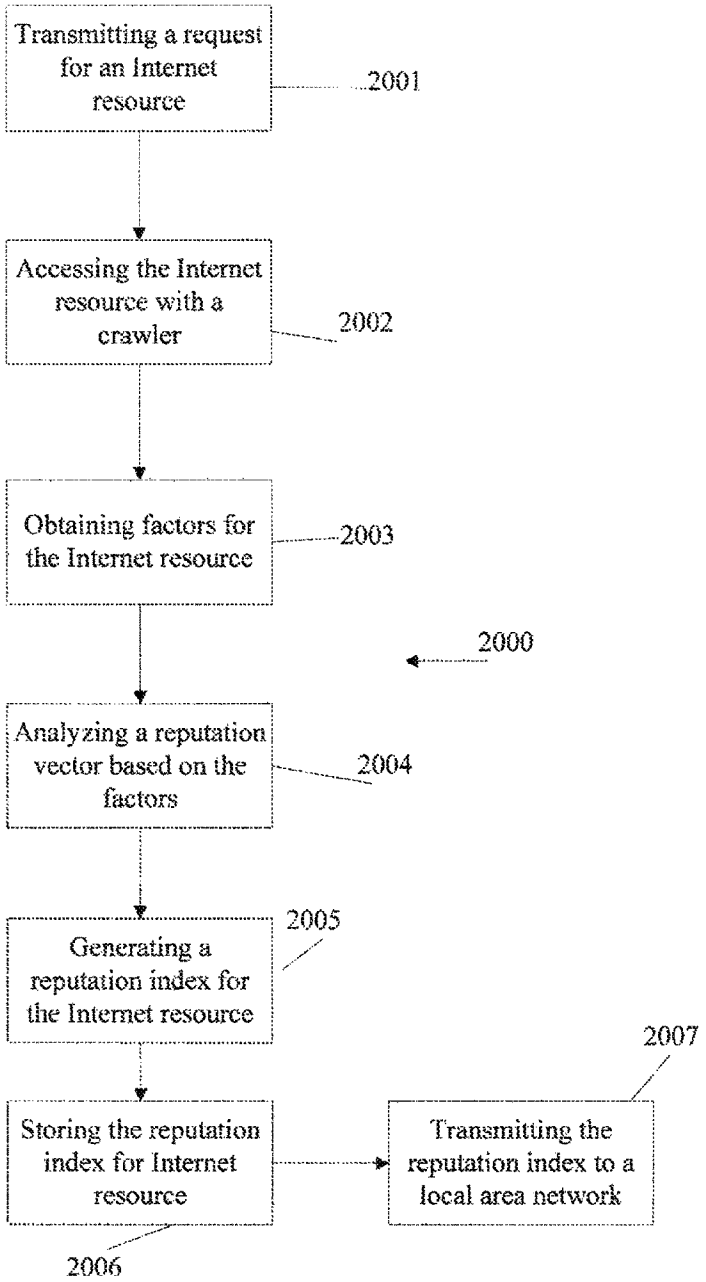

Transmitting a request for an Internet resource
2001
Accessing the Internet resource with a crawler
2002
Obtaining factors for the Internet resource
2003
2000
Analyzing a reputation vector based on the factors
2004
Generating a reputation index for the Internet resource
2005
2007
Storing the reputation index for Internet resource
Transmitting the reputation index to a local area network
2006

FIG. 7

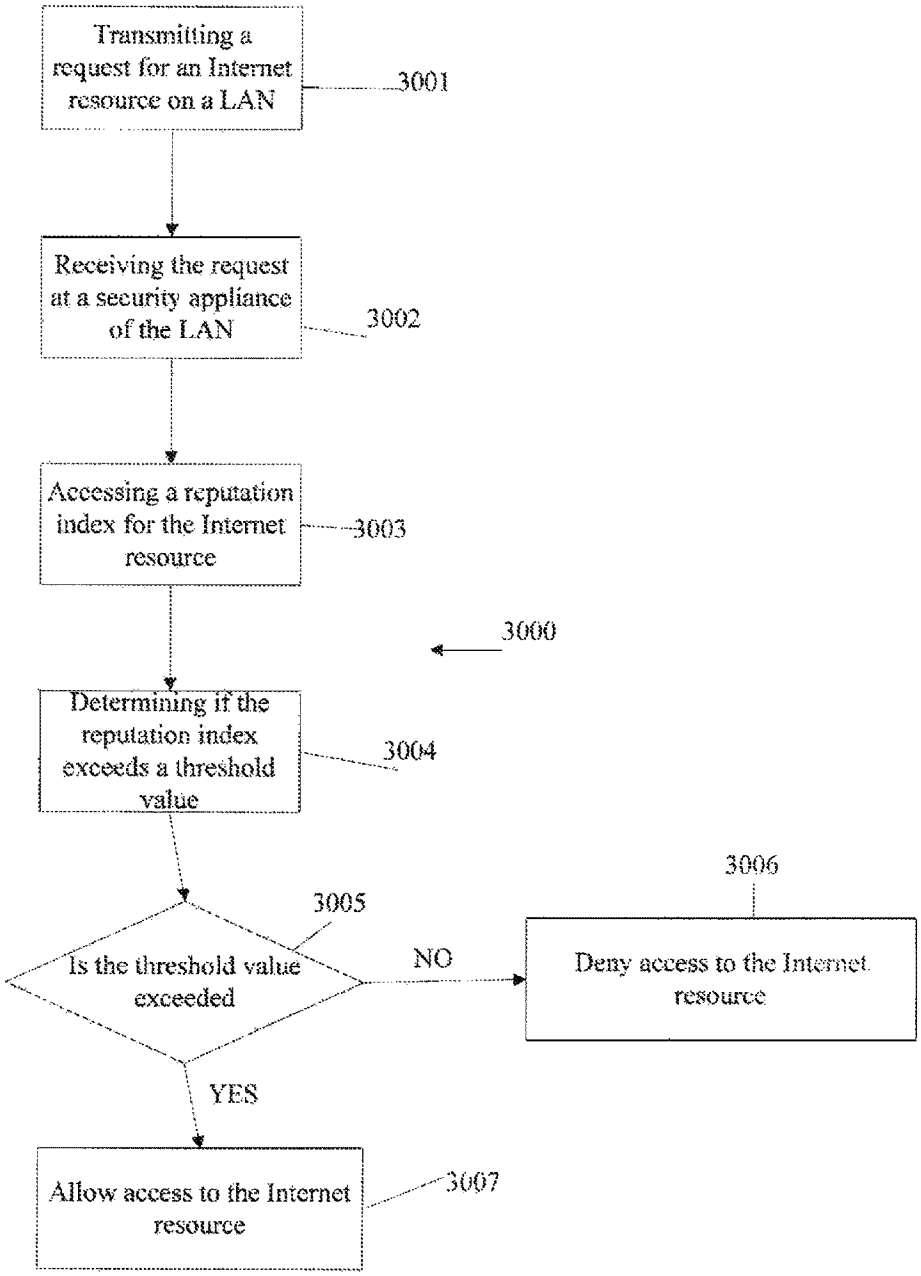

Transmitting a request for an Internet resource on a LAN
3001
Receiving the request at a security appliance of the LAN
3002
Accessing a reputation index for the Internet resource
3003
3000
Determining if the reputation index exceeds a threshold value
3004
3006
3005
Is the threshold value exceeded
NO
Deny access to the Internet resource
YES
Allow access to the Internet resource
3007

FIG. 8

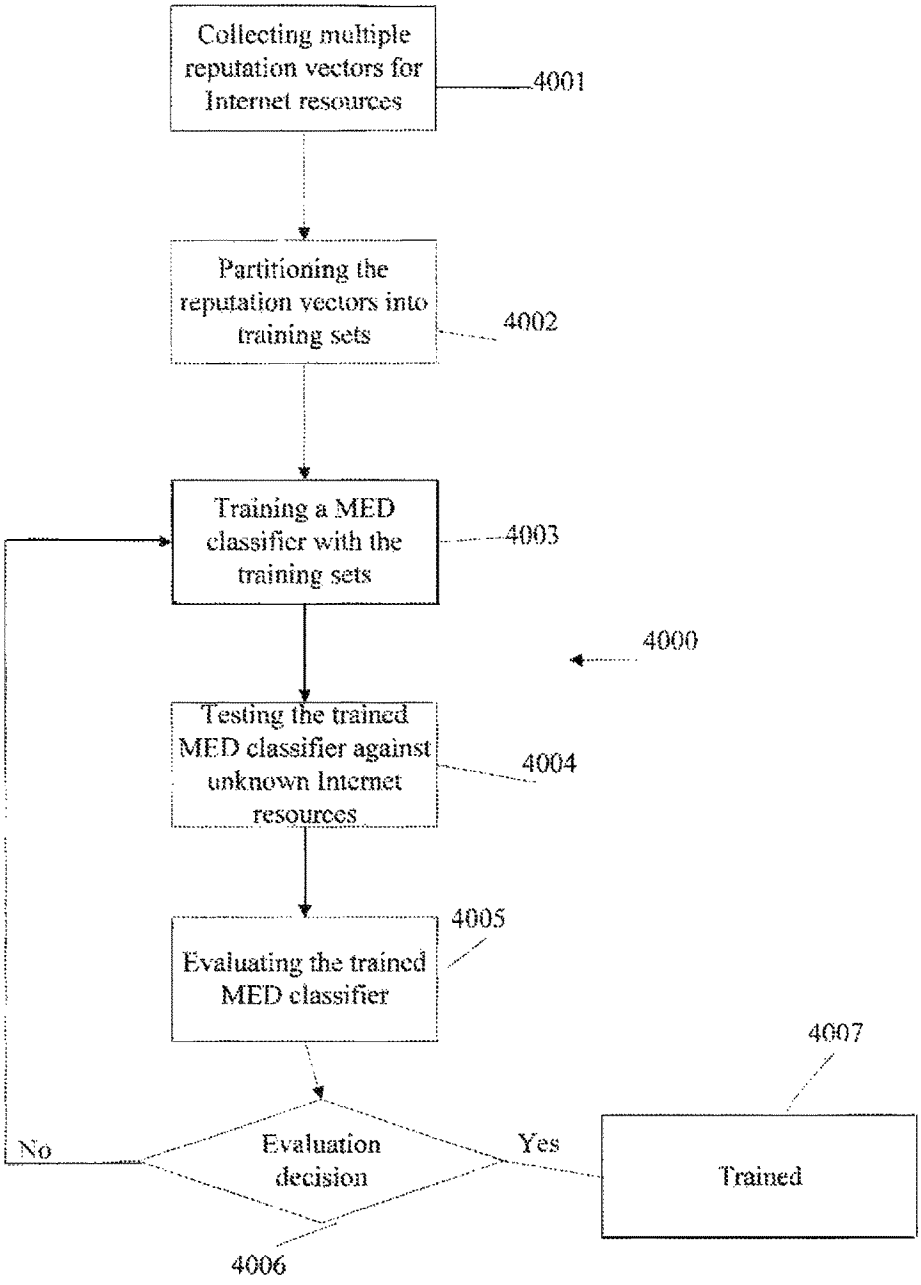

Collecting multiple reputation vectors for Internet resources
4001
Partitioning the reputation vectors into training sets
4002
Training a MED classifier with the training sets
4003
4000
Testing the trained MED classifier against unknown Internet resources
4004
Evaluating the trained MED classifier
4005
4007
Evaluation decision
No
Yes
Trained
4006

FIG. 9

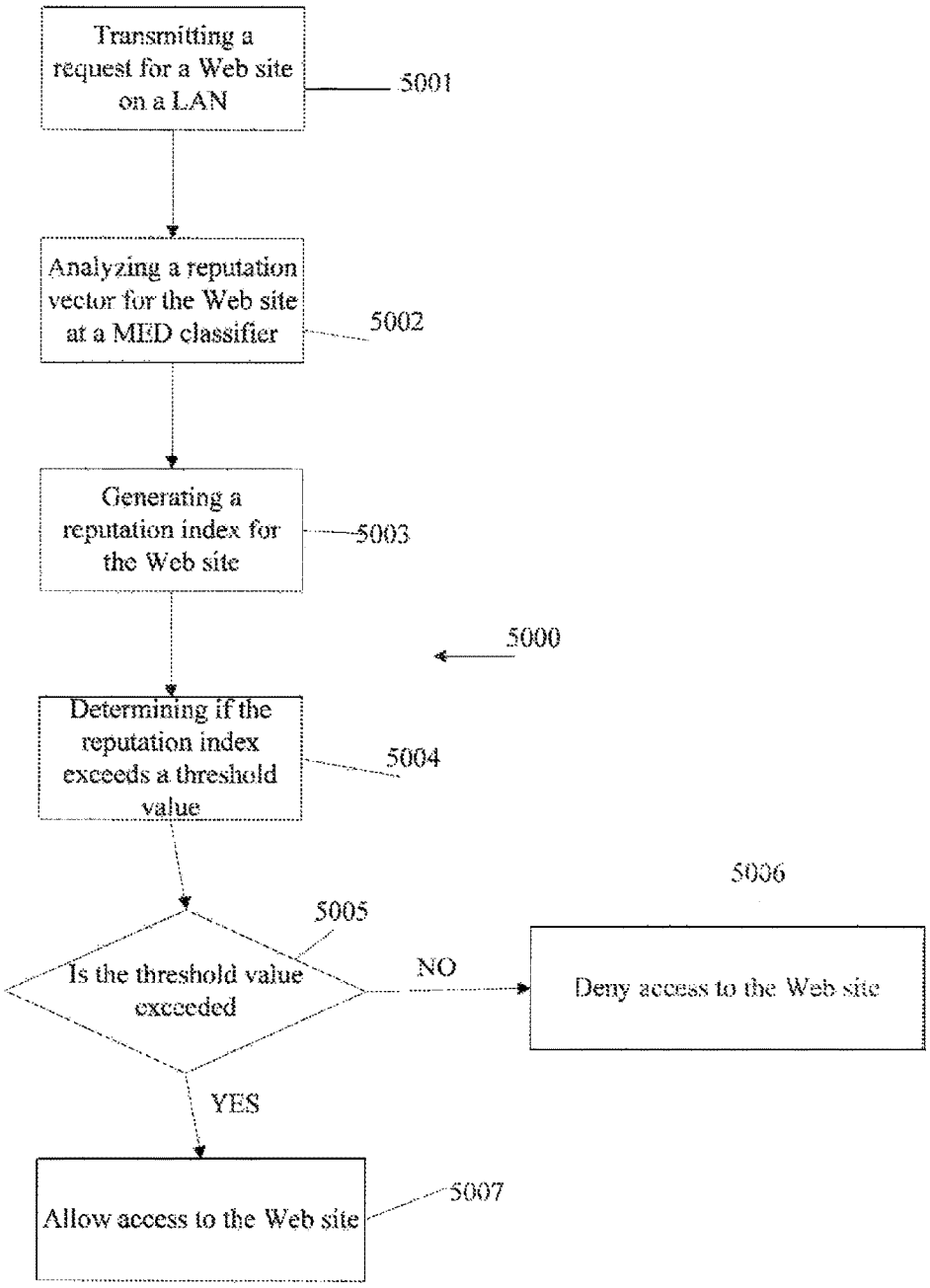

Transmitting a request for a Web site on a LAN
5001
Analyzing a reputation vector for the Web site at a MED classifier
5002
Generating a reputation index for the Web site
5003
5000
Determining if the reputation index exceeds a threshold value
5004
5006
5005
Is the threshold value exceeded
NO
Deny access to the Web site
YES
Allow access to the Web site
5007

FIG. 10

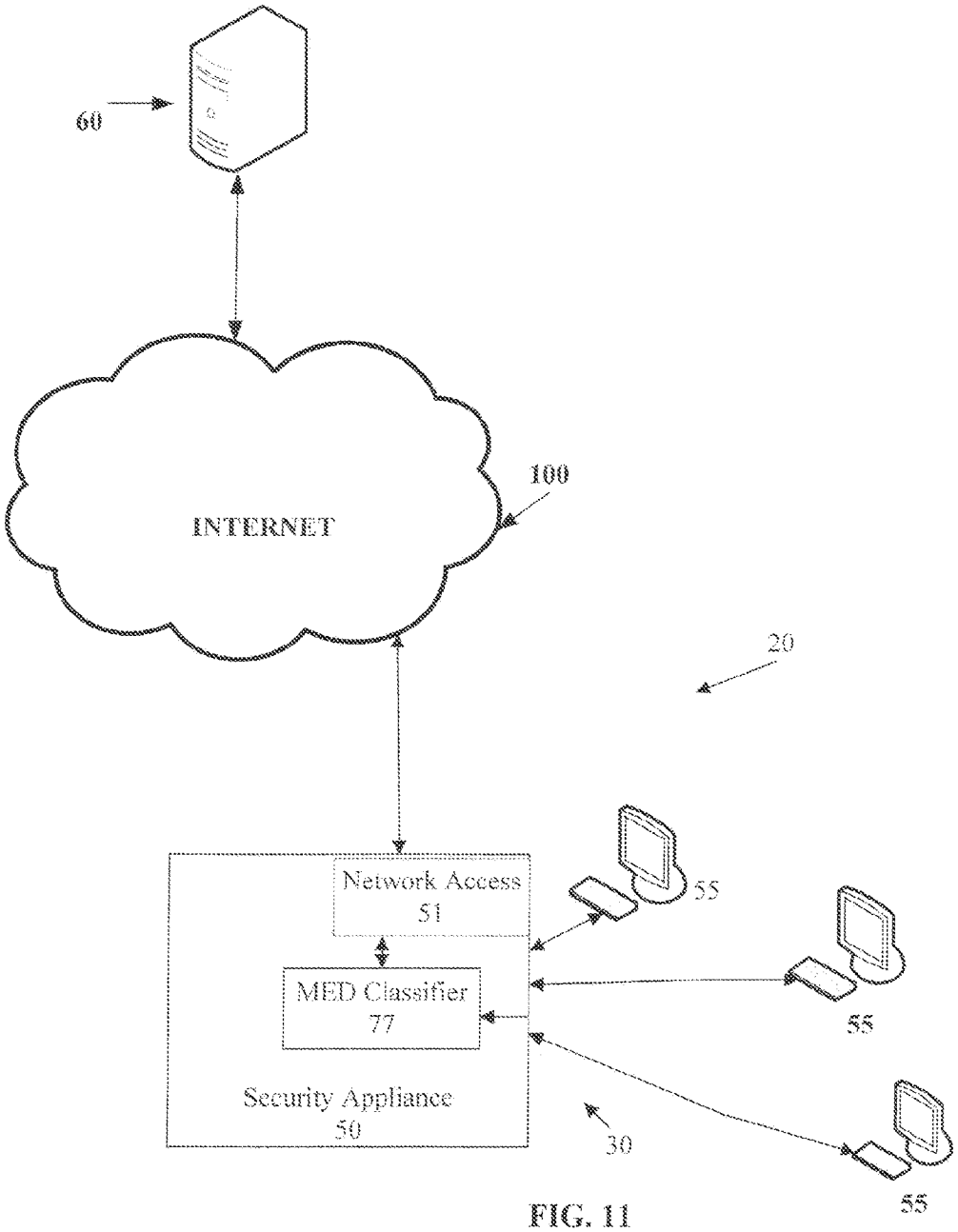

60
INTERNET
100
20
Network Access
51
MED Classifier
77
Security Appliance
50
30
55
55
55

FIG. 11

SYSTEM AND METHOD FOR DEVELOPING A RISK PROFILE FOR AN INTERNET SERVICE

CROSS REFERENCE TO RELATED APPLICATION

The present application claims priority to U.S. Provisional Patent Application No. 61/241,389, filed on Sep. 10, 2009 and U.S. Provisional Patent Application No. 61/171,264, filed on Apr. 21, 2009, both of which are hereby incorporated by reference in their entireties.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not Applicable

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is related to assessing risk profiles of Internet resources. More specifically, the present invention is related to a system and method for developing a risk profile for an Internet resource by generating a reputation index, based on attributes of the resource collectively referred to as the reputation vector of the resource.

2. Description of the Related Art

Management of internet access, particularly to Web sites, has been accomplished in the past using "Content Filtering", where Web sites are organized into categories and requests for Web content are matched against per-category policies and either allowed or blocked. This type of management focuses on the subject matter of a Web site, and provides visibility into, for example, how employees spend their time, and their company's network bandwidth usage, during the course of the day. These solutions also allow companies to enforce established internet usage policy (IUP) by blocking Web sites whose subject matter violates their IUP.

Security solutions, such as anti-virus products, examine file or Web page content to discover known patterns or signatures that represent security threats to users, computers, or corporate networks. These focus not on the subject matter of a site, but look for viruses and other 'malware' that are currently infecting the site. However, current solutions to management of Internet resources fail to measure the security risk associated with accessing an Internet resource in a more predictive way, before infections are isolated and signatures are identified and distributed.

A possible analogy to the reputation of an Internet resource is the credit score of an individual. A Web user would want to be informed of the reputation of a Web site before visiting it, just as a lender would want to know the reputation, the financial reputation at least, of a borrower of the lender's money.

A credit score is based on a variety of fairly tightly related factors, such as existing debt, available credit lines, on-time payments, existing credit balances, etc.

In the United States, a credit score is a number based on a statistical analysis of a person's credit files that represents the creditworthiness of that person, which is the likelihood that the person will pay their bills. A credit score is primarily based on credit information, typically from one of the three major credit agencies.

There are different methods of calculating credit scores. The best known one, FICO, is a credit score developed by the Fair Isaac Corporation. FICO is used by many mortgage lenders that use a risk-based system to determine the possibility that the borrower may default on financial obligations to the mortgage lender.

FICO® scores are provided to lenders by the three major credit reporting agencies: Equifax, Experian and TransUnion. When lenders order your credit report, they can also buy a FICO® score that is based on the information in the report. That FICO® score is calculated by a mathematical equation that evaluates many types of information from the borrower's credit report at that agency. In order for a FICO® score to be calculated on the borrower's credit report, the report must contain sufficient information—and sufficient recent information—on which to base a score. Generally, that means the borrower must have at least one account that has been open for six months or longer, and at least one account that has been reported to the credit reporting agency within the last six months.

FICO scores provide a reliable guide to future risk based solely on credit report data. FICO® scores have a 300-850® score range. The higher the score, the lower the risk. But no score says whether a specific individual will be a "good" or "bad" customer. And while many lenders use FICO® scores to help them make lending decisions, each lender has its own strategy to determine if a potential borrower is a good customer. Although FICO won't reveal exactly how it determines a credit score, it considers the following factors: payment history (35%); outstanding debt (30%); length of credit history (15%); types of credit (10%); and new credit (10%).

Returning to Internet resources, attackers have been using the Internet to attack the computers and other devices of users of the Internet. Attackers continue to take advantage of flaws in traditional security measures and bypass reputation-based systems to increase attack effectiveness.

In 2008, massive attacks were conducted that compromised hundreds of thousands of legitimate Web sites with good reputations worldwide with data-stealing malicious code. The attacks included sites from MSNBC, ZDNet, Wired, the United Nations, a large UK government site, and more. In the attacks, when a user's browser opened one of the thousands of compromised sites, a carefully crafted iframe HTML tag redirected users to a malicious site rife with exploits. As a result, malicious code, designed to steal confidential information, was launched on vulnerable machines. In addition to Web exploits, email spammers are also taking advantage of the reputation of popular email services like Yahoo! and Gmail to bypass anti-spam systems.

Also, spammers use sophisticated tools and bots to break the "CAPTCHA-" systems that were developed to keep email and other services safe from spammers and other malicious activity. MICROSOFT Live Mail, GOOGLE's popular Gmail service and Yahoo! mail services were all compromised by this breakthrough method. Subsequently, spammers have been able to sign up for the free email accounts on a mass basis and send out spam from email accounts with good reputations. With a free signup process, access to a wide portfolio of services and domains that are unlikely to be blacklisted given their reputation, spammers have been able to launch attacks on millions of users worldwide while maintaining anonymity.

Thus, prior art solutions have focused on security when accessing known infected sites in the Internet from a network such as a local area network or a wide area network.

Hegli et al., U.S. Pat. No. 7,483,982 for Filtering Techniques For Managing Access To Internet Sites Or Other Software Applications discloses a system and method for control-

ling an end user's access to the Internet by blocking certain categorized sites or limiting access based on bandwidth usage.

Hegli et al., U.S. Pat. No. 6,606,659 for a System And Method For Controlling Access To Internet Sites discloses a system and method for controlling an end user's access to the Internet by blocking certain categorized sites or limiting the number of times the end user can access an Internet site.

Yavatkar et al., U.S. Pat. No. 6,973,488 for Providing Policy Information To A Remote Device discloses a method for distributing high level policy information to remote network devices using a low-level configuration.

Turley et al., U.S. Patent Publication Number 2005/0204050 for a Method And System For Controlling Network Access discloses a system and method for controlling access to a specific site by using a gateway that assigns incoming traffic to specific sections of the site.

Shull et al., U.S. Pat. No. 7,493,403 for Domain Name Validation discloses accessing domain name registries to determine the ownership of a domain and monitoring the domain and registry.

Roy et al., U.S. Pat. No. 7,406,466 for a Reputation Based Search discloses using a search engine to present search results associated with measures of reputation to overcome the problem of META tags skewing the search results.

Hailpern et al., U.S. Pat. No. 7,383,299 for a System And Method For Providing Service For Searching Web Site Addresses discloses

Moore et al., U.S. Pat. No. 7,467,206, for a Reputation System For Web Services discloses a system and method for selecting a Web service from a search engine list which is ranked based on reputation information for each Web service.

Definitions for various terms are set forth below.

FTP or File Transfer Protocol is a protocol for moving files over the Internet from one computer to another.

HyperText Markup Language (HTML) is a method of mixing text and other content with layout and appearance commands in a text file, so that a browser can generate a displayed image from the file.

Hypertext Transfer Protocol (HTTP) is a set of conventions for controlling the transfer of information via the Internet from a Web server computer to a client computer, and also from a client computer to a Web server.

Internet is the worldwide, decentralized totality of server computers and data-transmission paths which can supply information to a connected and browser-equipped client computer, and can receive and forward information entered from the client computer.

JavaScript is an object-based programming language. JavaScript is an interpreted language, not a compiled language. JavaScript is generally designed for writing software routines that operate within a client computer on the Internet. Generally, the software routines are downloaded to the client computer at the beginning of the interactive session, if they are not already cached on the client computer. JavaScript is discussed in greater detail below.

Parser is a component of a compiler that analyzes a sequence of tokens to determine its grammatical structure with respect to a given formal grammar. Parsing transforms input text into a data structure, usually a tree, which is suitable for later processing and which captures the implied hierarchy of the input. XML Parsers ensure that an XML document follows the rules of XML markup syntax correctly.

URL or Uniform Resource Locator is a address on the World Wide Web.

Web-Browser is a complex software program, resident in a client computer, that is capable of loading and displaying text and images and exhibiting behaviors as encoded in HTML (HyperText Markup Language) from the Internet, and also from the client computer's memory. Major browsers include MICROSOFT INTERNET EXPLORER, NETSCAPE, APPLE SAFARI, MOZILLA FIREFOX, and OPERA.

Web-Server is a computer able to simultaneously manage many Internet information-exchange processes at the same time. Normally, server computers are more powerful than client computers, and are administratively and/or geographically centralized. An interactive-form information-collection process generally is controlled from a server computer, to which the sponsor of the process has access. Servers usually contain one or more processors (CPUs), memories, storage devices and network interface cards. Servers typically store the HTML documents and/or execute code that generates Web-pages that are sent to clients upon request. An interactive-form information-collection process generally is controlled from a server computer, to which the sponsor of the process has access.

World Wide Web Consortium (W3C) is an unofficial standards body which creates and oversees the development of web technologies and the application of those technologies.

XHTML (Extensible Hypertext Markup Language) is a language for describing the content of hypertext documents intended to be viewed or read in a browser.

XML (Extensible Markup Language) is a W3C standard for text document markup, and it is not a language but a set of rules for creating other markup languages.

The prior art fails to provide solutions to the problems with accessing the Internet.

BRIEF SUMMARY OF THE INVENTION

The present invention provides a predictive approach based on a statistical model built on a broad sampling of Internet resources with varying degrees of risk. The present invention focuses on the reputation of a Web site, or any Internet-based service or resource. The reputation incorporates many factors that are relevant to the overall safety of visiting a site. The reputation assesses the over-time track record of the site and the provider that operates the web site, the current characteristics of the pages and related files composing the site, and reputations of sites linked to the site and of referrers to the site. The overall assessment is expressed as a score, not unlike a FICO score, that predicts the potential risk of visiting the site which can be used to protect users from inadvertently visiting or utilizing higher-risk sites or services within the Internet.

There are many components of reputation available within the Internet. Much like other scoring mechanisms, such as credit scoring, the factors to be considered must be decided upon, and the weight that each factor will have in the overall "score" must be determined.

The present invention provides a system and method for defining a reputation of an Internet service such as a Web site.

A basic element of reputation is how long a domain has been registered to a particular company/entity. In addition, a domain which frequently changes hands is also interesting in a negative way relative to reputation.

Preferred steps of the invention are: evaluation of the important features to be included in the collection of reputation-relevant features referred to as the reputation vector; collection of the reputation vectors for a large sample of Internet resource; training of a classifier based on training sets of known high and low reputation services/sites; testing of a model against a wide variety of random samples; run-time evaluation of requests for the Internet resource using the

developed classifier and responding to reputation index information requests from clients which enforce network security policy.

The present invention preferably protects users against threats which are typically are not related to the subject matter of the service, or site. The present invention preferably protects users and networks from zero-day threats which have not been characterized or included in anti-virus signature files. The present invention preferably allows network managers to protect users and infrastructure without having to restrict access to particular categories of content. The present invention preferably allows higher security which is independent of cultural or moral biases related to many categories of content.

One aspect of the present invention is a method for controlling access to a Web site. The method includes transmitting a request for a Web site from a browser on a client-side device of a local area network. The Web site resides at a first server. The method also includes receiving the request for the Web site at a security appliance of the local area network prior to transmission of the request over the Internet. The method also includes analyzing a reputation vector for the Web site at the security appliance. The reputation vector includes a plurality of factors for the Web site comprising at least one or more of country of domain registration, country of service hosting, country of an internet protocol address block, age of a domain registration, popularity rank, internet protocol address, number of hosts, top-level domain, a plurality of run-time behaviors, JavaScript block count, picture count, immediate redirect and response latency. The method also includes generating a reputation index for the Web site based on the analysis of the plurality of factors. The method also includes determining if the reputation index for the Web site is above a threshold value established for the local area network. The method also includes transmitting a decision transmission to the browser of the client-side device.

If the reputation index for the Web site is above the threshold value, the method further includes transmitting the request for the Web-site over the Internet to a server for the Web site and receiving a Web page for the Web site at the local area network. In this situation, the decision transmission is the Web page for the Web site. If the reputation index for the Web site is at or below the threshold value, the decision transmission is a Web page from the local area network

The method can further include obtaining the plurality of factors for the Web site. Obtaining the plurality of factors for the Web site comprises accessing the Web site, analyzing a plurality of HTML documents for the Web site by crawling the Web site. Accessing the Web site comprises rendering a page for the Web site. Analyzing the plurality of HTML documents comprises determining the JavaScript block count and the picture count of each of the HTML documents.

Another aspect of the present invention is a system for controlling access to a Web site. The system includes a network, a Web site and a local area network. The network is the Internet. The Web site is hosted at a first server and accessible over the Internet. The local area network includes a plurality of client-side devices and a security appliance. Each of the client side devices has a browser. The security appliance controls access to the Internet by each of the plurality of client-side devices. The security appliance has a service engine for analyzing a reputation vector for the Web site and generating a reputation index for the Web site from the reputation vector. The reputation vector is based on a plurality of factors for the Web site. The plurality of factors comprises at least one or more of country of domain registration, country of service hosting, country of an internet protocol address block, age of a domain registration, popularity rank, internet protocol address, number of hosts, top-level domain, a plurality of run-time behaviors, JavaScript block count, picture count, immediate redirect and response latency. Access to the Web site by any of the plurality of client-side devices is determined on the reputation index exceeding a threshold value established for the local area network.

Another aspect of the present invention is a method for controlling access to an Internet resource utilizing a reputation generating site. The method includes transmitting a request for an Internet resource from a browser for a client-side device of a local area network. The Internet resource resides at a first server. The method also includes receiving the request for the Internet resource at reputation generating site prior to transmission of the request over the Internet to the first server. The method also includes analyzing a reputation vector for the Internet resource at the reputation generating site. The reputation vector includes a plurality of dimensions for the Internet resource comprising at least two of country of domain registration, country of service hosting, country of an internet protocol address block, age of a domain registration, popularity rank, internet protocol address, number of hosts, to-level domain, a plurality of run-time behaviors, JavaScript block count, picture count, immediate redirect and response latency. The method also includes generating a reputation index for the Internet resource based on the analysis of the plurality of factors. The method also includes determining if the reputation index for the Internet resource is above a threshold value established for the local area network. The method also includes transmitting a decision transmission to the browser of the client-side device.

Another aspect of the present invention is a method for controlling access to an Internet resource. The method includes transmitting a request for an Internet resource from an Internet-enabled client application from a client-side device of a local area network. The Internet resource resides at a first server. The method also includes receiving the request for the Internet resource at a security appliance of the local area network prior to transmission of the request over the Internet. The method also includes determining if a reputation index for the Internet resource is at or above a threshold value established for the local area network. The reputation index is generated from a reputation vector for the Internet resource. The reputation vector comprises a plurality of factors for the Internet resource comprising security history, legitimacy, behavior, associations and location. The reputation index preferably resides in a database file at the security appliance, which is immediately accessible by the security appliance for determining whether or not to allow access to the Internet resource. Alternatively, the reputation index is generated in real-time at a data collection site accessible by the security appliance over the Internet, and the reputation index is forwarded to the security appliance from the data collection site upon request. The method also includes transmitting a decision transmission to the Internet-enabled client application of the client-side device. The decision transmission allows or denies access to the Internet resource.

Yet another aspect of the present invention is a method for controlling access to an Internet resource. The method includes transmitting a request for an Internet resource from a Web browser for a client-side device of a local area network. The Internet resource resides at a first server. The method also includes receiving the request for the Internet resource at a security appliance of the local area network prior to transmission of the request over the Internet. The method also includes constructing a reputation vector for the Internet resource at the security appliance. The reputation vector comprises a

plurality of factors for the Internet resource comprising security history, legitimacy, behavior, associations and location. The method also includes analyzing the reputation vector to generate a reputation index for the Internet resource based on the analysis of the plurality of factors and the reputation classifier. The method also includes determining if the reputation index for the Internet resource is at or above a threshold value established for the local area network. The method also includes transmitting a decision transmission to the Web browser of the client-side device. The decision transmission allows or denies access to the Internet resource.

Yet another aspect of the present invention is a method for building a reputation database for Internet resources. The method includes collecting a plurality of factors for an Internet resource site to populate a reputation vector for the Internet resource to perform reputation analysis of the Internet resource. The method also includes receiving the plurality of factors for the Internet resource at a data collection site. The method also includes constructing a reputation vector for the Internet resource at the data collection site. The reputation vector comprises a plurality of factors for the Internet resource comprising security history, legitimacy, behavior, associations and location. The method also includes analyzing the reputation vector to generate a reputation index for the Internet resource based on the analysis of the plurality of factors and the reputation classifier. The method also includes storing the reputation index for the Internet resource at the data collection site. The method also includes transmitting the stored reputation index to a local area network upon request for managing access to the Internet resource.

The method further includes weighting each of the plurality of factors based on empirical knowledge of each of the plurality of factors. The method further includes obtaining the plurality of factors for the Internet resource using a crawler. Obtaining the plurality of factors for the Internet resource preferably comprises accessing the Internet service, analyzing a plurality of HTML documents for the Internet resource, and crawling a plurality of linked Internet resources of the plurality of HTML documents for Internet resource. Analyzing the plurality of HTML documents preferably comprises determining the JavaScript block count and the picture count of each of the HTML documents, browser hijacking, file downloads and a subject matter.

Yet another aspect of the present invention is a method for controlling access to an Internet resource. The method includes collecting a first plurality of Internet resource reputation vectors. The method also includes partitioning the first plurality of Internet resource reputation vectors into a plurality of training sets. The method also includes training a maximum entropy discrimination classifier with the plurality of training sets, the maximum entropy discrimination classifier trained for a specific local area network. The method also includes testing the trained maximum entropy discrimination classifier using a second plurality of Internet resource reputation vectors. Each of the second plurality of Internet resource reputation vectors is unknown to the trained maximum entropy discrimination classifier. The method also includes evaluating the tested maximum entropy discrimination classifier. The method also includes providing feedback to the evaluated maximum entropy discrimination classifier. The method also includes utilizing the reputation index at a local area network for managing access to an Internet resource.

Preferably, each of the first plurality of Internet resource reputation vectors comprises a plurality of dimensions for the Internet resource comprising security history, legitimacy, behavior, associations and location, and the method further comprises weighting each of the plurality of dimensions.

Yet another aspect of the present invention is a method for training a MED classifier for controlling access to an Internet resource. The method includes collecting a plurality of reputation vectors for Internet resources. The method also includes partitioning the plurality of reputation vectors into training sets. The method also includes training a MED classifier with the training sets. The method also includes testing the trained MED classifier against unknown Internet resources. The method also includes evaluating the trained MED classifier. The method also includes determining if the trained MED classifier has been adequately trained.

Having briefly described the present invention, the above and further objects, features and advantages thereof will be recognized by those skilled in the pertinent art from the following detailed description of the invention when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

FIG. **1** is a block diagram of a system for controlling access to a Web site.

FIG. **2** is a block diagram of a system for controlling access to a Web site.

FIG. **3** is a flow chart of a method for controlling access to a Web site.

FIG. **4** is a Web page for a requested Web site.

FIG. **5** is a page for a local area network informing a requestor of the denial of access to a Web site.

FIG. **6** is a block diagram of an Internet resource having a HTML document that is accessed by a crawler.

FIG. **7** is a flow chart of a method for generating a reputation index.

FIG. **8** is a flow chart of a method for controlling access to an Internet resource.

FIG. **9** is a flow chart of a method for utilizing a MED classifier for controlling access to an Internet resource.

FIG. **10** is flow chart of a method for controlling access to an Internet resource utilizing a MED classifier.

FIG. **11** is a block diagram of a system for utilizing a MED classifier for controlling access to an Internet resource.

DETAILED DESCRIPTION OF THE INVENTION

Reputation is a qualitative assessment of the safety of a website, expressed as a quantitative value that can be used in managing internet usage. Internet resources, such as Web sites, are safe, or of high reputation, if the Internet resource preferably has: a reputable ownership and registration; a consistent history; had consistent content during that history; associated with other high reputation sites; from a geographically safe region; the Internet service provider ("ISP") is well-known and reputable; not been known to be a source of malware infection; and worked cooperatively with the end-user and the end-user's Web browser application.

While security threats are transitory since they come up suddenly and are mitigated as quickly as possible, reputation is built up over a period of time and is a more enduring quality. Reputation can be lost, or become 'bad', over a period of time with repeated security events, bad associations, and bad behavior. For that reason, the occurrence of a single security breach (the site gets hacked and is a danger to visitors) does not dramatically lower the reputation of a site. Repeated occurrences over time, however, will destroy the reputation of the site.

Competitive reputation products include social considerations in their definitions, such that a highly reputable site, a site "held in high regard", preferably has these characteristics: established record of Web presence; not a source of network security risk; no introduction of malware; no popup ads; no persistent ad infection; is not pornographic or obscene; and has no illegal content.

The reputation of an Internet resource is preferably determined by security, legitimacy, behavior, geographic location, associations and additional factors. Legitimacy is determined by the top-level domain, the investment in the Internet resource (virtual hosting with non-affiliated sites, multiple hosting and SSL security), the traffic volume, the category age and the popularity rank. Legitimacy is also preferably determined by any or all of the following: the consistency between the registering and hosting city region or country; and city, region or country associated with the IP address. Behaviors include the use of popup ads, browser hijacking and the use of auto-redirecting. Associations include the number of sites linking into the site, the reputations of the linked in sites and the reputations of the linked-to sites. The geographic location includes the hosting country, the registration country, the region and the city. The geographic location also preferably includes the consistency between the registering and hosting country and the country associated with the IP address.

In a most preferred embodiment discussed below, machine learning technologies are utilized for controlling access to an Internet resource. A variation on support vector machine techniques called Maximum Entropy Discrimination ("MED") is a preferred machine learning technology. MED allows a computer to be trained to recognize the relative reputation of an Internet resource based on the features of the Internet resource. The set of features which characterize the reputation of the Internet resource is its reputation vector. Once trained, the computer uses the reputation vector for a requested Internet service to evaluate its reputation index, a score which can be used with empirically developed threshold values to block access where the reputation index is deemed to be too low to be safe.

A predictive security assessment for an Internet resource is provided based on known facts about the Internet resource, which is more secure than relying only on knowledge of previously experienced security attacks.

The system preferably provides classification of each Internet resource at run-time given a Uniform Resource Identifier (URI) and the reputation vector of the Internet resource. The system returns a score, or index, expressing the results on a relative scale for use by requesting clients, typically a security product which integrates the reputation assessment as a service.

The reputation vector preferably comprises a combination of some or all of the following: country of domain registration; country of service hosting; country of IP Address block; age of domain registration; time known to the assessor site; subject matter; classification age (time since last re-categorization); rank (popularity); IP Address; virtual hosting; number of hosts; top-level domain (.com, .biz, .ru, etc); security history; run-time behaviors; popup ads; downloadable executables; virus-infected executables; JavaScript block count; picture count; immediate redirect; and response latency. These features are collected and evaluated for all model training samples and at run-time on a per-user-request basis. Those skilled in the pertinent art will recognize that other factors may be utilized which are relevant to the security as determined by an assessor.

As shown in FIG. **1**, a system for controlling access to an Internet service is generally designated **20**. The system **20** preferably comprises a local area network **30**, the Internet **100** and an Internet service located at a remote server **60**. The Internet resource is preferably a Web site. A local area network **30** preferably comprises a security appliance **50** and a plurality of client-side devices **55**. The plurality of client-side devices preferably comprises desktop computers, laptop computers, personal digital assistants, smartphones and the like. Each of the client-side devices **55** preferably has a Web-browser for accessing the Internet from the client side device **55**. The security appliance **50** preferably comprises a network access **51** for permitting access to the Internet from the local area network **30**, and a service engine **52** for determining if a requested Internet resource has a reputation index that meets a threshold established for the local area network **30**.

A method **1000** for controlling access to a Web site is shown in FIG. **3**. At block **1001**, a request for a Web site is transmitted from a browser for a client-side device of a local area network which is received at a security appliance of the local area network prior to transmission of the request over the Internet. At block **1002**, a reputation index for the Web site is obtained at the security appliance. The reputation index is calculated from a reputation vector which preferably includes a plurality of factors for the Web site comprising country of domain registration, country of service hosting, country of an internet protocol address block, age of a domain registration, popularity rank, internet protocol address, number of hosts, to-level domain, a plurality of run-time behaviors, JavaScript block count, picture count, immediate redirect and response latency. At block **1004**, a determination is made if the reputation index for the Web site is above a threshold value established for the local area network. At decision **1005**, if the reputation index is not above the threshold, then at block **1006** access to the Web site is denied and a transmission of the denial is sent to the client-side device, preferably as a page **500** as shown in FIG. **5**. If at decision **1005** the reputation index for the Web site is above the threshold, then the access to the Web site by the client-side device is permitted by the security appliance, and preferably, as shown in FIG. **4**, a Web page **400** is provided to the client-side device.

An alternative embodiment of the system **20** is illustrated in FIG. **2**. The system **20** preferably comprises a local area network **30**, the Internet **100**, an Internet service located at a remote server **60** and a reputation generating site **70** preferably having a crawler **71** and a database **72**. The Internet service is preferably a Web site. A local area network **30** preferably comprises a security appliance **50** and a plurality of client-side devices **55**. Each of the client-side devices **55** preferably has a Web-browser for accessing the Internet from the client side device **55**. The security appliance **50** preferably comprises a network access **51** for permitting access to the Internet from the local area network **30**, and a service engine **52** for determining if a requested Internet service has a reputation index that meets a threshold establish for the local area network **30**. The reputation generating site **70** provides reputation indices to service engine **52** of the security appliance **50**. The reputation generating site **70** preferably utilizes the crawler **71** and other means to access Internet resources such as the Internet resource located at Web server **60**. The other means preferably includes publicly available data feeds, purchased databases, proprietary database, zone files from WHOIS database.

A flow chart for a method **2000** for generating a reputation index is shown in FIG. **7**. At block **2001**, a HTTP request is transmitted from a reputation generating site **70** for an Internet resource. From the HTTP request, a crawler **71** of the

reputation generating site accesses the Internet resource. In accessing the Internet resource, as shown in FIG. **6**, the crawler **71** preferably accesses at least one HTML document **91** of a plurality of HTML documents of the Internet resource **90**. At block **2003**, from the HTML documents and links within the HTML documents, the crawler **71** obtains information concerning the Internet resource **90**. The reputation vector for the Internet resource **90** is based on some of this information obtained by the crawler **71**. At block **2004**, the reputation vector for the Internet resource is analyzed at the reputation generating site **70**. At block **2005**, a reputation index for the Internet resource **90** is generated at the data collection site. At block **2006**, the reputation index for the Internet resource **90** is stored in a database **72** of the reputation generating site **70**. The reputation for the Internet resource is available to the security appliance as updates or individual requests. At block **2007**, the reputation index for the Internet resource **90** is transmitted to a LAN **30** for storage in a service engine **52** of a security appliance **50**.

A flow chart for a method **3000** for controlling access to an Internet resource is shown in FIG. **8**. At block **3001**, a request for an Internet resource is transmitted from an Internet-enabled client application for a client-side device **55** of a LAN **30**. At block **3002**, the request is received at a security appliance **50** of the LAN **30** prior to transmission of the request over the Internet **100**. At block **3003**, a reputation index for the Internet resource is accessed from a database of a service engine **52** of the security appliance **50**. The reputation index is based on a reputation vector which includes a plurality of factors for the Internet resource comprising at least two or more of country of domain registration, country of service hosting, country of an internet protocol address block, age of a domain registration, security history, popularity rank, internet protocol address, number of hosts, to-level domain, a plurality of run-time behaviors, JavaScript block count, picture count, immediate redirect and response latency. At block **3004**, a determination is made if the reputation index for the Internet resource is at or above a threshold value established for the LAN **30**. At decision **3005**, if the reputation index is below the threshold value, then at block **3006** access to the Internet resource is denied and a transmission of the denial is sent to the client-side device **55**. If at decision **3005** the reputation index for the Web site is at or above the threshold value, then the access to the Internet resource by the client-side device **55** is permitted by the security appliance **50**.

Table One provides a list of the attributes for the reputation vector and a description of each of the attributes.

TABLE ONE

Attribute	Description
Country	2-letter code, 3-letter code or full name of country based on IP block
Top-level Domain	.com, .biz, .org, .gov, etc.
Domain Age	Number of months in existence on zone lists, or no less than the classification age
Database Age	Months since Authority was entered into database
Classification Age	Months that the Authority has held its current classification
Hosts	Number of IP's associated with the Authority
Virtually Hosted	T/F if the other authorities share associated IP's
Popups	T/F if the page opens new browser windows on its own
Hijack	T/F does the default page alter the browser configuration
JavaScript	Count of <SCRIPT> blocks in default pages
Executables	T/F does the authorities download executables to client
Pictures	Count of pictures on default page
Latency	Number of milliseconds to return default page
Rank	Numerical ranking, used as T if <2,000,000, F otherwise in modeling
Infected	T/F were infected download files found by AV tools during site analysis
Security Trend	Number of malware infections in past 12 months
Total Security Count	Total number of malware infections known
Redirect	Authority redirects to another Authority
IP Address	Analysis of IP address for known threat sources, reserved IP ranges, and legacy IP address assignments
ISP	Internet service provider
City	
Region	

Table Two is an example of a "good" Internet resource.

TABLE TWO

Attribute	Value
Authority	USmoney.gov
Country	USA
Top Level Domain	Gov
Domain Age	18
Hosts	2
Virtual Hosts	0
Rank	1
Infected	0
Security Events	0
Recent Events	0
PublicCoIP	0
GovernmentIP	1
Hijack	0
JavaScript	0
Executables	0
Pictures	0
Latency	0
Redirect	0

Table Three is an example of a "bad" Internet resource.

TABLE THREE

Attribute	Value
Authority	www.c.reditcan.cn
Country	CN
Top Level Domain	CN
Domain Age	3
Hosts	1
Virtual Hosts	1
Rank	0
Infected	0
Security Events	1
Recent Events	1
PublicCoIP	0
GovernmentIP	0
Hijack	0
JavaScript	13
Executables	1
Pictures	14
Latency	826
Redirect	0

Depending on the threshold value established by the administrator of the LAN, the Internet resource of www.c.reditcan.cn with an reputation index value of 51, is not available for access by a user based on its reputation index, and the Internet resource of www.USmoney.GOV is available for access by a user based on its reputation index 95. Thus, even if the Internet resource of www.c.reditcan.cn is not a known source of malware or viruses, the present invention

would prevent an end user client from accessing the Internet resource since its reputation index is deemed unsafe.

Another embodiment uses a MED algorithm to build a statistical model on a Web page based on good and bad Internet samples. This embodiment uses a unique optimization algorithm for training, as well as two other optimization steps for calibrating the outputs to be probabilities, in a process that tolerates some input errors while still yielding reliable outputs. Training process feedback loops guide the implementer to improve the model data through splitting data into sets for holdout, training, and testing guided by two criteria: most violating examples, and least understood examples. The implementer using the criteria iteratively improves the quality of the training set which also reduces classifier errors and is exponentially faster than having the implementer manually verify or check the example assignments to categories in random or haphazard order. The examples are randomly reassigned before every training iteration to improve generalization. Sparse matrix math during the classification process improves processing speeds to enable a modest computer to classify millions of URLs per day. The implementation allows for a multiple of dimensions, each representing a fact about the Internet resource, to be included in the reputation model, while classification speed of any particular Internet resource is independent of the number of total dimensions in its reputation vector.

This embodiment is preferred since classifying a large percentage of existing Web sites into reputation risk assessments quickly and efficiently requires an automated process because the number of humans required is too large to be practical or economical. Further, defining automated classification rules by hand is very hard and requires writing many thousands of extremely specific as well as vague rules. All of these rules will interact with each other in an exponential number of ways, making human-based rule construction a daunting effort as well. The machine learning approach of this embodiment solves the problem by having humans define "training sets" or examples of each topic for the classifier, which then "trains" by optimizing the weights each factor should have in order to reduce classification error the most.

In addition to providing a good implementation of the learning algorithm, this embodiment efficiently utilizes the human efforts in identifying examples of good and bad reputations.

This embodiment preferably applies an effective learning formulation based on the principles and theory of MED with an efficient optimization algorithm based on the principles of Platt's sequential minimization optimization ("SMO"), in conjunction with an overall optimization of tunable parameters and calibrated confidence scores, to solve the learning problem given proper examples of Web sites of good and bad reputation.

The process then involves having humans examine a list of "most violating" examples, the examples which were marked as being good reputations but received extremely low confidence scores from the classifier (and vice-versa), as well as "least understood" examples, the examples which receive a confidence score close to the prior probability of the reputation.

By spending human time examining these two classes of examples, the classifier benefits from having egregiously misclassified examples being put into the proper reputation (good or bad) as well as providing the classifier with the largest amount of new information as quickly as possible. This combination improves the classifier's real-world effectiveness very quickly with minimal human effort. Thus, this embodiment efficiently combines human and automated work to solve the problem of automated reputation classification of Internet resources.

In one method, an evaluation of multiple factors (such as discussed above) is included in determining a reputation vector for an Internet resource. This process is done for multiple Internet resources. Next, reputation vectors for a large sample of Internet resources are collected at a data collection site. Next, a MED classifier is trained using the collection of reputation vectors based on training sets of known high reputation Internet resources and low reputation Internet resources. Next, a MED-based model for classification is tested against a wide variety of random samples of Internet resources. Next, a security appliance is deployed at a LAN. Next, a run-time evaluation of Internet resource requests is performed in using the developed MED classifier for responding to reputation index information requests from clients based on a LAN security policy. The MED-based model for classification is preferably utilized at run-time to calculate a reputation index. In this manner, this embodiment provides a predictive security assessment based on known facts about an Internet resource, which is more secure than relying only on knowledge of previously experienced security attacks. This embodiment provides a LAN real-time updates, real-time classification of non-cached URLs and a real-time feedback loop.

A flow chart of a method **4000** for utilizing a MED classifier for controlling access to an Internet resource is shown in FIG. **9**. At block **4001**, multiple reputation vectors for a large sample of Internet resources are collected preferably at a reputation generating site. The reputation vectors for the Internet resources are previously generated as discussed above. At block **4002**, the reputation vectors are partitioned into multiple training sets. The training sets comprise at least two training sets divided into high reputation Internet resources and low reputation Internet resources. At block **4003**, a MED classifier is trained using the training sets of high reputation Internet resources and low reputation Internet resources to create a trained MED classifier. At block **4004**, the trained MED classifier is tested against a wide variety of Internet resources which are not grouped into training sets and the reputation index is unknown to the trained MED classifier. At block **4005**, the tested MED classifier is evaluated to determine the accuracy of the tested MED classifier and to determine the most violating examples of either a wrongly categorized high reputation Internet resource or low reputation Internet resource, and the least understood Internet resources. At decision block **4006**, an evaluation of the testing is performed. If the testing was performed correctly, then at block **4007** the MED classifier is considered trained and ready for operations. If the testing was inadequate feedback is provided to the MED classifier concerning the wrongly categorized high reputation Internet resources or low reputation Internet resources, and the least understood Internet resources. The process is continued at block **4003** again until the MED classifier is properly trained.

In another embodiment, a reputation index is returned immediately from a stored set of reputation indexes calculated prior to the user's request. As shown in FIG. **10**, a method for controlling access to an Internet resource utilizing a MED classifier is generally designated **5000**. At block **5001**, a request for an Internet resource is transmitted from an Internet-enabled client application for a client-side device **55** of a LAN **30**. At block **5002**, a reputation vector for the Internet resource is analyzed preferably at a MED classifier or at a security appliance for the LAN. At block **5003**, a reputation index for the Internet resource is accessed/generated

from a database of a service engine **52** of the security appliance **50**. The reputation index is preferably based on a reputation vector which includes a plurality of factors for the Internet resource comprising at least two or more of country of domain registration, country of service hosting, country of an internet protocol address block, age of a domain registration, security history, popularity rank, internet protocol address, number of hosts, to-level domain, a plurality of run-time behaviors, JavaScript block count, picture count, immediate redirect and response latency. At block **5004**, a determination is made if the reputation index for the Internet resource is at or above a threshold value established for the LAN **30**. At decision **5005**, if the reputation index is below the threshold value, then at block **5006** access to the Internet resource is denied and a transmission of the denial is sent to the client-side device **55**. If at decision **5005** the reputation index for the Web site is at or above the threshold value, then the access to the Internet resource by the client-side device **55** is permitted by the security appliance **50**. In such an embodiment, a pre-calculated reputation index residing on the LAN or quickly available to the security appliance of the LAN provides for a much faster response (if not immediate response) as to the accessibility of the Internet resource.

FIG. **11** illustrates a system **20** for controlling access to an Internet resource utilizing a MED classifier site **77**. The system **20** preferably comprises a local area network **30**, the Internet **100**, a MED classifier site **77**, and an Internet service located at a remote server **60**. The Internet resource is preferably a Web site. A local area network **30** preferably comprises a security appliance **50** and a plurality of client-side devices **55**. Each of the client-side devices **55** preferably has a Web-browser for accessing the Internet from the client side device **55**. The security appliance **50** preferably comprises a network access **51** for permitting access to the Internet from the local area network **30**, based on data from the MED classifier site **77**, which determines if a requested Internet resource has a reputation index that meets a threshold establish for the local area network **30**.

Table Four provides an example of some dimensions and the sorted model weights of the MED classifier.

TABLE FOUR

Identification	Dimension	Sorted Model weights
1966272070	Domain age	3.785360
2307717	gov	1.969750
1906396306	paris	0.647784
1477426223	Hijack	−19.887100

From the foregoing it is believed that those skilled in the pertinent art will recognize the meritorious advancement of this invention and will readily understand that while the present invention has been described in association with a preferred embodiment thereof, and other embodiments illustrated in the accompanying drawings, numerous changes modification and substitutions of equivalents may be made therein without departing from the spirit and scope of this invention which is intended to be unlimited by the foregoing except as may appear in the following appended claim. Therefore, the embodiments of the invention in which an exclusive property or privilege is claimed are defined in the following appended claims.

We claim as our invention:

1. A method for controlling access to an Internet resource, the method comprising:

transmitting a request for an Internet resource from an Internet-enabled client application from a client-side device of a local area network, the Internet resource residing at a first server;

receiving the request for the Internet resource at a security appliance of the local area network prior to transmission of the request over the Internet;

determining if a reputation index for the Internet resource is at or above a threshold value established for the local area network, the reputation index generated from a reputation vector for the Internet resource, the reputation vector comprising a plurality of factors for the Internet resource comprising security history, legitimacy, behavior, associations and location, wherein the location factor of the plurality of factors for the reputation vector comprises two or more of: country of domain registration, country of service hosting and country of an internet protocol address block, wherein the legitimacy factor of the plurality of factors for the reputation vector comprises two or more of: age of a domain registration, popularity rank, internet protocol address, number of hosts, top-level domain, and wherein the behavior factor of the plurality of factors for the reputation vector comprises at least one of: plurality of run-time behaviors, script block count, picture count, immediate redirect and response latency; and

transmitting a decision transmission to the Internet-enabled client application of the client-side device, the decision transmission allowing or denying access to the Internet resource.

2. The method according to claim **1**, wherein the reputation index for the Internet resource is at or above the threshold value and wherein the decision transmission is access to the Internet resource, the method further comprises transmitting the request for the Internet resource over the Internet to the first server for the Internet resource and receiving the Internet resource at the local area network.

3. The method according to claim **1**, wherein the reputation index for the Internet resource is below the threshold value, and wherein the decision transmission is a Web page from the local area network containing information that accessing the Internet resource is against an Internet policy established on the local area network.

4. The method according to claim **1**, wherein the Internet resource is one of a Web site, an email server and a voice communications server.

5. The method according to claim **1**, wherein the reputation index is received from a data collection site accessible by the security appliance over the Internet.

6. The method according to claim **1**, wherein the reputation index is stored at a remote reputation generating site.

7. A system for controlling access to a Web site, the system comprising:

a network, the network comprising the Internet;

an Internet resource hosted at a first server and accessible over the Internet; and

a local area network, the local area network comprising:

a plurality of client-side devices, each of the client-side devices having a browser; and

a security appliance, the security appliance controlling access to the Internet by each of the plurality of client-side devices, the security appliance having a service engine for analyzing a reputation vector for the Web site and generating a reputation index for the Web site from the reputation vector, the reputation vector comprising a plurality of factors for the Internet resource comprising security history, legitimacy, behavior,

associations and location, wherein the location factor of the plurality of factors for the reputation vector comprises two or more of: country of domain registration, country of service hosting and country of an internet protocol address block, wherein the legitimacy factor of the plurality of factors for the reputation vector comprises two or more of: age of a domain registration, popularity rank, internet protocol address, number of hosts, top-level domain, and wherein the behavior factor of the plurality of factors for the reputation vector comprises at least one of: plurality of run-time behaviors, script block count, picture count, immediate redirect and response latency;

wherein access to the Internet resource by any of the plurality of client-side devices is determined based on the reputation index exceeding a threshold value established for the local area network.

8. The system according to claim **7**, wherein the client-side device is a personal computer or a PDA.

9. The system according to claim **7**, further comprising a data collection site accessible over the Internet, the security site having a crawler for collecting a plurality of factors from a plurality of Internet resources.

10. The system according to claim **9**, wherein the data collection site has a database containing reputation indices for a plurality of Internet resources.

11. A method for controlling access to an Internet resource, the method comprising:

transmitting a request for an Internet resource from a Web browser for a client-side device of a local area network, the Internet resource residing at a first server;

receiving the request for the Internet resource at a security appliance of the local area network prior to transmission of the request over the Internet;

accessing a reputation index for the Internet resource, the reputation index based on a reputation vector for the Internet resource comprising security history, legitimacy, behavior, associations and location, wherein the location factor of the plurality of factors for the reputation vector comprises two or more of: country of domain registration, country of service hosting and country of an internet protocol address block, wherein the legitimacy factor of the plurality of factors for the reputation vector comprises two or more of: age of a domain registration, popularity rank, internet protocol address, number of hosts, top-level domain, and wherein the behavior factor of the plurality of factors for the reputation vector comprises at least one of: plurality of run-time behaviours, script block count, picture count, immediate redirect and response latency;

determining if the reputation index for the Internet resource is at or above a threshold value established for the local area network; and

transmitting a decision transmission to the Web browser of the client-side device, the decision transmission allowing or denying access to the Internet resource.

12. The method according to claim **11**, further comprising transmitting the reputation index over the Internet to the local area network.

13. A method for building a reputation database for Internet resources, the method comprising:

collecting a plurality of factors for an Internet resource site to populate a reputation vector for the Internet resource to perform reputation analysis of the Internet resource;

receiving the plurality of factors for the Internet resource at a data collection site;

constructing a reputation vector for the Internet resource at the data collection site, the reputation vector comprising a plurality of factors for the Internet resource comprising security history, legitimacy, behavior, associations and location, wherein the location factor of the plurality of factors for the reputation vector comprises two or more of: country of domain registration, country of service hosting and country of an internet protocol address block, wherein the legitimacy factor of the plurality of factors for the reputation vector comprises two or more of age of: a domain registration, popularity rank, internet protocol address, number of hosts, top-level domain, and wherein the behavior factor of the plurality of factors for the reputation vector comprises at least one of: plurality of run-time behaviours, script block count, picture count, immediate redirect and response latency;

analyzing the reputation vector to generate a reputation index for the Internet resource based on the analysis of the plurality of factors and the reputation classifier;

storing the reputation index for the Internet resource at the data collection site; and

transmitting the stored reputation index to a local area network upon request for managing access to the Internet resource.

14. The method according to claim **13**, further comprising weighting each of the plurality of factors based on empirical knowledge of each of the plurality of factors.

15. The method according to claim **13**, further comprising obtaining the plurality of factors for the Internet resource using a crawler.

16. The method according to claim **15**, wherein obtaining the plurality of factors for the Internet resource comprises accessing the Internet service, analyzing a plurality of HTML documents for the Internet resource, and crawling a plurality of linked Internet resources of the plurality of HTML documents for Internet resource.

17. The method according to claim **15**, wherein analyzing the plurality of HTML documents comprises determining the script block count and the picture count of each of the HTML documents, browser hijacking, file downloads and a subject matter.

18. The method according to claim **13**, wherein the stored reputation index is transmitted over the Internet to a local area network.

19. A method for controlling access to an Internet resource, the method comprising:

collecting a first plurality of Internet resource reputation vectors;

partitioning the first plurality of Internet resource reputation vectors into a plurality of training sets;

training a maximum entropy discrimination classifier with the plurality of training sets, the maximum entropy discrimination classifier trained for a specific local area network;

testing the trained maximum entropy discrimination classifier using a second plurality of Internet resource reputation vectors, the second plurality of Internet resource reputation vectors unknown to the trained maximum entropy discrimination classifier;

evaluating the tested maximum entropy discrimination classifier;

providing feedback to the evaluated maximum entropy discrimination classifier; and

utilizing the maximum entropy discrimination classifier at a local area network for managing access to an Internet resource.

20. The method according to claim **19**, wherein each of the first plurality of Internet resource reputation vectors comprises a plurality of dimensions for the Internet resource comprising security history, legitimacy, behavior, associations and location, and the method further comprises weighting each of the plurality of dimensions.

21. The method according to claim **19**, wherein at least one of collecting, training, and testing is performed external to the local area network.

22. A method for training a maximum entropy discrimination (MED) classifier for controlling access to an Internet resource, the method comprising:

collecting a first plurality of reputation vectors for Internet resources;

partitioning the first plurality of reputation vectors for the Internet resources into one or more training sets;

training a MED classifier with the plurality of training sets, the maximum entropy discrimination classifier being trained for a specific local area network;

testing the trained MED classifier using a second plurality of Internet resource reputation vectors, the second plurality of Internet resource reputation vectors unknown to the trained MED classifier;

evaluating the tested MED classifier; and

determining if the tested MED classifier has been adequately trained.

23. The method according to claim **22**, wherein the training set comprises good reputation Internet resources and bad reputation Internet resources.

24. The method according to claim **22**, further comprising locating the trained MED classifier at a security appliance of the local area network.

25. The method according to claim **22**, wherein at least one of collecting, training, and testing is performed external to the local area network.

26. The method according to claim **22**, wherein each of the first plurality of Internet resource reputation vectors comprises a plurality of dimensions for the Internet resource comprising security history, legitimacy, behavior, associations and location.

* * * * *

Exhibit 4

U 9164950

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

October 07, 2021

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *9,413,721*
ISSUE DATE: *August 09, 2016*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

R GLOVER
Certifying Officer


US009413721B2

(12) **United States Patent**
Morris et al.

(10) **Patent No.: US 9,413,721 B2**
(45) **Date of Patent: Aug. 9, 2016**

(54) **METHODS AND APPARATUS FOR DEALING WITH MALWARE**

(75) Inventors: **Melvyn Morris**, Belper (GB); **Joseph Jaroch**, Elk Grove Village, IL (US)

(73) Assignee: **WEBROOT INC.**, Broomfield, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **13/372,375**

(22) Filed: **Feb. 13, 2012**

(65) **Prior Publication Data**

US 2012/0260340 A1 Oct. 11, 2012

Related U.S. Application Data

(60) Provisional application No. 61/443,095, filed on Feb. 15, 2011.

(51) **Int. Cl.**
H04L 29/06 (2006.01)
G06F 21/56 (2013.01)

(52) **U.S. Cl.**
CPC ***H04L 63/0263*** (2013.01); ***G06F 21/56*** (2013.01); ***G06F 21/566*** (2013.01); ***H04L 63/14*** (2013.01); ***H04L 63/145*** (2013.01)

(58) **Field of Classification Search**
USPC 726/23
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,175,869 B1 1/2001 Ahuja et al.
6,219,786 B1 4/2001 Cunningham et al.
6,842,878 B1 1/2005 Johnson et al.
7,051,365 B1 5/2006 Bellovin
7,188,173 B2 * 3/2007 Anderson et al. 709/225
7,231,440 B1 * 6/2007 Kouznetsov et al. 709/224
7,343,626 B1 3/2008 Gallagher
7,555,476 B2 6/2009 Holbrook
7,707,634 B2 4/2010 Sandu et al.
7,761,912 B2 7/2010 Yee et al.
7,814,056 B2 * 10/2010 McGrattan et al. 707/640
(Continued)

FOREIGN PATENT DOCUMENTS

EP 1315066 A1 5/2003
JP H08-44677 2/1996
JP 2009-500706 1/2009

OTHER PUBLICATIONS

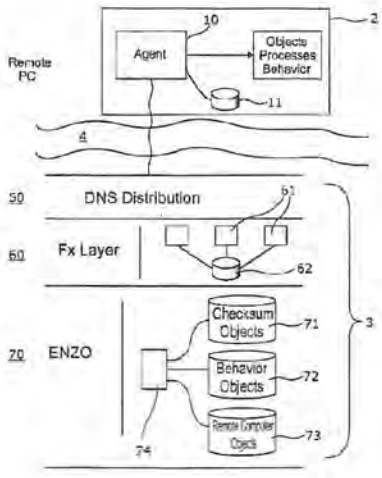
Morris, et al., "Specification for related U.S. Appl. No. 13/372,433," filed Feb. 13, 2012, p. 58 to be published in: US.
(Continued)

Primary Examiner — Dede Zecher
Assistant Examiner — Jason C Chiang
(74) *Attorney, Agent, or Firm* — Merchant & Gould P.C.

(57) **ABSTRACT**

Methods for classifying computer objects as malware and the associated apparatus are disclosed. An exemplary method includes, at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored or processed wherein the base computer comprises plural threat servers arranged to receive the data from the plural remote computers and apply rules or heuristics against the data in real time to determine whether or not the object is malware and to communicate the determination to the remote computers. The base computer includes at least one central server in communication with the threat servers and arranged to receive the data about objects from the threat servers to maintain a master database of data received about objects from all threat servers.

15 Claims, 15 Drawing Sheets



Copy provided by USPTO from the PIRS Image Database on 10-05-2021

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,921,063	B1 *	4/2011	Quinlan	706/12
7,921,459	B2	4/2011	Houston et al.	
8,042,184	B1	10/2011	Batenin	
8,116,732	B2 *	2/2012	Issa et al.	455/410
8,146,135	B2	3/2012	Waissbein et al.	
8,195,938	B2 *	6/2012	Fanton et al.	713/165
8,225,218	B2	7/2012	Frayman et al.	
8,234,709	B2 *	7/2012	Viljoen et al.	726/24
8,276,202	B1 *	9/2012	Dubrovsky et al.	726/22
8,296,178	B2 *	10/2012	Hudis et al.	370/401
8,301,904	B1 *	10/2012	Gryaznov	713/188
8,307,435	B1	11/2012	Mann et al.	
8,321,540	B2 *	11/2012	Webb-Johnson	709/221
8,321,944	B1 *	11/2012	Mayer et al.	726/25
8,332,946	B1 *	12/2012	Boisjolie et al.	726/24
8,336,100	B1 *	12/2012	Glick et al.	726/23
8,347,375	B2 *	1/2013	Graham et al.	726/13
8,347,386	B2 *	1/2013	Mahaffey et al.	726/23
8,352,484	B1 *	1/2013	Schneider	707/758
8,407,790	B2	3/2013	Mony	
8,443,447	B1 *	5/2013	Chen et al.	726/24
8,448,243	B1 *	5/2013	Sankruthi	726/22
8,453,244	B2 *	5/2013	Dai et al.	726/24
8,499,283	B2	7/2013	Mony	
8,543,694	B2	9/2013	Petersen et al.	
2003/0093429	A1 *	5/2003	Nishikawa et al.	707/10
2003/0188194	A1	10/2003	Currie et al.	
2004/0103315	A1	5/2004	Cooper et al.	
2005/0131959	A1	6/2005	Thorman et al.	
2005/0172338	A1	8/2005	Sandu et al.	
2005/0187740	A1	8/2005	Marinescu	
2006/0069912	A1	3/2006	Zheng et al.	
2006/0080637	A1	4/2006	Treit et al.	
2006/0090073	A1	4/2006	Steinberg et al.	
2006/0101334	A1	5/2006	Liao et al.	
2006/0117055	A1	6/2006	Doyle	
2006/0161856	A1	7/2006	Heir	
2006/0272020	A1	11/2006	Gardner	
2007/0016953	A1	1/2007	Morris et al.	
2007/0028304	A1 *	2/2007	Brennan	726/24
2007/0174429	A1	7/2007	Mazzaferri et al.	
2007/0220043	A1	9/2007	Oliver et al.	
2008/0109473	A1	5/2008	Dixon et al.	
2008/0115190	A1	5/2008	Aaron	
2008/0148381	A1	6/2008	Aaron	
2008/0229422	A1	9/2008	Hudis et al.	
2009/0070304	A1 *	3/2009	Chen et al.	707/3
2009/0089040	A1	4/2009	Monastyrsky et al.	
2009/0172774	A1	7/2009	Koti et al.	
2009/0292791	A1	11/2009	Livshits et al.	
2009/0300764	A1	12/2009	Freeman	
2010/0043072	A1	2/2010	Rothwell	
2010/0064368	A1	3/2010	Stolfo et al.	
2010/0306851	A1	12/2010	Zhou	
2010/0313183	A1	12/2010	Ellen et al.	
2010/0332593	A1 *	12/2010	Barash et al.	709/203
2011/0047594	A1	2/2011	Mahaffey et al.	
2011/0083183	A1	4/2011	Freeman	
2011/0138361	A1	6/2011	McEntee et al.	
2011/0191299	A1 *	8/2011	Huynh Huu et al.	707/646
2011/0191341	A1 *	8/2011	Meyer et al.	707/736
2011/0197177	A1	8/2011	Mony	
2011/0197272	A1	8/2011	Mony	
2012/0017200	A1	1/2012	Ghosh et al.	
2012/0260304	A1	10/2012	Morris et al.	
2012/0266208	A1	10/2012	Morris et al.	

OTHER PUBLICATIONS

Morris, et al., "Specification for related U.S. Appl. No. 13/372,439," filed Feb. 13, 2012, p. 54 to be published in: US.

Sogno-Pabis, Elzbieta, "Partial International Search Report re application PCT/EP2012/052497", May 4, 2012, p. 7, Published in: NL.

Clarke, "Fuzzing for Software Vulnerability Discovery—Technical Report RHUL-MA-2009-04," Department of Mathematics, Royal Holloway, University of London, Feb. 17, 2009, 178 pages. Retrieved from: www.ma.rhul.ac.uk/static/techrep/2009/RHUL-MA-2009-04.pdf.

Su et al., "The Essence of Command Injection Attacks in Web Applications," Conference record of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '06), 2006, pp. 372-382.

Official Action for U.S. Appl. No. 12/703,074 mailed Sep. 12, 2012, 16 pages.

Notice of Allowance for U.S. Appl. No. 12/703,074 mailed Nov. 21, 2012, 5 pages.

Notice of Allowance for U.S. Appl. No. 13/010,639 mailed Mar. 20, 2013, 15 pages.

Official Action for U.S. Appl. No. 13/372,433 mailed Apr. 24, 2013, 6 pages.

Official Action for U.S. Appl. No. 13/372,439 mailed Apr. 22, 2013, 6 pages.

Extended European Search Report for EP Application No. 14161844.7, mailed Jul. 25, 2014, 5 pages.

Extended European Search Report for EP Application No. 14161847.0, mailed Jul. 25, 2014, 5 pages.

International Search Report and Written Opinion for International (PCT) Patent Application No. PCT/EP2012/052497 mailed Jul. 13, 2012, 22 pages.

International Preliminary Report on Patentability for International (PCT) Patent Application No. PCT/EP2012/052497 mailed Aug. 29, 2013, 16 pages.

"Network Security Task Manager: Manual—Installation—Configuration," A. & M. Neuber Software GmbH, 2008, 65 pages.

Chacksfield, "Facebook apps ask for your permission before data mining," TechRadar.com, Jul. 1, 2010 [retrieved on Jun. 14, 2013], 5 pages. Retrieved from: www.techradar.com/us/news/broadband/web/internet/facebook-apps-ask-for-your-permission-before-data-mining-700540.

Cogswell et al., "RootkitRevealer v1.71," Nov. 1, 2006, [Retrieved on Jun. 29, 2012], 5 pages. Retrieved from: http://technet.microsoft.com/en-us/sysinternals/bb897445.aspx.

Srinivasan, "Protecting Anti-Virus Software Under Viral Attacks," Master of Science thesis, Arizona State University, Aug. 2007, 82 pages.

Official Action for U.S. Appl. No. 13/372,433 mailed Aug. 9, 2013, 18 pages.

Official Action for U.S. Appl. No. 13/372,439 mailed Jul. 3, 2013, 16 pages.

Dietrich (2003) Code Project "Event Logging, Part I: XEventLog—Basic NT Event Logging," 10 pages. Available at: http://www.codeproject.com/Articles/3995/Event-Logging-Part-l-XEventLog-Basic-NT-Event-Logg.

Wikipedia as crawled by Wayback Machine on Mar. 1, 2010 "Event Viewer" 6 pages.

Chinese Patent Application No. 201280018306.9, First Office Action mailed Jun. 2, 2015, 26 pages. (No English translation available).

U.S. Appl. No. 12/703,074, Amendment and Response dated Nov. 6, 2012, 8 pages.

U.S. Appl. No. 12/703,074, Amendment after Allowance dated Dec. 7, 2012, 7 pages.

U.S. Appl. No. 13/372,433, Amendment and Response dated Jan. 6, 2014, 13 pages.

U.S. Appl. No. 13/372,433, Final Rejection mailed May 9, 2014, 22 pages.

U.S. Appl. No. 13/372,433, Amendment and Response dated Oct. 9, 2014, 14 pages.

U.S. Appl. No. 13/372,433, Non-Final Rejection mailed Sep. 10, 2015, 16 pages.

U.S. Appl. No. 13/372,439, Amendment and Response dated Nov. 4, 2013, 12 pages.

U.S. Appl. No. 13/372,439, Final Rejection mailed Feb. 24, 2014, 15 pages.

U.S. Appl. No. 13/372,433, Amendment and Response filed Feb. 10, 2016, 18 pages.

Japanese Patent Application No. 2013-553909, Office Action mailed Jan. 26, 2016, 6 pages (English Translation).

Chinese Patent Application No. 201280018306.9, Second Office Action mailed Jan. 11, 2016, 11 pages (English Translation).

Australian Patent Application No. 2012217181, First Examination Report mailed Apr. 11, 2016, 4 pages.

Japanese Patent Application No. 2013-553909, Notice of Allowance mailed May 24, 2016, 3 pages (No English translation available).

U.S. Appl. No. 13/372,433, Final Rejection mailed Jun. 1, 2016, 17 pages.

* cited by examiner

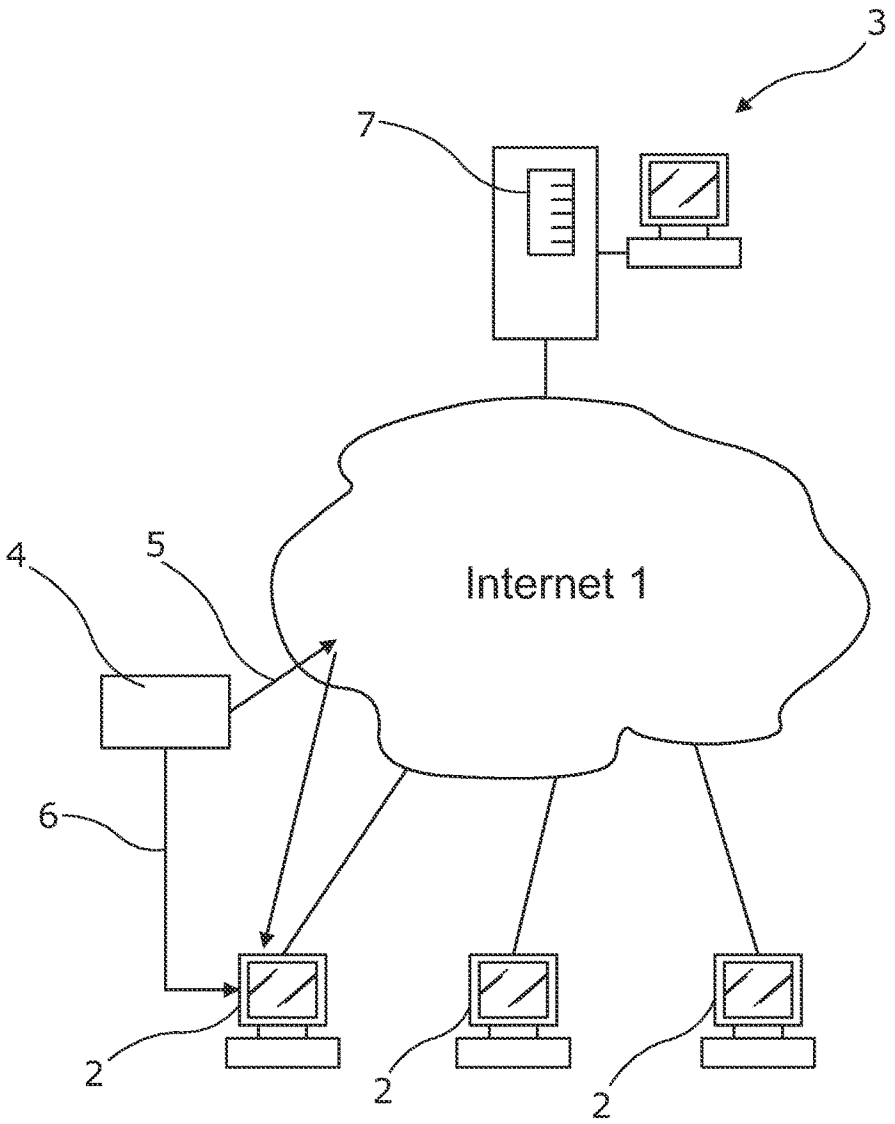


Fig. 1
(Prior Art)

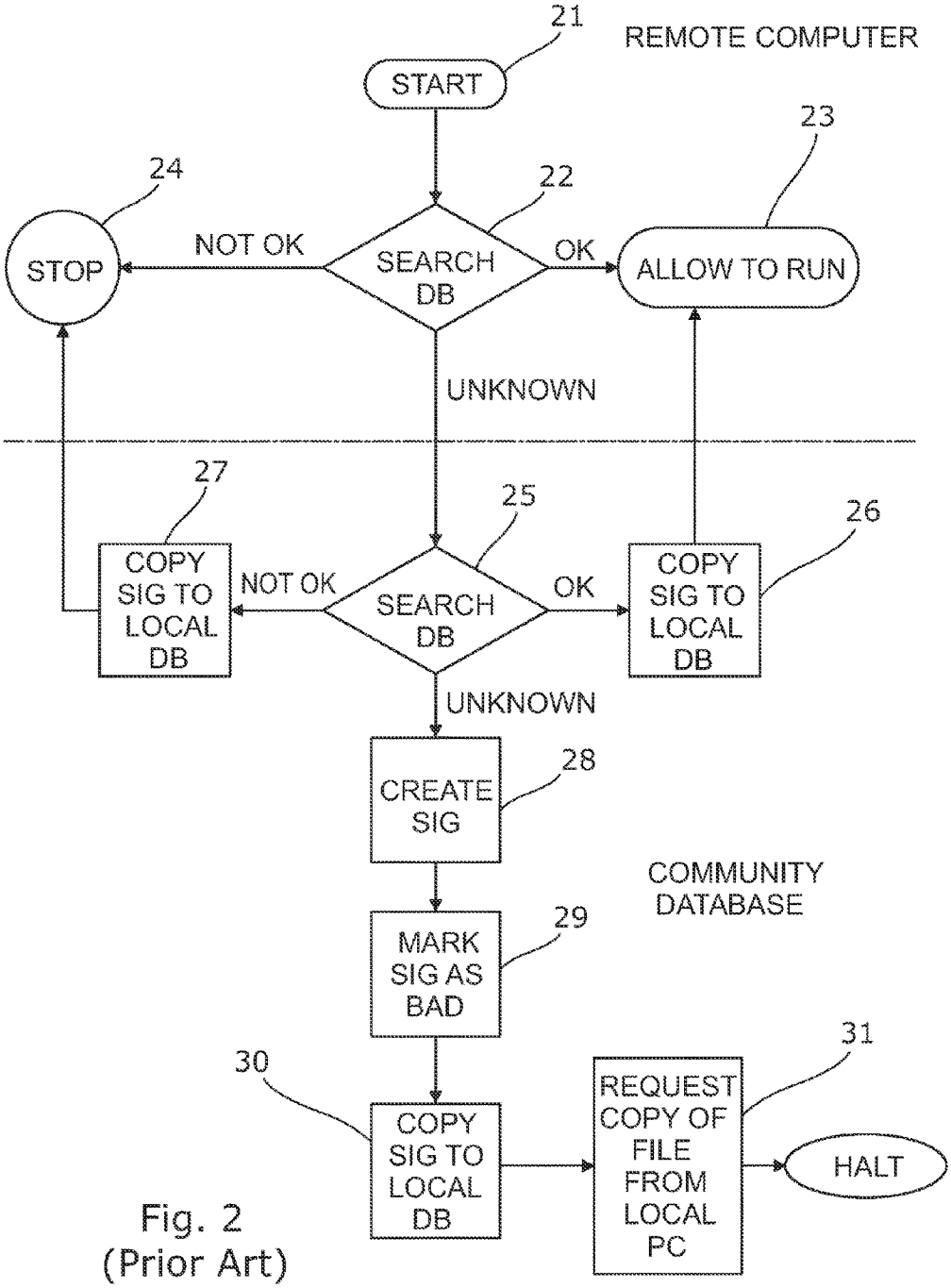


Fig. 2
(Prior Art)

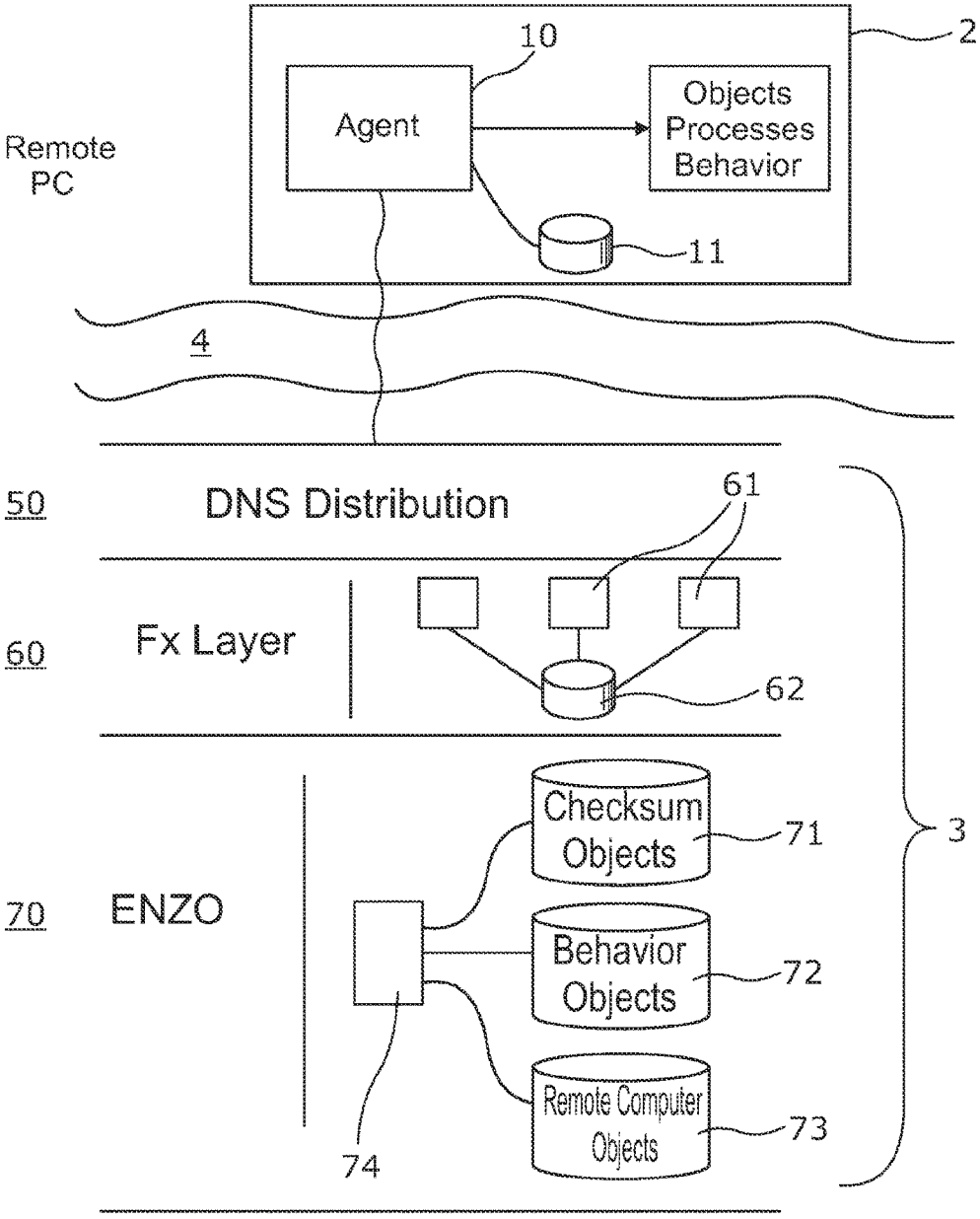

Remote PC
10
Agent
Objects Processes Behavior
2
11
4
50
DNS Distribution
61
60
Fx Layer
62
70
ENZO
Checksum Objects
71
Behavior Objects
72
Remote Computer Objects
73
74
3

Fig. 3

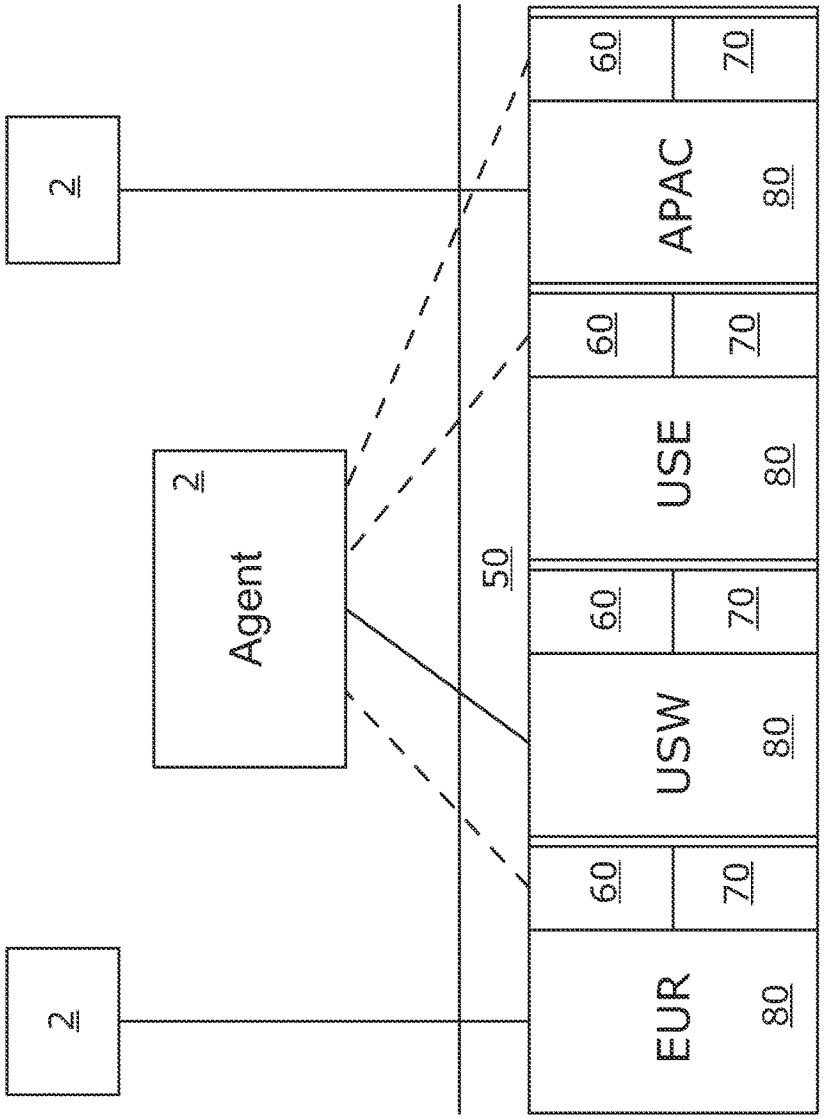


Fig. 4

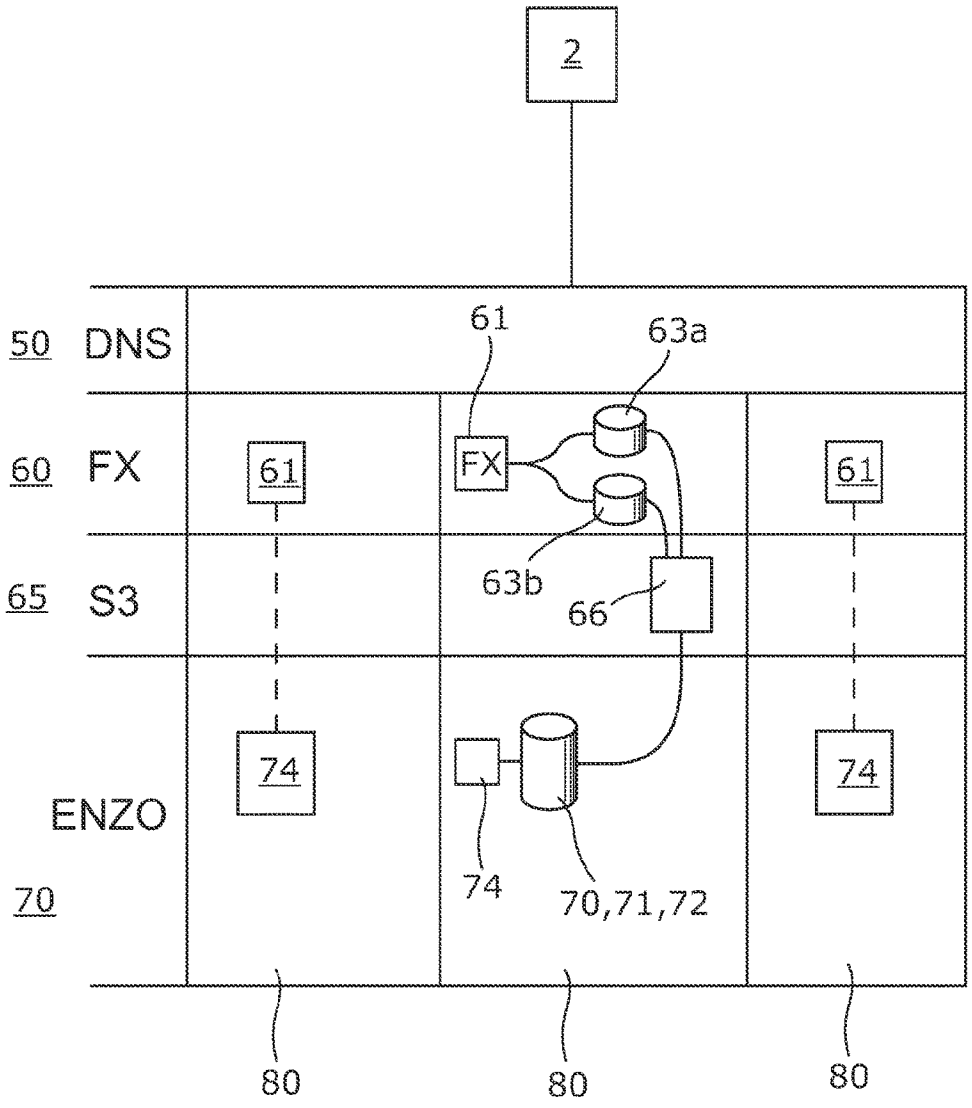

2
50 DNS
60 FX
65 S3
ENZO
70
61
63a
FX
61
61
63b
66
74
74
74
70,71,72
80
80
80

Fig. 5

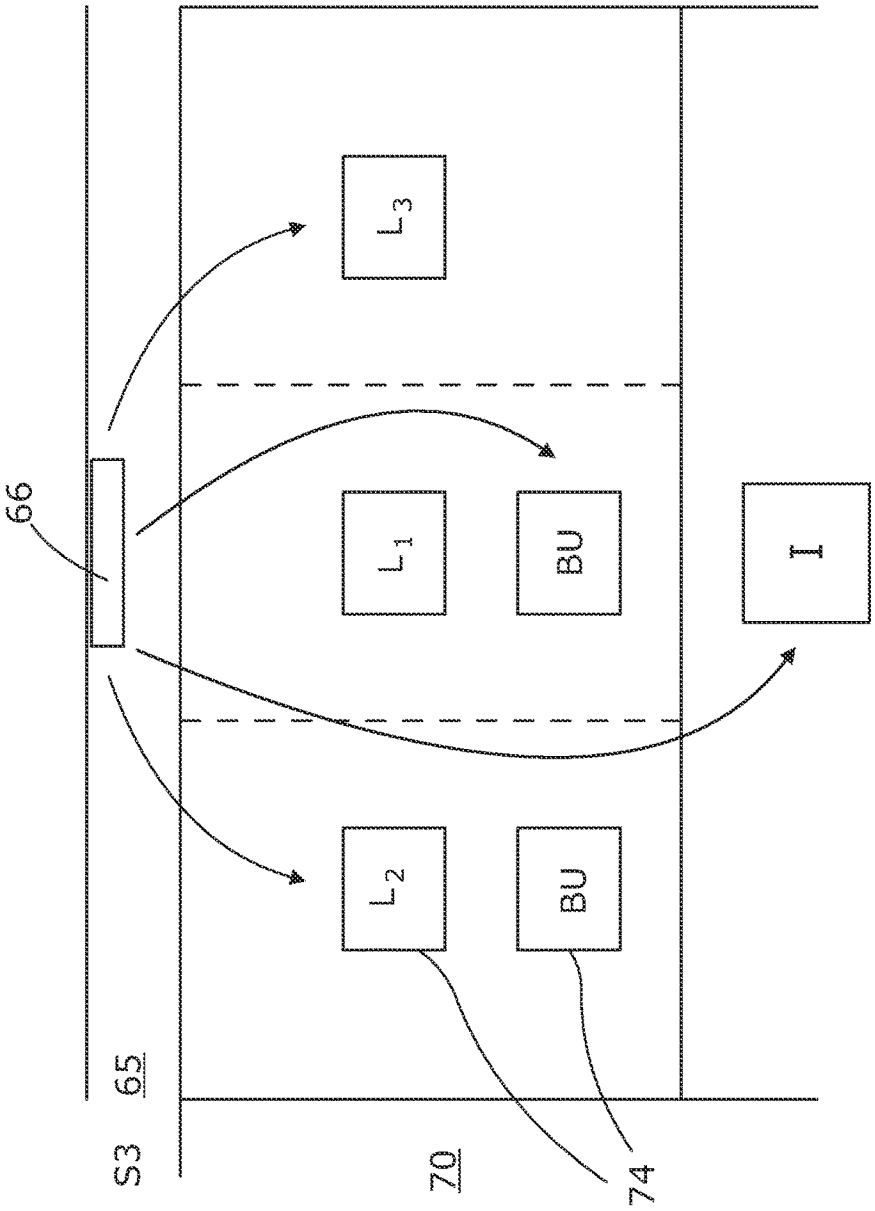


Fig. 6

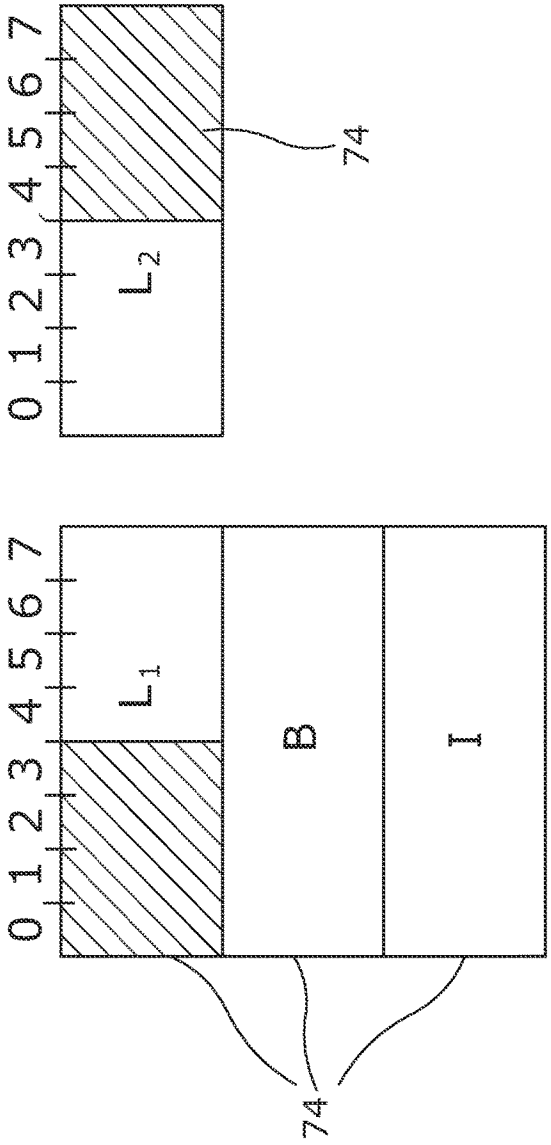


Fig. 7

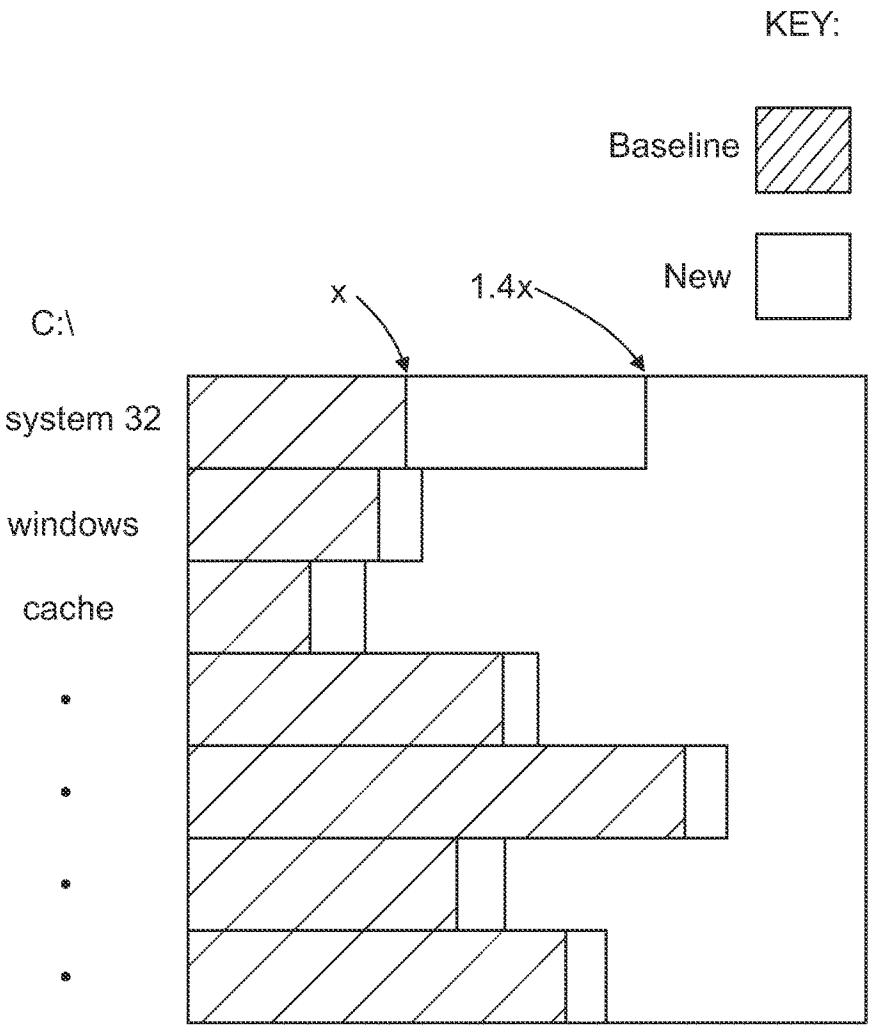


Fig. 8

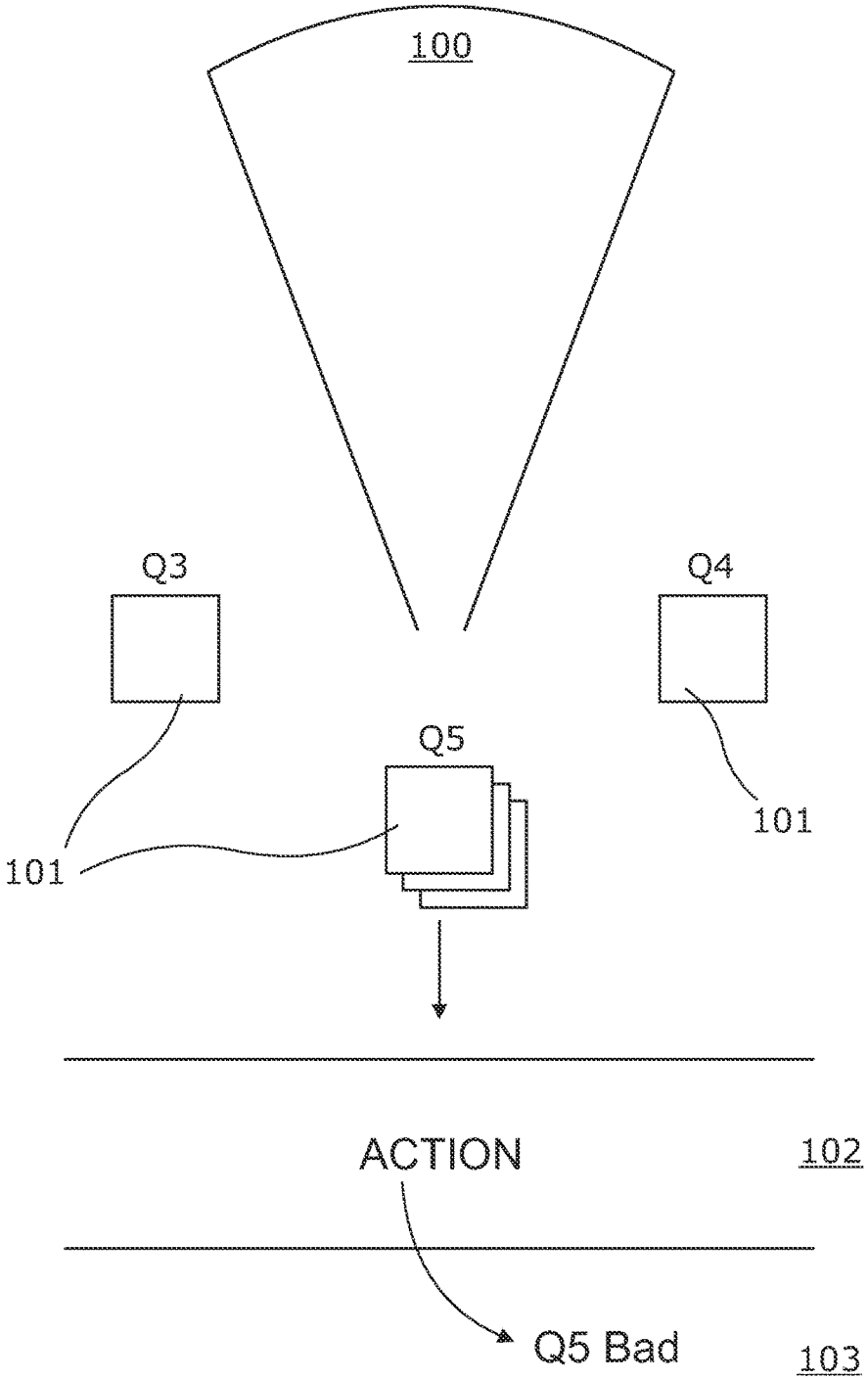


Fig. 9

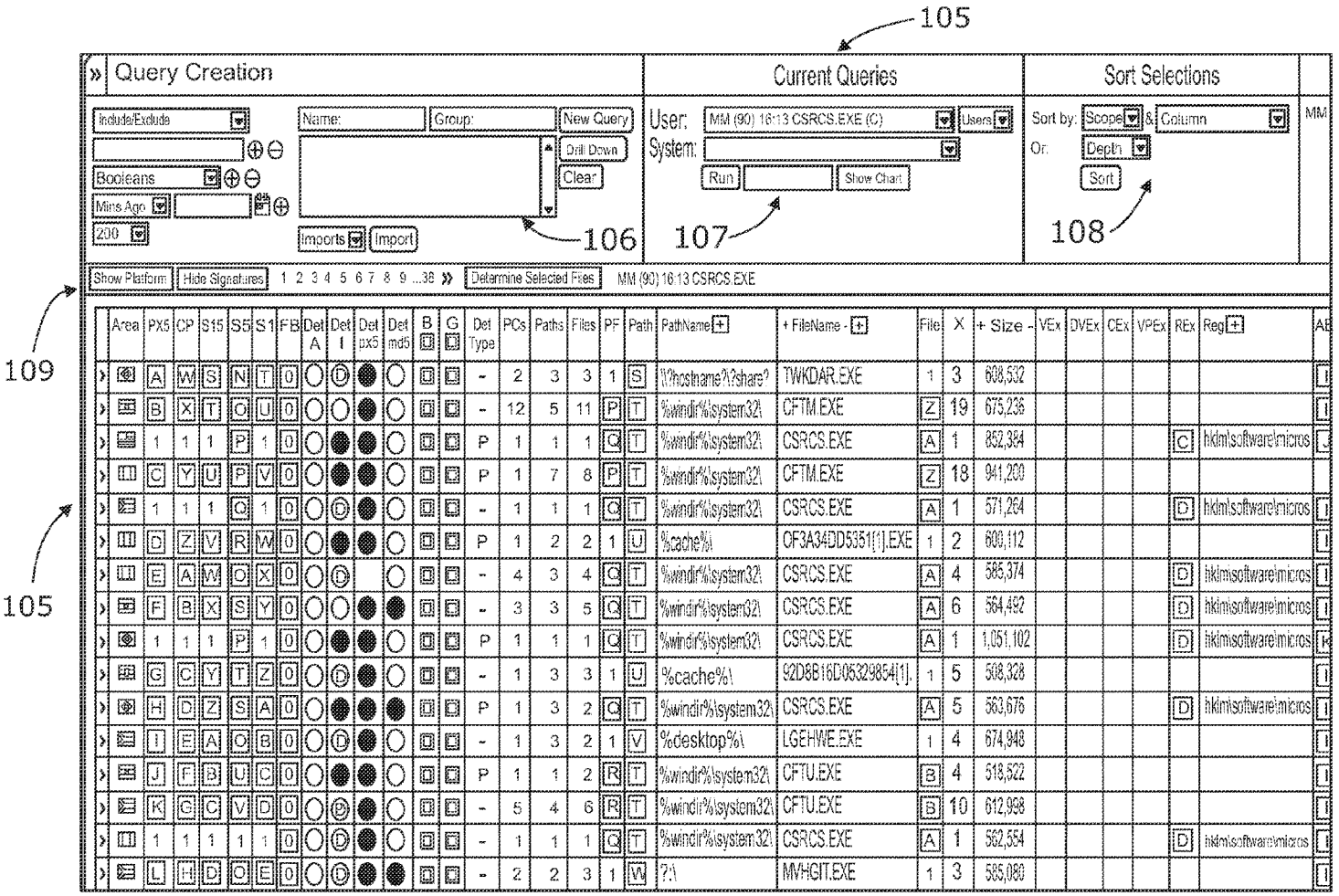

105
Query Creation
Current Queries
Sort Selections
Include/Exclude
Name:
Group:
New Query
Drill Down
Clear
Booleans
Mins Ago
200
Imports
Import
106
User:
MM (90) 16:13 CSRCS.EXE (C)
Users
System:
Run
Show Chart
107
Sort by:
Scope
&
Column
Or:
Depth
Sort
108
MM
Show Platform
Hide Signatures
1 2 3 4 5 6 7 8 9 ...38
Determine Selected Files
MM (90) 16:13 CSRCS.EXE
109
105
Area PX5 CP S15 S5 S1 FB Det A Det I Det px5 Det md5 B G Det Type PCs Paths Files PF Path PathName + FileName - + File X + Size - VEx DVEx CEx VPEx REx Reg + AE
%?hostname%?share% TWKDAR.EXE 698,532
%windir%\system32\ CFTIM.EXE 675,236
%windir%\system32\ CSRCS.EXE 852,364 hklm\software\micros
%windir%\system32\ CFTIM.EXE 941,200
%windir%\system32\ CSRCS.EXE 571,264 hklm\software\micros
%cache%\ OF3A34DD5351[1].EXE 608,112
%windir%\system32\ CSRCS.EXE 585,374 hklm\software\micros
%windir%\system32\ CSRCS.EXE 564,492 hklm\software\micros
%windir%\system32\ CSRCS.EXE 1,051,102 hklm\software\micros
%cache%\ 92D8B16D05329854[1]. 508,328
%windir%\system32\ CSRCS.EXE 563,676 hklm\software\micros
%desktop%\ LGEHWE.EXE 674,948
%windir%\system32\ CFTU.EXE 518,522
%windir%\system32\ CFTU.EXE 612,998
%windir%\system32\ CSRCS.EXE 562,554 hklm\software\micros
?:\ MVHGIT.EXE 585,080

Fig. 10

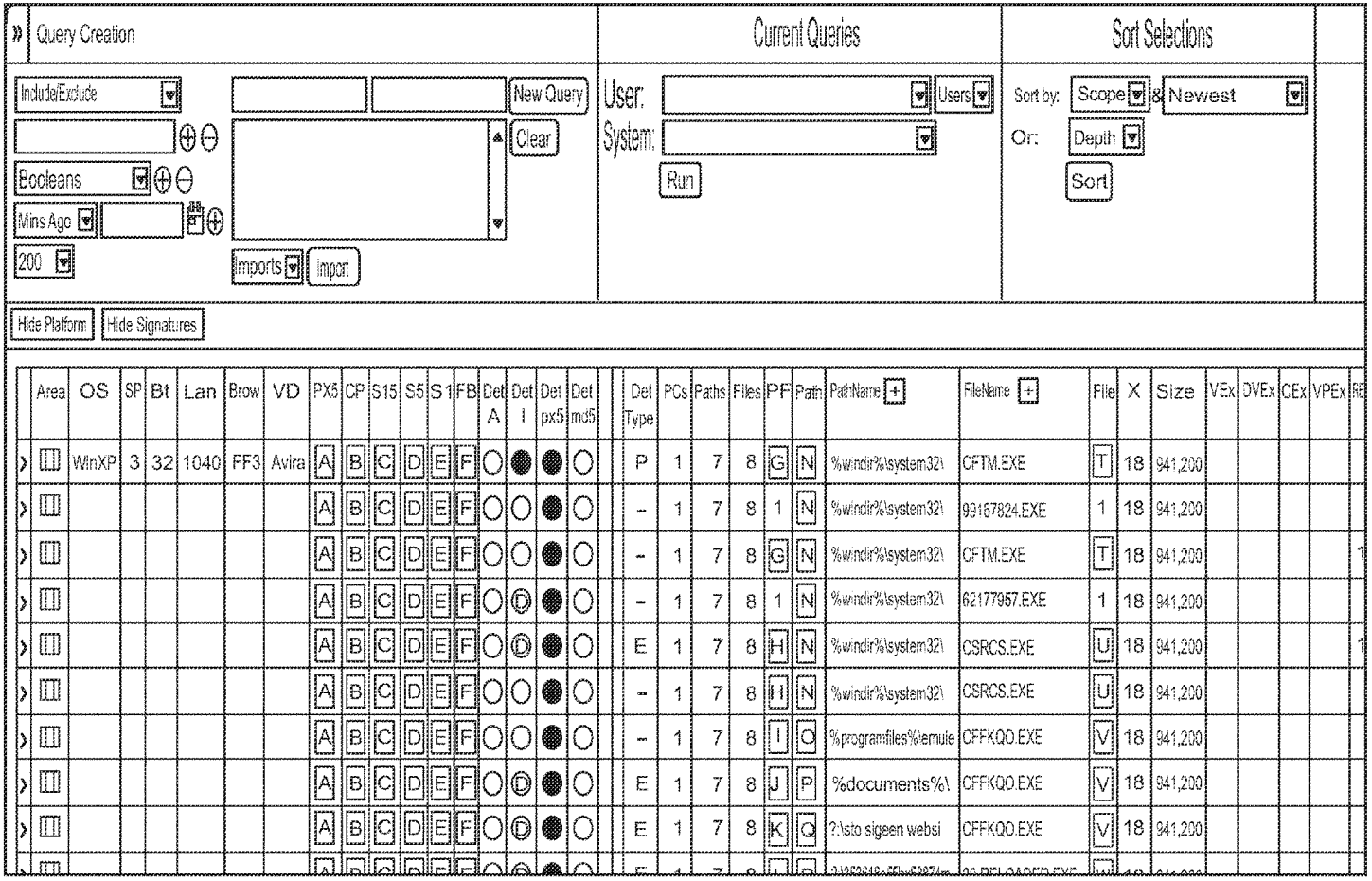

Query Creation
Current Queries
Sort Selections
Include/Exclude
New Query
Clear
Booleans
Mins Ago
200
Imports
Import
User:
Users
System:
Run
Sort by:
Scope
&
Newest
On:
Depth
Sort
Hide Platform
Hide Signatures
Area OS SP Bt Lan Brow VD PX5 CP S15 S5 S1 FB Det A Det I Det px5 Det md5 Det Type PCs Paths Files PF Path PathName FileName File X Size VEx DVEx CEx VPEx
WinXP 3 32 1040 FF3 Avira
%windir%\system32\ CFTM.EXE 99167824.EXE 62177957.EXE CSRCS.EXE %programfiles%\emule CFFKQO.EXE %documents%\ ?:\sto sigeen websi
941,200

Fig. 11

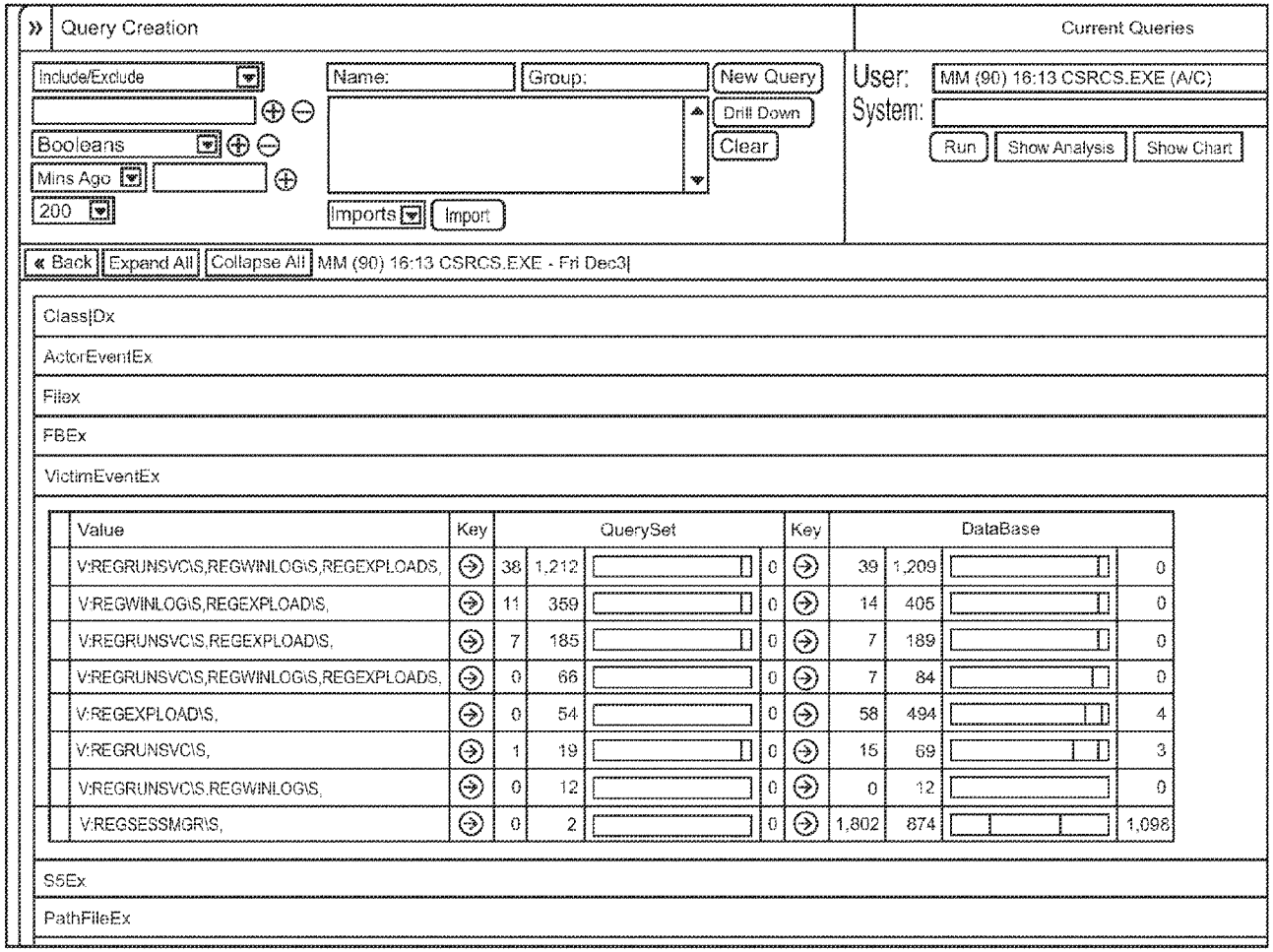

Query Creation
Current Queries
Include/Exclude
Booleans
Mins Ago
200
Name:
Group:
New Query
Drill Down
Clear
Imports
Import
User:
MM (90) 16:13 CSRCS.EXE (A/C)
System:
Run
Show Analysis
Show Chart
« Back
Expand All
Collapse All
MM (90) 16:13 CSRCS.EXE - Fri Dec3
Class|Dx
ActorEventEx
Filex
FBEx
VictimEventEx
Value
Key
QuerySet
Key
DataBase
V:\REGRUNSVC\S,\REGWINLOG\S,\REGEXPLOADS,
V:\REGWINLOG\S,\REGEXPLOAD\S,
V:\REGRUNSVC\S,\REGEXPLOAD\S,
V:\REGRUNSVC\S,\REGWINLOG\S,\REGEXPLOADS,
V:\REGEXPLOAD\S,
V:\REGRUNSVC\S,
V:\REGRUNSVC\S,\REGWINLOG\S,
V:\REGSESSMGR\S,
S5Ex
PathFileEx

Fig. 12

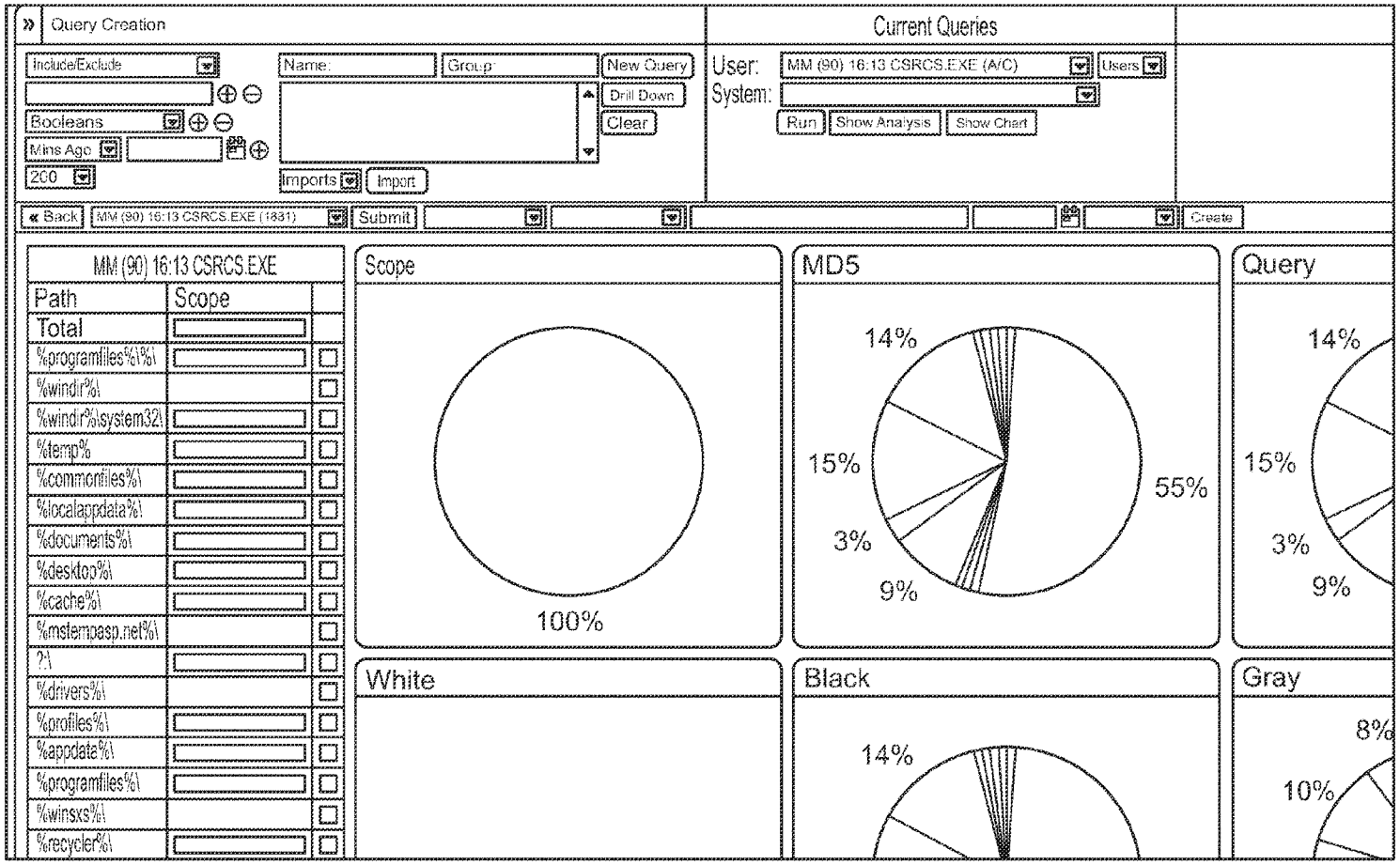

Query Creation
Current Queries
Include/Exclude
Booleans
Mins Ago
200
Name:
Group:
New Query
Drill Down
Clear
Imports
Import
User:
MM (90) 16:13 CSRCS.EXE (A/C)
Users
System:
Run
Show Analysis
Show Chart
« Back
MM (90) 16:13 CSRCS.EXE (1881)
Submit
Create
MM (90) 16:13 CSRCS.EXE
Path
Scope
Total
%programfiles%\%\
%windir%\
%windir%\system32\
%temp%
%commonfiles%\
%localappdata%\
%documents%\
%desktop%\
%cache%\
%mstempasp.net%\
?:\
%drivers%\
%profiles%\
%appdata%\
%programfiles%\
%winsxs%\
%recycler%\
Scope
100%
MD5
14%
15%
55%
3%
9%
Query
14%
15%
3%
9%
White
Black
14%
Gray
8%
10%

Fig. 13

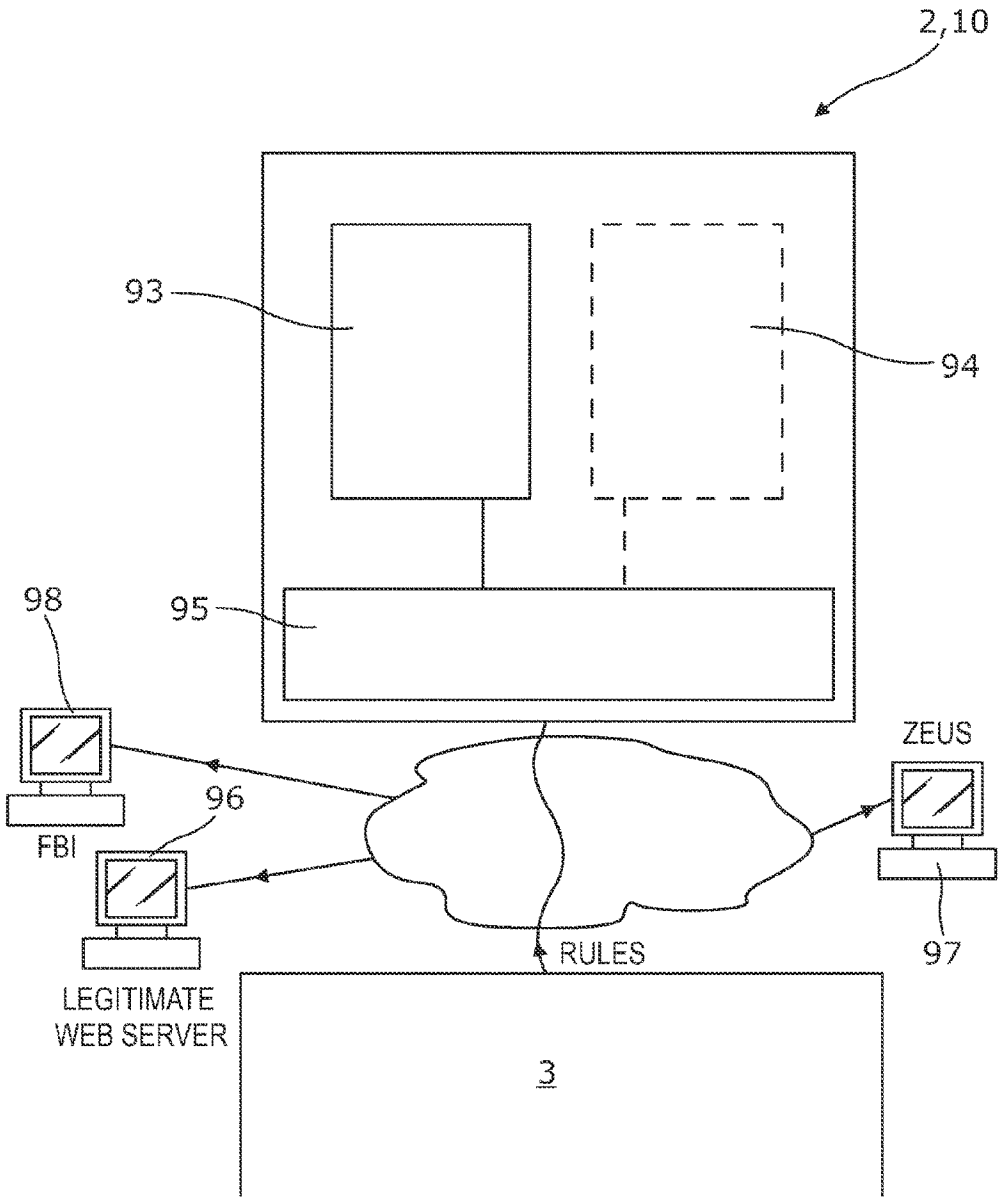


Fig. 14

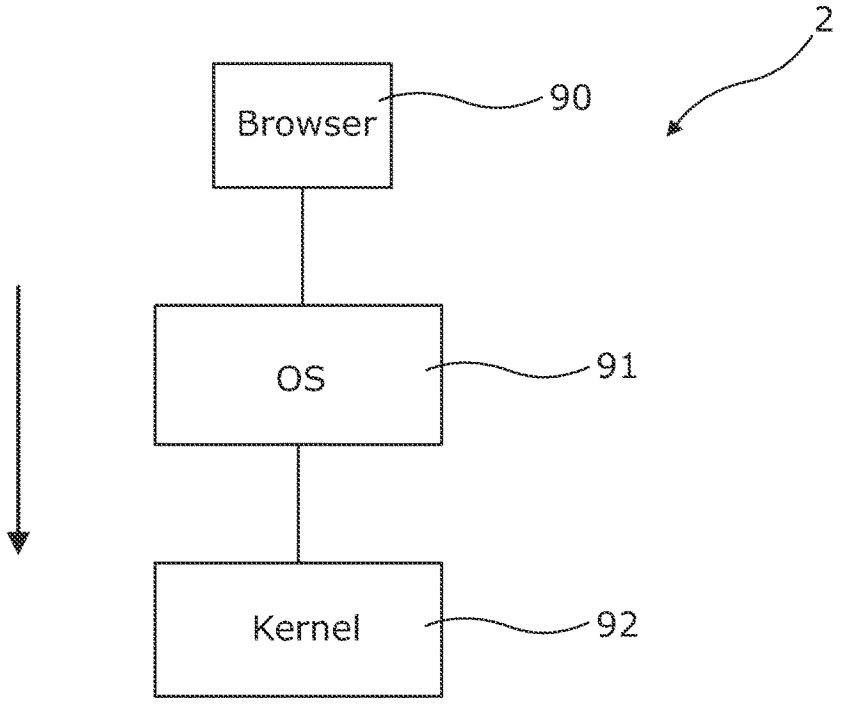


Fig. 15

METHODS AND APPARATUS FOR DEALING WITH MALWARE

CLAIM OF PRIORITY UNDER 35 U.S.C. §119

The present Application for Patent claims priority to Provisional Application No. 61/443,095 entitled "METHODS AND APPARATUS FOR DEALING WITH MALWARE" filed Feb. 15, 2011, and assigned to the assignee hereof and hereby expressly incorporated by reference herein.

REFERENCE TO CO-PENDING APPLICATIONS FOR PATENT

The present Application for Patent is related to the following U.S. Patent Applications: co-pending U.S. patent application Ser. No. 13/372,433, entitled "METHODS AND APPARATUS FOR MALWARE THREAT RESEARCH," assigned to the assignee hereof, and expressly incorporated by reference herein; and U.S. patent application Ser. No. 13/372,439, entitled "METHODS AND APPARATUS FOR AGENT-BASED MALWARE MANAGEMENT", now abandoned, assigned to the assignee hereof, and expressly incorporated by reference herein.

BACKGROUND

1. Field

The present invention relates generally to methods and apparatus for dealing with malware. And more specifically, systems and methods for protection against malware.

2. Background

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

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

These systems work well for protecting against known malicious objects. However, since they rely on signature files being generated and/or updated, there is inevitably a delay between a new piece of malware coming into existence or being released and the signatures for identifying that malware being generated or updated and supplied to users. Thus, users are at risk from new malware for a certain period of time which might be up to a week or even more.

More recently, so-called "cloud" based techniques have been developed for fighting malware/viruses. In these techniques, protection is provided by signatures that are stored in the cloud, i.e. in a central server to which the remote computers are connected. Thus, a remote computer can be given protection as soon as a new malware object is spotted and its signature stored in the central server, so that the remote computer is protected from it without the need to wait for the latest signatures to be downloaded and installed on the remote computer. This technique can also give the advantage of moving the processing burden from the remote computer to the central server. However, this technique is limited by sending only signature information or very basic information about an object to the central server. Therefore, in order to analyse whether or not an unknown object is malware, a copy of that object is normally sent to the central server where it is investigated by a human analyst. This is a time consuming and laborious task introducing considerable delay in classifying malware as safe or unsafe. Also, given the considerable volume of new objects that can be seen daily across a community, it is unrealistic to have a skilled human analyst investigate each new object thoroughly. Accordingly, malevolent objects may escape investigation and detection for considerable periods of time during which time they can carry out their malevolent activity in the community.

We refer in the following to our previous application US-A-2007/0016953, published 18 Jan. 2007, entitled "METHODS AND APPARATUS FOR DEALING WITH MALWARE," the entire contents of which are hereby incorporated by reference. In this document, various new and advantageous cloud-based strategies for fighting malware are disclosed. In particular a cloud-based approach is outlined where the central server receives information about objects and their behaviour on remote computers throughout the community and builds a picture of objects and their behaviour seen throughout the community. This information is used to make comparisons with this data across the community in developing and applying various heuristics and or rules to determine whether a newly seen object is malevolent or not.

This approach to fighting malware involves communicating, storing and managing vast amounts of data at the central server, which is a challenging problem in itself. It is also challenging to develop new schemes for more accurately and more efficiently detecting malware given the vast amount of data collected about objects seen in the community and the constantly evolving strategies used by malware writers to evade detection. It is also desirable to improve the processes for analysing the data to make the best use of the time and specialised skills of the human malware analysts.

Malware is also becoming increasingly adept at self-defence by interfering with the operation of security programs installed on a computer. This is another problem that security software must contend with.

SUMMARY

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

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

wherein the base computer comprises plural threat servers arranged to receive the data from the plural remote computers and apply rules and or heuristics against that data in real time to determine whether or not the object is malware and to communicate the determination to the remote computers;

the base computer further comprising at least one central server in communication with the threat servers and arranged to receive the data about objects from the threat servers to maintain a master database of data received about objects from all of the threat servers.

According to another aspect, an apparatus is provided for classifying a computer object as malware, the apparatus includes a base computer arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored or processed.

The base computer includes plural threat servers arranged to receive the data from the plural remote computers and apply rules or heuristics against that data in real time to determine whether or not the object is malware and to communicate the determination to the remote computers. The base computer also includes at least one central server in communication with the threat servers and arranged to receive the data about objects from the servers to maintain a master database of data received about objects from all threat servers.

According to yet another aspect, a method is provided for connecting a remote computer to one of a plurality of servers. The method includes installing an agent program on the remote computer, the agent program having an algorithm for allocating the remote computer to a server determined by the time when the agent program was installed; and connecting the agent program to the allocated server.

As will become apparent in view of the following disclosure, the various aspects and embodiments of the invention can be combined.

BRIEF DESCRIPTION OF THE DRAWINGS

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

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

FIG. **3** shows a more detailed view of an example of a base computer **3** according to an embodiment of the present invention;

FIG. **4** shows an example where web servers are located in different geographical locations;

FIG. **5** shows schematically the apparatus of FIG. **3** in more detail and in particular the way in which data can be moved from the FX layer to the ENZO layer in an embodiment;

FIG. **6** shows schematically an example in which a new instance of an ENZO server is created in accordance with an embodiment of the invention;

FIG. **7** shows schematically an example of apparatus by which an ENZO server shares out its workload between plural servers in accordance with an embodiment of the invention;

FIG. **8** is a chart showing an example of a scheme for analysing data about objects according to an embodiment of the present invention;

FIG. **9** show shows schematically an example of a scheme for processing data about objects according to an embodiment of the invention;

FIGS. **10** to **13** show graphical user interfaces of an example of a computer program for researching objects according to an embodiment of the present invention;

FIG. **14** shows an example of an agent program on a remote computer according to an embodiment of the present invention; and

FIG. **15** shows the hierarchy on a computer system.

DETAILED DESCRIPTION

Overview

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

In one exemplary embodiment, the base computer **3** is in communication with a database **7** with which the remote computers **2** can interact when the remote computers **2** run an object **4** to determine whether the object **4** is safe or unsafe. The community database **7** is populated, over time, with information relating to each object run on all of the connected remote computers **2**. As will be discussed further below, data representative of each object **4** may take the form of a so-called signature or key relating to the object, its attributes and behaviour.

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

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

Furthermore, the agent software at the remote computer **2** may be arranged to receive rules or heuristics from the base computer **3** for classifying objects as safe or unsafe. If the object is unknown locally, the agent software may apply the rules or heuristics to the details of the object to try to classify the object as safe or unsafe. If a classification is made, details of the object and classification are passed to the base computer **3** to be stored in the community database **7**. This means that the agent software is capable of providing protection against previously unseen objects even if it is "offline", e.g. it is unable to connect to the base computer for any reason. This mechanism is described in more detail later in the disclosure.

If the object is still not classified locally, then details of the object are passed over the Internet **1** or other network to the base computer **3** for storing in the community database **7** and for further analysis at the base computer **3**. In that case, the community database **7** is then searched at step **25** for a signature for that object that has already been stored in the community database **7**. The community database **7** is supplied with signatures representative of objects, such as pro-

grams or processes, run by each monitored remote computer **2**. In a typical implementation in the field, there may be several thousands or even millions of remote computers **2** connected or connectable to the base computer **3** and so any objects that are newly released upon the Internet **1** or that otherwise are found on any of these remote computers **2** will soon be found and signatures created and sent to the base computer **3** by the respective remote computers **2**.

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

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

If after the entire community database **7** has been searched the object is still unknown, then it is assumed that this is an entirely new object which has never been seen before in the field. A signature is therefore created representative of the object at step **28**, or a signature sent by the remote computer **2** is used for this purpose.

At this point, rules or heuristics may be applied by the base computer **3** to the details of the object to try to classify the object as safe or unsafe. If a classification is made, the signature is marked as safe or unsafe accordingly in the community database **7**. The signature is copied to the local database of the remote computer **2** that first ran the object. This mechanism is described in more detail later in the disclosure.

If the object is still not classified, the signature may be initially marked as bad or unsafe in the community database **7** at step **29**. The signature is copied to the local database of the remote computer **2** that first ran the object at step **30**. A message may then be passed to the remote computer **2** to instruct the remote computer **2** not to run the object or alternatively the user may be given informed consent as to whether to allow the object to run or not. In addition, a copy of the object itself may be requested at step **31** by the community database **7** from the remote computer **2**.

If the user at the remote computer **2** chooses to run a process that is considered unsafe because it is too new, then that process may be monitored by the remote computer **2** and/or community database **7** and, if no ill effect occurs or is exhibited after a period of time of n days for example, it may then be considered to be safe. Alternatively, the community database **7** may keep a log of each instance of the process which is found by the many remote computers **2** forming part of the network and after a particular number of instances have been recorded, possibly with another particular number of instances or the process being allowed to run and running safely, the signature in the community database **7** may then be marked as safe rather than unsafe. Many other variations of monitoring safety may be done within this concept.

The database **7** may further include a behaviour mask for the object **4** that sets out the parameters of the object's performance and operation. If an object is allowed to run on a remote computer **2**, even if the initial signature search **22** indicates that the object is safe, then operation of that object may be monitored within the parameters of the mask. Any behaviour that extends beyond that permitted by the mask is identified and can be used to continually assess whether the object continues to be safe or not.

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

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

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

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

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

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

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

This allows a core checksum to be generated by viewing only the executable elements of the checksum and making a comparison between two executables that share common executable code.

For the type 1 checksum mentioned above, three signature processes may be used. The first defines the entire file and will change with almost any change to the file's content. In particular, the first defines a representative sampling of the contents of the program in that if any fundamental change is made to the program, the checksum will change, but trivial changes can be identified as such, allowing correlation back to the original program body. The second attempts to define only the processing instructions of the process, which changes much less. The third utilises the file's size, which massively reduces the potential of collisions for objects of differing sizes. By tracking the occurrences of all signatures individually appearing with different counterparts, it is possible to identify processes that have been changed or have been created from a

common point but that have been edited to perform new, possibly malevolent functionality.

As well as checksum data, in many embodiments metadata about the object is captured and sent to the base computer **3**. Amongst other types, the types of metadata captured and sent to the base computer **3** might be:

"Events": these define the actions or behaviours of an object acting upon another object or some other entity. The event may include three principal components: the key of the object performing the act (the "Actor"), the act being performed (the "Event Type"), and the key of the object or identity of another entity upon which the act is being performed (the "Victim"). For example, the event data might capture the identity, e.g. the IP address or URL, of a network entity with which an object is communicating, another program acting on the object or being acted on by the object, a database or IP registry entry being written to by the object, etc. While simple, this structure allows a limitless series of behaviours and relationships to be defined. Examples of the three components of an event might be:

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

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

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

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

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

Thus, the metadata derived from the remote computers **2** can be used at the community database **7** to define the behaviour of a process across the community. As mentioned, the data may include at least one of the elements mentioned above (file size, location, etc.) or two or three or four or five or six or all seven (or more elements not specifically mentioned here). The data stored in the community database **7** provides an extensive corollary of an object's creation, configuration, execution, behaviour, identities and relationships to other objects or entities that either act upon it or are acted upon by it. This may be used accordingly to model, test and create new automated rules and heuristics for use in the community database **7** and as rules that may be added to those held and used in the local database of the remote computers **2** to identify and determine the response of the remote computers **2** to new or unknown processes and process activity.

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

Distributed Architecture

The system as described so far is generally as described in our previous application US-A-2007/0016953. There now follows a description of new, advantageous schemes in operating such a system. It will be appreciated that in the description in relation to FIGS. **1** and **2** the central computer and the community database are presented for convenience as single entities. As will be apparent from the following discussion, the base computer **3** can be comprised of multiple computers and servers, etc. and the community database **7** can be made up multiple databases and storage distributed around this central system.

FIG. **3** shows an example of an arrangement for the base computer **3**. The remote computer **2** has an agent program **10**, which generally includes the same functionality as the agent program described above in relation to FIGS. **1** and **2**. In short, the agent program **10** the monitors objects on that computer **2** and their behaviour and communicates with the base computer **3** to send details of new objects and behaviour found on the remote computer **2** and to receive determinations of whether the objects are safe or not. The agent program **10** optionally communicates with a local database **11**, which holds a local copy of signatures relating to the objects found on the remote computer **2**.

The agent program **10** can be comparatively small compared with other commercially available anti-malware software. The download size for the agent program **10** can be less than 1 MB and occupy about 3 MB of memory when running, using more memory only to temporarily hold an image of a file(s) being scanned. In comparison, other anti-malware packages will typically have a download of 50 MB and occupy 50 MB to 200 MB of computer memory whilst scanning files. Thus, the exemplary architecture can occupy less that 2% of system resources compared with other products.

This is achieved primarily by the exemplary agent program **10** being developed in a low-level language, having direct access to system resources such as the video display, disk, memory, the network(s), and without the incorporation of many standard code or dynamic linked libraries to perform these functions. Memory usage is optimised by storing data in

the local database structure that provides the ability to refer to objects by a unique identifier rather than requiring full filenames or signatures. All unnecessary dynamic link libraries are unloaded from the process immediately as they are identified as no longer being used, and background threads are merged to reduce CPU usage. A small, efficient agent can be deployed more quickly and can used alongside other programs, including other security programs, with less load on or impact on the computer's performance. This approach also has the advantage of having less surface area for attack by malware making it inherently more secure.

The agent program **10** communicates with the base computer **3** over the Internet **1** via the Internet's Domain Name Resolution System (DNS) **50**.

The base computer **3** has a first layer comprising one or more servers **61**, which in this example are web servers. In the present disclosure, the web servers **61** are referred to as "FX servers" or "threat servers" and the first layer is referred to as the "FX layer". However, it will be appreciated that any type of suitable server may be used according to need. The remote computers **2** are allocated to one of the web servers **61** as explained in more detail below.

The FX layer **60** makes real time decisions as to whether or not an object is benign or malevolent based on the details of that object sent from the remote computer **2**. Each web server **61** of the FX layer **60** is connected to a database **62** which contains entries for all of the objects known to the base computer **3** and a classification of whether the object is safe or unsafe. The database **62** also stores rules for deciding whether unknown objects are safe or unsafe based on the information received from the remote computer **2**. The web server **61** first searches the database **62** for matching signatures to determine if the object is safe or unsafe or unknown. As described above, the signature can be a function of the hashes (checksum data) derived from file itself, event data involving the object and or metadata about the object in any combination. If the object is unknown, the web server **61** then determines if any of the rules classify the object as unsafe. The web server **61** then communicates back to the remote computer **2** whether the object is safe and can be allowed to run, unsafe and is to be prevented from running, or unknown in which case the object can be allowed to run or not according to user preference.

Thus, the FX layer **60** reacts in real time to threats to determine whether or not malware should be allowed to run on a remote computer **2**.

Sitting behind the FX layer **60** is a second layer (called the "ENZO layer" **70** herein). Information about objects received by the FX layer **60** from the remote computers **2** is sent to the ENZO layer **70** such that a master record is maintained at the ENZO layer **70** of all data received from all remote computers **2**. This process is described in more detail below.

The ENZO layer **70** has one or more servers **74** (referred to as "central servers" or "ENZO servers" in this disclosure) connected to one or more "master" databases which aggregate all of the information received from all web servers **61** from all remote computers **2**. The databases may comprise any combination of the following three databases and in one exemplary embodiment has all three.

1) An object database **71**, which has entries containing object signatures (e.g. their MD5 checksums) and including metadata received about objects, e.g. file name, file location, or any other metadata collected by the system as described above.

2) A behaviour database **72** for capturing details of the behaviour of objects observed on a remote computer **2**. This database **72** is populated by the event data sent from the remote computers **2** and allows a picture to be built up of the behaviour and relationships an object has across the community.

3) A computer-object database **73** which associates remote computers **2** in the community with objects observed on those remote computers **2** by associating objects with identification codes sent from the remote computer **2** during object detail transmission. The computer may be identified by at least the following three identifiers sent from the remote computer: one which relates to the physical system, one which relates to the operating system instance and one which relates to the logged on user.

As will be appreciated, in a community having for example 10 million or more remote computers **2**, each having on average hundreds to thousands of objects, each having a large number of behaviours and associations with other objects, the amount of data held in the databases **71**,**72**,**73** is enormous.

The ENZO servers **74** can query the databases **71**,**72**,**73** and are operable, either by automation or by input from a human analyst or both, in monitoring object behaviour across the community and in investigating objects and developing rules for spotting malware. These rules are fed back to the FX layer **60** and or the agent program **10** running on remote computers **10** and, as discussed elsewhere in this document, are used by the web servers **61** or the agent program **10** in real time to stop malware from running on remote computers **2**. The performance of those rules in accurately stopping malware on the remote computers **2** is fed back to the ENZO layer **70** and can be used to refine the rules. The techniques used are described further in the following disclosure.

In an exemplary embodiment, some or all of the servers **61**,**74** and databases **62**,**71**,**72**,**73** of the FX and ENZO layers **60**,**70** are implemented using cloud computing. Cloud computing is a means of providing location-independent computing, whereby shared servers provide resources, software, and data to computers and other devices on demand. Generally, cloud computing customers do not own the physical infrastructure, instead avoiding capital expenditure by renting usage from a third-party provider. Cloud computing users avoid capital expenditure on hardware, software, and services when they pay a provider only for what they use. New resources can quickly be put on line. This provides a large degree of flexibility for the user.

An example of cloud computing is the Amazon Elastic Compute Cloud (EC2), which is a central part of the Amazon.com cloud computing platform, Amazon Web Services (AWS). Another example is the Windows Azure Platform, which is a Microsoft cloud platform that enables customers to deploy applications and data into the cloud. EC2 is used in the present example to provide cloud computing. Nonetheless, it will be appreciated that, in principle, any suitable cloud architecture could be used to implement the base computer **3**. Alternatively conventional data centres could be used to implement the base computer **3**, or a mixture of conventional data centres and cloud-based architecture could be used.

EC2 allows users to rent virtual computers on which to run their own computer applications. EC2 allows scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image to create a virtual machine, which Amazon calls an "instance," containing any software desired.

A user can create, launch, and terminate server instances as needed, paying by the hour for active servers, allowing the provision of "elastic" computing. EC2 provides users with control over the geographical location of instances, which allows for latency optimization and high levels of redundancy. For example, to minimize downtime, a user can set up

server instances in multiple zones which are insulated from each other for most causes of failure such that one backs up the other. In this way, the cloud provides complete control of a user's computing resources and its configuration, i.e. the operating system and software packages installed on the resource. Amazon EC2 allows the user to select a configuration of memory, CPU, instance storage, and the boot partition size that is optimal for the operating system and application. EC2 reduces the time required to obtain and boot new server instances to minutes, allowing the user to quickly scale capacity, both up and down, their computing requirements change.

EC2 maintains a number of data centres in different geographical regions, including: US East Coast, US West Coast, EU (Ireland), and Asia Pacific (APAC) (Singapore). As shown by FIG. **4**, in an exemplary embodiment, the servers and databases of the base computer **3** are implemented in plural of these geographical locations **80**.

The important question therefore arises of how to allocate remote computers **2** to a particular instance of web server **61** in the FX layer **60**. Important considerations in this decision are to minimise latency and to share the load evenly across the web servers. Ideally, the load on resources would be shared approximately equally across all instances of web servers in order to avoid web servers being under utilised and to minimise the number of web servers required. It is also desirable to implement a simple solution which does not add significantly to the speed or size of the agent program **10** on the remote computers, or place an overhead on the base computer **3**.

1) In an exemplary embodiment, the agent program **10** uses the geographical location of the remote computer **2** in allocating the remote computer **2** to a web server **61**, i.e. based on its proximity to a geographical region **80** of the cloud. The location of the remote computer can be determined by any convenient means, such as examining the IP address of the remote computer **2** against a database of known location ranges, or performing a network route analysis to determine the number of "hops" required to reach the destination server. This functionality can be performed by the agent program **10** running on the remote computer **2** when it is first installed on the computer **2**. And by allocating based on location, latency can be minimised.

2) In an exemplary embodiment, the agent program **10** generates a random number based on the date and time of its installation seeded with unique identifiers of the remote computer to decide to which web server **61** it should be mapped. This provides a simple, pseudo random way of mapping remote computers **2** to web servers **61**, which shares the load evenly between web servers **61** even if several thousand remote computers are being installed simultaneously.

3) The DNS distribution layer then maps the URL to the physical data centre. The Domain Name System (DNS) is a hierarchical naming system built on a distributed database for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. The DNS distribution network **50** allows load balancing to plural servers **61** making up the FX layer **60**.

Any combination of these techniques can be used by the agent program **10** to allocate a remote computer **2** to a web server **61**. And all three may be used in combination.

The agent program may also utilize a secondary algorithm to re-map the remote computer **2** to different URL in case the primary URL is unavailable for any reason, e.g. if the servers go down. Also, in the event that malware has been able to disable the DNS mechanism then the agent program **10** can make direct contact with the FX servers **61** by referring directly to a series of direct IP addresses reserved for this purpose.

Optionally, the agent program **10** is arranged so that it can reassign the remote computer **2** to a different server **61** by receiving feedback/instruction from the server **61** the computer is already connected to. This can be used to add central control to the allocation of agent program **10** to servers **61**, which adds more flexibility to the system.

Thus, this scheme allows a simple, agent managed way of dynamically load balancing between servers and providing resiliency, whilst dealing with geographical location. In known prior art arrangements, specialised hardware load balancers are employed in the network to divide load to other servers, or else agents are manually allocated to web servers **61** when the agent program **10** is installed.

Using a cloud-based distributed architecture for the base computer **3** presents challenges in trying to maintain integrity of data across the servers and in managing the update of data across the many servers. As will be clear from the foregoing, the web servers **61** in a typical implementation will be dealing with huge amounts of data. There will be large amounts of commonality in the data, which can be used for determining rules on what objects are malware. However, it is impractical to store all data on all web servers **61** on all regions. This presents a problem in determining whether or not the data is common in real time.

To address these issues, an exemplary embodiment adopts the following scheme. Each web server **61** in the FX layer **60** examines an incoming packet of data from a remote computer **2** about an object seen on that remote computer **2**. The agent program **10** generates a checksum of the data packet and sends it to the web server **61**. The web server **61** has a database **62** of objects that have previously been seen and which have already been sent to the ENZO server **74**. The web server **61** checks its database **62** for the checksum of the incoming data. If the checksum is found, then the data has already been seen by the ENZO server **74**. In this case, the web server **61** just increases the count associated with that checksum in its database of the number of times that data has been seen to assist in determining the popularity of a piece of software or the frequency of the occurrence of a specific event. This information can then be forwarded to the ENZO server **74**. If the checksum is not found in the FX database **62**, then the web server requests the full data packet from the remote computer and then forwards this to the ENZO server **74**. The ENZO server **74** then stores this data in the appropriate databases **71**,**72**,**73**. Thus, the ENZO layer **70** keeps a master list in the object database **71** of all data objects, their metadata and behaviour seen on all of the remote computers **2**. This master list is propagated back to all of the web servers **61** in all of the geographical locations **80**. Thus, all web servers **61** are updated with information about which objects have been seen by the community.

This scheme has been found to reduce workload and traffic in the network by a factor of about 50 compared with a conventional scheme where typically each piece of data on being received by a web server would immediately be propagated to all other servers in the system.

It is desirable that the data held in the master databases **71**,**72**,**73** in the ENZO layer **70** provide scalability, accessibility and resilience to the data for users. It will be appreciated that if just one live ENZO server **74** is used, then if that server becomes inaccessible, all live data is lost. For this reason, in many embodiments at least one ENZO server **74** resides in each region **80** of the cloud to provide redundancy. This means that the web servers **61** are not all directly linked to the

same ENZO server **74**, but rather the web servers **61** are only linked to the ENZO server **74** in their own region **80**. This creates a need for an extensible method of providing data to multiple database servers.

FIG. **5** shows schematically a scheme of updating data from a web server **61** to an ENZO server **74**. Each web server **61** has two temporary databases **63***a*, **63***b* linked to it. At an initial starting point, one database **63***a* is part full **63***b* (active) and the other is empty (inactive). As information about objects is received from a remote computer **2** by the web server **61**, it posts the information into the active database **63***a*. This information will be checksum information received from the remote computers **2** about objects or events seen on the remote computer. If the checksum is previously unseen at the web server, the associated data about that object or event is posted to the database along with its checksum.

Once the active database **63***a* reaches a predetermined size or some other predetermined criteria is met, the contents of the active database **63***a* are put into long term storage **66** together with a time stamp of the point in time when this happens. While this is occurring, the inactive database **63***b* is made active and starts to fill with new data as it arrives, and the formerly active database **63***a* is cleared and made inactive. Again, when this database is full, then the process of time stamping and moving the contents of the database to long term storage is repeated and the databases are swapped over. In some embodiments, the FX layer **60** has a S3 sub-layer **65**, which is responsible for managing input/output to the long term storage **66**. S3 (Simple Storage Service) is an online storage web service offered by Amazon Web Services. S3 is an Amazon service for the storage of high volume data in a secure and non-volatile environment through a simple web services interface. S3 is used to store the history of the FX server data being sent from the FX servers to the ENZO servers. When a block of data is put into long term storage **66**, the S3 layer **66** also forwards this data to the ENZO server **74** or servers so that the ENZO server can update its databases. All FX servers feed all ENZO servers via the S3 storage service.

The databases **63***a*,**63***b* may be emptied into long term storage **66** for example every few minutes or even every few seconds. However, as will be appreciated, the actual time period used can be selected according to the load and size of the databases and may be chosen dynamically based on for example the load on the database or the size of the data. Thus, the S3 layer **65** holds all of the data received by all of the web servers **61** in the community. This has a number of advantages as illustrated by FIG. **6**.

Firstly, this can be used to allow the master ENZO databases **71**,**72**,**73** to be rebuilt for example in the event that an ENZO server **74** develops a problem. Also, if the data held by an ENZO server **74** is corrupted somehow, for example by a problem in a software upgrade on the ENZO server **74**, it is possible to roll back the databases at the ENZO server **74** to a point before the corruption occurred. By using the timestamps associated with the blocks of data in the S3 layer **65**, the ENZO server **74** can request the data blocks in long term storage **66** at the S3 layer **65** to be resent to the ENZO server **74** so the server's databases **71**,**72**,**73** can be brought up to date.

Furthermore, using this scheme, the ENZO servers **74** in multiple regions **80** can create backup databases of the ENZO databases by receiving the data from the S3 layer. This can be done without affecting the live ENZO databases **74**, so these can carry on functioning seamlessly.

The alternative would be to post to databases at a single ENZO server which is designated the master server and to propagate the data from the master server to other servers in the community. However, this in effect creates one "live" server and one or more backups. The live server therefore does most of the work, e.g. pre-processing data received from the agents, updating the databases and analysing the data for threats, whilst the backup servers are under utilised. Thus, the live server must be more powerful than the backup servers. This tends to be wasteful and inefficient of resources. In comparison, the exemplary embodiment, shares the workload around the servers equally.

It is also desirable for operators, human malware analysts, etc., to have a copy of the data from a live server for research, etc. Image servers may be created in the same way as the live servers and the backup servers. The user of the system can take the image server offline when he needs to take an image for offline use. Once the image is taken, the image database can be brought up to date using the blocks of data from the S3 layer.

Cloud computing can recreate instances of web servers very quickly, e.g., in a matter of minutes. Cloud computing is therefore advantageous in implementations where the volume of transactions grows (or shrinks). In this situation, a new server instance can be quickly created to handle the new load. However, it is necessary to take an image of a live server to create the image. This is disruptive to the processes running on that instance during imaging. It is therefore clearly not acceptable to take a live server out of commission for this period. Therefore, in many embodiments, a special dedicated image server is maintained behind the ENZO server, which is updated in a similar way to the live and backup servers. In this case, the image server is taken off-line and imaged. This process might take for example 30 minutes. During this time, the image server is not receiving updates from the S3 server and therefore becomes out of date. However, since the exact time when the server went off line is known, it can be updated by requesting data blocks from the S3 server with a timestamp later than when it went off line. Thus, an image can be taken of an updated server for offline use etc. and brought back up to date without affecting the running of the system.

It is also desirable to be able to scale out the architecture of the ENZO layer **70**. For example, a live server **74** may be running out of processing power due to the increase in traffic it is seeing from web servers **61** or agents **10**. For example, more remote computers **2** may be connected to the system, or a common software package may be updated leading to a large volume of new data being sent to the ENZO layer **70**. In this situation, a second or further live server instance can be created in the ENZO layer **70** in order to scale out horizontally (as opposed to adding more CPU/RAM to an existing server). As described above, it is relatively straightforward to create an image of an ENZO server **74** in a cloud-computing environment. As can be seen from FIG. **7**, the data on each server **74** is divided into a plurality of sections. In this case the data on the server **74** is divided into eight sections **0**-**7**. Each section does a share of the processing on that server. When the server is divided, various sections are allocated to each server. So, for example, in the example of FIG. **7**, the original server L**1** processes data **0**-**3**, and the newly created server L**2** processes data **4**-**7**. Each server then handles pre-processing and storing its own data. For the purposes of analysing the data, in many embodiments the data from these multiple severs is aggregated so that the human analyst is working on the complete set of data when investigating malware (as described in more detail below). Of course, the data can be divided in different ways among the servers according to need.

So, instead of holding a single database and then having to dissect it if it is required to distribute the load across multiple

servers, in many embodiments the database is already dissected making it is much easier to make a transition to further distribution of the database workload across multiple servers.

The number of data sections may be selected to be eight, because this is enough to give reasonable flexibility and ability to expand. However, it is rare that a business needs to expand by more than 800% at any time. Nonetheless, other plural numbers of data sections can be used as appropriate.

This architecture allows vast amounts of data to be managed across multiple locations and servers. The architecture can be scaled and servers copied without incurring extra work or affecting the running of the system. The data can be exchanged across servers that can change dynamically.

Using Trend Data

Malware evolves at a significant pace. Historically, malware used a single infection (module or program) to infect many computers. As this allowed malware researchers to quickly become aware of the infection, signatures could be quickly deployed to identify and block the threat. In the last decade, malware authors have made their threats contain many varied modules, which shared a common purpose. To detect and block these new threats is a much greater challenge for malware research and to creating and distributing signatures to identify and block all of the variations, any one of which could infect one or more computers. Due to the varied and distributed nature of these modern threats malware researchers are often late in identifying that a threat even exists.

By collecting hashes, signatures and statistics of new programs and modules correlated by path and file details, and aggregating this information at a central location, it is possible to use a method of analysis that will allow faster identification of new threats.

On any given day the amount of new commercial programs and modules that are created can be considered almost constant. This data can be measured against any of the metadata or event data collected about the object. In one example, the data is measured against the Path (Folder) location of new files created. The data can also be measured against their filenames, and file types (e.g. .exe, .dll, .drv, etc.) along with details such as vendor, product and version information, registry keys, or any other system derived data forwarded to base computer. Any combination of these factors is possible. In whichever way it is chosen to group the data, by establishing a baseline for the pattern and distribution of this data across the groups it is possible to quickly identify outliers.

FIG. **8** shows an example of collected data over a time period grouped according to folders in the file system compared with the established normal distribution (base line). If the number of new programs in the windows system32 subfolder is normally x in any given time period t (e.g., in a given day) and in a particular time period t**1** under consideration it is 1.4x (or exceeds some other predetermined ratio) then it indicates an abnormal increase in the number of new files which signifies possible malware activity. This identification allows greater focus to be placed on researching these outliers. Processes can be readily developed that can use this outlier information to automatically group, identify or prioritise research effort and even make malware determinations. Where automated rules are used to identify malware, this information can be used to automatically heighten sensitivity of these rules when applied to objects in the group that has been identified as an outlier.

Another example of this technique is to focus on a filename and compare the number of new objects (as represented by its hash) that utilize that name, or the number of new objects all of which have new names. This is contrary to the general principles of most commercial software, which needs consistency to enable manageability of their products. Most commercial applications have only a few names and each file name has only a few variants as each new release is made available. In contrast, some malware has thousands of variants using the same, similar or similarly structured name and which are observed by the community utilising the related names within a very short window of time (e.g., seconds-weeks).

In principle, any length of time for the window can be selected. In practice, this is likely to be of the order of every 15 minutes, or every few hours, or a day, rather than longer periods like every week.

In some embodiments, techniques are provided to provide context for researchers. In detecting malware, human analysts and computer automated analysis have different strengths and are often used together to some degree to harness the capabilities of both. Computers are good at applying rules to and processing large amounts of data. In the present application, the amount of data received from agent program **10** running on remote computers **2** is vast and therefore suited to being processed by computer. It would be impractical to have human operators to look at all of this data to determine malware. On the other hand, experienced human analysts are highly adept at spotting malware given the appropriately focussed data about that object and similar objects and their context. Often a human analyst can "intuitively" spot malware, in a way that is difficult to reduce to a set of rules and teach to a machine. For example, kernel32.dll is a file associated with the Windows® operating system. A malware program may be called something similar, e.g. kernel64.dll, which would immediately be apparent to a user as being suspicious, but which it would be difficult to teach a machine to spot.

The pattern may be obvious in retrospect when an analyst is presented with the relevant data. However, this is a rear view mirror approach. The problem is finding the relevant data in the vast amount of data being received at the base computer **3** so that the analyst can focus his/her time on the important data. As illustrated by FIG. **9**, what is needed is some what of distilling the raw data **100** into groups of related data **101** to provide a focussed starting point for a malware researcher to investigate and take necessary action **102** to flag objects bad or safe **103** accordingly.

As will be appreciated, a single object on a remote computer **2** across the community can have a high degree of associations with other objects on remote computers **2**. Some tools currently exist which can map links between entities to create a network of associations. However, when used for a malware object and its associations with other objects, these will result in a diagram that is so dense and complicated that it is practically useless for the purpose of allowing a human analyst to identify malware. The problem is sifting this data so useful features are extracted for the analyst so that he can quickly focus on the likely candidates for being malware.

An exemplary embodiment, illustrated by FIGS. **10** to **13**, provides a software program running on the ENZO server **74** and giving the user a visual based tool for running queries against the databases. A query language such as TSQL may be used for example to query the database based on the user's input. As mentioned in the foregoing, the ENZO layer **70** has a complete record of the properties and activities of all objects seen on all remote computers **2** in their databases. FIG. **10** shows an example of an on-screen display and user interface including a table **105** in which the rows represent the objects (determined by their signatures) that match the results of the current query. The user interface provides tools for creating

new queries **106**, for displaying current and recent queries **107** and for sorting the data displayed in the table for the current query **108**. Clearly there may be more results than will fit on one screen. There may therefore be more than one page of results to which the user can navigate through appropriate controls **109** provided by the user interface.

The table **105** has a plurality of columns of data representing information about the objects. The information may include any of the data collected about an object, its metadata or its behaviour, sent to the base computer **3** by the remote computers **2**. This may include any of the following for example: the name of the file, and the location (path) of the file, the country where the object was first encountered, the identity of the remote computer that encountered the object, its operating system and version number, details of the web browser and any other security products installed on the computer, the creator of the file, the icon associated with the file, etc. The information may include metadata about the file, such as the author or the size of the object or information relating to executable instructions in the object. The data may contain "actor" and "victim" information and details of its registry key. The information may include information about whether or not the object has been identified as safe or unsafe at the remote computer **2** or at the FX server **61**.

The information displayed may contain counts of how often a particular attribute is present among all the objects returned by the query or across the community. In addition, the information may contain less easily understood information. As discussed above, the information collected about an object can include checksums of parts of the executable, e.g. the first 500 bytes of the executable part of the file. In many cases, the actual value of the checksum or other data is of little or no interest to the user. Therefore, in many implementations some or all of this information is presented to the user in a graphical way, utilising colour, shape and or different symbols, which makes spotting patterns in the data more intuitive and speedy for the user.

In some embodiments, at least one column of data is presented according to the following scheme:

1) if the value is unique in that column between the rows of data representing objects then the numeral "1" is presented.

2) if the value is replicated between the rows of data on the screen, or between all the rows of data matching that query, then a coloured symbol (e.g. a green triangle or a red letter "P" in a blue square, etc.) is allocated for that value and displayed in that column for all rows that share that value. Thus, without needing to know what the symbols represent, the user can see at a glance which rows on the screen have commonality of an attribute.

This provides a starting point for running further queries. For example, the user can quickly see commonality between objects, which look interesting. As explained elsewhere in this document, a typical trait of malware is that the objects use a large number of file names, file locations, similar file sizes, similar code portions, etc. The program allows a user to refine a query by for example clicking on columns in the table and using the query creating tools **106** to add a particular attribute value to the predicate of the query, or to start a new query with this column value in the predicate, etc.

The ultimate aim of this process is for the user to find a query that returns the maximum number of rows with commonality. By this process, the user identifies a set of criteria made up of hashes and other metadata definitions that can be used to accurately identify similar objects that share criteria. For example, FIG. **11** shows a user query that returns rows with a high degree of commonality representing objects that have commonality. The software also in many implementations allows the user to bring up a screen of statistics for the objects that have been returned from the query. In particular, as shown by FIG. **12**, the interface can display to the user the number of objects returned by the query that are known to the database as being malware, the number known as being not malware, and the number where it is not yet known whether or not the object is malware (represented in FIG. **12** by differently shaded/coloured bars). Other data about the results of a query or objects having a certain attribute selected by the user can be shown in graphical form, e.g. a bar graph, or a pie diagram as shown in FIG. **13**. The user can take this information into account as well as information shown in the table **105** when deciding whether or not the results of the query determine malware. The user can then take the appropriate action, such as tagging the objects returned by the query as malware, not malware, or suspicious and requiring further investigation. Thus, by being able to query and analyse the collective view of an object, i.e. its metadata and behaviours, across all agents **10** that have seen it, a more informed view can be derived, whether by human or computer, of the object. In addition it is possible to cross-group objects based on any of their criteria, i.e. metadata and behaviour.

To take an example, it might be a previously identified trait of certain types of malware that the malevolent object is located in the "recycle bin" of the operating system. This gives a starting point for the user to run a query to find all objects, or all objects encountered during a user specified time frame, that are located in the recycle bin by looking at their "path" attribute. This would display rows to the user corresponding to the different objects found (e.g. each row corresponding to a different MD5 signature). The user can then drill down through the rows by selecting interesting attributes by clicking on a value in a column. For example, the user may choose objects whose attributes show that the object's icon is the same as the icon used the "Notepad" program (which comes with Microsoft Windows® operating systems), but which are not Microsoft originating programs. In many implementations the system allows the user to apply Boolean operators in refining the query. The system allows the user to bring up statistics of how well the query cover objects that are known by the system to be malevolent or benign to give the user more information about useful the query is in classifying objects.

The users employed in grouping objects according to this scheme need not be highly skilled malware analysts. The users need only basic training in use of the tool and in grouping objects by running queries and looking for commonality. The final analysis of objects covered by the query can be taken by a skilled malware analyst. The advantage of this system is that the computer is doing the work in processing the raw data which is too large of a task to be practical for a human operator to complete. The human operators grouping the data need not be skilled, which reduces costs for the business. This reduces costs and leaves the skilled malware analysts to concentrate on queries that are found to be good candidates for further investigation. Thus, the skill of these operators is most effectively deployed in using their experience to identify malware.

Referring to FIG. **9**, if at stage **103** the user decides that a query (Q**5**) is deterministic in classifying the objects returned by a query as safe or unsafe, then the rule can be passed back to the web servers **61** and or the agent program **10** at the remote computers **2** and used in real time in deciding whether new objects seen at the web servers **61** or remote computer **2** are safe or unsafe.

If a new object that has not been seen before by the community is received by a remote computer **2** or a web server **61**

and is deemed by one of the rules run by the remote computer **2** or web server **61** to be unsafe, the object can be stopped from running on the remote computer **2** in real time, or the user of the remote computer **2** can be warned and given the option of continuing or not. Furthermore, the remote computer **2** (via the web server **61** to which it is assigned) or the web server **61** can report to the ENZO server **74** that a new object has been determined to be safe or unsafe by the rule. The ENZO server **74** can update its master database with this finding and this finding can be communicated to other remote computers **2** and or web servers **61** and used to update their databases that objects with that signature are unsafe and should be blocked in the future, or safe and should be allowed to run.

Alternatively, the rule can be used by human analyst at the ENZO layer to monitor suspicious objects. Objects covered by the rule can be displayed to a human analyst and the analyst can then decide for each object in turn whether it is malevolent or benign, and mark the object accordingly. These determinations for each object are sent back to the web servers **62** and or agent software **10** and used to decide whether or not the object is allowed to run when encountered at the remote computers in real time. This method does not react as quickly to new threats, compared with applying the rules in real time at the web servers **62** and or agent software **10**. However, it may be preferred in cases where it is desired to keep the number of "false positives" low, and or where a rule is found to be good at identifying suspicious objects which need further investigation before a firm determination can be made that they are malware.

Furthermore, this action can be semi automated. The application can monitor the groupings along with any and all actions taken such as groups being determined as malicious or as benign. By tracking these actions it is possible for the ENZO server **74** to learn from human analysts how to identify and determine objects automatically. The application can remember queries that have been run by users. If the same query is being run repeatedly by a user and returning malware objects, or if a researcher consistently takes the action of determining the objects to be malware, then the system can automatically identify this condition and create a rule of the criteria and the action to determined matching objects as malicious. This rule can be sent to the FX server layer **60** to enable real time actions to be made to catch objects that match the rule and determine them as malicious. The identity of these rules and of malware grouping applied by the rules can then be fed back to both the agent and the ENZO server **74**. In this way the FX server **61** is performing an action sanctioned and distributed by the ENZO server **74**, on its behalf. Similarly, these same processes can be applied to benign groups of objects resulting in rules that determine such objects as "good" which can be propagated back to the FX layer **60** in the same way for the rules for "bad" objects. Thus, the ENZO layer **70** allows the user to easily identify these groupings. The ENZO layer can then monitor the groupings along with any and all actions taken such as groups being determined as malicious or as benign. By tracking these actions it is possible for the ENZO layer **70** to learn from humans how to identify and determine objects automatically.

The ENZO layer **70** can also monitor trends and bring information to the human analysts' attention, such as the evolution in the behaviour of a previously known good program. For example, a legitimate version of the game Solitaire will be observed by the ENZO servers to have a particular behaviour profile from the actions performed by that program on remote computers over a period of time. If after a period of time that program starts exhibiting new, suspicious behaviour, such as trying to install a driver or accessing a web server in another country, this can be flagged for investigation.

Furthermore, an exemplary embodiment of the agent program **10** at the remote computer provides an interface that allows the user to report "false positives" to the system, i.e. where an object has been flagged as malware, which the user believes to be legitimate software. The system is arranged to feedback this information to the ENZO server **74**. At this point, the system can determine that the rule is no longer reliably deterministic in finding malware and can disable the rule. Alternatively the system can flag the rule for manual review and or modification by a human analyst. Alternatively, it can be decided that the rule is generally useful despite the false positive and that the false positive is an outlier. The conflicting benign object can then be specifically excluded from the rule at the FX layer **70** by placing it on a white list.

It should be noted that rules can be prioritised (applied in order by priority) and can be established to only apply against objects having a specific determination, location, popularity, age or any other criteria known by or derived by the database. For example, if a first rule determines the object originates from Microsoft, then it may be decided not to apply any further rules. In contrast, if a first rule determines that an object is suspicious, then it may be decided to apply further rules as appropriate.

Thus, many embodiments of the software application running on an ENZO server **74** provide powerful object behaviour monitoring and assessment. The software allows the user to obtain a thorough understanding of an object's behaviour on a remote computer and the contextual environment in which it is observed, in part through providing powerful data visualisation techniques. The software enables large research teams to work collaboratively and incrementally and leverages human research skills and experience through expert system automation.

Configurable Firewall

It is desirable to stop malware before a computer becomes infected, e.g., when the infected file is received by email, downloaded, or when it is installed or run for the first time.

A well known example of a malware program is the "Zeus" Trojan. This program runs in the background on an infected computer and steals login details as they are entered on screen by a user by monitoring the user's keystrokes. This information is then sent to the criminal's servers where it can be used to take control of the infected computer. An instant message is sent to the criminal over the network when the malware program detects logon information being entered on an online banking website, so the criminal receives this information in near real time. The Trojan can then take remote control of the user's computer using the Virtual Network Connection protocol, a protocol that allows a remote user to access and take control of a computer remotely. The criminal can then steal money from the victim's bank account.

While most anti-malware software attempts to stop the malware from infecting a host computer or stopping communications with the malware as soon as possible, not much attention has been shown to finding and prosecuting the criminals responsible. One difficulty is that it is illegal for the anti-malware or even the police to hack into the criminal's servers to find the criminal's data. Indeed, it a known ruse for the criminals to use so-called "botnets", which are "zombie networks" of legitimate users' computers which have been infected by malware and are thereby under the control of the criminals.

An exemplary embodiment illustrated by FIG. **14** implements a firewall **95** with the agent program **10** at the remote computer **2**. A firewall per se is well known. However, nor-

mally firewalls are configured by rules input by the user to govern what programs have permission to access the network for incoming/outgoing traffic. The exemplary embodiment additionally takes rules from the base computer **3** to limit, which objects on the remote computer **2** can communicate across the firewall and with which entities **96**,**97** on the network. For example, information can be sent from the base computer **3** to the firewall **95** that known active URLs of Zeus servers **97** are blocked from exchanging messages from all objects on the remote computer **2**. Thus, the firewall **95** can implement a rule blocking communications with the criminal's servers.

This provides a valuable second line of defence against malware. It is generally desirable to stop malware at the point of infection by preventing the malware from infecting a computer. The next line of defence is to stop the malware from communicating. The configurable firewall **95** can therefore leverage the rules and knowledge developed about malevolent objects and network URLs held at the base computer **3** to provide a next line of defence in fighting malware.

Furthermore, in one embodiment, the agent program **10** can be used to contact the authorities **98**, e.g., a law enforcement agency, if malevolent Internet traffic is detected. The IP traffic between the infected computer **2** and the criminal's server **97** can be intercepted by the firewall **95**, stored and sent to the authorities' server **98**, so the authorities can see what information the criminals were stealing through the malware. This information can be sent to the authorities' servers **98** in encrypted form for added security. In the exemplary embodiment, checksums are made and stored of the data that sent to the authorities' servers **98**, as there are very strict processes in place for forensic data handling and evidence retention that should be observed.

Legal constraints will mean in most instances it is necessary to obtain permission from the user of the infected computer **2** before this information is captured and sent to a third party. In this situation, the agent program **10** will open a dialogue with the user to obtain permission before proceeding. In some instances, the user may be happy to give blanket permission for data to be sent whenever a malware program or a particular class of malware program is detected on his computer **2**, so it will not be necessary to obtain specific permission every time malevolent objects or communication is detected.

Self Defence

Malware has evolved to the state where it will attack, disable or otherwise destroy security products installed on a computer. There are two principle vectors, which need to be considered.

A first approach is to target the methods used by malware to identify security products. In this specific case the malware will use behaviour monitoring of processes running on the remote computer that view, observe, record, examine or otherwise interfere any aspect of the malware code. Examples might be a malware entity watching for processes or tasks which look at the registry keys the malware has used to configure, control or record its operation. As will be appreciated, malware is by its nature surreptitious. From the malware's point of view, no processes on the remote computer should be monitoring or interacting with the malware object, since the existence of the malware object on the local computer should be unknown. If any processes are monitoring the malware, it is therefore likely to be security software or a system monitoring tool or a system tool being used by the user of the computer **2** to rid the computer **2** of the infection.

Thus, by finding processes that are doing this, the malware process can find the identity of the process and its location or locations on disk. This identification then allows it the ability to delete, disable or interfere with the process or processes it has identified as potential threats. In most cases these will indeed be security programs or system monitoring tools, both which it might consider threats.

Many embodiments guard against this possible interference by malware programs by preventing the behavioural detection of a security program when examining the malware. This is achieved by hiding behind, masquerading as, and/or leveraging core operating system components to examine the malware's state and components. In particular, the agent program **10** performs operations by impersonating the system's context when it is necessary to rely on standard system calls, or circumventing any system calls by accessing data at the raw level as it is stored on disk. For example, the agent program **10** can directly impersonate the kernel of the operating system when calling the ZwTerminateProcess function to terminate a malevolent process. In doing so, the malware cannot trace the call back to the agent program. The malware considers this behaviour as normal, as it expects the operating system to interact with it (e.g. it is necessary for the operating system to be able to terminate the process when the computer is shut down), and so the malware does not attempt to terminate the agent program process. Furthermore, the malware cannot react by killing the process as the process is only identifiable as the operating system itself. To kill the operating system would also prevent the malware from operating.

FIG. **15** shows a hierarchy of control on a computer system **2** going from the kernel level **92** at the lowest level, the operating system level **91**, and a web browser running **92** on the operating system **91**. A second consideration is that, for malware to assert maximum control over the operation of the computer, it will often attempt to install a driver, possibly a kernel mode driver, which has complete control over processes and actions performed. A type of malware commonly known as a "rootkit" operates in this way and can, among other things, be used to conceal other malware, notably password-stealing key loggers and computer viruses. In certain cases the malware will load a malicious driver and use this to disable security applications and drivers from starting when Windows starts. In particular, the malware may have a list of drivers or program components that it knows correspond to security products.

To prevent malware from using this approach to disabling security applications the agent program **10** dynamically creates a second or backup driver or component **94** which loads as a backup of main component **93** during installation or bootup in the event that the main components **93** are inaccessible (as shown by FIG. **14**). It would have identical functionality but would be modified so as to appear different by name and form. This can be randomised, so that the malware cannot learn to identify the security program. In this scenario should the primary component fail to load, most likely due to being killed by malware, the second driver **94** would be loaded to perform its function. If the primary component **93** loads successfully, the secondary component **94** would simply terminate to avoid conflict by identifying the presence of the primary component **93**. By randomising the filenames and checksums of the installed objects comprising the agent program **10**, the agent program **10** is able to evade heuristic detection by active threats that would attempt to terminate or remove critical agent components.

Later versions of Microsoft's Windows operating system provide enhanced security by requiring system and low level drivers to be signed. This policy, called the kernel mode code signing policy, disallows any unauthorized or malicious driver to be loaded. Some malware creations modify the

master boot record to nullify driver certification checks. The malware executed by an infected master boot record bypasses driver signing policy by changing the boot options of Microsoft boot programs that will allow an unsigned driver to load.

So, in these conditions the secondary driver or component could remove its own digital signature to evade detection by the malware. This means it would fail to load unless the master boot record had been modified to disable driver verification. So it represents no threat to an uninfected system and would never load unless the system was infected and the master boot record was modified.

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

Those of skill in the art would understand that information and signals may be represented using any of a variety of different technologies and techniques. For example, data, instructions, commands, information, signals, bits, symbols, and chips that may be referenced throughout the above description may be represented by voltages, currents, electromagnetic waves, magnetic fields or particles, optical fields or particles, or any combination thereof.

Those of skill would further appreciate that the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

One of ordinary skill in the art will appreciate that the various illustrative logical blocks, modules, and circuits described in connection with the embodiments disclosed herein may be implemented or performed with a general purpose processor, a digital signal processor (DSP), an application specific integrated circuit (ASIC), a field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, or any combination thereof designed to perform the functions described herein. A general purpose processor may be a microprocessor, but in the alternative, the processor may be any conventional processor, controller, microcontroller, or state machine. A processor may also be implemented as a combination of computing devices, e.g., a combination of a DSP and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration.

The steps of a method or algorithm described in connection with the embodiments disclosed herein may be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. A software module may reside in RAM memory, flash memory, ROM memory, EPROM memory, EEPROM memory, registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium known in the art. An exemplary storage medium is coupled to the processor such the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor. The processor and the storage medium may reside in an ASIC. The ASIC may reside in a user terminal. In the alternative, the processor and the storage medium may reside as discrete components in a user terminal.

The previous description of the disclosed embodiments is provided to enable any person skilled in the art to make or use the present invention. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments without departing from the spirit or scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

What is claimed is:

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

receiving, at a first threat server, details of a first computer object from a first remote computer, wherein the details of the first computer object include data uniquely identifying the first computer object;

determining, by the first threat server, whether the first computer object has been previously seen by comparing the data uniquely identifying the first computer object to a plurality of data uniquely identifying plural computer objects in a first database associated with the first threat server;

receiving additional information about the first computer object from the first remote computer when the first computer object has not been previously seen;

storing the details of the first computer object and the received additional information about the first computer object in a second database associated with the first threat server when the first computer object has not been previously seen;

providing contents of the second database to at least one database associated with a central server, wherein the contents comprise a signature of the first computer object, behavior information about the first computer object, and information about the first remote computer;

increasing a count associated with a number of times that the first computer object has been seen, and providing the increased count associated with the number of times that the first computer object has been seen to the central server; and

receiving, at a second threat server, at least a portion of the contents of the at least one database associated with the central server, wherein the at least a portion of the contents of the at least one database associated with the central server include a subset of the details of the first computer object stored in the second database.

2. The method according to claim **1**, further comprising storing at intervals, the contents of the second database in storage together with a timestamp and clearing the second database.

3. The method according to claim **2**, further comprising creating a backup central server by receiving at a second central server, all of the time-stamped blocks of data from the second database and incorporating all of the time-stamped blocks of data into at least one database associated with the second central server.

4. The method according to claim **2**, further comprising:

taking the central server off-line for a period of time such that the central server does not receive data from the first and second threat servers during that period of time;

after the period of time has elapsed, updating at least one database associated with the central server with time-stamped blocks of data from the storage that have a timestamp later than the time when the central server went off-line; and

bringing the central server back on line.

5. The method according to claim **2**, comprising:

rolling back at least one database associated with the central server to a point of time in the past;

updating the at least one database associated with the central server with time-stamped blocks of data from storage that have a timestamp later than the past point of time; and

bringing the central server back on line.

6. The method according to claim **2**, wherein the central server comprises:

a) an object database storing object signatures and metadata about objects;

b) a behavior database storing object behavior information; and

c) a computer-object database storing information about what objects are present on what remote computers.

7. The method according to claim **6**, wherein the threat and central servers are implemented using cloud computing.

8. The method according to claim **1**, further comprising:

receiving, at the second threat server, details of a second computer object from a second remote computer, wherein the details of the second computer object include data uniquely identifying the second computer object;

determining, by the second threat server, whether the second computer object has been previously seen by comparing the data uniquely identifying the second computer object to a plurality of data uniquely identifying plural computer objects in a third database associated with the second threat server;

determining that the second computer object has been seen before;

increasing a count associated with a number of times that the second computer object has been seen and providing the increased count associated with the number of times that the second computer object has been seen to the at least one central server; and

receiving, at the first threat server, a count associated with the number of times that the second computer object has been seen.

9. A system for classifying a computer object as malware, the system comprising:

a first threat server arranged to receive details of a computer object from a first remote computer, wherein the details of the first computer object include data uniquely identifying the first computer object, wherein the first threat server is further arranged to receive the details of the computer object from the first remote computer and determine whether the first computer object has been previously seen by comparing the data uniquely identifying the first computer object to a plurality of data uniquely identifying plural computer objects in a first database associated with the first threat server, wherein the first threat server is further arranged to receive additional information about the first computer object from the first remote computer when the first computer object has not been previously seen, store the details of the first computer object and the received additional information about the first computer object in a second database associated with the first threat server when the first computer object has not been previously seen, provide contents of the second database to at least one database associated with a central server wherein the contents comprise a signature of the first computer object, behavior information about the first computer object, and information about the first remote computer, and increase a count associated with a number of times that the first computer object has been seen;

the central server arranged to receive the increased count associated with the number of times that the first computer object has been seen; and

a second threat server arranged to receive at least a portion of the contents of the at least one database associated with the central server, wherein the at least a portion of the contents of the at least one database associated with the central server include a subset of the details of the first computer object stored in the second database.

10. The system according to claim **9**, wherein the first and second threat servers are arranged to store, at intervals, the contents of the second database in storage together with a timestamp and clear the database.

11. The system according to claim **10**, further comprising:

a backup central server having a database, the database of the backup central server being populated by receiving at the backup central server all of the time-stamped blocks of data from the storage and incorporating them into the database of the backup central server.

12. The system according to claim **10**, wherein, in the event that the central server is taken off-line for a period of time such that it does not receive updates of data from the first and second threat servers during that period of time, the central server is arranged to, after the period of time has elapsed, update at least one database with time-stamped blocks of data from the storage that have a timestamp later than the time when the central server went off-line.

13. The system according to claim **10**, wherein, in the event that at least one database of a central server is rolled back to a point of time in the past, the central server is arranged to update the at least one database with time-stamped blocks of data from storage that have a timestamp later than the point of time in the past.

14. The system according to claim **9**, wherein the central server comprises:

a) an object database storing object signatures and metadata about objects;

b) a behavior database storing object behavior information; and

c) a computer-object database storing information about what objects are present on what remote computers.

15. The system according to claim **14**, wherein the first and second threat servers and the central server are implemented using cloud computing.

* * * * *

Exhibit 5

US010025928B2

(12) **United States Patent**
Jaroch et al.

(10) **Patent No.: US 10,025,928 B2**
(45) **Date of Patent: Jul. 17, 2018**

(54) **PROACTIVE BROWSER CONTENT ANALYSIS**

(71) Applicant: **WEBROOT INC.**, Broomfield, CO (US)

(72) Inventors: **Joe Jaroch**, Elk Grove Village, IL (US); **Harry Murphey McCloy, III**, Longmont, CO (US); **Robert Edward Adams**, Sunnyvale, CA (US)

(73) Assignee: **WEBROOT INC.**, Broomfield, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **13/633,956**

(22) Filed: **Oct. 3, 2012**

(65) **Prior Publication Data**

US 2013/0086681 A1 Apr. 4, 2013

Related U.S. Application Data

(60) Provisional application No. 61/542,693, filed on Oct. 3, 2011.

(51) **Int. Cl.**
H04L 29/06 (2006.01)
G06F 21/56 (2013.01)

(52) **U.S. Cl.**
CPC ***G06F 21/566*** (2013.01)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,029,245 A * 2/2000 Scanlan 726/4
6,594,697 B1 * 7/2003 Praitis et al. 709/225
6,898,619 B1 * 5/2005 Tran 709/203
8,464,318 B1 * 6/2013 Hallak 726/4
2003/0037236 A1 * 2/2003 Simon et al. 713/161
2004/0073811 A1 * 4/2004 Sanin 713/201
2004/0260754 A1 * 12/2004 Olson et al. 709/200
2009/0125976 A1 * 5/2009 Wassermann et al. 726/1
2009/0193497 A1 * 7/2009 Kikuchi et al. 726/1
2011/0219446 A1 * 9/2011 Ichnowski 726/22
2011/0239288 A1 * 9/2011 Cross et al. 726/11
2013/0055386 A1 * 2/2013 Kim et al. 726/22

OTHER PUBLICATIONS

David Scott and Richard Sharp. 2002. Abstracting application-level web security. In Proceedings of the 11th international conference on World Wide Web (WWW '02). ACM, New York, NY, USA, 396-407. DOI=10.1145/511446.511498 http://doi.acm.org/10.1145/511446.511498.*

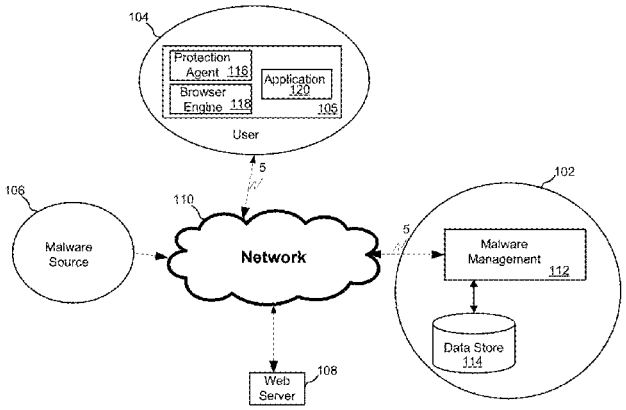
* cited by examiner

Primary Examiner — Jeffery Williams
(74) *Attorney, Agent, or Firm* — Merchant & Gould P.C.

(57) **ABSTRACT**

A protection module operates to analyze threats, at the protocol level (e.g., at the HTML level), by intercepting all requests that a browser engine resident in a computing device sends and receives, and the protection agent completes the requests without the help of the browser engine. And then the protection module analyzes and/or modifies the completed data before the browser engine has access to it, to, for example, display it. After performing all of its processing, removing, and/or adding any code as needed, the protection module provides the HTML content to the browser engine, and the browser engine receives responses from the protection agent as if it was speaking to an actual web server, when in fact, browser engine is speaking to an analysis engine of the protection module.

20 Claims, 4 Drawing Sheets



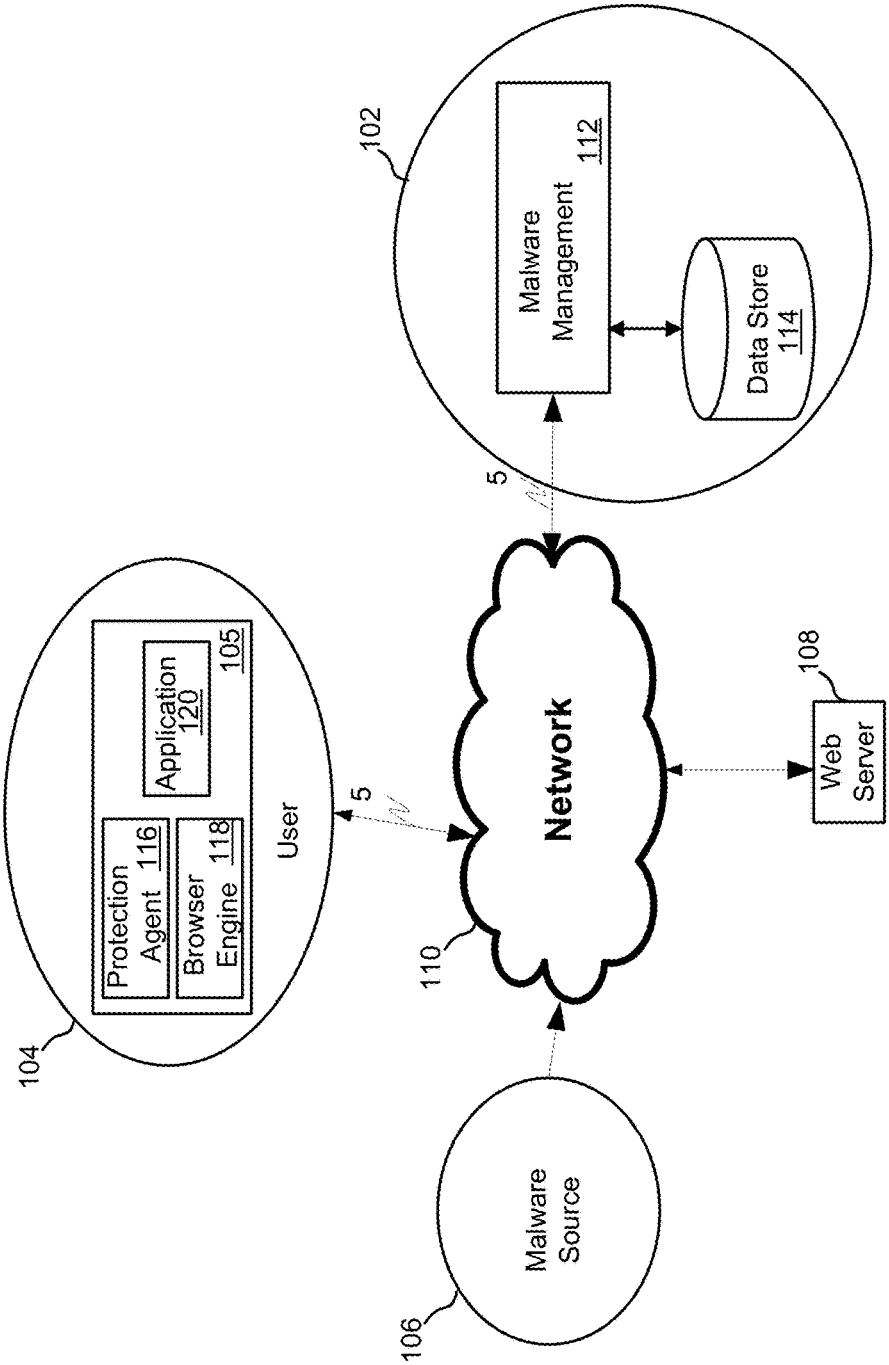

102
Malware Management 112
Data Store 114
5
Network
110
108
Web Server
User
104
Application 120
105
Protection Agent 116
Browser Engine 118
5
Malware Source
106

FIG. 1

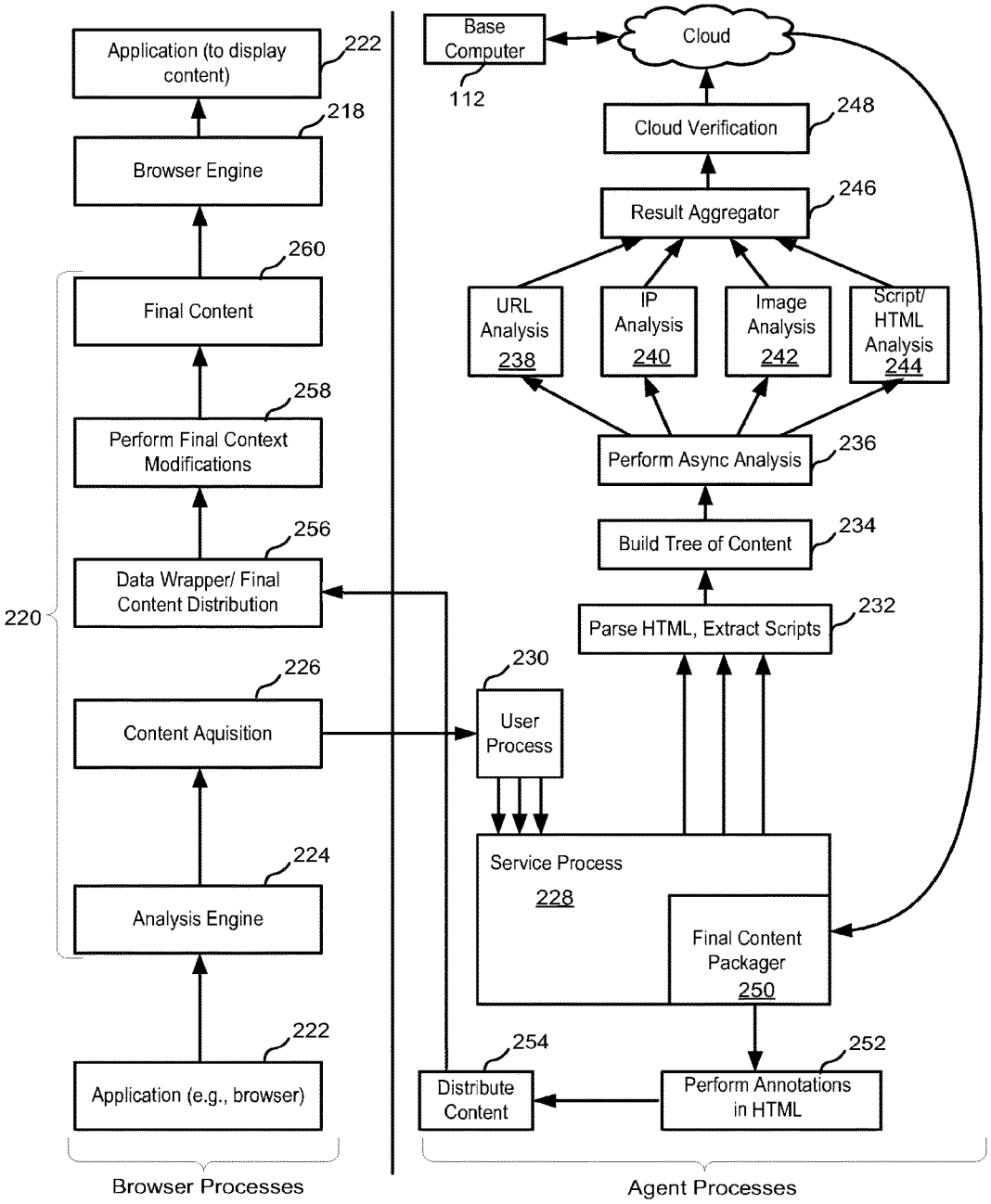

Application (to display content)
222
Browser Engine
218
Final Content
260
Perform Final Context Modifications
258
Data Wrapper/ Final Content Distribution
256
220
Content Aquisition
226
Analysis Engine
224
Application (e.g., browser)
222
Browser Processes
Base Computer
112
Cloud
Cloud Verification
248
Result Aggregator
246
URL Analysis 238
IP Analysis 240
Image Analysis 242
Script/ HTML Analysis 244
Perform Async Analysis
236
Build Tree of Content
234
Parse HTML, Extract Scripts
232
User Process
230
Service Process
228
Final Content Packager
250
Distribute Content
254
Perform Annotations in HTML
252
Agent Processes

FIG. 2

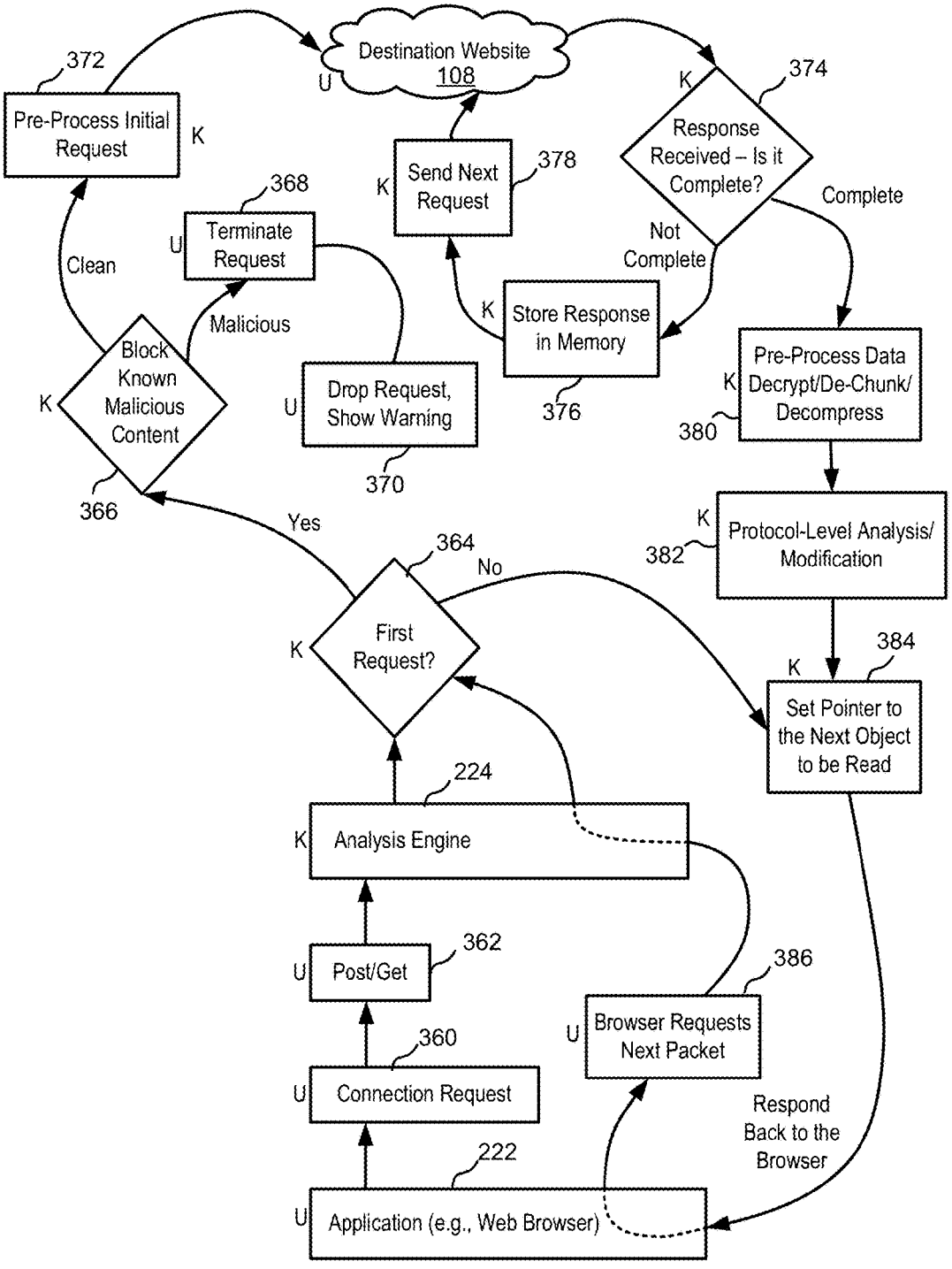

Destination Website
108
U
372
Pre-Process Initial Request
K
374
Response Received – Is it Complete?
K
Complete
Not Complete
378
Send Next Request
K
368
Terminate Request
U
Clean
Malicious
Block Known Malicious Content
K
366
Drop Request, Show Warning
U
370
Store Response in Memory
K
376
Pre-Process Data Decrypt/De-Chunk/ Decompress
K
380
Protocol-Level Analysis/ Modification
K
382
Yes
No
364
First Request?
K
384
Set Pointer to the Next Object to be Read
K
224
Analysis Engine
K
362
Post/Get
U
360
Connection Request
U
386
Browser Requests Next Packet
U
Respond Back to the Browser
222
Application (e.g., Web Browser)
U

FIG. 3

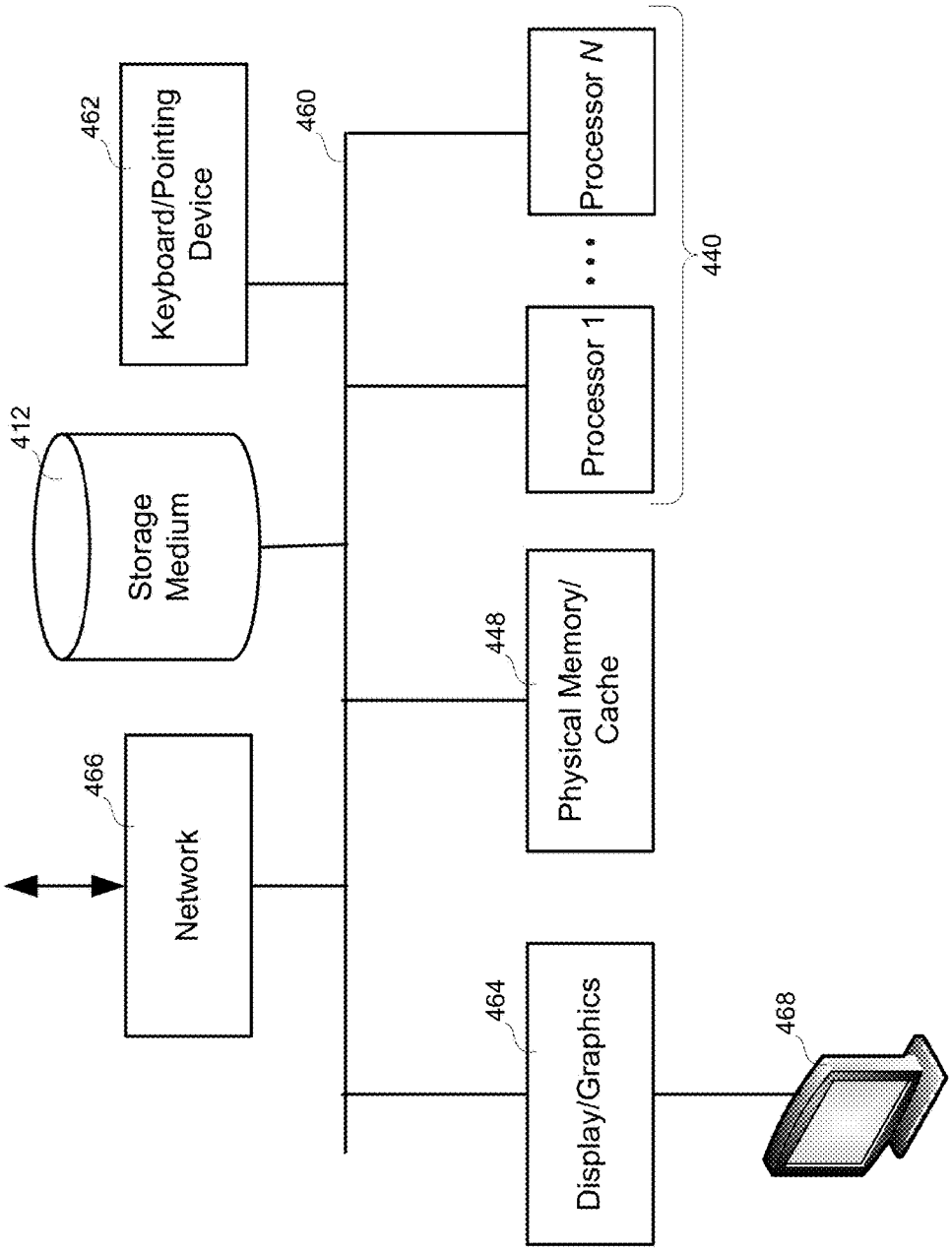

462
Keyboard/Pointing Device
460
Processor N
440
Processor 1
412
Storage Medium
448
Physical Memory/ Cache
466
Network
464
Display/Graphics
468

FIG. 4

PROACTIVE BROWSER CONTENT ANALYSIS

CROSS REFERENCE TO RELATED APPLICATION

The present application claims the benefits of and priority, under 35 U.S.C. § 119(e), to U.S. Provisional Application No. 61/542,693, filed Oct. 3, 2011, entitled "PROACTIVE BROWSER CONTENT ANALYSIS," which is incorporated herein by this reference in its entirety for all that is teaches and for all purposes.

FIELD

An exemplary aspect of the present invention generally relates to computer system management. In particular, but not by way of limitation, an exemplary aspect relates to systems and methods for controlling pestware or malware or other undesirable or unwanted applications and/or instructions.

BACKGROUND

Personal computers and business computers are continually attacked by viruses, trojans, spyware, adware, etc, collectively referred to as "malware" or "pestware." These types of programs generally act to gather information about a person or organization—often without the person or organization's knowledge. Some pestware is highly malicious. Other pestware is non-malicious but may cause issues with privacy or system performance. And yet other pestware is actually beneficial or wanted by the user. Wanted pestware is sometimes not characterized as "pestware" or "spyware." But, unless specified otherwise, "pestware" or "malware" as used herein refers to any program that is malicious in some way and/or collects and/or reports information about a person or an organization and any "watcher processes" related to the pestware or malware.

SUMMARY

In accordance with an exemplary aspect, a protection module operates to analyze threats, at the protocol level (e.g., at the HTML level), by intercepting all requests that a browser engine resident in a computing device sends and receives, and the protection agent completes the requests without the help of the browser engine.

And then the protection module analyzes and/or modifies the completed data before the browser engine has access to it, to, for example, display it. After performing all of its processing, removing, and/or adding any code as needed, the protection module provides the HTML content to the browser engine, and the browser engine receives responses from the protection agent as if it was speaking to an actual web server, when in fact, browser engine is speaking to an analysis engine of the protection module.

This allows the protection module to have control over what a browser engine "sees," providing means to remove any exploits, malware, and other threats dynamically. This also enables the protection module to add content into the browser stream at the HTML level, before receipt by the browser.

In some exemplary implementations, search engine results (e.g., results provided by Google®, Yahoo®, and Bing®) are annotated/updated/amended by the protection module—within the HTML code—to denote if a particular website is legitimate or malicious. For example, a legitimate link in the search results may be depicted in connection with a green check mark and a suspect link may be depicted with a red cross. (Of course other indicators could also be used that identify to a user whether or not a link is "good," "bad," or "unknown.) In addition to search result annotation, the protocol-level analysis approach may also be used in connection with anti-phishing and URL analysis among other types of analysis.

The differences between the disclosed protocol-level analysis approach compared to other prior anti-malware approaches are significant. In the context of search result annotation for example, the data (e.g., a web page of search results) is first analyzed and modified by an analysis engine of the protection module, which has control over every element of a web page before the web page is operated on by the browser engine. This is in contrast to prior approaches that just make high-level modifications to the content after the content has been rendered and displayed through a Browser Helper Object. With an exemplary aspect of the present protocol-level approach, there is virtually no performance overhead, and in many cases, there is actually a performance improvement when performing the browser content analysis.

When the protection module receives content from a web server, the protection module then, if necessary, decrypts and decompresses the web content and then assembles the requested web page (e.g., in a decrypted and decompressed HTML format that the web page existed in at the remote server). The protection module then analyzes the web page to determine whether the web page includes links that may lead to sites hosting malware or whether the web page itself includes malware. The analysis of the assembled web page may include communicating with a remote security center so that a malware management analysis may be performed to analyze one or more portions of the content of the assembled web page and/or the protection module itself may perform analysis of content of the assembled webpage. The analyzed webpage can then be forwarded to the web browser for display to a user.

The preceding is a simplified summary of the disclosure to provide an understanding of some aspects of the disclosure. This summary is neither an extensive nor exhaustive overview of the disclosure and its various aspects, embodiments, and/or configurations. It is intended neither to identify key or critical elements of the disclosure nor to delineate the scope of the disclosure but to present selected concepts of the disclosure in a simplified form as an introduction to the more detailed description presented below. As will be appreciated, other aspects, embodiments, and/or configurations of the disclosure are possible utilizing, alone or in combination, one or more of the features set forth above or described in detail below.

BRIEF DESCRIPTION OF THE FIGURES

FIG. **1** illustrates an exemplary embodiment of a computing environment according to an exemplary embodiment.

FIG. **2** illustrates an exemplary embodiment of systems and operations at the remote computer.

FIG. **3** is a flowchart illustrating exemplary process flow at the remote computer.

FIG. **4** illustrates an exemplary embodiment of one of the computers in FIG. **1**.

DETAILED DESCRIPTION

Referring now to the drawings, where like or similar elements are designated with identical reference numerals

throughout the several views, and referring in particular to FIG. **1**, it is a block diagram depicting an environment in which several embodiments of the invention may be implemented. As shown, a security center **102** (also referred to herein as a "central" or "base" computer), remote user **104** operating a remote computer **105**, a malware source **106**, and a web server **108** are all communicatively coupled through one or more networks (e.g., the Internet and/or local or wide area networks) **110** and links **5**. Although only one remote user **104**, remote computer **105**, malware source **106**, web server **108**, and security center **102** are depicted, each of these logically represents a potentially unlimited number of persons, entities and/or computers or computing resources.

The remote user **104** may be an individual or a business enterprise that operates the remote computer **105**, which may each be a personal computer, a server of any type, a PDA, mobile phone, tablet, netbook, an interactive television, or any other device capable of loading and operating computer objects.

In the depicted environment, the malware source **106** generally represents a source of malware that ends up or is strategically placed at the web server **108**, which may or may not be suspected of hosting malware. For example, the malware source **106** may generate a malware object in a variety of forms including in a scripting language such as ECMAscript-based scripting languages (e.g., JavaScript or Adobe Flash), but the malware source may generate other types of objects such as computer files, part of a file or a sub-program, an instruction(s), macro, web page or any other piece of code to be operated by or on the computer, or any other event whether executed, emulated, simulated or interpreted.

As depicted, the security center **102** is disposed and configured to be accessible to the user **104** so that, as discussed further herein, the security center **102** may facilitate the management of malware on the remote computer **104**. In many implementations, the security center **102** operates according to a Software as a Service (SaaS) business model to generally provide Web security services "in the cloud."

As depicted in FIG. **1**, the exemplary security center **102** includes a malware management portion **112** that is coupled to a data store **114**. Although not depicted, the security center **102** may also include components that provide other services (e.g., internet policy enforcement, in/outbound content control, application control, compliance-related services, etc.).

The security center **102** is generally configured to obtain information about malware threats and to be a resource for the remote computer **105** to enable the remote computer to manage malware threats more effectively and efficiently. It should be noted that that the malware management component **112** and data store **114** are presented for convenience as single entities, but the security center **102** can be scaled and comprised of multiple geographically distributed computers and servers, etc., and the data store can be made up multiple databases and storage distributed around this central system and/or be located in a cloud-type environment.

Although not required, the malware management **112** component of the security center **102** may maintain the data store **114** as a community database that is populated, over time, with information relating to each object run on all of the connected remote computers as disclosed in US A 2007/0016953, published 18 Jan. 2007, entitled "METHODS AND APPARATUS FOR DEALING WITH MALWARE," the entire contents of which are hereby incorporated herein by reference. As discussed in the above-identified application, data representative of each malware object may take the form of a so-called signature or key relating to and/or identifying the object, its attributes and/or behavior(s).

In operation, the protection agent **116** in this embodiment operates to analyze threats, at the protocol level (e.g., at the HTML level), by intercepting all requests that the browser engine **118** sends and receives, and the protection agent **116** completes the requests without the help of the browser engine **118**. And then the protection agent **116** analyzes and/or modifies the completed data before the browser engine **118** has access to it. After performing all of its processing, removing, and/or adding any code as needed, the protection agent feeds the HTML content back to the browser engine **118**, and the browser engine **118** receives responses from the protection agent **116** as if it was "speaking" to an actual web server (e.g., web server **108**) when in fact, it is speaking to an analysis engine of the protection agent **116**. This allows the protection agent **116** to have full control over what the browser engine **118** "sees," providing means to remove any exploits, malware, and other threats dynamically. This also enables the protection agent **116** to add content into the browser stream at the HTML level. Stated another way, the protection agent **116** caches web content requested by a browser, analyzes and/or modifies the retrieved web content, and provides a clean or sanitized version of the web content, free of malware, to the browser.

In some optional implementations for example, search engine results (e.g., results provided by Google, Yahoo, and Bing) are annotated by the protection agent **116**—within the HTML code—to denote if a particular website is legitimate or malicious. For example, a legitimate link in the search results may be depicted in connection with a green check mark and a suspect link may be depicted with a red cross. In addition to search result annotation, the protocol-level analysis approach may also be used in connection with anti-phishing and URL analysis among other types of analysis.

The differences between the protocol-level analysis approach disclosed herein as compared to other prior anti-malware approaches are significant. In the context of search result annotation for example, the data (e.g., a web page of search results) is first analyzed and modified by an analysis engine of the protection agent **116**, which has full control over every element of the web page before the page is operated on by the browser engine **118**. This is in contrast to prior approaches that just make high-level modifications to the content after it has been displayed through a Browser Helper Object. With the present protocol-level approach, there is virtually no performance overhead, and in many cases, there is actually a performance improvement when performing the browser content analysis.

When the protection agent **116** receives content from the web server **108**, the protection agent **116** then, if necessary, decrypts and decompresses the web content and then assembles the requested web page (e.g., in a decrypted and decompressed HTML format that the web page existed in at the remote server **108**). The protection agent **116** then analyzes the web page to determine whether the web page includes links that may lead to sites hosting malware or whether the web page itself includes malware. The analysis of the assembled web page may include communicating with the security center **102** so that the malware management component **112** may analyze one or more portions of

the content of the assembled web page and/or the protection agent **116** itself may perform analysis of content of the assembled webpage.

In many embodiments, the cleaning process that the protection agent **116** carries out takes place through a highly optimized routine written in, for example, raw C with an inline assembler reducing processing effort within the browser engine **118** itself by decrypting/decompressing/de-encoding any of the content outside of the browser engine **118**. Having full control at this level also means that complex inferential algorithms can be applied to the browser content as a whole, taking into account any external script/image links, to build an in-memory picture of the final content before it is rendered to the user by the browser on a display (not shown). This general operation can be extended to remove/modify any form of content, whether illicit images, ads, fake password request forms, malicious exploits, cross site scripting attacks (XSS), etc.

Referring to FIG. **2**, shown is a block diagram depicting exemplary functional components and operations that reside and take place on the remote computer **105** at the location of the remote user **104**. In the depiction of FIG. **2**, the browser-related processes and protection-agent-related processes are separated so that the interaction between the two types of processes may be more clearly understood. While referring to FIG. **2**, simultaneous reference is also made to FIG. **3**, which depicts exemplary aspects of the process flow that occurs in connection with the components depicted in FIG. **2**. Depicted next to each of the blocks in FIG. **3** is either a "K" or a "U," which indicate that the operation is implemented at the kernel or user-level, respectively. It should be recognized, however, that the operations described with reference to FIGS. **2** and **3** may be implemented with some kernel-mode operations being implemented at the user-level and vice versa. Moreover, the illustrated arrangement of components in FIGS. **2** and **3** is logical, and is not meant to be an actual hardware diagram. Thus, many of the components can be combined and/or further separated in an actual implementation.

As shown, the browser processes in this embodiment include agent processes **220** that are installed on the remote computer **105** in connection with the protection agent **116** so that functions of the protection agent **116** are integrated with a typical browser engine **218**. In other words, the agent processes **220** of the browser processes are implemented by additional code that is wrapped around a typical browser engine **218** to intercept what is requested and received. As discussed further herein, these agent processes **220** enable all content that is requested by an application **222** (e.g., a browser) and received by the browser engine **218** to be intercepted.

More specifically as shown, when a user initiates a request via the application **222** (e.g., web browser or other application that requests web content), a connection request **360** is initiated as a POST/GET request **362** to a website (e.g., hosted by the webserver **108**) and the analysis engine **224** looks at the context of the request to assess whether the request is the first request in a session (Block **364**), and if the request is the first request, a determination is made as to whether the request is associated with known, malicious content (Block **366**), and if so, the request is blocked (Blocks **368**, **370**).

In some embodiments, the analysis engine **224** accesses the security center **102** via the Internet and the security center **102** is utilized to facilitate whether the request is a request for known malicious content (e.g., the URL of the request may be compared to a black list of URLs). But the analysis engine **224** may also include some malware checking functionality locally. As shown, if the request is not blocked, the request may be pre-processed by the content acquisition component **226** (Block **372**) (e.g., to set aside sufficient memory in RAM in anticipation of the content from the website being received) before the request is sent to the destination website **108**.

And as shown, when the first response is received by the content acquisition component **226** (Block **374**), if the response is not complete, the response is stored in memory (Block **376**), and the next request is sent to the destination website **108** (Block **378**). In this way, the content acquisition component **226** continues to obtain content from the website **108** (Blocks **374**, **376**, **378**) until the web page is complete, and the complete page is held in memory by the content acquisition component **226**.

Thus, in short, the initial request **360** by the application (e.g., browser) is intercepted and if the request does not appear to be a request for malicious content, the content acquisition component **226** iteratively sends requests and receives content (Blocks **374**, **376**, **378**) from the website **108** until the requested content has been completely received. This is very different from the ordinary operation of the browser engine **218**, which would (if unaltered by the protection agent **116**) obtain the content from the webserver **108** itself by way of a series of GET requests.

As shown in FIG. **2**, once the web content is completely gathered by the content acquisition component **226**, the content is passed along to the service process component **228** via the user process component **230**, and the data is then preprocessed (Block **380**), if necessary, to decrypt, remove chunks, and decompress the gathered content. When a web page is received from the webserver **108**, it may be compressed, chunked encoded, and encrypted, and decompressing, decoding and unencrypting the content enables a complete picture of the webpage looked like before if any of the these forms of obfuscation were applied to the content by the webserver **108**. One of ordinary skill in the art will appreciate that various layers of encoding and compression may be applied to content, but for simplicity, these known details are not included herein for clarity.

And after the data is decrypted and decompressed, the data is in an HTML format, so at this point, the protection has the content in the same HTML format that the website **108** had the content in. And at this point the browser engine **218** is unaware that the requested content (e.g., an entire webpage) has been received. As a consequence, the browser page may be analyzed by the protocol-level analysis/modification component **382** depicted in FIG. **3** outside of the context of the browser engine **218** before the browser engine **218** has operated on the webpage. For example, all the links in the webpage, all the pictures in the webpage, and any scripts in the webpage may be analyzed at a low level of granularity. And code may be changed, enhanced, and removed according to default modes and/or user-configurable mode of operation. As one particular example, if a malicious script is found, it may be commented out before the content is handed to the browser engine.

In addition, in many modes of operation the protocol-level analysis/modification component **382** modifies and/or annotates the content—as HTML within the content—to provide the user with textual, audible and/or graphical indicators of risk associated with the content. In the context of a webpage that include search results from a user's search query, for example, a green check may be added within the HTML code next to a result that is a low risk link, and a red X may be added within the HTML code next to a result the

is a high risk link. For example, an image tag and/or text may be added within the HTML to indicate a risk of being exposed to malware. Beneficially, making modifications to the content (e.g., to include risk indicators) at the HTML level avoids having to address the differences that different browsers introduce into the presentation of rendered content.

This approach of modifying content at the HTML level (e.g., to add annotations) is very different than the prior approaches of rendering (e.g., using a Browser Helper Object) annotations on top of a page that has already been parsed and rendered by a browser engine. This prior approach is problematic because it allows the browser engine to potentially execute malicious scripts or perform malicious actions while it is parsing and rendering the code. And in addition, because the annotations are added after rending, the annotation process must account for the rendering differences (e.g., differences in how and where content is displayed) that different browsers (e.g., Firefox, Safari, Chrome, Internet Explorer, etc.) exhibit.

It should be recognized that although search results annotation is one application for the protocol-level (e.g., HTML level) handling of content, it is certainly contemplated that there are other anti-malware-related applications of the protocol-level handling of content. One application for example, is an anti-phishing application, which may automatically modify HTML data so that user need not type in a password and expose the password to a keylogger object.

As shown in FIG. **2**, more details of the protocol-level analysis/modification component **382** are depicted. As shown, once a complete collection of content is received, the service process **228** is utilized to initiate parsing of the HTML content and extraction of any scripts **230** in the content before building a tree representation of the content **234**, which is then asynchronously analyzed **236**. As shown in FIG. **2** for example, the analysis may include URL analysis **238**, IP analysis **240**, image analysis (e.g., to analyze phishing threat) **242**, and script/HTML analysis **244**. The results of the analysis are then aggregated by the result aggregator **246** so that cloud verification **248** may be performed on the single aggregated collection of data by the base computer **112** of the security center **102** to determine malware vulnerabilities.

But in optional embodiments, the verification may be effectuated locally or in combination with the security center **102**. As shown, the results of the cloud verification are provided to the final content packager **250** of the service process **228**, and then in some modes of operation, annotations are performed **252** on the packaged HTML code before being distributed **254** back to the data wrapper component **256**, which sits on top of the response receive component of the browser engine **218** and performs final context modifications **258** to ensure content is properly displayed before being passed to the browser engine **218**. As shown, final content modifications are performed at **258** before the final content **260** is provided to the browser engine **218** for parsing and rendering of the HTML content before being displayed by the application **222**.

Referring again to FIG. **3**, shown is a loop (Blocks **384**, **386**, **364**) that depicts the cleaned and/or annotated content being fed back to the browser engine **218**. As shown, as the browser engine requests more content, the pointer is set to the next byte to be read (Block **384**), and once all the content is on hand, the browser engine requests content (Block **386**), and the cleaned and/or annotated content is fed to the browser engine **218** in the manner the browser engine **218** would have requested and received the content.

In other words, from the browser engine's **218** perspective, the browser engine **218** is obtaining the content from the webserver **108** directly. In other words, the protection agent **116** operates as an emulated server in memory to provide the clean content in the way the web server **108** would have provided the content to the browser engine **118**, **218** if the browser received the content form the web server **108** directly.

In the exemplary embodiment however, for operational speed, the clean/modified content is not encoded or encrypted once it is cleaned. As a consequence, the packet headers (e.g., length headers) are modified to reflect that the content being provided to the browser engine **118**, **218** is in a decrypted, decompressed, and/or de-chunked format. In this way, the browser is not expecting to receive encrypted, encoded, compressed content. Beneficially, the decryption and decompression is carried out with code implemented in raw C (and most browsers are written in a high level language), so the decryption and decompression are actually carried out faster than an ordinary browser would do so. Additionally, the decryption and decoding is generally carried out after the request is complete, so the decryption and decompression is more efficient than handling the decryption and decompression over several passes.

Referring next to FIG. **4**, shown is a block diagram depicting hardware components in an exemplary embodiment of the protected computer described with reference to FIG. **1**. As shown, the N processing components **140** described with reference to FIG. **1** are depicted as N processors **440** that are coupled to a bus **460**, and also coupled to the bus **460** are a memory **438** (corresponding to memory **138**), storage medium **412** (corresponding to the storage medium **112**), a keyboard/pointing device **462**, a display/graphics adapter **464**, and a network interface **466**. In addition, a display **468** is coupled to the display/graphics adapter **464**.

The storage medium **412** may be any device capable of holding substantial amounts of data, such as a hard drive, flash memory, or some other form of fixed or removable storage device. And the storage medium **412** in this embodiment stores processor-readable code with instructions to effectuate the functions described herein (e.g., the functions of the components in FIG. **1** depicted in the user **102** and kernel **104** environments). The processors **440** generally function to execute code and process other information that resides in memory and may be any specific or general-purpose processor such as an INTEL x86 or POWERPC-compatible central processing unit (CPU), and each may include one or multiple (e.g., four) cores. The memory **438** may include several gigabytes of random access memory, but this is merely exemplary and other memory types and sizes may be utilized. As one of ordinarily skill will appreciate, an operating system (e.g., LINUX® or WINDOWS®) may also reside in the storage medium **412** and memory **438** and function (e.g., when executed by one or more of the processors **440**) to enable the components to operate as described with reference to FIG. **1**.

As one of ordinary skill in the art in light of this disclosure will appreciate, FIG. **4** depicts only an exemplary embodiment, and the processes presented herein are not inherently related to any particular computing device or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct a more specialized apparatus to perform the desired method. In addition, embodiments of the present invention are not described with reference to any particular programming language. It will be appreciated that

a variety of programming languages may be used to implement the teachings of the invention as described herein. In addition, it should be understood that operations, capabilities, and features described herein may be implemented with any combination embodied in firmware, software, application-specific integrated circuits (ASICs), and/or programmable logic devices.

It is to be appreciated that a lesser or more equipped computer system than the example described above may be desirable for certain implementations. Therefore, the configuration of the system illustrated in the figure can vary from implementation to implementation depending upon numerous factors, such as its intended use, price constraints, performance requirements, storage requirements, technological improvements, and/or other circumstances, or the like.

It should also be noted that while the embodiments and methods described herein may be performed and used with a computer similar to the one described herein, other embodiments and variations can be used with computer that vary from the described example. Therefore, nothing disclosed herein concerning the configuration of the illustrated computer should be construed as limiting the disclosure to a particular embodiment wherein the recited operations are performed by a specific combination of hardware components.

The various embodiments and variations thereof illustrated in the accompanying Figures and/or in the totality of this document are merely exemplary and are not meant to limit the scope of the invention. It is to be appreciated that numerous variations of the invention have been contemplated as would be obvious to one of ordinary skill in the art with the benefit of this disclosure. Additionally, while certain features may be categorized under one or more headings to assist with readability, it is to be appreciated that the feature(s) described under a particular heading may be used in associating with other portions of the specification and/or feature(s) described herein. Similarly, while certain embodiments are discussed in relation to specific languages, it is to be appreciated that the techniques disclosed herein can be used with any software language(s).

While the above described methodology has been discussed in relation to a particular sequence of events, it should be appreciated that minor changes to this sequence can occur without materially effecting the operation of the invention.

The above-described system and methodology, as has been indicated herein, can be implemented on a computing device, such as a personal computer, server, dedicated computing device, distributed processing system, in a cloud, or the like, or a separately programmed general purpose computer. Additionally, the systems and methods of this invention can be implemented on a special purpose computer, a programmed microprocessor or microcontroller and peripheral integrated circuit element(s), an ASIC or other integrated circuit, a digital signal processor, a hard-wired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FPGA, PAL, or the like, in fuzzy logic, artificial intelligence and/or neural networks. In general, any device(s) or module, which can be any combination of hardware and/or software, capable of implementing a state machine that is in turn capable of implementing the processes described herein can be used to implement this invention.

Furthermore, the disclosed methods may readily implemented in software using, for example, object or object-oriented software development environments that provide portable source code that can be used on a variety of computer or workstation and/or server platforms. The software can be stored on a non-transitory computer-readable medium, with the software including one or more processor executable instructions. The disclosed system and methodology may also be implemented partially or fully in hardware using standard logic circuits or, for example, a VLSI design. Whether software or hardware is used to implement the systems in accordance with this invention is dependent on the speed and/or efficiency requirements of the system, the particular function, and the particular software or hardware systems or microprocessor or microcomputer systems being utilized. The systems and methods illustrated herein can be readily implemented in hardware and/or software using any suitable systems, means, structures, devices and/or the functionality stored on an appropriate information storage medium, by those of ordinary skill in the applicable art from the functional description provided herein and with a basic general knowledge of the computer and software arts.

While the embodiments illustrated herein may show some of the various components collocated, it is to be appreciated that the various components of the system can be located at distant portions of a distributed network, such as a communications network and/or the Internet and/or within a dedicated network. Thus, it should be appreciated that the various components can be combined into one or more devices or collocated on a particular node of a distributed network and/or in a cloud. As will be appreciated from the description, and for reasons of computational efficiency, the components can be arranged at any location within a distributed network without affecting the operation of the system.

Furthermore, it should be appreciated that various links connecting elements can be wired or wireless links, or a combination thereof, or any known or later developed element(s) that is capable of supplying and/or communicating data to and from the elements.

The present disclosure, in various aspects, embodiments, and/or configurations, includes components, methods, processes, systems and/or apparatus substantially as depicted and described herein, including various aspects, embodiments, configurations embodiments, subcombinations, and/or subsets thereof. Those of skill in the art will understand how to make and use the disclosed aspects, embodiments, and/or configurations after understanding the present disclosure. The present disclosure, in various aspects, embodiments, and/or configurations, includes providing devices and processes in the absence of items not depicted and/or described herein or in various aspects, embodiments, and/or configurations hereof, including in the absence of such items as may have been used in previous devices or processes, e.g., for improving performance, achieving ease and\or reducing cost of implementation.

The foregoing discussion has been presented for purposes of illustration and description. The foregoing is not intended to limit the disclosure to the form or forms disclosed herein. In the foregoing Detailed Description for example, various features of the disclosure are grouped together in one or more aspects, embodiments, and/or configurations for the purpose of streamlining the disclosure. The features of the aspects, embodiments, and/or configurations of the disclosure may be combined in alternate aspects, embodiments, and/or configurations other than those discussed above. This method of disclosure is not to be interpreted as reflecting an intention that the claims require more features than are expressly recited in each claim. Rather, as the following

claims reflect, inventive aspects lie in less than all features of a single foregoing disclosed aspect, embodiment, and/or configuration. Thus, the following claims are hereby incorporated into this Detailed Description, with each claim standing on its own as a separate exemplary, and separately claimable, embodiment of the disclosure.

While exemplary aspects have been described in conjunction with a number of embodiments, it is evident that many alternatives, modifications and variations would be or are apparent to those of ordinary skill in the applicable arts. Accordingly, this disclosure is intended to embrace all such alternatives, modifications, equivalents and variations that are within the spirit and scope of this disclosure.

The invention claimed is:

1. A content analysis and malware prevention method comprising:

intercepting, by a protection agent, a request from a web browser;

determining, by the protection agent, whether the request is associated with known malicious content;

when the request is associated with known malicious content, blocking, by the protection agent, the request; and

when the request is not associated with known malicious content:

sending, by the protection agent, the request to one or more web servers;

receiving, at the protection agent on a remote computer, web content from the one or more web servers, wherein web content comprises data for assembling a web page, and wherein the web content is received in response to the request;

identifying, by the protection agent, a malware threat within the web content;

in response to identifying the malware threat, modifying, by the protection agent, the web content, wherein modifying the web content comprises modifying the web content at protocol level to remove the malware threat from the web content;

assembling, by the protection agent, a modified version of the web page, wherein the modified web page is assembled using the modified web content, and wherein the modified version of the web page does not comprise the malware threat; and

providing, by the protection agent, the modified version of the web page to the web browser application on the remote computer for rendering and display.

2. The method of claim **1**, further comprising supplementing, by the protection agent, the web content with indicators regarding a vulnerability of one or more links within the web content.

3. The method of claim **1**, wherein an analysis to identify a malware threat within the web content includes one or more of a URL (Uniform Resource Locator) analysis, an IP (Internet Protocol) analysis, an image analysis, and a script/HTML analysis.

4. The method of claim **1**, further comprising determining, by the protection agent, whether a request for the web content from the web browser application is a first request.

5. The method of claim **1**, further comprising performing one or more of pre-process data decryption, de-chunking and decompressing.

6. The method of claim **1**, wherein a protocol-level analysis is utilized to identify the malware threat within the web content.

7. The method of claim **3**, further comprising performing cloud verification of the web content, wherein cloud verification determines one or more malware vulnerabilities based on an aggregation of the analyses.

8. The method of claim **1**, further comprising allocating memory for caching the web content.

9. A content analysis and malware prevention system comprising:

at least one processor; and

memory encoding computer executable instructions that, when executed by the at least one processor, perform a method comprising:

intercepting a request from a web browser;

determining whether the request is associated with known malicious content;

when the request is associated with known malicious content, blocking, by a protection agent, the request; and

when the request is not associated with known malicious content:

sending the request to one or more web servers;

receiving web content from one or more web servers, wherein web content comprises data for assembling a web page;

identifying a malware threat within the web content;

in response to identifying the malware threat, modifying the web content, wherein modifying the web content comprises modifying the web at protocol level to remove the malware threat from the web content;

assembling a modified version of the web page, wherein the modified web page is assembled using the modified web content, and wherein the modified version of the web page does not comprise the malware threat; and

providing the modified version of the web page to a web browser application on the remote computer for rendering and display.

10. The system of claim **9**, wherein the method further comprises supplementing the web content with indicators regarding a vulnerability of one or more links within the web content.

11. The system of claim **9**, wherein the method further comprises identifying the malware threat within the web content by performing one or more of a URL (Uniform Resource Locator) analysis, an IP (Internet Protocol) analysis, an image analysis, and a script/HTML analysis.

12. The system of claim **9**, wherein the method further comprises determining whether a request for the web content from the web browser application is a first request.

13. The system of claim **9**, wherein the method further comprises performing one or more of pre-process data decryption, de-chunking, and decompressing.

14. The system of claim **9**, wherein the method further comprises identifying the malware threat utilizing a protocol-level analysis.

15. The system of claim **9**, wherein the method further comprises caching the web content in allocated memory.

16. The method of claim **1**, wherein the malware threat comprises a Uniform Resource Locator (URL).

17. The system of claim **9**, wherein the method further comprises determining if a malware threat comprises a Uniform Resource Locator (URL).

18. The method of claim **1**, wherein sending the request to one or more web servers comprises iteratively sending a plurality of requests to the one or more web servers.

19. The method of claim **1**, further comprising processing the web content prior to identifying the malware threat within the web content.

20. The system of claim **9**, wherein sending the request to one or more web servers comprises iteratively sending a plurality of requests to the one or more web servers.

* * * * *

Exhibit 6

US011409869B2

(12) **United States Patent**
Schmidtler et al.

(10) **Patent No.: US 11,409,869 B2**
(45) **Date of Patent: *Aug. 9, 2022**

(54) **AUTOMATIC THREAT DETECTION OF EXECUTABLE FILES BASED ON STATIC DATA ANALYSIS**

(71) Applicant: **Webroot Inc.**, Broomfield, CO (US)

(72) Inventors: **Mauritius Schmidtler**, Escondido, CA (US); **Gaurav Dalal**, San Jose, CA (US); **Reza Yoosoofmiya**, San Diego, CA (US)

(73) Assignee: **Webroot Inc.**, Broomfield, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **16/791,649**

(22) Filed: **Feb. 14, 2020**

(65) **Prior Publication Data**

US 2020/0184073 A1 Jun. 11, 2020

Related U.S. Application Data

(63) Continuation of application No. 14/709,875, filed on May 12, 2015, now Pat. No. 10,599,844.

(51) **Int. Cl.**
G06F 21/56 (2013.01)
G06N 5/04 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06F 21/565*** (2013.01); ***G06F 21/562*** (2013.01); ***G06N 20/00*** (2019.01);
(Continued)

(58) **Field of Classification Search**
CPC G06F 21/51; G06F 21/53; G06F 21/562; G06F 21/56; G06F 21/565; G06F 21/566;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,161,548 B1 * 4/2012 Wan G06K 9/6228 726/22
8,510,836 B1 * 8/2013 Nachenberg H04L 63/145 726/23

(Continued)

OTHER PUBLICATIONS

Bai, Jinrong, Junfeng Wang, and Guozhong Zou. "A malware detection scheme based on mining format information." The Scientific World Journal 2014 (2014). (Year: 2014).*

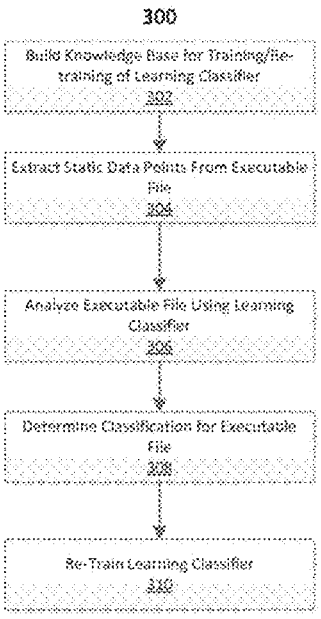
(Continued)

Primary Examiner — Michael Simitoski
(74) *Attorney, Agent, or Firm* — Sprinkle IP Law Group

(57) **ABSTRACT**

Aspects of the present disclosure relate to threat detection of executable files. A plurality of static data points may be extracted from an executable file without decrypting or unpacking the executable file. The executable file may then be analyzed without decrypting or unpacking the executable file. Analysis of the executable file may comprise applying a classifier to the plurality of extracted static data points. The classifier may be trained from data comprising known malicious executable files, known benign executable files and known unwanted executable files. Based upon analysis of the executable file, a determination can be made as to whether the executable file is harmful.

59 Claims, 4 Drawing Sheets



(51) **Int. Cl.**

G06N 20/10	(2019.01)
G06N 20/00	(2019.01)
G06F 11/30	(2006.01)
G06F 8/53	(2018.01)

(52) **U.S. Cl.**
CPC ***G06N 20/10*** (2019.01); *G06F 8/53* (2013.01); *G06F 2221/033* (2013.01)

(58) **Field of Classification Search**
CPC G06F 2221/033; G06F 8/53; G06N 20/00; G06N 20/10; H04L 63/145
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

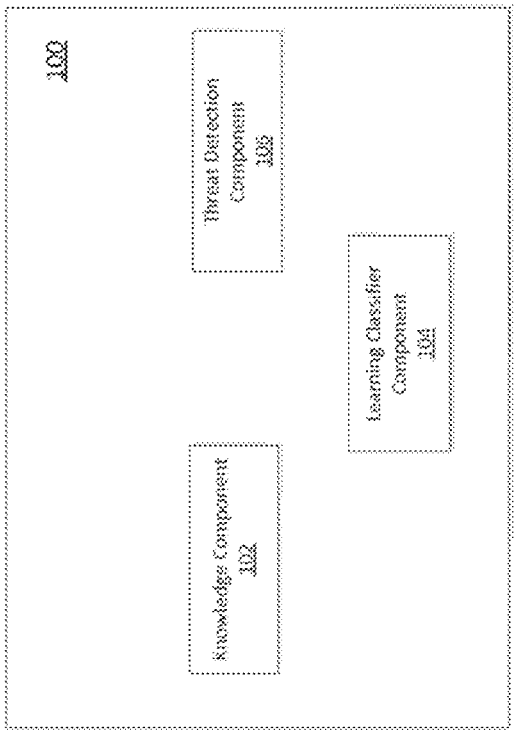
8,719,924	B1	5/2014	Williamson et al.	
9,197,663	B1 *	11/2015	Gilbert	G06F 21/552
9,465,940	B1 *	10/2016	Wojnowicz	G06F 21/565
10,116,688	B1 *	10/2018	Yun	H04L 63/145
10,599,844	B2	3/2020	Schmidtler et al.	
2006/0026675	A1 *	2/2006	Cai	G06F 21/562 726/22
2006/0037080	A1 *	2/2006	Maloof	G06F 21/562 726/24
2007/0240220	A1	10/2007	Tuvell et al.	
2009/0013405	A1 *	1/2009	Schipka	G06F 21/562 726/22
2009/0274376	A1 *	11/2009	Selvaraj	G06K 9/6269 382/224
2009/0300765	A1 *	12/2009	Moskovitch	G06K 9/6256 726/24
2010/0082642	A1 *	4/2010	Forman	G06F 16/35 707/749
2010/0192222	A1	7/2010	Stokes et al.	
2010/0293273	A1 *	11/2010	Basarrate	H04L 61/2567 709/224
2011/0154490	A1 *	6/2011	DeLuca	G06F 21/56 726/23
2011/0172504	A1 *	7/2011	Wegerich	G16H 50/30 600/301
2012/0079596	A1 *	3/2012	Thomas	G06F 21/566 726/24
2012/0227105	A1 *	9/2012	Friedrichs	G06F 21/564 726/22
2013/0291111	A1 *	10/2013	Zhou	G06F 21/562 726/23
2014/0165203	A1	6/2014	Friedrichs et al.	
2014/0310517	A1 *	10/2014	Christodorescu	G06N 20/00 713/160
2015/0213365	A1 *	7/2015	Ideses	H04L 67/125 706/12
2016/0021174	A1 *	1/2016	De Los Santos Vilchez	H04W 12/128 709/201
2016/0156646	A1 *	6/2016	Hsueh	G06N 20/00 726/1
2016/0225017	A1 *	8/2016	Wong	G06Q 30/0247
2016/0335435	A1	11/2016	Schmidtler et al.	
2017/0262633	A1 *	9/2017	Miserendino	G06F 21/564
2018/0365573	A1 *	12/2018	Sultana	G06N 20/00

OTHER PUBLICATIONS

Office Action issued for U.S. Appl. No. 14/709,875, dated Dec. 8, 2016, 19 pages.
Shabtai et al., "Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-of-the-Art Survery," Information Security Technical Report, Elsevier Advanced Technology, Amsterdam NL, vol. 14, No. 1, Feb. 1, 2009, pp. 16-29, XP026144093, ISSN: 1363-4127.
Office Action issued for U.S. Appl. No. 14/709,875, dated Mar. 24, 2017, 21 pages.
Office Action issued for U.S. Appl. No. 14/709,875, dated Apr. 20, 2018, 22 pages.
Office Action issued for U.S. Appl. No. 14/709,875, dated Jan. 24, 2019, 21 pages.
Office Action issued for U.S. Appl. No. 14/709,875, dated Jul. 17, 2019, 12 pages.
Bost, et al., "Machine learning classification over encrypted data." N DSS. vol. 4324. 2015. (Year: 2015), 14 pages.
Choi, et al., "Efficient Malicious Code Detection Using N-Gram Analysis and SVM," 2011 14th International Conference on Network-Based Information Systems. IEEE, 2011, 4 pages.
Devi, Dhruwajita et al., "Detection of Packed Malware," Proceedings of the First International Conference on Security of Internet of Things, ACM, 2012, 5 pages.
Dube, et al., "Malware Target Recognition Via Static Heuristics," Computers & Security 31.1, 2012, 11 pages.
Merkel, et al., Statistical Detection of Malicious Pe-Executables for Fast Offline Analysis, IFIP International Conference on Communications and Multimedia Security, Springer, Berlin, 2010, 12 pages.
Singla, et al., "A Novel Approach to Malware Detection Using Static Classification," International Journal of Computer Science and Information Security 13.3, 2015, 5 pages.
Tabish, et al., "Malware Detection Using Statistical Analysis of Byte-Level File Content," Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, ACM, 2009, 9 pages.
Treadwell, et al., A Heuristic Approach for Detection of Obfuscated Malware, 2009 IEEE International Conference on Intelligence and Security Informatics, 2009, 9 pages.
Notice of Allowance issued for U.S. Appl. No. 14/709,875, dated Nov. 14, 2019, 11 pages.
Corrected Notice of Allowance issued for U.S. Appl. No. 14/709,875, dated Dec. 23, 2019, 8 pages.
Corrected Notice of Allowance issued for U.S. Appl. No. 14/709,875, dated Jan. 24, 2020, 8 pages.
International Patent Application No. PCT/US2016/032090, International Search Report and Written Opinion dated Jul. 29, 2016, 13 pages.
Perdisci et al., "Classification of Packed Executables for Accurate Computer Virus Detection," Pattern Recognition Letters 29, 2008, 6 pages.
Perdisci et al., "McBoost: Boosting Scalability in Malware Collection and Analysis Using Statistical Classification of Executables," Annual Computer Security Applications, 2008, 10 pages.
Khan, et al., "Determining Malicious Executable Distinguishing Attributes and Low-Complexity Detection", J. ComputVirol, 7.2 (2011), 11 pgs.
Kolter, et al., "Leaming to Detect and Classift Malicious Executables in the Wild", Journal of Machine Learning Research 7 (2006), 24 pages.

* cited by examiner

FIG. 1


100
Threat Detection Component 108
Learning Classifier Component 104
Knowledge Component 102

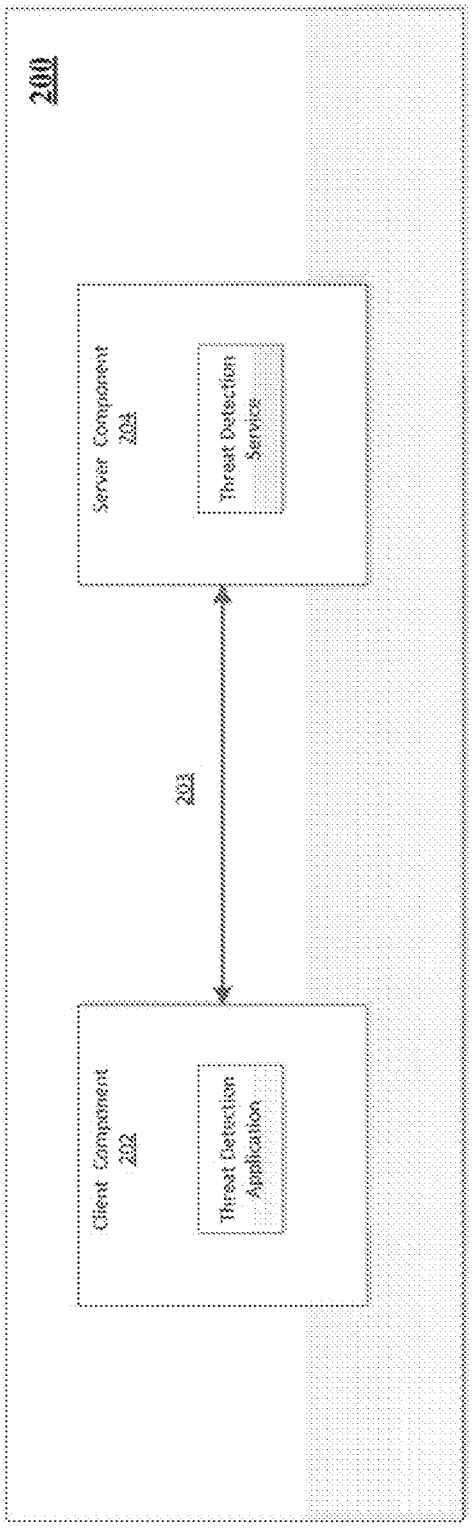
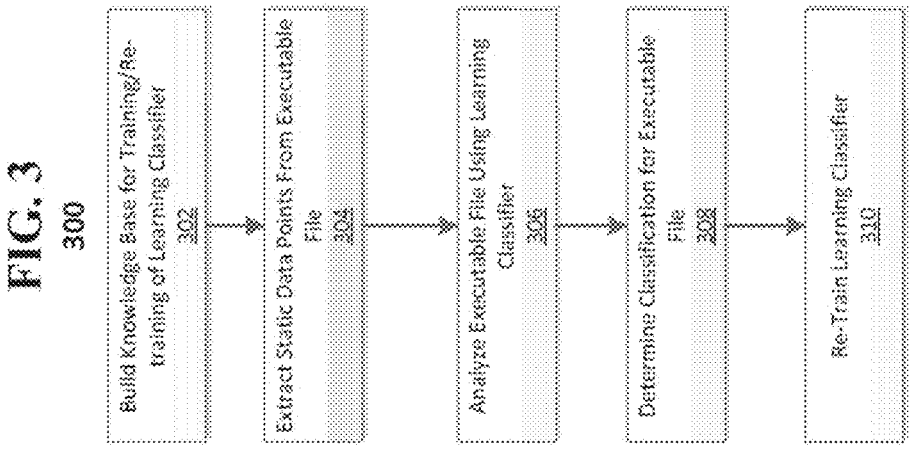

200
Server Component
204
Threat Detection
Service
203
Client Component
202
Threat Detection
Application

FIG. 2

FIG. 3


300
Build Knowledge Base for Training/Re-training of Learning Classifier
302
Extract Static Data Points From Executable File
304
Analyze Executable File Using Learning Classifier
306
Determine Classification for Executable File
308
Re-Train Learning Classifier
310

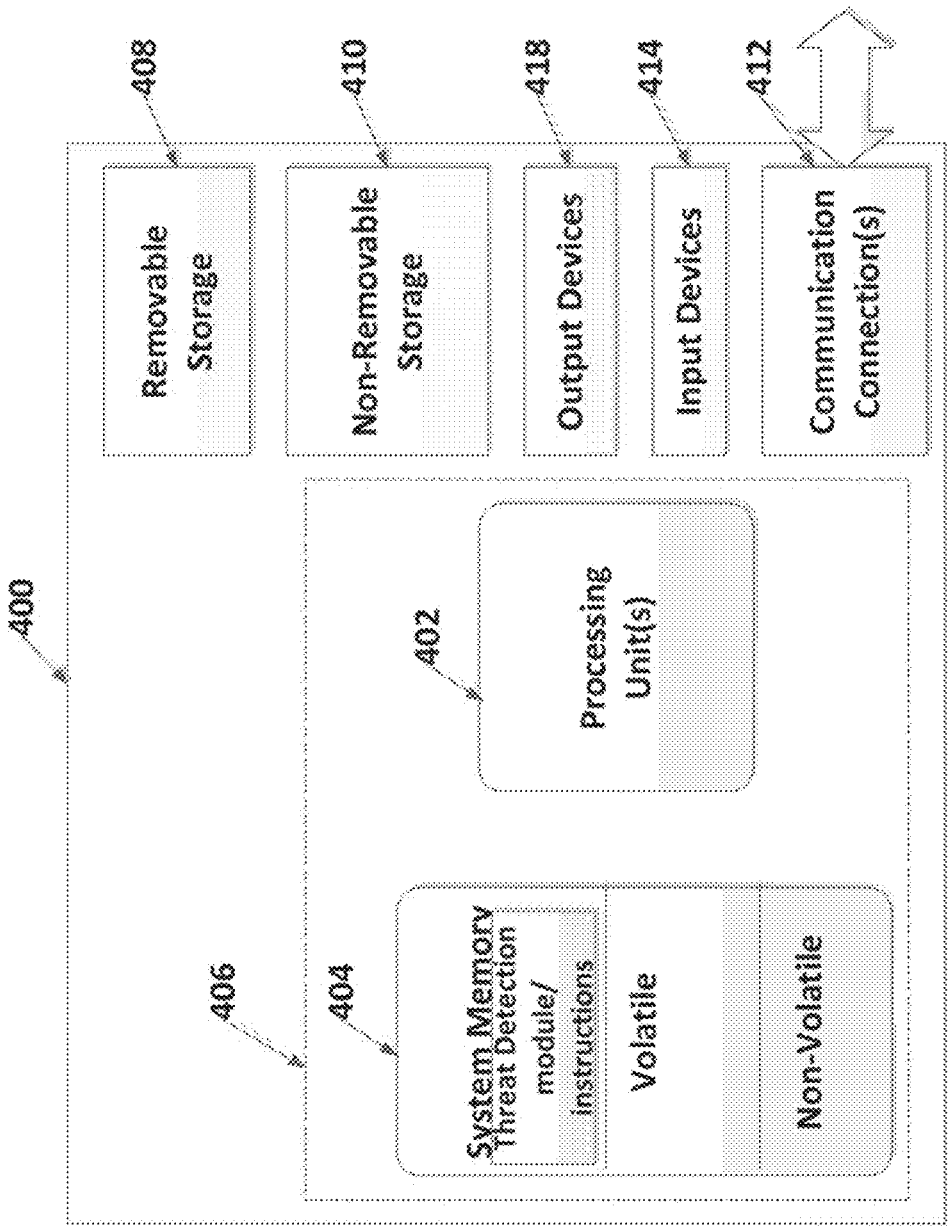

400
402
Processing Unit(s)
404
406
System Memory
Threat Detection module/ instructions
Volatile
Non-Volatile
408
Removable Storage
410
Non-Removable Storage
418
Output Devices
414
Input Devices
412
Communication Connection(s)

FIG. 4

AUTOMATIC THREAT DETECTION OF EXECUTABLE FILES BASED ON STATIC DATA ANALYSIS

CROSS-REFERENCE TO RELATED APPLICATION(S)

This application is a continuation of, and claims a benefit of priority from U.S. patent application Ser. No. 14/709,875, filed May 12, 2015, issued as U.S. Pat. No. 10,599,844, entitled "AUTOMATIC THREAT DETECTION OF EXECUTABLE FILES BASED ON STATIC DATA ANALYSIS," which is fully incorporated by reference herein.

BACKGROUND

Everyday new executable files are created and distributed across networks. A large portion of these distributed executable files are unknown. For instance, it is not known if such distributed executable files are malicious or not. Given the high volume of new unknown files distributed on a daily basis, it is important to determine threats contained in the set of new unknown files instantaneously and accurately. It is with respect to this general environment that aspects of the present technology disclosed herein have been contemplated.

SUMMARY

Aspects of the present disclosure relate to threat detection of executable files. A plurality of static data points are extracted from an executable file without decrypting or unpacking the executable file. The executable file may then be analyzed without decrypting or unpacking the executable file. Analyzing of the executable file comprises applying a classifier to the plurality of static data points extracted from the executable file. The classifier is trained from data comprising known malicious executable files, known benign executable files and potentially unwanted executable files. Based upon the analysis of the executable file, a determination is made as to whether the executable file is harmful. In some examples, execution of the executable file is prevented when a determined probability value that the executable file is harmful exceeds a threshold value.

This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

Non-limiting and non-exhaustive examples are described with reference to the following figures. As a note, the same number represents the same element or same type of element in all drawings.

FIG. **1** illustrates an exemplary system **100** showing interaction of components for implementation threat detection as described herein.

FIG. **2** illustrates an exemplary distributed system **200** showing interaction of components for implementation of exemplary threat detection as described herein.

FIG. **3** illustrates an exemplary method **300** for implementation of threat detection systems and methods described herein.

FIG. **4** illustrates one example of a suitable operating environment **400** in which one or more of the present examples may be implemented.

Non-limiting examples of the present disclosure relate to the threat detection of executable files. The examples disclosed herein may also be employed to detect zero day threats from unknown executable files. However, one skilled in the art will recognize that the present disclosure is not limited to detection of executable files that may present zero day threats and can be applicable to any unknown file that is attempting to execute on a system (e.g., processing device). The present disclosure is able to detect whether an executable file is harmful or benign before the executable file is actually executed on a processing device. In examples, machine learning processing applies a classifier to evaluate an executable file based on static points data collected from the executable file. The classifier is trained from a collection of data comprising known malicious files, potentially unwanted files and benign files. The classifier is designed and trained such that it can handle encrypted and/or compressed files without decrypting and/or decompressing the files.

Approaches to detecting threats typically focus on finding malicious code blocks within a file and analyzing the behavior of the file. Such approaches are expensive and time-consuming operations that require decrypting encrypted files, dissembling code, analyzing behavior of malware, among other things. Additionally, behavioral detection requires the execution of the potentially malicious code, thereby presenting an opportunity for the malicious code to harm the computing system is executing on. The present disclosure achieves high and accurate classification rates for potentially malicious executables without the need to employ time consuming processing steps like decryption, unpacking or executing unknown files while maintaining a controlled and safe environment. The determination of unknown files is achieved by evaluating static data points extracted from an executable file using a trained classification system that is able to identify potential threats without analyzing executable behavior of a file. The present disclosure also provides for the creation of a training set that adaptively learns what static data points may indicate the existence of malicious code using training data that contains a statistically significant number of both encrypted and non-encrypted examples of malicious or unwanted executable files, among other information. By collecting a large set of training examples as they appear as publicly available and distributed over network connection (e.g., "in the wild"), the present disclosure ensures that its adaptive learning processing is robust enough to comprise sufficient representation of features (and distributions) from files that may be encrypted, not-encrypted, compressed, and uncompressed, among other examples.

A number of technical advantages are achieved based on the present disclosure including, but not limited to: enhanced security protection including automatic detection of threats, reduction or minimization of error rates in identification and marking of suspicious behavior or files (e.g., cut down on the number of false positives), ability to adapt over time to continuously and quickly detect new threats or potentially unwanted files/applications, improved efficiency in detection of malicious files, and improved usability and interaction for users by eliminating the need to continuously check for security threats, among other benefits that will be apparent to one of skill in the art.

FIG. **1** illustrates an exemplary system **100** showing an interaction of components for implementation threat detec-

tion as described herein. Exemplary system **100** may be a combination of interdependent components that interact to form an integrated whole for execution of threat detection and/or prevention operations. Components of the systems may be hardware components or software implemented on and/or executed by hardware components of the systems. In examples, system **100** may include any of hardware components (e.g., used to execute/run operating system (OS)), and software components (e.g., applications, application programming interfaces, modules, virtual machines, runtime libraries, etc.) running on hardware. In one example, an exemplary system **100** may provide an environment for software components to run, obey constraints set for operating, and/or makes use of resources or facilities of the system **100**, where components may be software (e.g., application, program, module, etc.) running on one or more processing devices. For instance, threat detection operations (e.g., application, instructions, modules, etc.) may be run on a processing device such as a computer, a client device (e.g., mobile processing device, laptop, smartphone/phone, tablet, etc.) and/or any other electronic devices, where the components of the system may be executed on the processing device. In other examples, the components of systems disclosed herein may be spread across multiple devices. For instance, files to be evaluated may be present on a client device and information may be processed or accessed from other devices in a network such as, for example, one or more server devices that may be used to perform threat detection processing and/or evaluation a file before execution of the file by the client device.

As one example, system **100** comprises a knowledge component **102**, a learning classifier component **104**, and a threat determination component **106**, each having one or more additional components. The scale of systems such as system **100** may vary and include more or less components than those described in FIG. **1**. In alternative examples of system **100**, interfacing between components of the system **100** may occur remotely, for example where threat detection processing is implemented on a first device (e.g., server) that remotely monitors and controls process flow for threat detection and prevention of a second processing device (e.g., client).

As an example, threat detection may detection exploits that are executable files. However, one skilled in the art will recognize that the descriptions herein referring to executable files are just an example. Threat detection examples described herein can relate to any computer file. In one example, an executable file may be a portable executable (PE) file that is a file format for executables, object code, dynamic link library files (DLLs), and font library files, among other examples. However, one skilled in the art will recognize that executable files are not limited to PE files and can be any file or program that executes a task according to an encoded instruction. Knowledge component **102** described herein may collect data for use in building, training and/or re-training a learning classifier to evaluate executable files. The knowledge component **102** is one or more storages, memories, and/or modules that continuously collects and manages data that may be used to detect threats in files such as executable files. In one example, the knowledge component **102** maintains a robust collection of data comprising known malicious executable files, known benign executable files, and potentially unwanted executable files. As an example, Malicious executable files may be any type of code, data, objects, instructions, etc., that may cause harm or alter an intended function of a system and/or any system resources (e.g., memory, processes, peripherals, etc.) and/or applications running on a device such as an operating system (OS). A benign executable file is a file that upon execution, will not cause harm/damage or alter an intended system function. In other examples, a benign executable may cause harm/damage or alter an intended system; however, the potential harm/damage or alteration may be acceptable to the owner of the system or device. In examples, potentially unwanted executable files are files that may be installed on system **100** that may not be malicious, but a user of the device may not want such a file to execute and/or a file that is executed/installed unknowingly to a user. In one example, classification of executable files as malicious, benign or potentially unwanted are done by research and development support associated with developments and programming of a threat detection application/module. However, one skilled in the art will recognize that identification and classification of files into the above identified categories may be done by monitoring or evaluating a plurality of resources including but not limited to: network data, executable file libraries and information on previously known malicious executable files as well as benign executable files and potentially unwanted executable files, users/customers of threat detection/computer security software, network flow observed from use of threat detection processing and products, business associations (e.g., other existing threat detection services or partners), third-party feeds, and updates from threat detection performed using learning classifier of present disclosure, among other examples.

To classify executable files into one of the above identified categories, the knowledge component **102** collects static data on a large variety of executable files (e.g., PE files). Examples of different types of executable files collected and evaluated include but are not limited to: bit files (e.g., 32/64 bit files), operating system files (e.g., Windows, Apple, Linux, Unix, etc.), custom built files (e.g., internal tool files), corrupted files, partial downloaded files, packed files, encrypted files, obfuscated files, third party driver files, manually manipulated binary files, unicode files, infected files, and/or memory snapshots, among other examples. The data collected and maintained by the knowledge component **102** yields a knowledgebase that may be used to periodically train a classifier, e.g. learning classifier component **104** utilized by system **100**. The learning classifier component may be used to classify an executable file as one of the following classifications: malicious, benign or potentially unwanted. In some examples, classification may span two or more two or more of those classification categories. For example, an executable file may be benign in the sense that it is not harmful to system **100** but may also be classified as potentially unwanted as it might be installed without explicit user consent, for example. Data maintained by the knowledge component **102** may be continuously updated system **100** or a service that updates system **100** with new exploits to add to the training sample. For example, a research team may be employed to continuously collect new examples of harmful executables, benign executables, and potentially unwanted executables, as many unknown executable files are generated on a daily basis over the Internet. Executable files may be evaluated by the research team such as by applying applications or processing to evaluate executable files including data associated with the file and/or actions associated with the file (e.g., how it is installed and what a file does upon execution). Continuous update of the data maintained by the knowledge component **102** in conjunction with on-going re-learning/re-training of the learning classifier component **104** ensures that system **100** is up-to-date on current threats. Knowledge of the most current threats

improves the generalization capability of system **100** to new unknown threats. By incorporating malicious files, benign files and potentially unwanted files, the present disclosure greatly improves a knowledge base that may be used to in training a learning classifier thereby resulting in more accurate classifications as compared with other knowledge bases that are based on only malicious and/or benign files.

In examples, the collected data on the executable files is analyzed to identify static data points that may indicate one of a malicious file, a benign file or a potentially unwanted file. For instance, the knowledge component **102** may employ one or more programming operations to identify static data points for collection, and to associate the static data points with one of the categories of files (e.g., malicious, benign or potentially unwanted). Programming operations utilized by the knowledge component **102** include operations to collect file data (e.g., executable files or data points from executable files), parse the file data, and store extracted data points. In at least one example, the knowledge component **102** comprises one or more components to manage file data. For example, the knowledge component **102** may comprise one or more storages such as databases, and one or more additional components (e.g., processors executing programs, applications, application programming interfaces (APIs), etc.).

Further, the knowledge component **102** may be configured to continuously collect data, generate a more robust collection of file data to improve classification of file data and train a classifier used to detect whether an executable file is harmful, benign, or potentially unwanted. The identification of static data points to be collected and analyzed for executable files may be continuously updated as more information becomes available to the knowledge component **102**. A static data point may be a point of reference used to evaluate an executable file. As examples, static data points include, but are not limited to: header information, section information, import and export information, certificate information, resource information, string and flag information, legal information, comments, and/or program information (e.g., APIs and DLLs), among other examples. Static data points may be organized into categories that identify a type of static data point. Categories of static data points comprise, but are not limited to: numeric values, nominal values string sequences, byte sequences, and/or Boolean values, among other examples. Any number of static data points may be collected and analyzed for an executable file. In general, collecting and analyzing a greater number of static data points results in more accurate classification of an executable file. For example, eighty static data points may be identified and collected (or attempted to be collected) from an executable file during analysis of an executable file. While a specific number is static data points are provided herein, one of skill in the art will appreciate the more or fewer static data points may be collected without departing from the scope of this disclosure. As an example, the following table, Table 1.1, identifies some of the static data points identified for analysis of an executable file, where the static data points are organized by category:

TABLE 1.1

Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heapr	file OS	special build	Code Characteristics Anomaly
size of stackr	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

Examples of the present disclosure need not distinguish between encrypted files and non-encrypted files. The static data points may be extracted from files regardless of whether or not they are encrypted and/or compressed. Given that the training set contains a statistically significant number of encrypted and non-encrypted examples, as well as compressed and decompressed files, a learning classifier (e.g., learning classifier component **104**) may be trained to identify features in the extracted data to identify malicious files. Since a very large majority of the files "in the wild" use few tools to encrypt the files (e.g., encryption algorithm, packers, etc.), the distribution of the data found across large number of files is preserved after encryption, although the actual content of the data is transformed into something different. By collecting a large set of training examples as they appear "in the wild", a sufficient representation of features and distributions associated with the represented features exist in a training set that are both encrypted and not-encrypted. The present disclosure provides at least one benefit over other threat detection applications/programs by utilizing a very large and diverse training sample, thereby enabling a more intelligent and adaptive learning classifier that is able to achieve high detection rates for threats and low false positive rates, among other benefits.

In addition to collecting and managing information to train a learning classifier, the knowledge component **102** is used to evaluate an executable file and extract/collect static data points for evaluation of the file by the learning classifier before the file is executed. In one example, the system automatically identifies a new file (e.g., unknown file) for evaluation. As examples, identification of an executable file for evaluation may occur in examples such as: when a download of a file is requested, while a download is being performed, evaluation of streaming data, when a new file is detected as attempting to execute (e.g., potentially unwanted file), and before an unknown file or file containing executable code that was not previously checked attempts to execute, among other examples. In another example, a user of a device with which components of system **100** are operating may identify a file to be evaluated. The knowledge component **102** collects as many static data points as it can to evaluate the executable file using a learning classifier built by the learning classifier component **102**. In an exemplary extraction, the knowledge component **102** extracts each of the static data points identified in Table 1.1. The learning classifier component **102** intelligently builds a learning classifier to evaluate a file based on the extracted static data points of the file and the data managed by the knowledge component **102**.

The learning classifier component **104** is a component used to evaluate an executable file using the information provided by the knowledge component **102**. The learning classifier component **104** interfaces with the knowledge component **102** and a threat detection component **106** for the system **100** to evaluate an executable file as a possible threat. As an example, the learning classifier **104** applies programming operations or machine learning processing to evaluate static data points extracted from a file to be analyzed. In doing so, the learning classifier component **104** builds a learning classifier based on information from the knowledge component **102** including the extracted static data points of a file and the information used to train/re-train the learning classifier (e.g., static data on the robust collection of executable files including variety of file types of malicious executable files, benign executable files, and potentially unwanted executable files).

As an example, the learning classifier can adaptively set features of a learning classifier based on the static data points extracted for a file. For example, ranges of data can be identified for static data points that may enable the learning classifier to controllably select (e.g., turn on/off) features for evaluation by learning classifier. In one instance where a static data point extracted for evaluation is a file size (e.g., numeric value), the learning classifier may turn on/off features for evaluation based on whether a the file size of the file is within a certain range. In another example, the learning classifier may detect that the file does not contain "legal information" in the file. Legal information may be any identifying information indicating data that conveys rights to one or more parties including: timestamp data, licensing information, copyright information, indication of intellectual property protection, etc. In an example where the learning classifier detects the presence of legal information does not exist in the executable file, this may trigger the learning classifier to adaptively check for additional features that might be indicative of a threat or malicious file as well as turn off other features related to an evaluation of the "legal information." The learning classifier component **104** utilizes programming operations or machine-learning processing to train/re-train the learning classifier to uniquely evaluate any file/file type. One of skill in the art will appreciate different types of processing operations may be employed without departing from the spirit of this disclosure. For example, the learning classifier component **104** may utilize an artificial neural network (ANN), a decision tree, association rules, inductive logic, a support vector machine, clustering analysis, and Bayesian networks, among other examples.

The learning classifier component **104** processes and encodes collected/extracted data from a file to make the static data points suitable for processing operations. The processing and encoding executed by the learning classifier component **104** may vary depending on the identified categories and/or type (e.g., numeric values, nominal values, string/byte sequences, Boolean values, etc.) of static data points collected by the knowledge component **102**. As an example, string sequence data as well as byte sequence data may be parsed and processed as n-grams and/or n-gram word prediction (e.g., word-grams). For instance, for a given string all unigrams, bigrams and so forth up to a given length n are generated and the counts of the individual n-grams are determined. The resulting counts of the unique n-grams string and/or byte sequences are then used as input to a generated feature vector. In one example, strings are processed directly according to a bag of word model. A bag of word model is a simplifying representation used in natural language processing and information retrieval. In another example, numeric value static data points may be binned appropriately ensuring a good balance between information loss through data binning and available statistics. Nominal values as well Boolean values may be encoded using true/false flags. All of the different encoded data may be combined into one or more final feature vectors, for example after a complex vector coding/processing is performed (e.g., L2 norm).

As an application example of processing performed by the learning classifier component **104**, suppose extraction of static data points from a file identifies the following four (4) data points:

File size: 4982784
PEHeader Anomaly: False
Section name: .TEXT
Legal copyright: Copyright (C) Webroot Inc. 1997

Processing these 4 data points into a feature vector may require binning the file size, labeling the PE Header Anomaly, build n-grams for the section name and building word-gram for legal copyright. Encapsulation of special characters that sometimes appear in text (e.g., section names) may be desirable. In an example, strings of data may be transformed first into hex code representation and then build n-grams or word-grams. In this particular example, the section name is transformed into n-grams while the legal copyright text is transformed into word-grams. In evaluation using the classifier, the features related to the extracted data points may be weighted by the particular data point being evaluated or by category, for example. For instance, the training of the learning classifier may indicate that legal information (e.g., "legal copyright") provides a better indication that a file may be malicious than that of a section name. Features being evaluated (and their distributions) differ for benign files as compared to malicious files (e.g., harmful executables such as malware). For example, the data field "Legal Copyright" for benign files tends to have meaningful words, whereas in a malicious file, this data field is often left empty or it is filled with random characters. Commercial software files tend to have a valid certificate, whereas malware in general do not have valid certificates. Similarly, each data field for a static data point evaluated

provides further indication about the maliciousness of the file. The combined information from all these data fields coupled with machine learning enables accurate prediction determination as to whether a file is malicious, benign or potentially unwanted. As an example, the resulting feature vector in sparse form is shown below:

```
secname_002e00740065:0.204124145232
secname_0065:0.0944911182523
secname_0074006500780074:0.353553390593
secname_006500780074:0.408248290464
secname_00740065:0.144337567297
secname_002e007400650078:0.353553390593
secname_00650078:0.144337567297
secname_002e0074:0.144337567297
secname_002e:0.0944911182523
secname_00780074:0.433012701892
secname_0078:0.0944911182523
secname_007400650078:0.204124145232
secname_0074:0.472455591262
legalcopyright_0043006f00700079007200690067006800740:0.4472135955
legalcopyright_002800430029:0.4472135955
legalcopyright_0043006f00720070002e:0.4472135955
legalcopyright_0031003900390035:0.4472135955
legalcopyright_004d00690063007200 6f0073006f00660074:0.4472135955
filesize_9965:1.0
PEHeaderAnomaly_False:1.0
```

In examples, the learning classifier component **104** provides the one or more feature vectors as input to a support vector machine (SVM). The SVM may then perform data analysis and pattern recognition on the one or more feature vectors. In one example the SVM, may be a linear SVM. Given a set of training examples provided by the knowledge component **102**, an SVM may build a probabilistic model that indicates whether or not a file may be malicious. In another example, a hybrid approach may be elected that combines two or more individual linear SVM classifiers into a final classifier using ensemble methods. Specifically, the evaluated static data points may be subdivided into a set of families (e.g., sections, certificate, header data, and bytes sequences are used as different families). For each family, a linear SVM may be trained. The resulting classification scores generated by each linear SVM may then be combined into a final classification score using a decision tree. As an example, the decision tree may be trained using two-class logistic gradient boosting. In yet another example of feature classification evaluation, a classification may be subdivided into a three class classification problem defined by malicious files, potentially unwanted files/applications and benign files. The resulting three class problem is solved using multi-class classification (e.g., Directed Acyclic Graph (DAG) SVM) or, in case of the hybrid approach based on feature families, using a decision tree based on three-class logistic gradient boosting.

The threat detection component **106** is a component of system **100** that evaluates results of the processing performed by the learning classifier component **104** to make a determination as to whether the executable file is malicious, benign or a potentially unwanted file/application. As an example, a probabilistic value may be a final score determined from evaluating all static data points (or feature distributions) individually and determining an aggregate score that may be used for the evaluation of an executable file. In another example, correlation between static data points may be determined by the learning classification component **104** and a final score may be generated based on the correlation between static data points evaluated.

In one example, a classification as to whether a file is malicious or not may be based on comparison with a predetermined threshold value. For instance, a final score (e.g., probability value/evaluation) is determined based on feature vector processing performed by the learning classifier component **104** and compared with a probability threshold value for determining whether an executable file is malicious. As an example, a threshold value(s) may be set based on predetermined false positive range data where ranges may be set for one or more of the malicious executable files, the benign executable files and the potentially unwanted executable files. The threshold values for each of the different types may be the same or different. For instance, a threshold value may be set that indicates a confidence score in classifying an executable file as a malicious executable file. Ranges may be determined indicating how confident the system **100** is in predicting that a file is malicious. That may provide an indication of whether further evaluation of an executable file should occur. However, one skilled in the art will recognize that threshold determinations can be set in any way that be used to determine whether an executable file is malicious or not. Examples of further evaluation include but are not limited to: additional processing using a learning classifier, identification of an executable file as potentially malicious where services associated with the system **100** may follow-up, quarantining a file, and moving the file to a secure environment for execution (e.g., sandbox) among other examples. In other examples, retraining of the classifier may occur based on confidence scores. In examples, when a probability value exceeds or threshold value (or alternatively is less than a threshold value), it may be determined that an executable file may be identified as malicious. One skilled in the art will recognize that determination for predictive classification of executable files is not limited to threshold determinations. For example, any type of analytical, statistical or graphical analysis may be performed on data to classify an executable file/unknown executable file. As identified above, the threat detection component **106** may interface with the learning classifier component **104** to make a final determination as to how to classify an executable file as well as interface with the knowledge component **102** for training/retraining associated with the learning classifier.

FIG. **2** illustrates an exemplary distributed system **200** showing interaction of components for implementation of exemplary threat detection system as described herein. Where FIG. **1** illustrates an example system **100** having

components (hardware or software) operating on a client, FIG. **2** illustrates an exemplary distributed system **200** comprising a client component **202** connected with at least one server component **204** via communication line **203**. Communication line **203** represents the ability of the client component **202** to communicate with the server component **204**, for example, to send or receive information over a network connection. That is, client component **202** and server component **204** are connectable over a network connection (e.g., a connection Internet via, for example, a wireless connection, a mobile connection, a hotspot, a broadband connection, a dial-up, a digital subscriber line, a satellite connection, an integrated services digital network, etc.).

The client component **202** may be any hardware (e.g., processing device) or software (e.g., application/service or remote connection running on a processing device) that accesses a service made available by the server component **204**. The server component **204** may be any hardware or software (e.g., application or service running on a processing device) capable of communicating with the client component **202** for execution of threat detection processing (e.g., threat detection application/service). Threat detection processing may be used to evaluate a file before execution of the file as described in FIG. **1**. Threat detection application or services may be present on at least one of the client component **202** and the server component **204**. In other examples, applications or components (e.g., hardware or software) may be present on both the client component **202** and the server component **204** to enable processing by threat detection application/services when a network connection to the server cannot be established. In one example of system **200**, client component **202** (or server component **204**) may comprise one or more components for threat detection as described in system **100** including a knowledge component **102**, a learning classifier component **104** and/or a threat detection component **106**, as described in the description of FIG. **1**. In other examples, the client component **202** may transmit data to/from the server component **204** to enable threat detections services over distributed network **200**, for example as represented by communication line **203**. In one example, threat detection applications/services operating on the client component **202** may receive updates from the server component **204**. For instance, updates may be received by the client component **202** for re-training of a learning classifier used to evaluate an executable file on the client component **202**.

FIG. **3** illustrates an exemplary method **300** for performing threat detection. As an example, method **300** may be executed by an exemplary system such as system **100** of FIG. **1** and system **200** of FIG. **2**. In other examples, method **300** may be executed on a device comprising at least one processor configured to store and execute operations, programs or instructions. However, method **300** is not limited to such examples. Method **300** may be performed by any application or service that may include implementation of threat detection processing as described herein. The method **300** may be implemented using software, hardware or a combination of software and hardware.

Method **300** begins at operation **302** where a knowledge base is built for training/retraining of a learning classifier used to detect threats in executable files. Operation **302** builds the knowledge base from data collected and evaluated related to known malicious executable files, known benign executable files and potentially unwanted executable files as described in the description of FIG. **1** (e.g., knowledge component **102**). The knowledge base may be used to automatically train/re-train one or more learning classifiers used to evaluate threats in executable files based on the known malicious executable files, known benign executable files and the potentially unwanted executable files. In examples, the knowledge base may be maintained on at least one of a client component and a server component, and the knowledge base may be continuously updated with information from the resources described in FIG. **1** including update information based on threat detection processing evaluation performed on unknown executable files.

When an executable file is identified for evaluation, flow proceeds to operation **304** where a one or more static data points are extracted from an executable file. In operation **304**, an executable file is analyzed using machine learning processing as described with respect to the knowledge component **102** of FIG. **1** to collect/extract static data points from an executable file for evaluation. In examples, operation **304** occurs without decrypting or unpacking the executable file. However, in other examples, machine learning processing performed has the capability to evaluate static data points from extracted/unpacked content. In one example, machine learning processing is used to extract static data points from encrypted and/or compress versions of one or more files. In another example, machine learning processing is used to extract static data points from decrypted and/or unpacked versions of one or more files. In any example, extraction of static data points from different files (e.g. executable files) can be used to enhance training of a learning classifier, providing better results for extraction of static data points and classification of files.

In examples, operation **304** further comprises classifying extracted data according to a type of data extracted. For instance, the actions performed at operation **304** may be used to classify the plurality of static data points extracted into categories (categorical values) comprising numeric values, nominal values, string or byte sequences, and Boolean values, for example, as described with respect to FIG. **1**. In examples, data of an executable file (e.g., binary file data) may be parsed and compared against data maintained by a threat detection application/service as described in the present disclosure to determine static data points for extraction/collection.

In operation **306**, the executable file is analyzed for threats. As an example, operation **306** analyzes an executable without decrypting or unpacking the executable file. However, in other examples, the machine learning processing being performed has the capability to evaluate static data points from extracted/unpacked content. Operation **306** comprises applying a learning classifier (e.g., the learning classifier generated during performance of operation **302**) to the plurality of static data points extracted from the file. As discussed, the learning classifier may be built from data comprising known malicious executable files, known benign executable files and known unwanted executable files, for example. In one example, operation **306** comprises generating at least one feature vector from the plurality of static data points extracted using the learning classifier trained by the knowledge base. In order to generate the feature vector for the learning classifier, data may be parsed and encoded for machine learning processing.

In one example, generation of the feature vector may comprise selectively setting features of the learning classifier based on the one or more of static data points extracted. Features of the generated feature vector may be weighted based on classified categories identified by the knowledge base (as described above) and the plurality of static data points extracted from the file. As an example, one or more

features of the feature vector may be selectively turned on or off based on evaluation of whether a value of a static data point is within a predetermined range. However, one skilled in the art will recognize that the learning classifier can uniquely generate a feature vector for analysis of the executable file based on any data used to train/re-train the learning classifier. In examples, operation **306** further comprises evaluating the feature vector using linear or nonlinear support vector processing to determine a classification for the executable file, for example whether the executable file is harmful, benign, or unwanted.

Flow proceeds to operation **308**, where a determination is made as to a classification of the executable file. For example, operation **308** makes a final determination as to whether the executable file is harmful (e.g., malicious, malware) or not based on results of the analysis of the executable file (e.g., using machine learning processing by a learning classifier). In one example, results of the analysis of an executable file may be data obtained from learning classifier, such as, for example an SVM, processing data. In one example, operation **308** further comprises preventing execution of the executable file when a probability value that the executable file is harmful exceeds a threshold value. The probability value for the executable file may be determined based on applying the learning classifier to the executable file. As an example, the threshold value may be set based on predetermined false positive range data for identifying a malicious or harmful executable file. False positive range data may be determined from the analysis/evaluation of the known malicious executable files, known benign files and potentially unwanted executable files/applications, of the knowledge base. However, as acknowledged above, determining a classification of an executable file may be based on any type of analytical, statistical or graphical analysis, or machine learning processing. In one example, ranges can be based on evaluation of data during operation of the threat detection service as well as analysis related to unknown files, for example analytics performed to evaluate unknown files.

At any point in time, operation **310** may occur where a learning classifier used for threat detection processing is re-trained. Continuously re-training of the learning classifier may ensure that the threat detection application/service is up to date and able to accurately detect new threats. As identified above, re-training may occur through results of threat detection processing including updated information added to the knowledge base. In one example, training of a learning classifier can be based on evaluation of data during operation of the threat detection service as well as analysis related to unknown files, for example analytics performed to evaluate unknown files.

FIG. **4** and the additional discussion in the present specification are intended to provide a brief general description of a suitable computing environment in which the present invention and/or portions thereof may be implemented. Although not required, the embodiments described herein may be implemented as computer-executable instructions, such as by program modules, being executed by a computer, such as a client workstation or a server. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. Moreover, it should be appreciated that the invention and/or portions thereof may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

FIG. **4** illustrates one example of a suitable operating environment **400** in which one or more of the present embodiments may be implemented. This is only one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality. Other well-known computing systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics such as smart phones, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

In its most basic configuration, operating environment **400** typically includes at least one processing unit **402** and memory **404**. Depending on the exact configuration and type of computing device, memory **404** (storing, among other things, executable evaluation module(s), e.g., malware detection applications, APIs, programs etc. and/or other components or instructions to implement or perform the system and methods disclosed herein, etc.) may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.), or some combination of the two. This most basic configuration is illustrated in FIG. **4** by dashed line **406**. Further, environment **400** may also include storage devices (removable, **408**, and/or non-removable, **410**) including, but not limited to, magnetic or optical disks or tape. Similarly, environment **400** may also have input device(s) **414** such as keyboard, mouse, pen, voice input, etc. and/or output device(s) **416** such as a display, speakers, printer, etc. Also included in the environment may be one or more communication connections, **412**, such as LAN, WAN, point to point, etc.

Operating environment **400** typically includes at least some form of computer readable media. Computer readable media can be any available media that can be accessed by processing unit **402** or other devices comprising the operating environment. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other non-transitory medium which can be used to store the desired information. Computer storage media does not include communication media.

Communication media embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes

wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

The operating environment **400** may be a single computer operating in a networked environment using logical connections to one or more remote computers. The remote computer may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above as well as others not so mentioned. The logical connections may include any method supported by available communications media. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

The different aspects described herein may be employed using software, hardware, or a combination of software and hardware to implement and perform the systems and methods disclosed herein. Although specific devices have been recited throughout the disclosure as performing specific functions, one of skill in the art will appreciate that these devices are provided for illustrative purposes, and other devices may be employed to perform the functionality disclosed herein without departing from the scope of the disclosure.

As stated above, a number of program modules and data files may be stored in the system memory **404**. While executing on the processing unit **402**, program modules **408** (e.g., applications, Input/Output (I/O) management, and other utilities) may perform processes including, but not limited to, one or more of the stages of the operational methods described herein such as method **300** illustrated in FIG. **3**, for example.

Furthermore, examples of the invention may be practiced in an electrical circuit comprising discrete electronic elements, packaged or integrated electronic chips containing logic gates, a circuit utilizing a microprocessor, or on a single chip containing electronic elements or microprocessors. For example, examples of the invention may be practiced via a system-on-a-chip (SOC) where each or many of the components illustrated in FIG. **4** may be integrated onto a single integrated circuit. Such an SOC device may include one or more processing units, graphics units, communications units, system virtualization units and various application functionality all of which are integrated (or "burned") onto the chip substrate as a single integrated circuit. When operating via an SOC, the functionality described herein may be operated via application-specific logic integrated with other components of the operating environment **400** on the single integrated circuit (chip). Examples of the present disclosure may also be practiced using other technologies capable of performing logical operations such as, for example, AND, OR, and NOT, including but not limited to mechanical, optical, fluidic, and quantum technologies. In addition, examples of the invention may be practiced within a general purpose computer or in any other circuits or systems.

This disclosure described some aspects of the present technology with reference to the accompanying drawings, in which only some of the possible embodiments were shown. Other aspects may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein. Rather, these aspects were provided so that this disclosure was thorough and complete and fully conveyed the scope of the possible embodiments to those skilled in the art.

Although specific aspects were described herein, the scope of the technology is not limited to those specific embodiments. One skilled in the art will recognize other embodiments or improvements that are within the scope and spirit of the present technology. Therefore, the specific structure, acts, or media are disclosed only as illustrative embodiments. The scope of the technology is defined by the following claims and any equivalents therein.

What is claimed is:

1. A computer-implemented method comprising:

identifying, by a knowledge module, static data points that may be indicative of either a harmful or benign executable file;

associating, by the knowledge module, the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

identifying an executable file to be evaluated;

extracting, by the knowledge module, a plurality of static data points from the identified executable file;

generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

2. The computer-implemented method according to claim **1**, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

3. The computer-implemented method according to claim **1**, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.

4. The computer-implemented method according to claim **1**, wherein the probabilistic model indicates whether the executable file is harmful.

5. The computer-implemented method according to claim **1**, wherein the executable file is identified in response to a detected condition.

6. The computer-implemented method according to claim **5**, wherein the detected condition is user request for a file download.

7. The computer-implemented method according to claim **5**, wherein the detected condition is the detection of a new file attempting to execute.

8. The computer-implemented method according to claim **1**, wherein the plurality of static data points represent predefined character strings in the executable file.

9. The computer-implemented method according to claim **1**, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

10. A system comprising:

at least one memory; and

at least one processor operatively connected with the memory and configured to perform operation of:

identifying static data points that may be indicative of either a harmful or benign executable file;

associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

identifying an executable file to be evaluated;
extracting a plurality of static data points from the identified executable file; and
generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and
providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

11. The system according to claim **10**, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

12. The system according to claim **10**, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.

13. The system according to claim **10**, wherein the probabilistic model indicates whether the executable file is harmful.

14. The system according to claim **10**, wherein the plurality of static data points represent predefined character strings in the executable file.

15. A computer-readable storage device containing instructions, that when executed on at least one processor, causing the processor to execute a process comprising:
identifying static data points that may be indicative of either a harmful or benign executable file;
associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;
identifying an executable file to be evaluated;
extracting a plurality of static data points from the identified executable file;
generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and
providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

16. The computer-readable storage device according to claim **15**, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

17. The computer-readable storage device according to claim **15**, wherein the plurality of static data points represent predefined character strings in the executable file.

18. A computer-implemented method comprising:
identifying static data points that may be indicative of either a harmful or benign executable file;
associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;
identifying an executable file to be evaluated;
extracting a plurality of static data points from the executable file;
generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and
evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.

19. The computer-implemented method according to claim **18**, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

20. The computer-implemented method according to claim **18**, wherein the machine learning model comprises an artificial neural network.

21. The computer-implemented method according to claim **18**, wherein the machine learning model comprises a support vector machine.

22. The computer-implemented method according to claim **18**, wherein the machine learning model comprises a machine learning decision tree.

23. The computer-implemented method according to claim **18**, wherein the machine learning model comprises a Bayesian network.

24. The computer-implemented method according to claim **18**, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

25. The computer-implemented method according to claim **18**, wherein the machine learning model comprises a clustering model.

26. The computer-implemented method according to claim **18**, wherein the executable file is identified in response to a detected condition.

27. The computer-implemented method according to claim **26**, wherein the detected condition is user request for a file download.

28. The computer-implemented method according to claim **26**, wherein the detected condition is the detection of a new file attempting to execute.

29. The computer-implemented method according to claim **18**, wherein the plurality of static data points represent predefined character strings in the executable file.

30. The computer-implemented method according to claim **18**, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

31. The computer-implemented method according to claim **18**, further comprising preventing execution of the executable file based on classifying the executable file as harmful.

32. A computer-readable storage device containing instructions, that when executed on a processor, cause the processor to execute a process comprising:
identifying static data points that may be indicative of either a harmful or benign executable file;
associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;
identifying an executable file to be evaluated;
extracting a plurality of static data points from the executable file;

generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.

33. The computer-readable storage device according to claim **32**, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

34. The computer-readable storage device according to claim **32**, wherein the machine learning model comprises an artificial neural network.

35. The computer-readable storage device according to claim **32**, wherein the machine learning model comprises a support vector machine.

36. The computer-readable storage device according to claim **32**, wherein the machine learning model comprises a machine learning decision tree.

37. The computer-readable storage device according to claim **32**, wherein the machine learning model comprises a Bayesian network.

38. The computer-readable storage device according to claim **32**, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

39. The computer-readable storage device according to claim **32**, wherein the machine learning model comprises a clustering model.

40. The computer-readable storage device according to claim **32**, wherein the executable file is identified in response to a detected condition.

41. The computer-readable storage device according to claim **40**, wherein the detected condition is user request for a file download.

42. The computer-readable storage device according to claim **40**, wherein the detected condition is the detection of a new file attempting to execute.

43. The computer-readable storage device according to claim **32**, wherein the plurality of static data points represent predefined character strings in the executable file.

44. The computer-readable storage device according to claim **32**, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

45. The computer-readable storage device according to claim **32**, wherein the process further comprises preventing execution of the executable file based on classifying the executable file as harmful.

46. A system comprising:

at least one memory;

at least one processor configured to perform operation of:

identifying static data points that may be indicative of either a harmful or benign executable file; and

associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files; and

at least one processor configured to perform operation of:

identifying an executable file to be evaluated;

extracting a plurality of static data points from the executable file;

generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.

47. The system according to claim **46**, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

48. The system according to claim **46**, wherein the machine learning model comprises an artificial neural network.

49. The system according to claim **46**, wherein the machine learning model comprises a support vector machine.

50. The system according to claim **46**, wherein the machine learning model comprises a machine learning decision tree.

51. The system according to claim **46**, wherein the machine learning model comprises a Bayesian network.

52. The system according to claim **46**, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

53. The system according to claim **46**, wherein the machine learning model comprises a clustering model.

54. The system according to claim **46**, wherein the executable file is identified in response to a detected condition.

55. The system according to claim **54**, wherein the detected condition is user request for a file download.

56. The system according to claim **54**, wherein the detected condition is the detection of a new file attempting to execute.

57. The system according to claim **46**, wherein the plurality of static data points represent predefined character strings in the executable file.

58. The system according to claim **46**, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

59. The system according to claim **46**, further comprising at least one processor configured to perform operation of:

preventing execution of the executable file based on classifying the executable file as harmful.

* * * * *

Exhibit 7

8287799

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

August 30, 2022

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

PATENT NUMBER: *8,201,243*
ISSUE DATE: *June 12, 2012*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

John Burson
Certifying Officer

US008201243B2

(12) **United States Patent**
Boney

(10) **Patent No.: US 8,201,243 B2**
(45) **Date of Patent: Jun. 12, 2012**

(54) **BACKWARDS RESEARCHING ACTIVITY INDICATIVE OF PESTWARE**

(75) Inventor: **Matthew L. Boney**, Longmont, CO (US)

(73) Assignee: **Webroot Inc.**, Broomfield, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1150 days.

(21) Appl. No.: **11/408,146**

(22) Filed: **Apr. 20, 2006**

(65) **Prior Publication Data**

US 2007/0250817 A1 Oct. 25, 2007

(51) **Int. Cl.**
G06F 21/00 (2006.01)

(52) **U.S. Cl.** **726/22**; 713/164; 713/202; 380/279; 717/124; 726/27

(58) **Field of Classification Search** 726/22
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,623,600	A	4/1997	Ji et al.
5,920,696	A	7/1999	Brandt et al.
5,951,698	A	9/1999	Chen et al.
6,069,628	A	5/2000	Farry et al.
6,073,241	A	6/2000	Rosenberg et al.
6,092,194	A	7/2000	Touboul
6,154,844	A	11/2000	Touboul
6,167,520	A	12/2000	Touboul
6,310,630	B1	10/2001	Kulkarni et al.
6,397,264	B1	5/2002	Stasnick et al.
6,405,316	B1	6/2002	Krishnan et al.
6,460,060	B1	10/2002	Maddalozzo, Jr. et al.
6,480,962	B1	11/2002	Touboul
6,535,931	B1	3/2003	Celi, Jr.
6,611,878	B2	8/2003	De Armas et al.
6,633,835	B1	10/2003	Moran et al.
6,667,751	B1	12/2003	Wynn et al.
6,701,441	B1	3/2004	Balasubramaniam et al.
6,772,345	B1	8/2004	Shetty
6,785,732	B1	8/2004	Bates et al.
6,804,780	B1	10/2004	Touboul
6,813,711	B1	11/2004	Dimenstein
6,829,654	B1	12/2004	Jungek

(Continued)

FOREIGN PATENT DOCUMENTS

WO	WO 98/45778	10/1998

(Continued)

OTHER PUBLICATIONS

Elements of Computer Security|http://www.google.com/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=8&ved=0CEwQFjAH&url=http%3A%2F%2Fcms.vle.ase.md%2Fget%2Fpdf%2F7664&ei=_pzyTp3mMeLL0QG7-9zLAg&usg=AFQjCNEmOz7OxykRPyP1_ePiXGgJI1WGWA&sig2=XE_fLI7CTuchuObMdbGYtA|David Salomon|2010.*

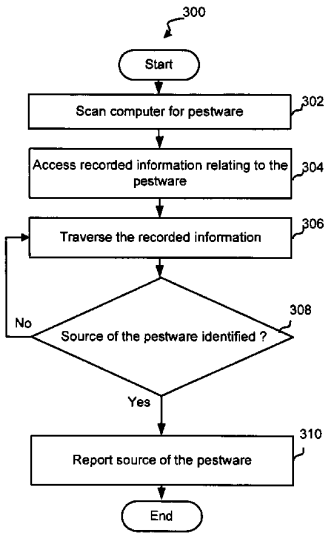
(Continued)

Primary Examiner — Taghi Arani
Assistant Examiner — Mahfuzur Rahman
(74) *Attorney, Agent, or Firm* — Cooley LLP

(57) **ABSTRACT**

A system and method for researching an identity of a source of activity that is indicative of pestware is described. In one embodiment the method comprises monitoring the computer for activity that is indicative of pestware, identifying, based upon the activity, an object residing on the computer that is a suspected pestware object; and accessing at least a portion of a recorded history of sources that the computer received files from so as to identify a reference to an identity of a particular source that the suspected pestware object originated from.

14 Claims, 5 Drawing Sheets



U.S. PATENT DOCUMENTS

6,910,134	B1	6/2005	Maher et al.	
6,965,968	B1	11/2005	Touboul	
7,058,822	B2	6/2006	Edery et al.	
7,058,976	B1 *	6/2006	Dark	726/23
7,107,617	B2	9/2006	Hursey et al.	
7,849,185	B1 *	12/2010	Rockwood	709/224
2002/0078381	A1 *	6/2002	Farley et al.	713/201
2002/0162015	A1	10/2002	Tang	
2002/0162017	A1	10/2002	Sorkin	
2002/0166063	A1	11/2002	Lachman et al.	
2003/0065943	A1 *	4/2003	Geis et al.	713/201
2003/0074581	A1	4/2003	Hursey et al.	
2003/0101381	A1	5/2003	Mateev et al.	
2003/0154399	A1 *	8/2003	Zuk et al.	713/201
2003/0159070	A1 *	8/2003	Mayer et al.	713/201
2003/0217287	A1	11/2003	Kruglenko	
2004/0024864	A1 *	2/2004	Porras et al.	709/224
2004/0030914	A1	2/2004	Kelley et al.	
2004/0034794	A1	2/2004	Mayer et al.	
2004/0064515	A1 *	4/2004	Hockey	709/206
2004/0064736	A1	4/2004	Obrecht et al.	
2004/0080529	A1	4/2004	Wojcik	
2004/0143763	A1	7/2004	Radatti	
2004/0187023	A1	9/2004	Alagna et al.	
2004/0225877	A1 *	11/2004	Huang	713/100
2005/0005160	A1	1/2005	Bates	
2005/0038697	A1	2/2005	Aaron	
2005/0120242	A1 *	6/2005	Mayer et al.	713/201
2005/0138433	A1	6/2005	Linetsky	
2005/0154885	A1	7/2005	Viscomi et al.	
2005/0268112	A1 *	12/2005	Wang et al.	713/188
2006/0085528	A1 *	4/2006	Thomas	709/223
2006/0230290	A1 *	10/2006	Burtscher	713/188
2007/0074289	A1 *	3/2007	Maddaloni	726/23
2007/0094725	A1 *	4/2007	Borders	726/22
2007/0168694	A1 *	7/2007	Maddaloni et al.	714/4
2007/0169198	A1 *	7/2007	Madddaloni et al.	726/24
2007/0220043	A1 *	9/2007	Oliver et al.	707/103 R

FOREIGN PATENT DOCUMENTS

WO	WO 9845778	A2 *	10/1998
WO	PCT/US2006/025378		9/2007
WO	PCT/US2007/067078		11/2007
WO	PCT/US2007/067084		11/2007
WO	PCT/US2006/041798		12/2007

OTHER PUBLICATIONS

PCT/US2007/067076 International Search Report and Written Opinion of the International Searching Authority mailed Nov. 2, 2007.
U.S. Appl. No. 10/956,573, filed Oct. 1, 2004, Steve Thomas.
U.S. Appl. No. 10/956,574, filed Oct. 1, 2004, Steve Thomas.
U.S. Appl. No. 10/956,818, filed Oct. 1, 2004, Matthew Boney.
U.S. Appl. No. 11/257,609, filed Oct. 25, 2005, Michael Burtscher.
U.S. Appl. No. 11/237,291, filed Sep. 28, 2005, Philip Maddaloni.
U.S. Appl. No. 11/408,215, filed Apr. 20, 2006, Matthew Boney.
U.S. Appl. No. 11/408,145, filed Apr. 20, 2006, Matthew Boney.
Codeguru, Three Ways to Inject Your Code Into Another Process, by Robert Kuster, Aug. 4, 2003, 22 pgs.
Codeguru, Managing Low-Level Keyboard Hooks With the Windows API for VB .Net, by Paul Kimmel, Apr. 18, 2004, 10 pgs.
Codeguru, Hooking the Keyboard, by Anoop Thomas, Dec. 13, 2001, 6 pgs.
Illusive Security, Wolves in Sheep's Clothing: malicious DLLs Injected Into trusted Host Applications, Author Unknown, http://home.arcor.de/scheinsicherheit/dll.htm 13 pgs.
DevX.com, Intercepting Systems API Calls, by Seung-Woo Kim, May 13, 2004, 6 pgs.
Microsoft.com, How to Subclass a Window in Windows 95, Article ID 125680, Jul. 11, 2005, 2 pgs.
MSDN, Win32 Hooks by Kyle Marsh, Jul. 29, 1993, 15 pgs.
PCT Search Report, PCT/US05/34874, Jul. 5, 2006, 7 Pages.
Yurcik, William et al., A Planning Framework for Implementing Virtual Private Networks, Jun. 2001, IT Pro, IEEE, pp. 41-44.

* cited by examiner

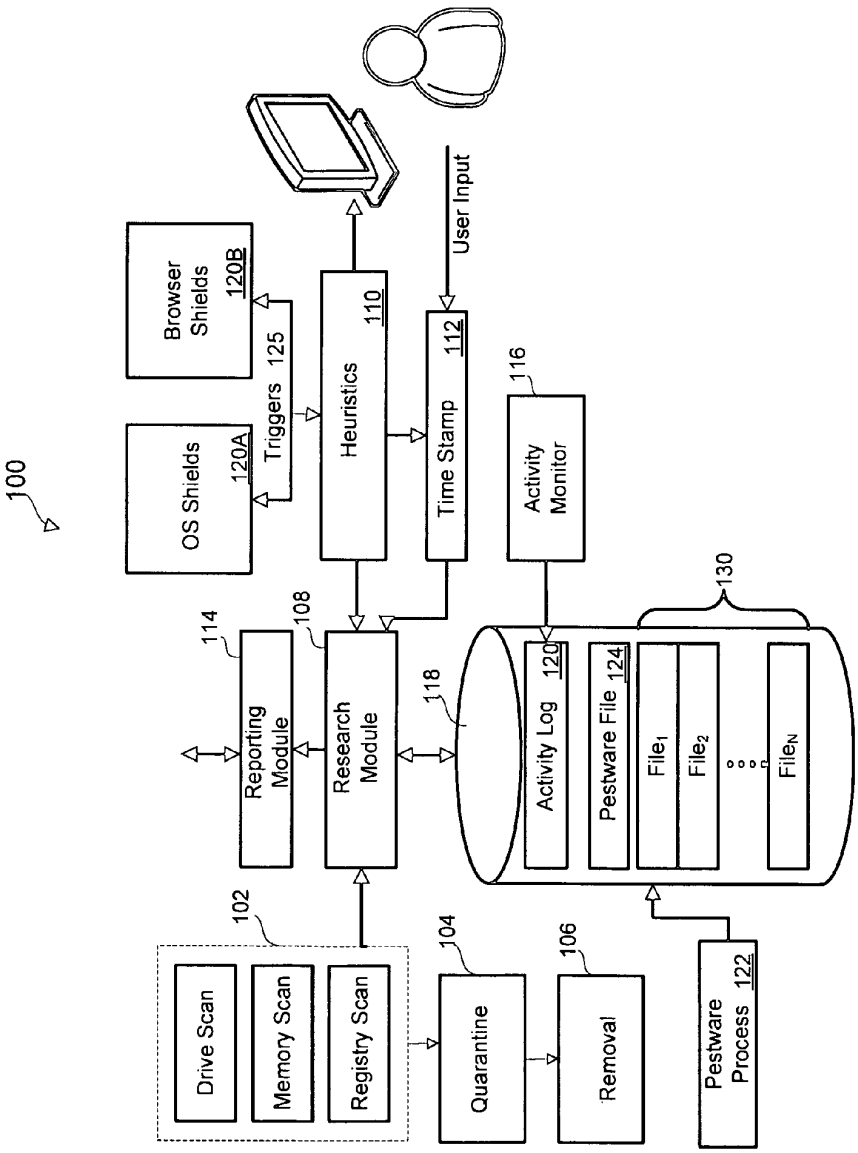

100
Browser Shields 120B
OS Shields 120A
Triggers 125
Heuristics 110
User Input
Time Stamp 112
Activity Monitor 116
130
Reporting Module 114
Research Module 108
118
Activity Log 120
Pestware File 124
File1
File2
FileN
102
Drive Scan
Memory Scan
Registry Scan
104
Quarantine
106
Removal
Pestware Process 122

FIGURE 1

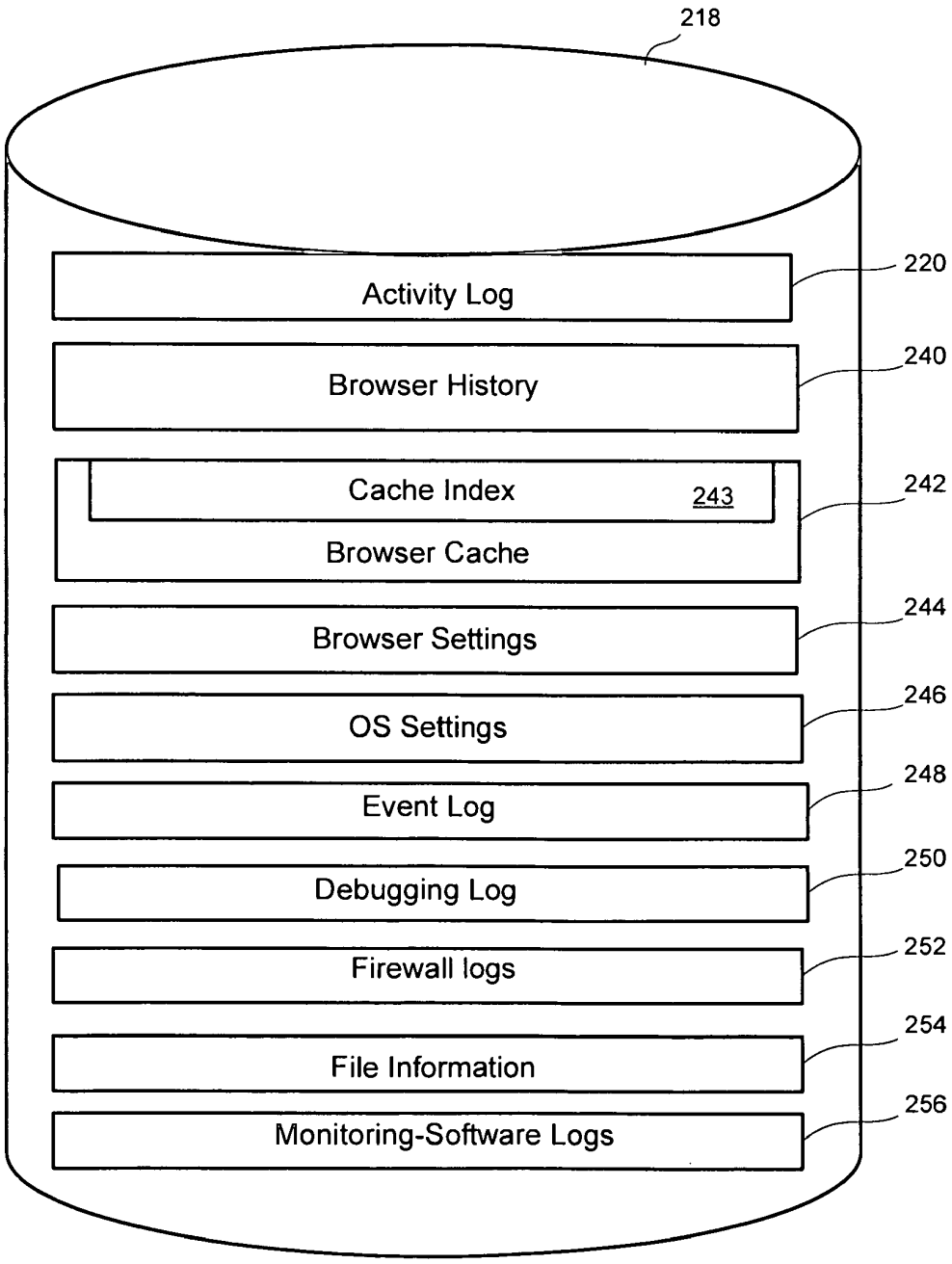

218
Activity Log
220
Browser History
240
Cache Index
243
Browser Cache
242
Browser Settings
244
OS Settings
246
Event Log
248
Debugging Log
250
Firewall logs
252
File Information
254
Monitoring-Software Logs
256

FIGURE 2

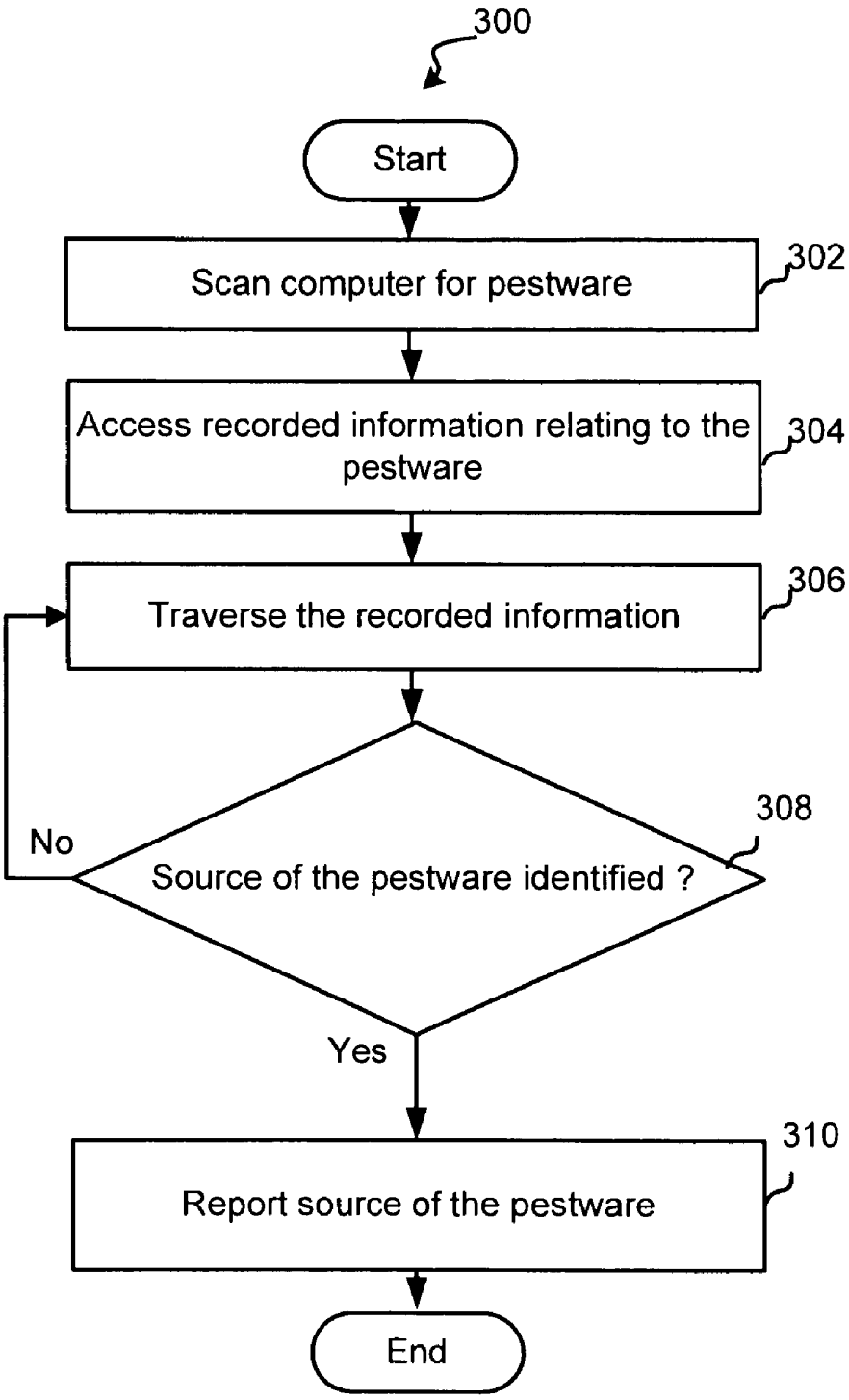

300
Start
Scan computer for pestware
302
Access recorded information relating to the pestware
304
Traverse the recorded information
306
Source of the pestware identified ?
308
No
Yes
Report source of the pestware
310
End

FIGURE 3

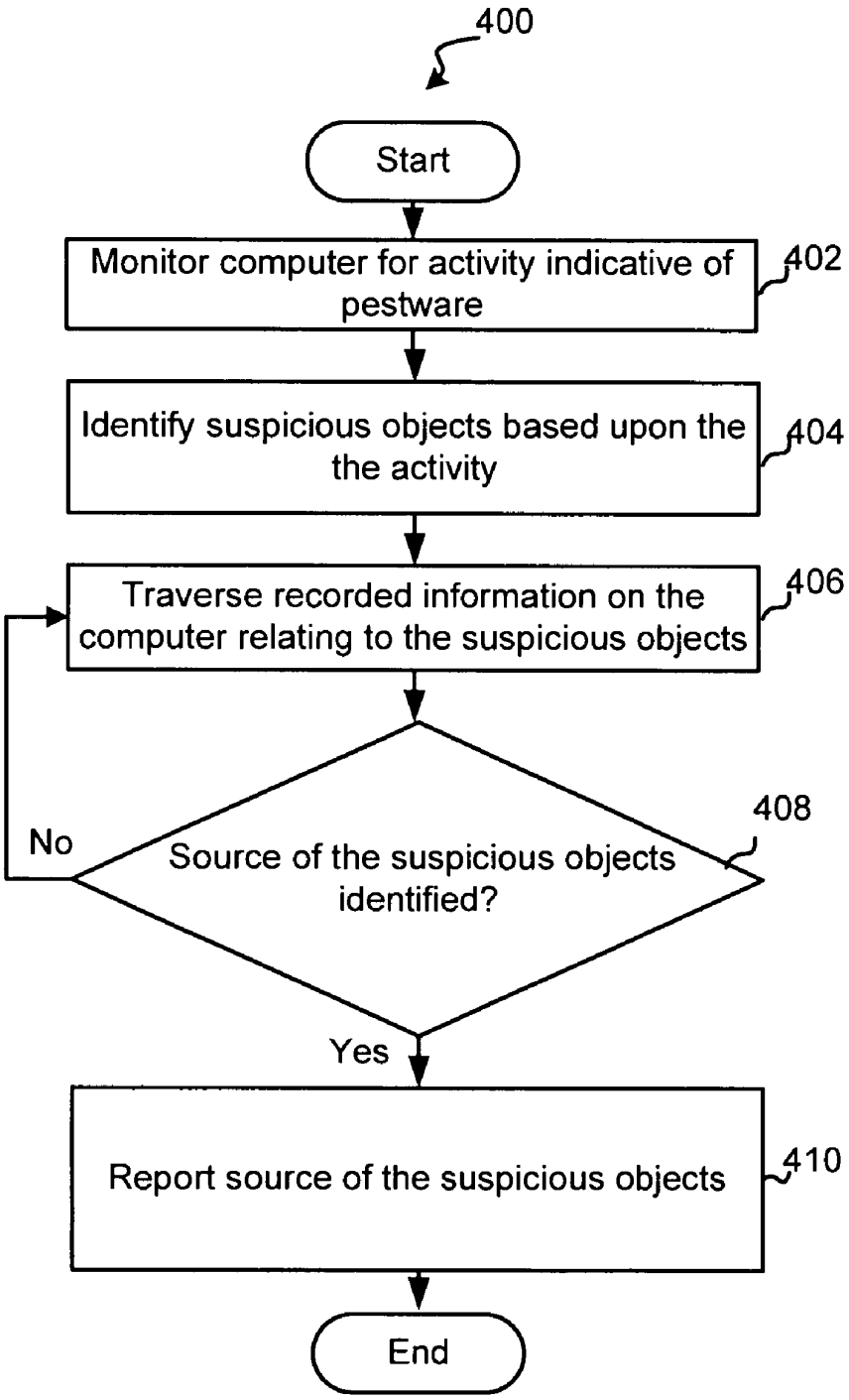

400
Start
Monitor computer for activity indicative of pestware
402
Identify suspicious objects based upon the the activity
404
Traverse recorded information on the computer relating to the suspicious objects
406
Source of the suspicious objects identified?
408
No
Yes
Report source of the suspicious objects
410
End

FIGURE 4

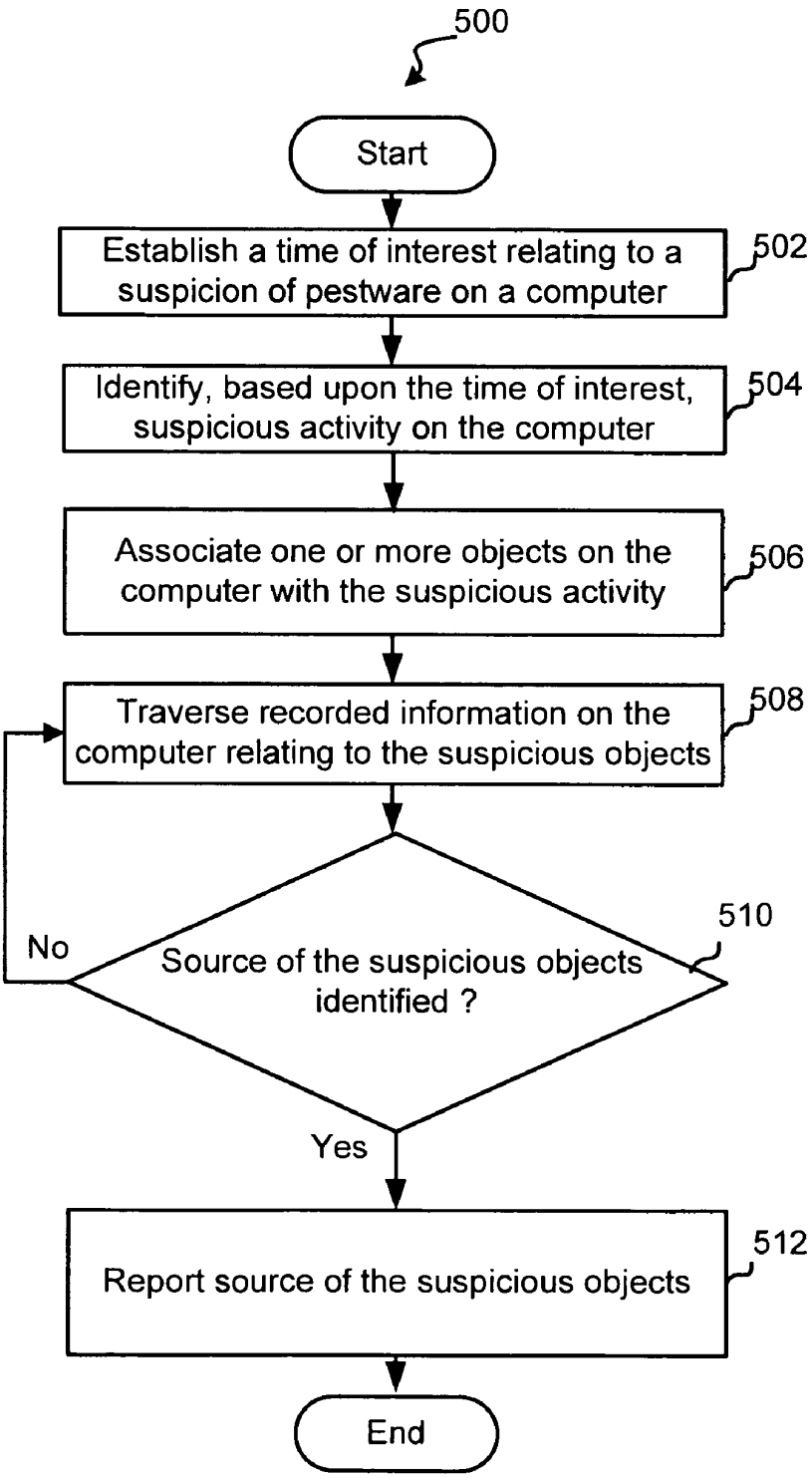

500
Start
Establish a time of interest relating to a suspicion of pestware on a computer
502
Identify, based upon the time of interest, suspicious activity on the computer
504
Associate one or more objects on the computer with the suspicious activity
506
Traverse recorded information on the computer relating to the suspicious objects
508
Source of the suspicious objects identified ?
510
No
Yes
Report source of the suspicious objects
512
End

FIGURE 5

BACKWARDS RESEARCHING ACTIVITY INDICATIVE OF PESTWARE

RELATED APPLICATIONS

The present application is related to commonly owned and assigned application Ser. No. 10/956,573, now U.S. Pat. No. 7,480,683, entitled System and Method For Heuristic Analysis to Identify Pestware; application Ser. No. 10/956,574, now U.S. Pat. No. 7,533,131, entitled System and Method for Pestware Detection and Removal; application Ser. No. 10/956,818, entitled System and Method for Locating Malware and Generating Malware Definitions; application Ser. No. 11/257,609, entitled System and Method for Kernel-Level Pestware Management; application Ser. No. 11/237,291, entitled Client Side Exploit Tracking; application Ser. No. 11/408,215, entitled Backward Researching Existing Pestware, filed herewith; and application Ser. No. 11,408,145, entitled Backward Researching Time Stamped Events to Find an Origin of Pestware, filed herewith, which are incorporated herein by reference.

COPYRIGHT

A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF THE INVENTION

The present invention relates to computer system management. In particular, but not by way of limitation, the present invention relates to systems and methods for detecting, controlling and/or removing pestware.

BACKGROUND OF THE INVENTION

Personal computers and business computers are continually attacked by trojans, spyware, and adware, collectively referred to as "malware," "spyware" or "pestware." These types of programs generally act to gather information about a person or organization—often without the person or organization's knowledge. Some pestware is highly malicious. Other pestware is non-malicious but may cause issues with privacy or system performance. And yet other pestware is actually beneficial or wanted by the user. Unless specified otherwise, "pestware" as used herein refers to any of these programs that collects information about a person or an organization.

Software is presently available to detect and remove pestware. But as it evolves, the software to detect and remove it must also evolve. Accordingly, current techniques and software for removing pestware are not always satisfactory and will most certainly not be satisfactory in the future. Additionally, because some pestware is actually valuable to a user, pestware-detection software should, in some cases, be able to handle differences between wanted and unwanted pestware.

Current pestware removal software uses definitions of known pestware to search for and remove files on a protected system. These definitions are often slow and cumbersome to create. Additionally, it is often difficult to initially locate the pestware in order to create the definitions. Accordingly, a

system and method are needed to address the shortfalls of present technology and to provide other new and innovative features.

SUMMARY OF THE INVENTION

Exemplary embodiments of the present invention that are shown in the drawings are summarized below. These and other embodiments are more fully described in the Detailed Description section. It is to be understood, however, that there is no intention to limit the invention to the forms described in this Summary of the Invention or in the Detailed Description. One skilled in the art can recognize that there are numerous modifications, equivalents and alternative constructions that fall within the spirit and scope of the invention as expressed in the claims.

The present invention can provide a system and method for researching an identity of a source of activity that is indicative of pestware. In one embodiment the method comprises monitoring the computer for activity that is indicative of pestware, identifying, based upon the activity, an object residing on the computer that is a suspected pestware object and accessing at least a portion of a recorded history of sources that the computer received files from so as to identify a reference to an identity of a particular source that the suspected pestware object originated from.

As previously stated, the above-described embodiments and implementations are for illustration purposes only. Numerous other embodiments, implementations, and details of the invention are easily recognized by those of skill in the art from the following descriptions and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Various objects and advantages and a more complete understanding of the present invention are apparent and more readily appreciated by reference to the following Detailed Description and to the appended claims when taken in conjunction with the accompanying Drawings wherein:

FIG. **1** illustrates a block diagram of one implementation of the present invention;

FIG. **2** is a block diagram depicting an embodiment of the file storage device of FIG. **1**;

FIG. **3** is a flowchart depicting a method according to an exemplary embodiment;

FIG. **4** is a flowchart depicting another method according to another embodiment;

FIG. **5** is a flowchart depicting yet another method according to yet another embodiment.

DETAILED DESCRIPTION

Referring now to the drawings, where like or similar elements are designated with identical reference numerals throughout the several views, and referring in particular to FIG. **1**, it illustrates a block diagram of one implementation of the present invention. Shown is a protected computer **100** that includes a detection module **102**, quarantine module **104**, removal module **106**, and shields **120**. In addition, a research module **108** is shown coupled to a heuristics module **110**, a time stamp module **112** and a reporting module **114**.

Each of these modules can be implemented in software or hardware. And if implemented in software, the modules can be implemented in a single software package or in multiple software packages. In addition, one of ordinary skill in the art will recognize that the software can be designed to operate on any type of computer system including WINDOWS and

Linux-based systems. Additionally, the software can be configured to operate on personal computers and/or servers. For convenience, embodiments of the present invention are generally described herein with relation to WINDOWS-based systems. Those of skill in the art can easily adapt these implementations for other types of operating systems or computer systems.

Also shown is a file storage device **118** that is coupled to the research module **108** and an activity monitor **116**. In this embodiment the file storage device includes an activity log **120**, a pestware file **124** and a collection of N files **130**. The file storage device **118** is described herein in several implementations as hard disk drive for convenience, but this is certainly not required, and one of ordinary skill in the art will recognize that other storage media may be utilized without departing from the scope of the present invention. In addition, one of ordinary skill in the art will recognize that the storage device **118**, which is depicted for convenience as a single storage device, may be realized by multiple (e.g., distributed) storage devices.

In the exemplary embodiment, the pestware file **124**, corresponds to (e.g., includes data relating to) a pestware process **122** operating in memory. The pestware process **122** is exemplary of pestware processes that are configured to make one or more unauthorized alterations to the computer **100**. For example, the pestware process **122** may make changes to either or both of the browser settings and/or operating system (OS) settings without approval and/or the knowledge of the user.

In accordance with several embodiments, the research module **108** is configured to receive an indication that either known pestware is residing on the computer or activities indicative of pestware have occurred or are occurring on the protected computer. In response, the research module **108** is configured to research the activity log **120** and/or the N files **130**, which include information (e.g., historical logs) relating to events on the computer **100**, to identify a source of the pestware **122**, **124** or pestware-related activities, which may be reported by the reporting module **114** to a centralized data store for subsequent pestware management purposes.

For example, the identities (e.g., I.P. address, URL, email client or program name) of sources (e.g., web sites, email or program) of the pestware may be collected in a centralized data store (not shown) and then subsequently reported to other users. In addition, the sources of the pestware or suspected pestware may be visited to further research how the pestware is being distributed from the URLs and/or to generate new definitions for pestware discovered at these sources.

As described further with reference to FIG. **2**, the N files **130** include one or more files with information that assist the research module **108** in tracing information in the N files **130** to an identity (e.g., URL) of the source of pestware **122**, **124** on the computer **100**. One or more of the N files **130** may be associated with an operating system of the protected computer and/or one or more applications of the protected computer, and may include information such as process IDs, registry entries, file names, cookies and URLs among other information that is used to trace from an identified pestware file, pestware process and/or pestware activity to an originating source (e.g., URL or IP address) of the infection. In one embodiment, one or more of the N files **130** is generated from an application that generates a log of data after examining the computer. An example of such an application is an application distributed under the name HijackThis.

In the exemplary embodiment depicted in FIG. **1**, the research module **108** is configured to receive indications that pestware may be present on the computer from each of the detection engine **102**, the heuristics engine **110** and the time stamp module **112**, but this is certainly not required. In other embodiments, for example, the research module **108** may be configured to communicate only with the detection engine **102** or only with the heuristics engine. In yet other embodiments, the research module **108** may be configured to receive only a time stamp from the time stamp module **112**.

According to several embodiments, pestware-detection functions operating on the protected computer **100** are represented by the detection engine **102**, the quarantine engine **104**, the removal engine **106**, the shields **120** and the heuristic engine **110**. The basic functions of the detection engine **102** is to compare files, processes and registry entries on the protected computer against known pestware definitions and characteristics. When a match is found, in addition to quarantining and removing a pestware object, the detection engine **102** informs the research module **108** of the pestware. Details associated with several embodiments of sweep, quarantine, and removal engines are found in the above-identified application entitled System and Method for Pestware Detection and Removal.

Pestware and pestware activity can also be identified by the shields **120**, which generally run in the background on the computer system. In the exemplary embodiment depicted in FIG. **1**, the shields **120** are divided into the operating system shields **120**A and the browser shields **120**B. The shields **120** are designed to watch for pestware and for typical pestware activity and includes two types of shields: behavior-monitoring shields and definition-based shields.

As an example, the shields **120** monitor the protected computer **100** for certain types of activities that generally correspond to pestware behavior. Particular examples of some of the types of activities that are monitored include a process spawning another process, an alteration to registry entries, communications with remote sites via the Internet, alterations to a start up folder, injection of a DLL into another process, and a change to the browser's home page and/or bookmarks. Some specific examples of shields are disclosed in the above-identified application entitled System and Method for Locating Malware and Generating Malware Definitions. In the exemplary embodiment, the shields **120** inform the heuristics engine **108** about the activities and the heuristics engine **108** determines whether the research module **108** should be informed and/or whether the activity should be blocked.

As an example, the heuristics module **110** may compare the activities on the computer with weighted factors to make decisions relative to activities at the protected computer. Each of these factors may be, for example, associated with a particular activity and each factor may be weighted by the likelihood that the activity is associated with pestware. If the sum of the weighted factors that match the activity history exceed a threshold, then the activity is identified as pestware activity and the heuristics module **110** prompts the research module **108** to initiate research into the origin of the pestware initiating the activity. It should be recognized that this type of heuristics operation is merely one example, and that the heuristics module **110** may use other techniques to analyze activity on the computer. Additional information related to heuristics-based scanning is found in the above-identified applications entitled System and Method For Heuristic Analysis to Identify Pestware and Client Side Exploit Tracking.

In the exemplary embodiment depicted in FIG. **1**, the time stamp module **112** is configured to send a time stamp to the research module **108** in response to a request from a user and/or in response to the heuristics module **108**. In some embodiments for example, the heuristics module **108** pro-

vides the user with information about suspect activity on the computer **100** so that the user has the option as to whether or not the research module **108** will attempt to identify the source of the activity.

In other embodiments, the heuristics module **110** prompts the time stamp module **112** to initiate the generation of a time stamp, without user intervention, in response to activity that is indicative of pestware activity. In one implementation, for example, the heuristics module **110** prompts the research module **108** to initiate tracing of the pestware activity to an origin of the pestware that is associated with the activity, and the heuristics module **110** also prompts the time stamp module **112** to send a time stamp **112** to the research module **108** so that the research module **108** is provided with a time reference as well as information about the pestware activities.

In yet other embodiments, the timestamp module **112** operates independently of the heuristics module **110**. For example, the timestamp module **112** may prompt the research module **108** to initiate a review of the activity log **120** and/or one or more of the N files in response to a user request.

The activity monitor **116** in several embodiments monitors activities on the computer and stores information about the activities in the activity log **120** so as to assist the research module **108** in identifying activities associated with pestware and the source of the pestware. In some embodiments, for example, the activity log **120** includes a list of processes that are running on the protected computer and the files that are associated with the processes. Although not depicted in FIG. **1**, the shields **120** may utilize the activity monitor **116** to detect pestware-related events and intercept efforts by pestware to spawn new processes or alter aspects of the protected computer **100**.

In some variations, the activity monitor **116** may inject code into existing processes so that when a process attempts to call a function of the computer's **100** operating system (not shown) the injected code can check the process to be started and raise a flag if the existing process is attempting to create a new pestware process or attempting to alter one or more settings (e.g., a registry setting) of the protected computer **100**.

In other embodiments, the activity monitor **116** is realized by a kernel-mode driver that may be loaded during a boot sequence for the protected computer or anytime later. In these embodiments, the activity monitor **116** is configured to log API calls in the activity log. In many variations for example, when a process (e.g., the pestware process) attempts to spawn a another pestware process or alter a registry entry, the API call utilized by the process is intercepted before it is carried out by an operating system of the protected computer. In this way, the attempt may be logged in the activity log **120** and the process may be analyzed to determine whether it is known to be a pestware-related process. Additional details of use of a kernel-level driver in connection with pestware management may be found in the above identified application entitled: System and Method for Kernel-Level Pestware Management.

It should be recognized that the block diagram in FIG. **1** depicts functional capabilities associated with several embodiments of the present invention. One of ordinary skill in the art will recognize that the functions described with reference to FIG. **1** may be realized by various implementations of software in connection with hardware or hardware alone. In these implementations several functions may be consolidated into a single module, for example, and as a consequence, may appear different from the block diagram in FIG. **1** without departing from the scope of the present invention.

Referring next to FIG. **2**, shown are exemplary resources, also referred to herein as historical logs, that are available to the research module **108** in accordance with one embodiment of the present invention. In general, these resources are logs of historical events that occurred on the computer, and each of the logs includes information that may be used to reference other information when, for example, activities, processes and files are traced so as to determine an origin of the pestware or suspected pestware. As shown, a file storage device **218** in this embodiment includes an activity log **220**, which may be generated by an activity monitor (e.g., the activity monitor **116**) and may include information about processes running on the computer **100** and files corresponding to the processes. In variations, the activity log **220** includes a history of API calls and respective times made by processes running on the protected computer, but this is certainly not required.

In addition, the file storage device **218** includes a browser history **240**, browser cache **242**, browser settings **244**, OS settings **246**, an event log **248**, a debugging log **250**, a firewall log **252**, file information **254** and monitoring software logs **256**. One or more of these exemplary files **240-256** may be implemented for one or more of the N files **130** described with reference to FIG. **1**.

The browser history **240** in this embodiment includes a listing of web sites and respective times that the web sites were visited by the user. The browser cache **242** in this embodiment includes files, cookies, and other objects cached in connection with use of a browser of the protected computer (e.g., Internet Explorer or Mozilla (not shown)). As depicted, the browser cache **242** includes a cache index **243** that includes a listing, which associates the content of the browser cache **242** with URLs and time stamps. As discussed further herein, the cache index **243** provides an efficient means for identifying the times objects were retrieved and the URLs that the objects were retrieved from.

The browser settings **244** include information about settings associated with a browser and may include a home page setting and list of user favorites. The browser settings **244** are monitored by the shields **120** for changes. Those settings also contain URLs that may be referenced in time stamped logs, firewall logs, browser histories, etc.

The operating system (OS) settings **246** may include registry entries, a start up folder and other information utilized by an operating system of the protected computer. As discussed further herein, data included in the OS settings **246** may include time stamps, which indicate when changes were made to the settings.

The event log **248** in this embodiment includes a log of events that is maintained by an operating system of the protected computer **100**. For example, the event log **248** may include event information including errors, user log-in history, a listing of processes that have been launched (and the users that launched the processes), path information, information about which process (and which users) accessed a secure area and other information relating to operations of the computer **100**.

Also shown in FIG. **2** is a debugging log **250** that includes application errors, which point to a process and the address where the error occurred. Some techniques employed by pestware forcibly shut down applications, or cause applications to crash when their memory space is injected with pestware code. The infected applications like "explorer.exe" will crash, or some times restart spontaneously. These events/occurrences show up in debugging logs. These are time stamped events, that also reference files on the system.

The firewall log **252** is this embodiment is a collection of information relating to network-related events on the pro-

tected computer. The firewall log **252**, may for example, include time stamps of network activities on the protected computer, which may be utilized by the research module **108** to locate pestware or indicia of pestware.

Also shown in the data storage device **218** is a collection of file information **254**, also known as the file system, which includes a database that the operating system uses to locate and store information about files. For example, the file information **254** may include the date and time a file is created, a date and time the file was last modified, the date and time the files was last accessed and the size of the file.

The monitoring-software logs **256** includes information collected from one or more pieces of monitoring software that are configured to monitor activity on the computer. As an example, the monitoring logs **256** may be generated from a filter driver, a module interfacing with a layer service provider and/or browser helper objects. It should be recognized that the logs **220**, **240-256** depicted in FIG. **2** are merely exemplary and these logs are by no means representative all the potential logs that may be accessed to research origins of pestware and/or suspected-pestware.

Referring next to FIG. **3**, shown is a flowchart depicting a method in accordance with one embodiment. Although reference will be made to FIGS. **1** and **2** for exemplary purposes, it should be recognized that the method described with reference to FIG. **3** is certainly not limited to the specific embodiments described with reference to FIGS. **1** and **2**. As shown, in this embodiment a scan of a computer (e.g., the computer **100**) for pestware is initially carried out (Block **302**). In several embodiments for example, the detection engine **102** scans the file storage device **118**, the operating system registry and executable memory of the protected computer for indicia of pestware (e.g., processes or files matching pestware definitions and/or alterations to a registry entry that are consistent with pestware).

As shown in FIG. **3**, if pestware indicia is found, then recorded information (e.g., the activity log **120** and/or the N files **130**) that may include traces of the pestware is accessed (Block **304**), and traversed to search for information leading to the identification of the source of the pestware (Block **306**). In some instances, the origin of the pestware may be identified by simply referencing one piece of data that is stored in connection with the identification of the pestware source. In other instances, however, it may be necessary to access several pieces of referential data, which may be located in one or more of the files **220-256** before arriving at the identity of the source of the pestware.

As an example, if a pestware file is found on the file storage device **118**, then the cache index **243** of the browser cache **242** may be searched to identify the name of the file, and if the originating URL is stored in connection with the file, the URL is identified as the source of the file. As another example, if a pestware process is identified, a search of the activity log **220** may lead to the identity of a second process that spawned the pestware process, and additional searching of the activity log **220** using the name of the second process may lead to the identification of a pestware file associated with the second process. In turn, a search of the cache index **243** for the name of the pestware file may lead to the URL from which the pestware file was downloaded.

As shown in FIG. **3**, if the source of the pestware is identified (Block **308**), then the source of the pestware is reported (Block **310**). In some embodiments, it is contemplate that several other computers configured in accordance with the protected computer **100** depicted in FIG. **1** may also report sources of pestware to a centralized location where the URLs may be added to a list of "bad URLs." In addition, the identified URLs may be actively searched to learn more about the pestware generated at the sites, which may help generate definitions for the pestware and may provide information about how pestware infections occur.

Referring next to FIG. **4**, shown is a flowchart depicting a method in accordance with another embodiment of the present invention. Again, reference will be made to FIGS. **1** and **2** for exemplary purposes, but it should be recognized that the method described with reference to FIG. **4** is certainly not limited to the specific embodiments described with reference to FIGS. **1** and **2**. As shown, in this embodiment activity on a computer is monitored for indicia of pestware (Block **402**), and if potential pestware-related activity is detected, recorded information related to the activity is searched to identify one or more suspicious objects (e.g., files and/or processes) that are related to the activity (Block **404**). In some embodiments, the shields **120** (described with reference to FIG. **1**) may monitor the protected computer **100** for activities, and if the activity is identified as potential pestware activity (e.g., by the heuristics module **110**), then the research module **108** searches the activity log **120** and/or one or more of the N files for information relating to the pestware-related activity.

As shown in FIG. **4**, recorded information (e.g., one or more of the N files) is then traversed to trace through information related to the suspicious objects that leads to an origin of the suspicious objects (Block **406**). In one embodiment for example, the suspicious activity leads to a search for suspicious processes and/or files related to the activity (e.g., using the activity log **120**), which then leads to a search of one or more of the N files **130** (e.g., the cache index **243**) for an indication of the source of the suspicious process and/or files.

As depicted in FIG. **4**, if the source of the suspicious object(s) is identified (Block **408**), then the source of the suspicious object(s) is then reported (e.g., to a pestware research entity). In this way, the suspicious objects and the web sites originating the suspicious objects may be further researched to establish the extent to which they may be a threat.

As an example of pestware-related activity that may trigger the search for a source of the activity, if a series of particular API calls is made in a pattern known to be associated with pestware, the process(es) making the calls may be identified using, for example, the activity log **120**. In turn, the activity log **120** may be used to identify the file(s) associated with the process(es), and the cache index **243** may be utilized to search for a source of the file(es). It should be recognized that this is merely one example of the type of activity that may trigger backwards researching of logs on a computer, and that patterns in process creation, downloaded files, changes to an operating system registry and browser settings, for example, may trigger a search of the computer's logs.

Referring next to FIG. **5**, shown is a flowchart depicting yet another method in accordance with another embodiment of the present invention. While referring to FIG. **5**, simultaneous reference will be made to FIGS. **1** and **2** for exemplary purposes, but it should be recognized that the method described with reference to FIG. **5** is certainly not limited to the specific embodiments described with reference to FIGS. **1** and **2**.

As shown, in this embodiment a time of interest is initially established based upon a suspicion that pestware has infected a computer (Block **502**). In some embodiments, for example, a user may establish the time of interest based upon events the user observed (e.g., pop-ups or a system crash). In one embodiment, as discussed with reference to FIG. **1**, in response to a user request, the time stamp module **112** may issue a time stamp and initiate research that is related to activity occurring at or around the time of the time stamp. In

variations, the user is provide with an alert in response to the heuristics module **110** identifying an activity that is suspicious and the user is given an option to initiate research at or around the time of interest.

In other embodiments, the time stamp module **112** automatically generates a time stamp in response to a report of suspicious activity (e.g., from the shields **120** or heuristics module **110**).

As shown in FIG. **5**, once a time of interest is established, suspicious activity is identified on the computer based upon the time of interest (Block **504**). The time of interest may be established, for example, to include a time period before the time stamp is issued so that a search for suspicious activity is limited to a particular time period. In some embodiments, the activity log **130** and/or one or more of the N files **130** are accessed and analyzed to determine whether any activity during the time of interest is suspicious (e.g., the activity indicates in some way that it may be associated with pestware). As an example, if any logged information (e.g., in the activity log **130** and/or one or more of the N files **130**) indicates that during the time of interest that, for example, access to the registry was carried out in connection with a downloaded file or the launch of a process, the activities may identified as being suspect and further research relative to the process and the file may carried out.

Beneficially, many of the logs accessed include time-stamped information, which enables an activity that occurred during the time of interest to be readily identified and analyzed either alone or in connection with other activities occurring during the time of interest. As an example, the activity log **220**, the browser history **240**, browser cache **242**, operating system setting **246**, the event log **248**, the debugging log **250** the firewall log **252**, the file information **254** and the monitoring software logs **256** include time stamped information that provides insight into the activities that occurred on the computer during the time of interest.

As depicted in FIG. **5**, once one or more activities are identified as being suspicious (Block **504**), one or more objects (e.g., processes or files) on the computer are associated with the suspicious activity (Block **506**). For example, the research module **108** may search the activity log **120** and/or one or more of the N files for information that associates the suspicious activity to one or more processes and the processes may be related to one or more files.

As shown in FIG. **5**, recorded information on the computer is then traversed in order to trace to an origin of one or more of the objects (Block **508**). For example, a search of one or more of the N files **130** (e.g., the cache index **243**) may be carried out to identify the source of a suspicious process and/or files. Once the source of the suspicious object(s) is identified (Block **510**), the source is then reported (e.g., to a remote research entity)(Block **512**).

It should be recognized that the methods described with reference to FIGS. **3**, **4** and **5** are merely exemplary and are certainly not the only modes of operation that are contemplated. As an example, the establishment of a time of interest, as discussed with reference to FIG. **5**, may be useful in the method described with reference to FIGS. **3** and **4** for identifying information that leads to the source of the pestware or pestware-related activities. Moreover, it is contemplated that aspects from all three of the methods described with reference to FIGS. **3**, **4** and **5** may be combined.

In conclusion, the present invention provides, among other things, a system and method for identifying a source of pestware or suspected pestware on a computer. Those skilled in the art can readily recognize that numerous variations and substitutions may be made in the invention, its use and its configuration to achieve substantially the same results as achieved by the embodiments described herein. Accordingly, there is no intention to limit the invention to the disclosed exemplary forms. Many variations, modifications and alternative constructions fall within the scope and spirit of the disclosed invention as expressed in the claims.

What is claimed is:

1. A method for identifying an origin of activity on a computer that is indicative of pestware comprising:

monitoring, using a kernel-mode driver, the computer for activity that is indicative of pestware, wherein the monitoring includes monitoring API calls and storing a history of at least a portion of the API calls in an activity log;

analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;

identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;

accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and

reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware.

2. The method of claim **1**, wherein the particular externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.

3. The method of claim **2**, wherein the identifying includes accessing an activity log that includes information that relates the activity to the suspected pestware object.

4. The method of claim **1**, wherein the recorded history resides in a browser cache, which includes information about files downloaded to the computer and a source of each of the files.

5. The method of claim **1**, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.

6. A system for identifying a source of activity on a computer that is indicative of pestware including:

an activity monitor configured to monitor API calls and to store a history of at least a portion of the API calls in an activity log, wherein the activity monitor includes a kernel-mode driver adapted to intercept the API calls;

a heuristics module configured to identify an activity on the computer that is indicative of pestware residing on the computer and to analyze the activity to determine whether one or more weighted factors associated with the activity exceeds a threshold; and

a research portion configured to access, in response to a prompt from the heuristics module, a first set of recorded information on the computer that relates the activity to at least one file residing on the computer, and wherein the research portion is configured to access a second set of recorded information on the computer that relates the at least one file to an externally networked source from which the file was received; and

a reporting portion configured to generate a report that identifies the externally networked source of the file and

to report an identity of the externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the externally networked source is a source of pestware.

7. The system of claim **6**, wherein activity monitor is configured to store an identity of each process that made each of the API calls in the stored history, and wherein the activity monitor is configured to store an identity of each file that corresponds to each the processes that made the API calls in the stored history so as to create a relation between each API call and at least one file.

8. The system of claim **6**, wherein the heuristics module is configured to receive information about the API calls from the activity monitor and determine whether the API calls are indicative of pestware.

9. The system of claim **6**, wherein the first set of recorded information and the second set of recorded information reside in the same log file on a file storage device of the computer.

10. The system of claim **6**, wherein the externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.

11. The system of claim **6** wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.

12. A non-transitory computer-readable medium including processor-executable instructions for identifying an origin of activity on a computer that is indicative of pestware, the instructions including instructions for:

monitoring, with a kernel-mode driver, the computer for activity that is indicative of pestware, wherein the instructions for monitoring include instructions for monitoring API calls and storing a history of at least a portion of the API calls in an activity log;

analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;

identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;

accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and

reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware.

13. The non-transitory computer-readable medium of claim **12**, wherein the particular externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.

14. The non-transitory computer-readable medium of claim **12**, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.

* * * * *

Exhibit 8

8287799

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

August 30, 2022

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

PATENT NUMBER: *8,719,932*
ISSUE DATE: *May 6, 2014*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

John Burson
Certifying Officer

US008719932B2

(12) **United States Patent**
Boney

(10) **Patent No.: US 8,719,932 B2**
(45) **Date of Patent: *May 6, 2014**

(54) **BACKWARDS RESEARCHING ACTIVITY INDICATIVE OF PESTWARE**

(75) Inventor: **Matthew L. Boney**, Longmont, CO (US)

(73) Assignee: **Webroot Inc.**, Broomfield, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 142 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/490,294**

(22) Filed: **Jun. 6, 2012**

(65) **Prior Publication Data**

US 2012/0246722 A1 Sep. 27, 2012

Related U.S. Application Data

(63) Continuation of application No. 11/408,146, filed on Apr. 20, 2006, now Pat. No. 8,201,243.

(51) **Int. Cl.**
G06F 21/56 (2013.01)

(52) **U.S. Cl.**
USPC **726/22**; 713/164; 713/202; 713/188; 709/206; 709/224; 380/279; 370/386; 726/27

(58) **Field of Classification Search**
CPC H04L 63/1416; G06F 21/56
USPC 726/22
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,490,352	B2 *	2/2009	Kramer et al.	726/22
8,312,479	B2 *	11/2012	Boillot	719/328
2005/0273858	A1 *	12/2005	Zadok et al.	726/24
2007/0016914	A1 *	1/2007	Yeap	719/328
2007/0174911	A1 *	7/2007	Kronenberg et al.	726/22
2011/0167050	A1 *	7/2011	Fanton et al.	707/698

OTHER PUBLICATIONS

The ethical and legal concerns of spyware|http://utopia.csis.pace.edu/dps/2007/jkile/2005%20-%20Spring/DCS823/Spyware/sipior.pdf|2005|Sipior et al.|pp. 39-49.*

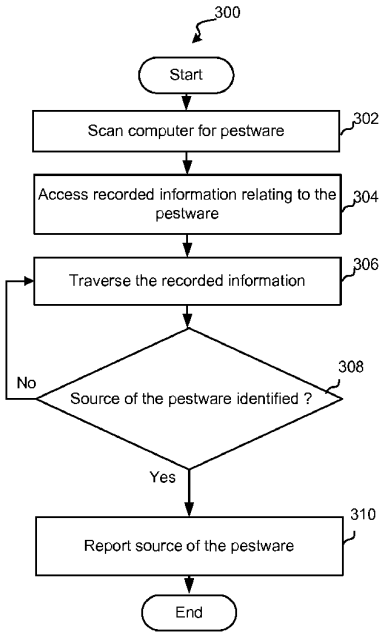
* cited by examiner

Primary Examiner — Mahfuzur Rahman
(74) *Attorney, Agent, or Firm* — Sheridan Ross P.C.

(57) **ABSTRACT**

A system and method for researching an identity of a source of activity that is indicative of pestware is described. In one embodiment the method comprises monitoring, using a kernel-mode driver, API call activity on the computer; storing information related to the API call activity in a log; analyzing, heuristically, the API call activity to determine whether one or more weighted factors associated with the API call activity exceeds a threshold; identifying, based upon the API call activity, a suspected pestware object on the computer; identifying, in response to the identifying the suspected pestware object, a reference to an identity of an externally networked source of the suspected pestware object; and reporting the identity of the externally networked source to an externally networked pestware research entity.

15 Claims, 5 Drawing Sheets



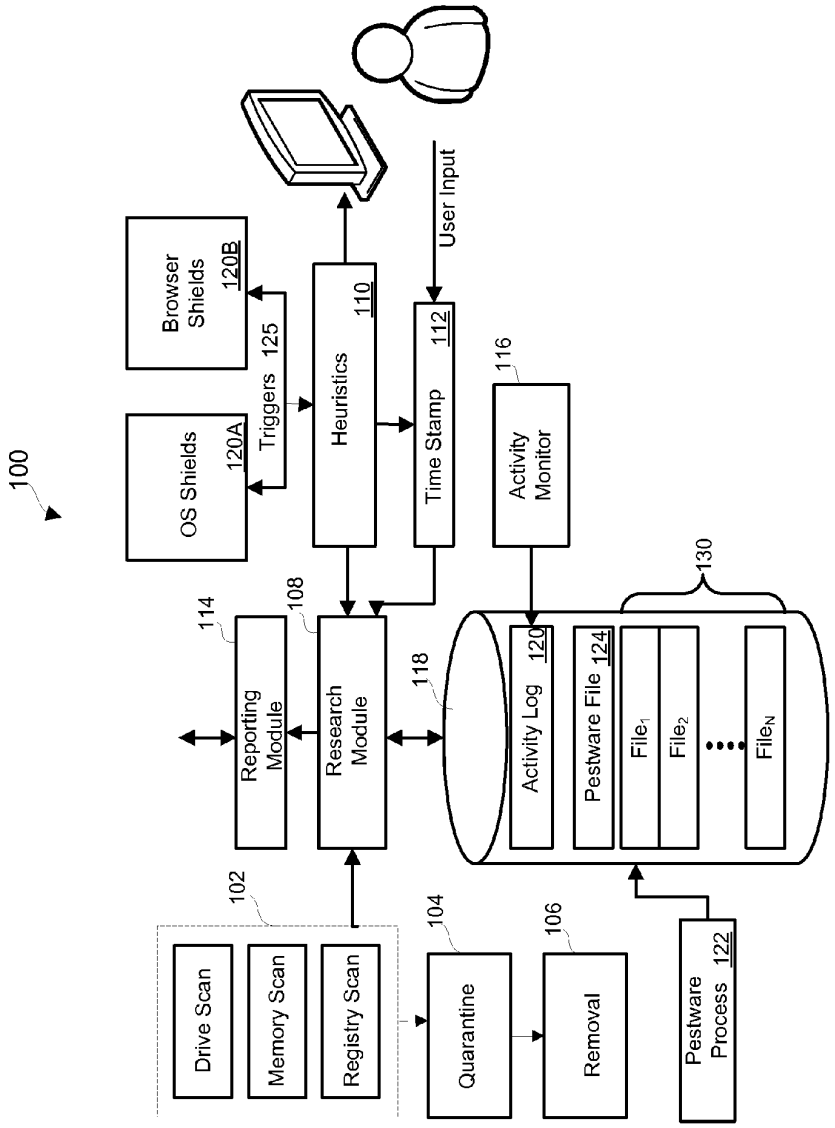

100
User Input
Browser Shields 120B
OS Shields 120A
Triggers 125
Heuristics 110
Time Stamp 112
Activity Monitor 116
130
114
108
118
Reporting Module
Research Module
Activity Log 120
Pestware File 124
File1
File2
FileN
102
104
106
Drive Scan
Memory Scan
Registry Scan
Quarantine
Removal
Pestware Process 122

FIGURE 1

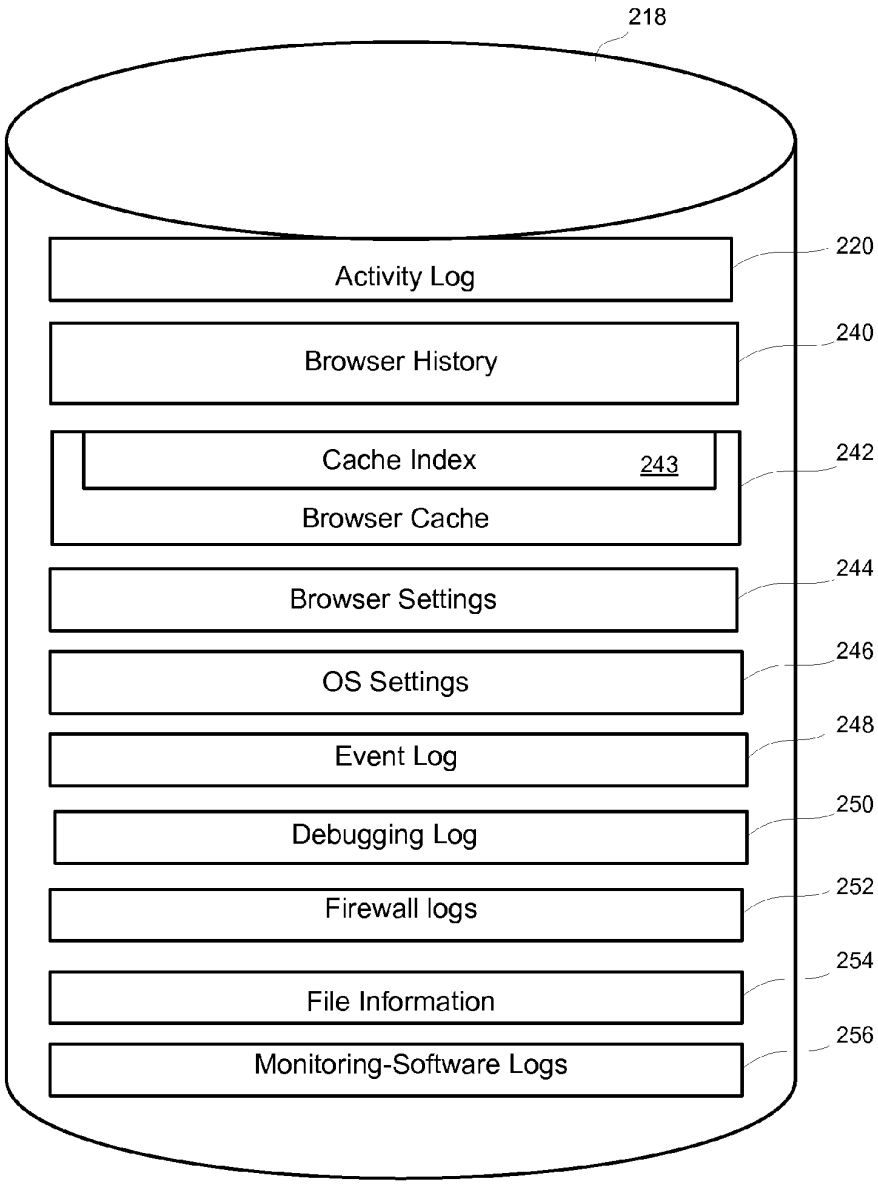

218
Activity Log
220
Browser History
240
Cache Index
243
Browser Cache
242
Browser Settings
244
OS Settings
246
Event Log
248
Debugging Log
250
Firewall logs
252
File Information
254
Monitoring-Software Logs
256

FIGURE 2

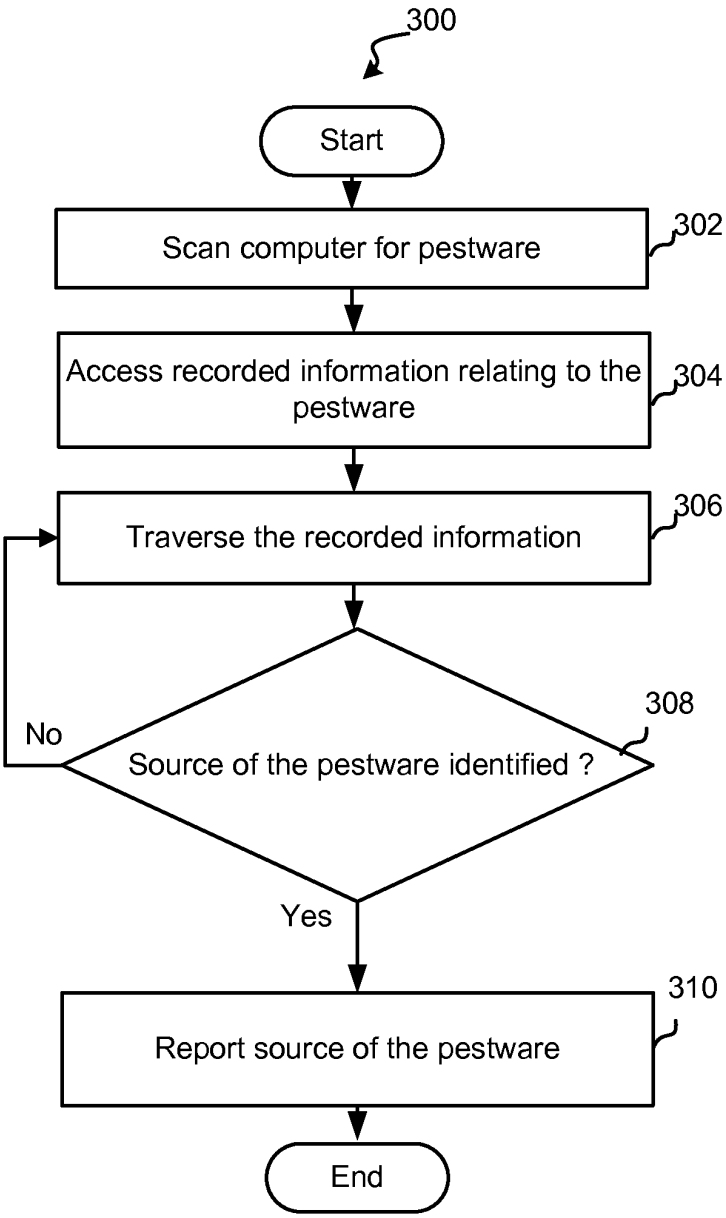

300
Start
Scan computer for pestware
302
Access recorded information relating to the pestware
304
Traverse the recorded information
306
Source of the pestware identified ?
308
No
Yes
Report source of the pestware
310
End

FIGURE 3

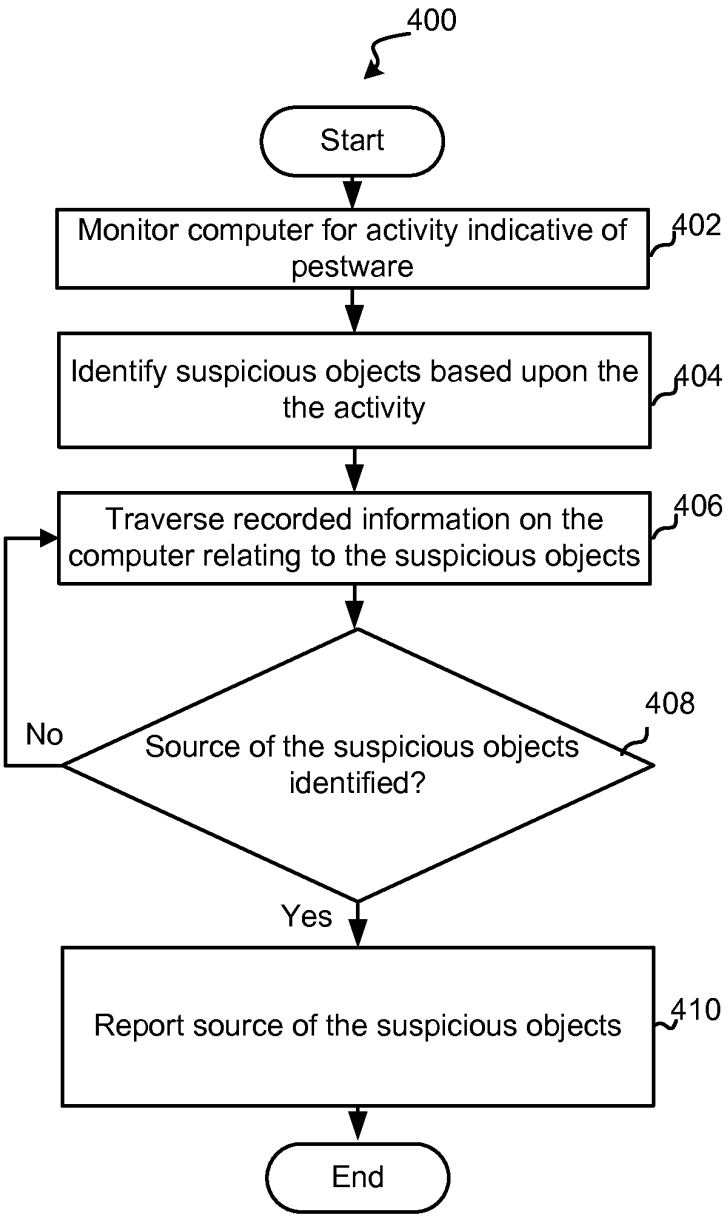

400
Start
Monitor computer for activity indicative of pestware
402
Identify suspicious objects based upon the the activity
404
Traverse recorded information on the computer relating to the suspicious objects
406
Source of the suspicious objects identified?
408
No
Yes
Report source of the suspicious objects
410
End

FIGURE 4

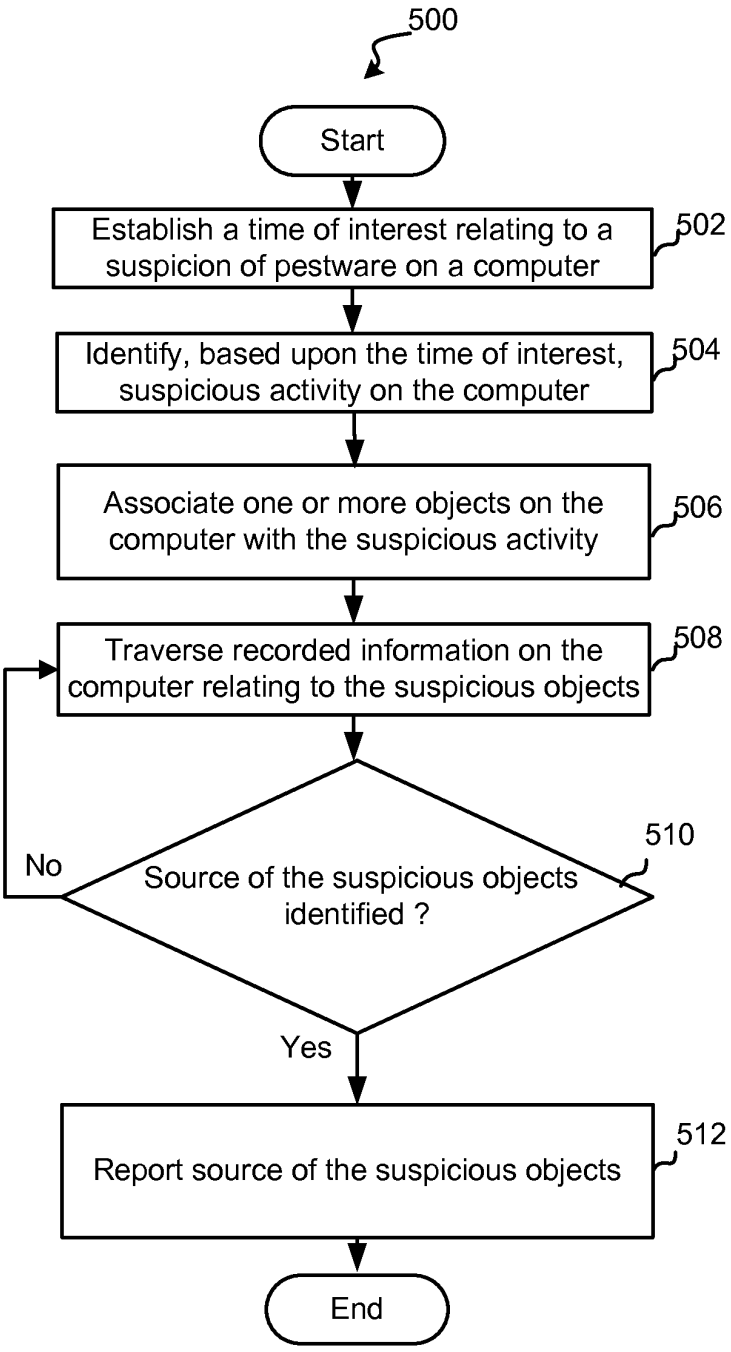

500
Start
Establish a time of interest relating to a suspicion of pestware on a computer
502
Identify, based upon the time of interest, suspicious activity on the computer
504
Associate one or more objects on the computer with the suspicious activity
506
Traverse recorded information on the computer relating to the suspicious objects
508
Source of the suspicious objects identified ?
510
No
Yes
Report source of the suspicious objects
512
End

FIGURE 5

BACKWARDS RESEARCHING ACTIVITY INDICATIVE OF PESTWARE

PRIORITY

The present application is a continuation of commonly-owned and assigned application Ser. No. 11/408,146, entitled Backwards Researching Activity Indicative of Pestware, filed Apr. 20, 2006, which is incorporated herein by reference in its entirety and for all purposes.

RELATED APPLICATIONS

The present application is related to commonly-owned and assigned application Ser. No. 10/956,573, entitled System and Method For Heuristic Analysis to Identify Pestware, which issued as U.S. Pat. No. 7,480,683; application Ser. No. 10/956,574, entitled System and Method for Pestware Detection and Removal which issued as U.S. Pat. No. 7,533,131; application Ser. No. 11/237,291, entitled System and Method for Locating Malware and Generating Malware Definitions; application Ser. No. 11/257,609, entitled System and Method for Kernel-Level Pestware Management; application Ser. No. 11/237,291, entitled Client Side Exploit Tracking; application Ser. No. 11/408,215, entitled Backward Researching Existing Pestware; and application Ser. No. 11/408,145, entitled Backward Researching Time Stamped Events to Find an Origin of Pestware, which are incorporated herein by reference.

COPYRIGHT

A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF THE INVENTION

The present invention relates to computer system management. In particular, but not by way of limitation, the present invention relates to systems and methods for detecting, controlling and/or removing pestware.

BACKGROUND OF THE INVENTION

Personal computers and business computers are continually attacked by trojans, spyware, and adware, collectively referred to as "malware," "spyware" or "pestware." These types of programs generally act to gather information about a person or organization—often without the person or organization's knowledge. Some pestware is highly malicious. Other pestware is non-malicious but may cause issues with privacy or system performance. And yet other pestware is actually beneficial or wanted by the user. Unless specified otherwise, "pestware" as used herein refers to any of these programs that collects information about a person or an organization.

Software is presently available to detect and remove pestware. But as it evolves, the software to detect and remove it must also evolve. Accordingly, current techniques and software for removing pestware are not always satisfactory and will most certainly not be satisfactory in the future. Additionally, because some pestware is actually valuable to a user, pestware-detection software should, in some cases, be able to handle differences between wanted and unwanted pestware.

Current pestware removal software uses definitions of known pestware to search for and remove files on a protected system. These definitions are often slow and cumbersome to create. Additionally, it is often difficult to initially locate the pestware in order to create the definitions. Accordingly, a system and method are needed to address the shortfalls of present technology and to provide other new and innovative features.

SUMMARY OF THE INVENTION

Exemplary embodiments of the present invention that are shown in the drawings are summarized below. These and other embodiments are more fully described in the Detailed Description section. It is to be understood, however, that there is no intention to limit the invention to the forms described in this Summary of the Invention or in the Detailed Description. One skilled in the art can recognize that there are numerous modifications, equivalents and alternative constructions that fall within the spirit and scope of the invention as expressed in the claims.

The present invention can provide a system and method for researching an identity of a source of activity that is indicative of pestware. In one embodiment the method comprises monitoring, using a kernel-mode driver, API call activity on the computer; storing information related to the API call activity in a log; analyzing, heuristically, the API call activity to determine whether one or more weighted factors associated with the API call activity exceeds a threshold; identifying, based upon the API call activity, a suspected pestware object on the computer; identifying, in response to the identifying the suspected pestware object, a reference to an identity of an externally networked source of the suspected pestware object; and reporting the identity of the externally networked source to an externally networked pestware research entity.

As previously stated, the above-described embodiments and implementations are for illustration purposes only. Numerous other embodiments, implementations, and details of the invention are easily recognized by those of skill in the art from the following descriptions and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Various objects and advantages and a more complete understanding of the present invention are apparent and more readily appreciated by reference to the following Detailed Description and to the appended claims when taken in conjunction with the accompanying Drawings wherein:

FIG. **1** illustrates a block diagram of one implementation of the present invention;

FIG. **2** is a block diagram depicting an embodiment of the file storage device of FIG. **1**;

FIG. **3** is a flowchart depicting a method according to an exemplary embodiment;

FIG. **4** is a flowchart depicting another method according to another embodiment;

FIG. **5** is a flowchart depicting yet another method according to yet another embodiment.

DETAILED DESCRIPTION

Referring now to the drawings, where like or similar elements are designated with identical reference numerals throughout the several views, and referring in particular to FIG. **1**, it illustrates a block diagram of one implementation of

the present invention. Shown is a protected computer **100** that includes a detection module **102**, quarantine module **104**, removal module **106**, and shields **120**. In addition, a research module **108** is shown coupled to a heuristics module **110**, a time stamp module **112** and a reporting module **114**.

Each of these modules can be implemented in software or hardware. And if implemented in software, the modules can be implemented in a single software package or in multiple software packages. In addition, one of ordinary skill in the art will recognize that the software can be designed to operate on any type of computer system including WINDOWS and Linux-based systems. Additionally, the software can be configured to operate on personal computers and/or servers. For convenience, embodiments of the present invention are generally described herein with relation to WINDOWS-based systems. Those of skill in the art can easily adapt these implementations for other types of operating systems or computer systems.

Also shown is a file storage device **118** that is coupled to the research module **108** and an activity monitor **116**. In this embodiment the file storage device includes an activity log **120**, a pestware file **124** and a collection of N files **130**. The file storage device **118** is described herein in several implementations as hard disk drive for convenience, but this is certainly not required, and one of ordinary skill in the art will recognize that other storage media may be utilized without departing from the scope of the present invention. In addition, one of ordinary skill in the art will recognize that the storage device **118**, which is depicted for convenience as a single storage device, may be realized by multiple (e.g., distributed) storage devices.

In the exemplary embodiment, the pestware file **124**, corresponds to (e.g., includes data relating to) a pestware process **122** operating in memory. The pestware process **122** is exemplary of pestware processes that are configured to make one or more unauthorized alterations to the computer **100**. For example, the pestware process **122** may make changes to either or both of the browser settings and/or operating system (OS) settings without approval and/or the knowledge of the user.

In accordance with several embodiments, the research module **108** is configured to receive an indication that either known pestware is residing on the computer or activities indicative of pestware have occurred or are occurring on the protected computer. In response, the research module **108** is configured to research the activity log **120** and/or the N files **130**, which include information (e.g., historical logs) relating to events on the computer **100**, to identify a source of the pestware **122**, **124** or pestware-related activities, which may be reported by the reporting module **114** to a centralized data store for subsequent pestware management purposes.

For example, the identities (e.g., I.P. address, URL, email client or program name) of sources (e.g., web sites, email or program) of the pestware may be collected in a centralized data store (not shown) and then subsequently reported to other users. In addition, the sources of the pestware or suspected pestware may be visited to further research how the pestware is being distributed from the URLs and/or to generate new definitions for pestware discovered at these sources.

As described further with reference to FIG. **2**, the N files **130** include one or more files with information that assist the research module **108** in tracing information in the N files **130** to an identity (e.g., URL) of the source of pestware **122**, **124** on the computer **100**. One or more of the N files **130** may be associated with an operating system of the protected computer and/or one or more applications of the protected computer, and may include information such as process IDs, registry entries, file names, cookies and URLs among other information that is used to trace from an identified pestware file, pestware process and/or pestware activity to an originating source (e.g., URL or IP address) of the infection. In one embodiment, one or more of the N files **130** is generated from an application that generates a log of data after examining the computer. An example of such an application is an application distributed under the name HijackThis.

In the exemplary embodiment depicted in FIG. **1**, the research module **108** is configured to receive indications that pestware may be present on the computer from each of the detection engine **102**, the heuristics engine **110** and the time stamp module **112**, but this is certainly not required. In other embodiments, for example, the research module **108** may be configured to communicate only with the detection engine **102** or only with the heuristics engine. In yet other embodiments, the research module **108** may be configured to receive only a time stamp from the time stamp module **112**.

According to several embodiments, pestware-detection functions operating on the protected computer **100** are represented by the detection engine **102**, the quarantine engine **104**, the removal engine **106**, the shields **120** and the heuristic engine **110**. The basic functions of the detection engine **102** is to compare files, processes and registry entries on the protected computer against known pestware definitions and characteristics. When a match is found, in addition to quarantining and removing a pestware object, the detection engine **102** informs the research module **108** of the pestware. Details associated with several embodiments of sweep, quarantine, and removal engines are found in the above-identified application entitled System and Method for Pestware Detection and Removal.

Pestware and pestware activity can also be identified by the shields **120**, which generally run in the background on the computer system. In the exemplary embodiment depicted in FIG. **1**, the shields **120** are divided into the operating system shields **120**A and the browser shields **120**B. The shields **120** are designed to watch for pestware and for typical pestware activity and includes two types of shields: behavior-monitoring shields and definition-based shields.

As an example, the shields **120** monitor the protected computer **100** for certain types of activities that generally correspond to pestware behavior. Particular examples of some of the types of activities that are monitored include a process spawning another process, an alteration to registry entries, communications with remote sites via the Internet, alterations to a start up folder, injection of a DLL into another process, and a change to the browser's home page and/or bookmarks. Some specific examples of shields are disclosed in the above-identified application entitled System and Method for Locating Malware and Generating Malware Definitions. In the exemplary embodiment, the shields **120** inform the heuristics engine **110** about the activities and the heuristics engine **110** determines whether the research module **108** should be informed and/or whether the activity should be blocked.

As an example, the heuristics module **110** may compare the activities on the computer with weighted factors to make decisions relative to activities at the protected computer. Each of these factors may be, for example, associated with a particular activity and each factor may be weighted by the likelihood that the activity is associated with pestware. If the sum of the weighted factors that match the activity history exceed a threshold, then the activity is identified as pestware activity and the heuristics module **110** prompts the research module **108** to initiate research into the origin of the pestware initiating the activity. It should be recognized that this type of heuristics operation is merely one example, and that the heu-

ristics module **110** may use other techniques to analyze activity on the computer. Additional information related to heuristics-based scanning is found in the above-identified applications entitled System and Method For Heuristic Analysis to Identify Pestware and Client Side Exploit Tracking.

In the exemplary embodiment depicted in FIG. **1**, the time stamp module **112** is configured to send a time stamp to the research module **108** in response to a request from a user and/or in response to the heuristics module **110**. In some embodiments for example, the heuristics module **110** provides the user with information about suspect activity on the computer **100** so that the user has the option as to whether or not the research module **108** will attempt to identify the source of the activity.

In other embodiments, the heuristics module **110** prompts the time stamp module **112** to initiate the generation of a time stamp, without user intervention, in response to activity that is indicative of pestware activity. In one implementation, for example, the heuristics module **110** prompts the research module **108** to initiate tracing of the pestware activity to an origin of the pestware that is associated with the activity, and the heuristics module **110** also prompts the time stamp module **112** to send a time stamp to the research module **108** so that the research module **108** is provided with a time reference as well as information about the pestware activities.

In yet other embodiments, the timestamp module **112** operates independently of the heuristics module **110**. For example, the timestamp module **112** may prompt the research module **108** to initiate a review of the activity log **120** and/or one or more of the N files in response to a user request.

The activity monitor **116** in several embodiments monitors activities on the computer and stores information about the activities in the activity log **120** so as to assist the research module **108** in identifying activities associated with pestware and the source of the pestware. In some embodiments, for example, the activity log **120** includes a list of processes that are running on the protected computer and the files that are associated with the processes. Although not depicted in FIG. **1**, the shields **120** may utilize the activity monitor **116** to detect pestware-related events and intercept efforts by pestware to spawn new processes or alter aspects of the protected computer **100**.

In some variations, the activity monitor **116** may inject code into existing processes so that when a process attempts to call a function of the computer's **100** operating system (not shown) the injected code can check the process to be started and raise a flag if the existing process is attempting to create a new pestware process or attempting to alter one or more settings (e.g., a registry setting) of the protected computer **100**.

In other embodiments, the activity monitor **116** is realized by a kernel-mode driver that may be loaded during a boot sequence for the protected computer or anytime later. In these embodiments, the activity monitor **116** is configured to log API calls in the activity log. In many variations for example, when a process (e.g., the pestware process) attempts to spawn a another pestware process or alter a registry entry, the API call utilized by the process is intercepted before it is carried out by an operating system of the protected computer. In this way, the attempt may be logged in the activity log **120** and the process may be analyzed to determine whether it is known to be a pestware-related process. Additional details of use of a kernel-level driver in connection with pestware management may be found in the above identified application entitled: System and Method for Kernel-Level Pestware Management.

It should be recognized that the block diagram in FIG. **1** depicts functional capabilities associated with several embodiments of the present invention. One of ordinary skill in the art will recognize that the functions described with reference to FIG. **1** may be realized by various implementations of software in connection with hardware or hardware alone. In these implementations several functions may be consolidated into a single module, for example, and as a consequence, may appear different from the block diagram in FIG. **1** without departing from the scope of the present invention.

Referring next to FIG. **2**, shown are exemplary resources, also referred to herein as historical logs, that are available to the research module **108** in accordance with one embodiment of the present invention. In general, these resources are logs of historical events that occurred on the computer, and each of the logs includes information that may be used to reference other information when, for example, activities, processes and files are traced so as to determine an origin of the pestware or suspected pestware. As shown, a file storage device **218** in this embodiment includes an activity log **220**, which may be generated by an activity monitor (e.g., the activity monitor **116**) and may include information about processes running on the computer **100** and files corresponding to the processes. In variations, the activity log **220** includes a history of API calls and respective times made by processes running on the protected computer, but this is certainly not required.

In addition, the file storage device **218** includes a browser history **240**, browser cache **242**, browser settings **244**, OS settings **246**, an event log **248**, a debugging log **250**, a firewall log **252**, file information **254** and monitoring software logs **256**. One or more of these exemplary files **240**-**256** may be implemented for one or more of the N files **130** described with reference to FIG. **1**.

The browser history **240** in this embodiment includes a listing of web sites and respective times that the web sites were visited by the user. The browser cache **242** in this embodiment includes files, cookies, and other objects cached in connection with use of a browser of the protected computer (e.g., Internet Explorer or Mozilla (not shown)). As depicted, the browser cache **242** includes a cache index **243** that includes a listing, which associates the content of the browser cache **242** with URLs and time stamps. As discussed further herein, the cache index **243** provides an efficient means for identifying the times objects were retrieved and the URLs that the objects were retrieved from.

The browser settings **244** include information about settings associated with a browser and may include a home page setting and list of user favorites. The browser settings **244** are monitored by the shields **120** for changes. Those settings also contain URLs that may be referenced in time stamped logs, firewall logs, browser histories, etc.

The operating system (OS) settings **246** may include registry entries, a start up folder and other information utilized by an operating system of the protected computer. As discussed further herein, data included in the OS settings **246** may include time stamps, which indicate when changes were made to the settings.

The event log **248** in this embodiment includes a log of events that is maintained by an operating system of the protected computer **100**. For example, the event log **248** may include event information including errors, user log-in history, a listing of processes that have been launched (and the users that launched the processes), path information, information about which process (and which users) accessed a secure area and other information relating to operations of the computer **100**.

Also shown in FIG. **2** is a debugging log **250** that includes application errors, which point to a process and the address where the error occurred. Some techniques employed by pestware forcibly shut down applications, or cause applications to crash when their memory space is injected with pestware code. The infected applications like "explorer.exe" will crash, or some times restart spontaneously. These events/occurrences show up in debugging logs. These are time stamped events, that also reference files on the system.

The firewall log **252** is this embodiment is a collection of information relating to network-related events on the protected computer. The firewall log **252**, may for example, include time stamps of network activities on the protected computer, which may be utilized by the research module **108** to locate pestware or indicia of pestware.

Also shown in the data storage device **218** is a collection of file information **254**, also known as the file system, which includes a database that the operating system uses to locate and store information about files. For example, the file information **254** may include the date and time a file is created, a date and time the file was last modified, the date and time the files was last accessed and the size of the file.

The monitoring-software logs **256** includes information collected from one or more pieces of monitoring software that are configured to monitor activity on the computer. As an example, the monitoring logs **256** may be generated from a filter driver, a module interfacing with a layer service provider and/or browser helper objects. It should be recognized that the logs **220**, **240-256** depicted in FIG. **2** are merely exemplary and these logs are by no means representative all the potential logs that may be accessed to research origins of pestware and/or suspected-pestware.

Referring next to FIG. **3**, shown is a flowchart depicting a method in accordance with one embodiment. Although reference will be made to FIGS. **1** and **2** for exemplary purposes, it should be recognized that the method described with reference to FIG. **3** is certainly not limited to the specific embodiments described with reference to FIGS. **1** and **2**. As shown, in this embodiment a scan of a computer (e.g., the computer **100**) for pestware is initially carried out (Block **302**). In several embodiments for example, the detection engine **102** scans the file storage device **118**, the operating system registry and executable memory of the protected computer for indicia of pestware (e.g., processes or files matching pestware definitions and/or alterations to a registry entry that are consistent with pestware).

As shown in FIG. **3**, if pestware indicia is found, then recorded information (e.g., the activity log **120** and/or the N files **130**) that may include traces of the pestware is accessed (Block **304**), and traversed to search for information leading to the identification of the source of the pestware (Block **306**). In some instances, the origin of the pestware may be identified by simply referencing one piece of data that is stored in connection with the identification of the pestware source. In other instances, however, it may be necessary to access several pieces of referential data, which may be located in one or more of the files **220-256** before arriving at the identity of the source of the pestware.

As an example, if a pestware file is found on the file storage device **118**, then the cache index **243** of the browser cache **242** may be searched to identify the name of the file, and if the originating URL is stored in connection with the file, the URL is identified as the source of the file. As another example, if a pestware process is identified, a search of the activity log **220** may lead to the identity of a second process that spawned the pestware process, and additional searching of the activity log **220** using the name of the second process may lead to the identification of a pestware file associated with the second process. In turn, a search of the cache index **243** for the name of the pestware file may lead to the URL from which the pestware file was downloaded.

As shown in FIG. **3**, if the source of the pestware is identified (Block **308**), then the source of the pestware is reported (Block **310**). In some embodiments, it is contemplate that several other computers configured in accordance with the protected computer **100** depicted in FIG. **1** may also report sources of pestware to a centralized location where the URLs may be added to a list of "bad URLs." In addition, the identified URLs may be actively searched to learn more about the pestware generated at the sites, which may help generate definitions for the pestware and may provide information about how pestware infections occur.

Referring next to FIG. **4**, shown is a flowchart depicting a method in accordance with another embodiment of the present invention. Again, reference will be made to FIGS. **1** and **2** for exemplary purposes, but it should be recognized that the method described with reference to FIG. **4** is certainly not limited to the specific embodiments described with reference to FIGS. **1** and **2**. As shown, in this embodiment activity on a computer is monitored for indicia of pestware (Block **402**), and if potential pestware-related activity is detected, recorded information related to the activity is searched to identify one or more suspicious objects (e.g., files and/or processes) that are related to the activity (Block **404**). In some embodiments, the shields **120** (described with reference to FIG. **1**) may monitor the protected computer **100** for activities, and if the activity is identified as potential pestware activity (e.g., by the heuristics module **110**), then the research module **108** searches the activity log **120** and/or one or more of the N files for information relating to the pestware-related activity.

As shown in FIG. **4**, recorded information (e.g., one or more of the N files) is then traversed to trace through information related to the suspicious objects that leads to an origin of the suspicious objects (Block **406**). In one embodiment for example, the suspicious activity leads to a search for suspicious processes and/or files related to the activity (e.g., using the activity log **120**), which then leads to a search of one or more of the N files **130** (e.g., the cache index **243**) for an indication of the source of the suspicious process and/or files.

As depicted in FIG. **4**, if the source of the suspicious object(s) is identified (Block **408**), then the source of the suspicious object(s) is then reported (e.g., to a pestware research entity). In this way, the suspicious objects and the web sites originating the suspicious objects may be further researched to establish the extent to which they may be a threat.

As an example of pestware-related activity that may trigger the search for a source of the activity, if a series of particular API calls is made in a pattern known to be associated with pestware, the process(es) making the calls may be identified using, for example, the activity log **120**. In turn, the activity log **120** may be used to identify the file(s) associated with the process(es), and the cache index **243** may be utilized to search for a source of the file(es). It should be recognized that this is merely one example of the type of activity that may trigger backwards researching of logs on a computer, and that patterns in process creation, downloaded files, changes to an operating system registry and browser settings, for example, may trigger a search of the computer's logs.

Referring next to FIG. **5**, shown is a flowchart depicting yet another method in accordance with another embodiment of the present invention. While referring to FIG. **5**, simultaneous reference will be made to FIGS. **1** and **2** for exemplary purposes, but it should be recognized that the method described

with reference to FIG. **5** is certainly not limited to the specific embodiments described with reference to FIGS. **1** and **2**.

As shown, in this embodiment a time of interest is initially established based upon a suspicion that pestware has infected a computer (Block **502**). In some embodiments, for example, a user may establish the time of interest based upon events the user observed (e.g., pop-ups or a system crash). In one embodiment, as discussed with reference to FIG. **1**, in response to a user request, the time stamp module **112** may issue a time stamp and initiate research that is related to activity occurring at or around the time of the time stamp. In variations, the user is provide with an alert in response to the heuristics module **110** identifying an activity that is suspicious and the user is given an option to initiate research at or around the time of interest.

In other embodiments, the time stamp module **112** automatically generates a time stamp in response to a report of suspicious activity (e.g., from the shields **120** or heuristics module **110**).

As shown in FIG. **5**, once a time of interest is established, suspicious activity is identified on the computer based upon the time of interest (Block **504**). The time of interest may be established, for example, to include a time period before the time stamp is issued so that a search for suspicious activity is limited to a particular time period. In some embodiments, the activity log **130** and/or one or more of the N files **130** are accessed and analyzed to determine whether any activity during the time of interest is suspicious (e.g., the activity indicates in some way that it may be associated with pestware). As an example, if any logged information (e.g., in the activity log **130** and/or one or more of the N files **130**) indicates that during the time of interest that, for example, access to the registry was carried out in connection with a downloaded file or the launch of a process, the activities may identified as being suspect and further research relative to the process and the file may carried out.

Beneficially, many of the logs accessed include time-stamped information, which enables an activity that occurred during the time of interest to be readily identified and analyzed either alone or in connection with other activities occurring during the time of interest. As an example, the activity log **220**, the browser history **240**, browser cache **242**, operating system setting **246**, the event log **248**, the debugging log **250** the firewall log **252**, the file information **254** and the monitoring software logs **256** include time stamped information that provides insight into the activities that occurred on the computer during the time of interest.

As depicted in FIG. **5**, once one or more activities are identified as being suspicious (Block **504**), one or more objects (e.g., processes or files) on the computer are associated with the suspicious activity (Block **506**). For example, the research module **108** may search the activity log **120** and/or one or more of the N files for information that associates the suspicious activity to one or more processes and the processes may be related to one or more files.

As shown in FIG. **5**, recorded information on the computer is then traversed in order to trace to an origin of one or more of the objects (Block **508**). For example, a search of one or more of the N files **130** (e.g., the cache index **243**) may be carried out to identify the source of a suspicious process and/or files. Once the source of the suspicious object(s) is identified (Block **510**), the source is then reported (e.g., to a remote research entity)(Block **512**).

It should be recognized that the methods described with reference to FIGS. **3**, **4** and **5** are merely exemplary and are certainly not the only modes of operation that are contemplated. As an example, the establishment of a time of interest, as discussed with reference to FIG. **5**, may be useful in the method described with reference to FIGS. **3** and **4** for identifying information that leads to the source of the pestware or pestware-related activities. Moreover, it is contemplated that aspects from all three of the methods described with reference to FIGS. **3**, **4** and **5** may be combined.

In conclusion, the present invention provides, among other things, a system and method for identifying a source of pestware or suspected pestware on a computer. Those skilled in the art can readily recognize that numerous variations and substitutions may be made in the invention, its use and its configuration to achieve substantially the same results as achieved by the embodiments described herein. Accordingly, there is no intention to limit the invention to the disclosed exemplary forms. Many variations, modifications and alternative constructions fall within the scope and spirit of the disclosed invention as expressed in the claims.

What is claimed is:

1. A method for identifying an origin of activity on a computer that is indicative of pestware comprising:

monitoring, using a kernel-mode driver, API call activity on the computer;

storing information related to the API call activity in a log;

analyzing, heuristically, the API call activity to determine whether one or more weighted factors associated with the API call activity exceeds a threshold;

identifying, based upon the API call activity, a suspected pestware object on the computer;

identifying, in response to the identifying the suspected pestware object, a reference to an identity of an externally networked source of the suspected pestware object; and

reporting the identity of the externally networked source to an externally networked pestware research entity, wherein

the identity of the externally networked source is selected from a group consisting of an I.P. address, a URL, an email client and a program.

2. The method of claim **1**, wherein the identifying the suspected pestware object includes accessing an activity log that includes information that relates the activity to the suspected pestware object.

3. The method of claim **1**, wherein the identifying a reference includes accessing at least a portion of a recorded history of externally networked sources.

4. The method of claim **3**, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.

5. A system for identifying a source of activity on a computer that is indicative of pestware including:

a kernel-level monitor configured to monitor API call activity on the computer and to store information related to the API call activity in a log;

a heuristics module configured to analyze the API call activity to determine whether one or more weighted factors associated with the API call activity exceeds a threshold;

a research portion configured to identify, in response to a prompt from the heuristics module, a suspected pestware object on the computer; and

a reporting portion configured to generate a report including a reference to an identity of an externally networked source of the suspected pestware object and to report the

identity of the externally networked source to an externally networked pestware research entity, wherein

the kernel-level monitor is configured to store information about a relationship between at least one API call of the API call activity and a file.

6. The system of claim **5**, wherein the information comprises an identity of at least one process that made at least one API call in the log.

7. The system of claim **5**, wherein the heuristics module is configured to receive information about the API call activity from the kernel-level monitor and to determine whether the API call activity is indicative of pestware.

8. The system of claim **5**, wherein the kernel-level monitor is configured to intercept at least one API call.

9. The system of claim **5**, wherein the identity of the externally networked source is selected from the group consisting of an I.P. address, a URL, an email client and a program name.

10. The system of claim **5**, wherein the research portion is configured to access at least a portion of a recorded history of externally networked sources.

11. The system of claim **10**, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.

12. A non-transitory computer-readable medium including processor-executable instructions for identifying an origin of activity on a computer that is indicative of pestware, the instructions including instructions for:

monitoring, using a kernel-mode driver, API call activity on the computer;

storing information related to the API call activity in a log;

analyzing, heuristically, the API call activity to determine whether one or more weighted factors associated with the API call activity exceeds a threshold;

identifying, based upon the API call activity, a suspected pestware object on the computer;

identifying, in response to the identifying the suspected pestware object, a reference to an identity of an externally networked source of the suspected pestware object; and

reporting the identity of the externally networked source to an externally networked pestware research entity, wherein

the identity of the externally networked source is selected by an identifier selected from a group consisting of an I.P. address, a URL, an email client and a program.

13. The non-transitory computer-readable medium of claim **12**, wherein the instructions for identifying the suspected pestware object include instructions for accessing an activity log that includes information that relates the activity to the suspected pestware object.

14. The non-transitory computer-readable medium of claim **12**, wherein the instructions for identifying a reference include instructions for accessing at least a portion of a recorded history of externally networked sources.

15. The non-transitory computer-readable medium of claim **14**, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.

* * * * *