

(19) World Intellectual Property Organization
International Bureau



(10) International Publication Number
WO 2011/112348 A1

(43) International Publication Date
15 September 2011 (15.09.2011)

- (51) **International Patent Classification:**
G06F 21/00 (2006.01)
- (21) **International Application Number:**
PCT/US201 1/025683
- (22) **International Filing Date:**
22 February 201 1 (22.02.201 1)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
12/719,6 14 8 March 2010 (08.03.2010) US
- (71) **Applicant (for all designated States except US):**
RAYTHEON COMPANY [US/US]; 870 Winter Street,
Waltham, MA 0245 1-1449 (US).
- (72) **Inventors; and**
- (75) **Inventors/ Applicants (for US only):** MC DOUGAL,
Monty, D. [US/US]; 1352 Heidi Drive, Piano, TX 75025
(US). SMITH, Brian, N. [US/US]; 430 Buckingham

Road, Apartment 612, Richardson, TX 75081 (US).
KALKBRENNER, Keven, K. [US/US]; 5733 Caruth
 Haven Lane, Apartment 220, Dallas, TX 75206 (US).
FORD, Bradley, T. [US/US]; 1309 Iron Horse Street,
 Wylie, TX 75098 (US). **JENNINGS, Randy, S.**
 [US/US]; 3 112 Charring Cross, Piano, TX 75025 (US).
STERN, William, E. [US/US]; 3209 Admiral Drive,
 Wylie, TX 75098 (US).

(74) **Agent:** WILLIAMS, Bradley, P.; Baker Botts, LLP,
 2001 Ross Avenue, Suite 600, Dallas, TX 75201 (US).

(81) **Designated States (unless otherwise indicated, for every kind of national protection available):** AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD,

[Continued on next page]

(54) **Title:** SYSTEM AND METHOD FOR HOST-LEVEL MALWARE DETECTION

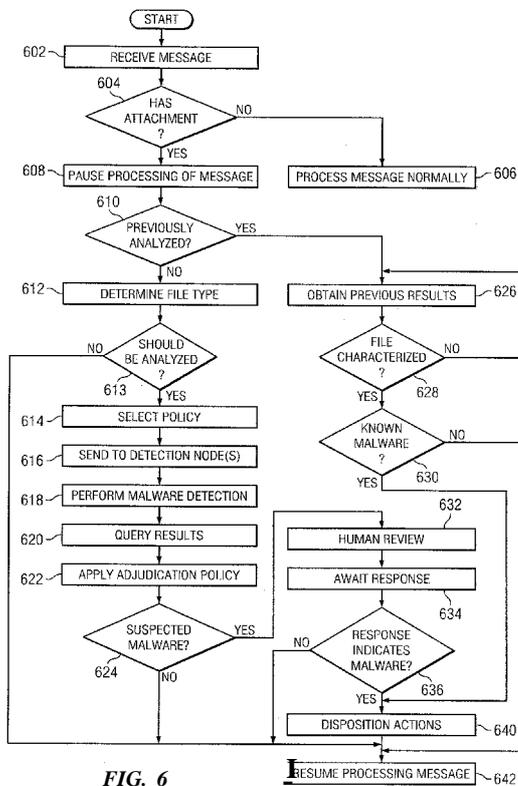


FIG. 6

(57) **Abstract:** According to one embodiment, a computer-implemented method includes: accessing a set of configuration parameters, accessing a set of identifiers of files known not to be malware, and accessing a set of identifiers of files known to be malware. Further, the method includes: comparing a first file to the set of configuration parameters, determining that a first hash of the first file is not in the set of identifiers of files known not to be malware and that the first hash is not in the set of identifiers of files known to be malware, and sending the at least one file and information related to the at least one file to be analyzed for malware. The method includes deleting the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known to be malware after sending the first file.

W^o 2011/112348 A1

SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) **Designated States** (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(H))
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(Hi))

Published:

- with international search report (Art. 21(3))

SYSTEM AND METHOD FOR HOST-LEVEL MALWARE DETECTION

TECHNICAL FIELD

5 This disclosure relates generally to computer security and more particularly to a system and method for host-level malware detection.

BACKGROUND

10 Malware (such as viruses, trojans, and other malicious software) has become increasingly more difficult to protect against. Various methods have been used to combat malware but more sophisticated malware continues to abound. Methods of detection have grown more complex, but have also taken longer to execute as a result of this complexity. Malware protection at host systems are susceptible to being detected and, thus, their efforts thwarted. Current, host-level malware protection
15 systems search for files they can detect as malware using signatures or other forms of identifying known malware. This leaves open the possibility for zero-day attacks.

SUMMARY

20 According to one embodiment, a computer-implemented method includes accessing a set of configuration parameters. The method also includes accessing a set of identifiers of files known not to be malware. In addition, the method includes accessing a set of identifiers of files known to be malware. Further, the method includes determining that a first file in a first host of a plurality of hosts should be analyzed by comparing the first file to the set of configuration parameters. Also, the
25 method includes generating a first hash of the first file in response to determining that a first file should be analyzed. In addition, the method includes determining that the first hash is not in the set of identifiers of files known not to be malware and that the first hash is not in the set of identifiers of files known to be malware. Further, the method includes determining that the first file is not present in a set of hosts of the
30 plurality of hosts in response to determining that the first hash is not in the set of identifiers of files known not to be malware and is not in the set of identifiers of files known to be malware. The method also includes sending the at least one file and

information related to the at least one file to be analyzed for malware in response to determining that the first file is not present in a set of hosts of the plurality of hosts. Further, the method includes deleting the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known
5 to be malware after sending the first file.

In various embodiments, identifiers of files known to be malware may include hashes of files known to be malware. Also, identifiers of files known not to be malware may include hashes of files known not to be malware. In addition, determining that the first file should be analyzed may include determining that the
10 first file should be analyzed in response to detecting that the first file has entered the first host through a removable, computer-readable, storage medium. Further, information related to the first file may include metadata related to the first file. Deleting the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known to be malware may include
15 using secure deletion techniques.

According to one embodiment, a system for malware detection includes at least one computer-readable storage medium comprising instructions that, when executed by at least one processor, are operable to access a set of identifiers of files known not to be malware. The instructions are also operable to access a set of
20 identifiers of files known to be malware. In addition, the instructions are operable to generate at least one hash of at least one file associated with a first process running on a host of a plurality of hosts. Further, instructions are operable to determine that the at least one hash is not in the set of identifiers of files known not to be malware and that the at least one hash is not in the set of identifiers of files known to be malware.
25 The instructions are also operable to monitor the first process in response to determining that the at least one hash is not in the set of identifiers of files known not to be malware and that the at least one hash is not in the set of identifiers of files known to be malware. Also, the instructions are operable to send the at least one file, information related to the at least one file, and information related to the first process
30 to be analyzed for malware in response to comparing a system call associated with the first process to a set of rules.

Depending on the specific features implemented, particular embodiments may exhibit some, none, or all of the following technical advantages. Entire file systems of hosts may be scanned rapidly and without utilizing a large amount of resources. Deletion (including, in some cases, secure deletion) of files related to the scan may avoid detection by malware attempting to thwart scanners. This may also prevent malware from discovering information that is being used to detect the malware's presence. Other technical advantages will be readily apparent to one skilled in the art from the following figures, description and claims.

10 BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram illustrating one embodiment of a system that is operable to detect malware;

FIGURE 2 illustrates one embodiment of a control module;

FIGURE 3 illustrates one embodiment of a detection subsystem;

15 FIGURE 4 illustrates one embodiment of a disposition subsystem;

FIGURE 5 illustrates an example computer system suitable for implementing one or more portions of particular embodiments;

FIGURE 6 is a flowchart illustrating one embodiment processing an e-mail attachment in a malware detection system operating in an active mode;

20 FIGURE 7 is a flowchart illustrating one embodiment of processing an e-mail attachment in a malware detection system operating in a passive mode;

FIGURE 8 is a flowchart illustrating one embodiment of a malware detection system dynamically changing between an active and passive mode;

25 FIGURE 9 is a flowchart illustrating one embodiment of dynamic scheduling in a malware detection system;

FIGURE 10 is a flowchart illustrating one embodiment of applying a behavior-based malware detection scheme to a file;

FIGURE 11 is a block diagram illustrating one embodiment of a system that is operable to detect host-level malware;

30 FIGURE 12 is a flowchart illustrating one embodiment of scanning a file system of a host; and

FIGURE 13 is a flowchart illustrating one embodiment of monitoring a host.

DETAILED DESCRIPTION

FIGURE 1 is a block diagram illustrating one embodiment of system 100 that is operable to detect malware. The block diagram includes external network 110 and internal network 120. Internal network 120 couples ingest block 130, scheduler block 140, detection block 150, disposition block 160, analysis console 170 and server 180. By being coupled to internal network 120, blocks 130-180 may communicate with each other. In various embodiments, communication between blocks 130-180 utilizing internal network 120 may be indirect. For example, one or more of blocks 130-180 may be in a demilitarized zone (DMZ) that may limit direct communication between blocks 130-180. In various embodiments, blocks 130-180 may represent hardware, software or a combination of the two. Blocks 130-180 may be implemented on the same or different hardware in various suitable combinations. Further details regarding the implementation and/or operation of blocks 130-180 may be determined by referencing FIGURES 2-10 and their accompanying descriptions below.

Networks 110 and 120, in some embodiments, may be communicative platforms operable to exchange data or information. In various embodiments, the networks may include packet data networks offering communication interfaces or exchanges between any two nodes in system 100. Networks 110 and 120 may each be or include any local area network (LAN), metropolitan area network (MAN), wide area network (WAN), wireless local area network (WLAN), virtual private network (VPN), intranet, Internet, plain old telephone system (POTS), or any other appropriate architecture or system that facilitates communications in a network or telephonic environment, including a combination of any networks or systems described above. The networks may also include wireless technology such as WiMAX, LTE, or any of a variety of cellular data networks. In various embodiments, networks 110 and 120 may each include firewalls or other equipment or software that create security zones and/or secure networks such as DMZs. One or more of blocks 130-180 may be located in one more of such security zones and/or secure networks in some embodiments. For example, analysis console 170 may be in a DMZ that is separate

from ingest block 130 and detection block 150. As another example, ingest block 130 may be in a different security zone than detection block 150.

As described above, embodiments of the present invention are operable to detect malware. In particular, embodiments of the present invention are operable to
5 detect malware in one or more files. For purposes of this disclosure, a file may include one or more of the following in any suitable combination: (1) a data file; (2) a data record; (3) an email message; (4) an attachment to an email message; (5) a Uniform Resource Locator (URL) or other suitable address identifier; and (6) any other suitable target object for which malware detection may be appropriate.

10 In operation, in various embodiments, ingest block 130 may receive a file or set of files from network 110. In some embodiments, a file may include several files embedded within the file. For example, a ZIP file or other archive or compressed file type, may be received by ingest block 130. Ingest block 130 may extract and cause to be analyzed the files that are contained within the received ZIP file. Ingest block 130
15 may analyze the file(s) and determine one or more tests that should be performed on the file(s) in order to determine whether the file is malware. Detection block 150 may perform the test(s) that were determined by ingest block 130 in accordance with scheduling performed by scheduler block 140. Disposition block 160 may utilize the results of the tests performed by detection block 150 to determine what should be
20 done with the file(s). In some cases, disposition block 160 will send the file(s) to analysis console 170 for further review by a human analyst. In various embodiments, disposition block 160 may respond to the results of detection block 150 regardless of whether it sends the file(s) to analysis console 170. For example, disposition block 160 may determine that the file(s) should be quarantined and send the file to server
25 block 180 to quarantine the file. In some embodiments, disposition block 160 may also determine that the file(s) are not malware and may perform suitable corresponding actions with nodes coupled to external network 110. For example, disposition block 160 may cause the delivery of a message to which the analyzed file(s) were attached in response to the determination by disposition block 160 that the
30 file(s) are not malware.

Some or all of system 100, in various embodiments, may be utilized to detect malware in a variety of contexts. In some embodiments, accessing a "file" may

comprise accessing one or more URLs. For example, aspects of system 100 may be used to detect malware in a messaging environment. Suitable messaging environments include, but are not limited to, an e-mail environment, an instant messaging environment, a peer-to-peer messaging environment, a chat environment, or other messaging environments. Aspects of system 100, in various embodiments, may also be used in contexts where files are uploaded. For example, some or all of system 100 may be utilized in File Transfer Protocol (FTP) sessions, Wiki sessions, online message boards (such as forums), or other interactive environments. Aspects of system 100 may also be used in a Web browsing context. In such a context, some or all of system 100 may analyze requested Web pages and determine if they are malicious. System 100 may be used to detect malware in these and other contexts utilizing suitable configurations of ingest block 130. For example, if the context is an e-mail context, then ingest block 130 may be configured as a Mail Transfer Agent (MTA) that is able to handle e-mail messages. Configured in this manner, ingest block 130 may analyze a message, an attachment to the message, or both the message and any attachment(s) to the message. When some or all of system 100 operates in a context where files are uploaded, such as in an FTP session, ingest block 130 may be configured to access the file(s) associated with the FTP session and perform analysis on the file(s). When aspects of system 100 are used in a Web browsing context, ingest block 130 may be configured to retrieve an address associated with an Internet site or files associated with the Internet site and analyze such items.

In some embodiments, some or all of system 100 may be provided as a service to various agents. For example, an agent analyzing traffic passing through a particular boundary of a network may transmit certain traffic (such as one or more files) to aspects of system 100 for analysis and these or other aspects of system 100 may report to the agent the results of the analysis. As another example, an agent residing on a workstation or laptop computer used by end users may similarly transmit one or more files to some or all of system 100 for analysis and receive the results of the analysis. A particular example of such an agent is a client-located module capable of analyzing the downloaded contents of web-based email (e.g., GMAIL by GOOGLE; HOTMAIL by WINDOWS, or any other suitable web-based email program). As another example, a collaborative environment such as one

associated with a cross-domain document management system (e.g., RAYTHEON 's CHAIN environment) may utilize this service to check files hosted on the platform for malware. In some embodiments, aspects of system 100 may be offered as a service using Web service technology, such as Javascript-based Application Programming Interfaces (APIs), Simple Object Access Protocol (SOAP) Remote Procedure Calls (RPCs), and/or other suitable network based APIs.

System 100, in various embodiments, may be configured to operate in one or more of an active mode, a passive mode, and a hybrid mode. Each of these example modes is described below in greater detail.

When operating in an active mode, system 100 may delay one or more operations while the malware process is proceeding, depending on the context in which system 100 is operating. For example, as discussed above, system 100 may operate in a messaging context such as analyzing e-mail messages. If in the active mode, system 100 may prevent the delivery of an e-mail until an attachment to the e-mail has been analyzed for malware using blocks 140-180. As another example, when system 100 is operating in an active mode in a context where files are being uploaded, system 100 may stop a file from being uploaded until the malware detection process applied to the file has been completed. Hence, in some embodiments, if a file was being uploaded to a database, the file would not be entered into the database and made available to other users or processes until the file has been analyzed by the malware detection scheme of system 100. If system 100 operates in a Web browsing context and is operating in the active mode, an Internet site may not be transmitted to a Web browser requesting the site until the site has been fully analyzed for malware by system 100. In some embodiments, ingest block 130 may be used to prevent the completion of operations until malware detection has been completed.

System 100, in various embodiments, may operate in a passive mode. While in the passive mode, system 100 may not prevent the operation of processes while a file is being detected for malware. For example, when system 100 is handling e-mail messages or their attachments and operating in the passive mode, the e-mails may continue to be processed and delivered even though the malware detection process has not been completed. As another example, when system 100 operates in the passive mode and is operating on files in a file uploading context, a file to be analyzed for

malware may be uploaded, stored, and made available to other users or processes even though the system 100 has not completed analyzing the file for malware.

System 100, in various embodiments, may operate in a hybrid mode, which may comprise an ability incorporate aspects of both the passive and active modes. In one example hybrid mode, system 100 may operate in a hybrid mode for detection purposes. When a file is determined to be known malware, system 100 may switch to an active mode, preventing or delaying one or more operations while the malware process is proceeding, depending on the context in which system 100 is operating. As a particular example, when in the hybrid mode, system 100 may allow emails to be processed and delivered during the detection process; however, when an email is determined to be or include known malware, system 100 may prevent the email and/or its contents from being delivered in an active manner.

FIGURE 2 illustrates one embodiment of control module 200. Control module 200 includes ingest module 210 and scheduler module 250. Ingest module 210 includes file type module 220, duplicate module 230, and policy module 240. Scheduler module 250 includes schedule module 260, and monitor module 270. Ingest module 210 is coupled to jobs database 290 and ingest policy database 295. Scheduler module 250 is also coupled to jobs database 290. Using ingest module 210 and scheduler module 250, control module 200 may receive a file and schedule the file for one or more malware detection schemes as further described below.

File type module 220 may be implemented using software and/or hardware and may reside on the same hardware that ingest module 210 resides upon or on different hardware. File type module 220, in some embodiments, may be configured to determine the type of file that ingest module 210 receives. File type module 220 may determine the type of a file using any suitable methods in combination or separately. For example, file type module 220 may examine an extension associated with the file to determine the type of the file. As another example, file type module 220 may examine portions of the file in order to determine its type. File type module 220 may look at characters in a header of a file to determine its type. Such characters may be referred to as magic numbers or magic bits. In this manner, in some embodiments, file type module 220 may detect the correct type of the file even if the file's extension has been removed or changed. As another example, for certain types

of files (e.g., MICROSOFT OFFICE files), may determine the file type based on both magic number(s) and the file extension, possibly examining the magic number(s) prior to examining the file extension.

In various embodiments, duplicate module 230 may be implemented using software and/or hardware and may reside on the same hardware as ingest module 210 or on different hardware. Duplicate module 230 may be configured to determine whether a file has been previously analyzed for malware. Duplicate module 230 may use one or more techniques to determine if a file has been previously analyzed for malware. For example, duplicate module 230 may perform generate one or more hashes of the file (such as a checksum, an MD5 hash, and/or a SHA1 hash). These value(s) may be compared to a database containing hash values of previously analyzed files. If the hash value is in the database duplicate module 230 may determine that the file has been previously analyzed. If the hash value is not present in the database duplicate module 230 may determine that the file has not been previously analyzed. In some embodiments, duplicate module 230 may use the name of the file and/or its extension, as well as variations on those items, to determine if the file has been previously analyzed.

If it is determined that a file has been analyzed previously, malware detection schemes may not be applied to the file; instead, the results of the previous analysis of the file may be determined using a database that contains results of a previous analysis of the file. If the results indicate that the file is known not to be malware, then the analysis of the file may end. If it is determined that the file was previously determined to be malware, then it may be determined that the file should be quarantined. If it is determined that the file has been previously received and is currently going through analysis (including review by human analysts), then action may be taken once the outcome of the ongoing analysis is known. In some embodiments, this may allow for more efficient utilization of the resources that perform the malware detection schemes on the files and may reduce the workload of the human analyst.

Policy module 240, in various embodiments, may be implemented using software, hardware or a combination of the two. Policy module 240 may reside on the same hardware as ingest module 210 or it may reside on separate hardware. In

some embodiments, policy module 240 may be used to determine whether a file received by ingest module 210 should undergo malware detection and/or which malware detection schemes should be applied to a file received by ingest module 210. Policy module 240 may be configured to access ingest policy database 295 to make such determinations, in various embodiments.

For example, policy module 240 may determine that a file received by ingest module 210 is a plain text file. Policy module 240 may then access ingest policy database 295 and retrieve one or more policies associated with plain text files. A retrieved policy may indicate that plain text files are not to be analyzed for malware. As a result, the plain text file may be ignored.

As another example, policy module 240 may determine that a file received by ingest module 210 is a document created by the MICROSOFT WORD application. Policy module 240 may then access ingest policy database 295 and retrieve one or more policies associated with MICROSOFT WORD documents. Policy module 240 may examine the retrieved policy or policies and determine that the received file should be analyzed for malware. Policy module 240 may also examine the retrieved policy or policies and determine the malware detection schemes that should be applied to the MICROSOFT WORD document. Policy module 240 may then create entries in jobs database 290 consistent with the determined malware detection schemes.

In some embodiments, policy module 240 may also determine that the file should be accessed by different versions of software. Versions of software may include different versions of application used to access the file or may include using different versions of an operating system used to access the file. For example, if the file is a MICROSOFT WORD document, it may be determined that a different version of MICROSOFT WORD should be used to access the file. As another example, if the file is an image file such as a JPEG file, then it may be determined to access the file using different applications such as ADOBE PHOTOSHOP and MICROSOFT INTERNET EXPLORER. As another example, if the file is a URL, then it may be determined to access the file using different vendor's version of a browser such as MICROSOFT INTERNET EXPLORER, MOZILLA FIREFOX, APPLE SAFARI, and any other suitable browsers. In some embodiments,

determining to access the file using a variety of versions of software may provide for a more robust malware detection scheme.

Schedule module 260, in various embodiments, may be implemented using software, hardware or a combination of the two. Schedule module 260 may reside on the same hardware as scheduler module 250 or it may reside on separate hardware. Schedule module 260, in some embodiments, may determine the order in which jobs in jobs database 290 are performed. Schedule module 260 may maintain a list of detection nodes that can receive the jobs and may assign the jobs using any suitable method. For example, schedule module 260 may use a first-in-first-out (FIFO) algorithm to assign jobs to detection nodes. Jobs may also be prioritized. For example, schedule module may use a FIFO approach to schedule jobs initially, but it may be determined to prioritize one job over another in response to results provided by the detection nodes. Schedule policies may be used by schedule module 260 to determine how to schedule jobs and how to prioritize jobs. Priorities for jobs, in some embodiments, may also be determined based on the context associated with the file. For example, if the file undergoing analysis was part of an e-mail attachment, it may be prioritized higher than other files.

In some embodiments, when adding a new malware detection scheme, policies used by schedule module 260 may need to be modified. The interface may require information to be entered regarding how the malware detection scheme should be applied. Such information may be entered using a tabbed interface, a wizard-style interface, or other interfaces for entering information. The information required to be entered may include how jobs should be prioritized based on responses from detection nodes, the context associated with the file, what detection nodes are involved in implementing the malware detection scheme, and/or other items associated with applying a malware detection scheme.

Monitor module 270, in various embodiments, may be implemented using any suitable combination of software and hardware. Monitor module 270 may reside on the same hardware as scheduler module 250 or it may reside on separate hardware. Monitor module 270 may be able to monitor the capacity of detection nodes that apply various malware detection schemes to a file. In some embodiments, monitor module 270 may query the detection nodes to determine their status. Example status

responses may include "offline" and/or "busy." In some embodiments, monitor module 270 may determine if a detection node has taken too long to analyze a file. Monitor module 270 may be able to reboot or restore aspects of detection nodes. In various embodiments, monitor module 270 may be able to reconfigure a detection node so that the detection node may perform a different type of analysis. This reconfiguration may be performed in response to the types of files that are scheduled to be analyzed. In some embodiments, this reconfiguration capability may provide an advantage in that detection nodes may be utilized efficiently and files may be analyzed faster.

Monitor module 270 may also be able to determine when the capacity of the detection nodes reaches a critical threshold. Schedule module 260 and/or monitor module 270 may ensure that detection nodes do not pass critical thresholds. Monitor module 270 may also be able to determine when the load on the detection nodes decreased below a critical threshold. Monitor module 270 may also be able to establish various thresholds regarding the capacity of detection nodes and may be able to determine when the load of the various detection nodes has gone below any of the thresholds. For example, the ability to monitor thresholds and take appropriate action in response to the monitoring may be particularly useful in determining when to switch between one or more modes of operation, such as the active, passive, and hybrid modes. As a particular example, if a particular detection node is determined to be too busy (e.g., exceeds a capacity threshold) and that particular detection node is in active mode, it may be appropriate to switch the particular detection node to passive mode to avoid significant processing delays.

Databases 290 and 295 may be implemented using a variety of database structures. The databases may be row-oriented or column-oriented. They may include relational databases and/or object-oriented databases. Databases 290 and 295 may be indexed and accessed in a variety of manners, including by utilizing an operating system or through one or more networks. In some embodiments, databases 290 and 295 may be implemented in one database such that separate tables within the database may be used to store the information contained in databases 290 and 295.

In some embodiments, jobs database 290 stores jobs related to apply malware detection schemes to a file. A job stored in the database may include information

such as an identifier associated with the file, information regarding what type of malware detection test that should be applied to the file, and the priority associated with the job. Jobs database 290 may be configured so that it can be accessed by ingest module 210 and scheduler module 250. Ingest module 210 may insert jobs into the job database and scheduler module 250 may read the jobs database to determine which jobs need to be assigned to malware detection nodes.

In some embodiments, ingest policy database 295 may contain a set of policies that determine what malware detection schemes should be applied to various types of files. The policies may be applied by type of file or other suitable criteria. Malware detection schemes employed by the policies may include classification schemes (such as recognizing generic suspect patterns), signature-based detection schemes (such as those looking for specific byte patterns), meta-data detection schemes, as well as behavior-based detection schemes. The behavior-based detection scheme specified by the policies may include accessing the file in a virtual or real environment.

In various embodiments, the policies specify the type of access applied to the files based on the file type. For example, if the file type is an executable file type accessing the file may include executing the file within a virtual environment. As another example, if the file type is a MICROSOFT WORD document the policy may specify that accessing the file within the virtual environment includes opening the file using MICROSOFT WORD or using various versions of MICROSOFT WORD or using various versions of other word processors that are also able to access a MICROSOFT WORD document. If handling URLs, for example, accessing the file may include using one or more different browsers, such as MOZILLA FIREFOX and OPERA, to access a web page located at the address provided by the URL. A variety of suitable applications may be used to access the files. Policies may also specify different types of virtual environments that should be used when accessing the file. For example, policies may specify that the file should be accessed within one or more versions of the WINDOWS operating system, within a version of an operating system using the LINUX kernel, within a version of the MAC OS operating system, within versions of operating systems using the BSD kernel, within a version of an embedded operating system, within a version of a non-conventional operating system (e.g., a mobile operating systems such as those used in smart phones), and/or within other

versions of any other suitable operating systems or computing environments. Other suitable forms of malware detection schemes can be stored and applied in the policies.

In some embodiments, when adding a new malware detection scheme, an interface may be provided to modify one or more policies of ingest policy database 295. The interface may require information to be entered regarding how the malware detection scheme should be applied. Such information may be entered using a tabbed interface, a wizard-style interface, or other suitable interfaces for entering information. The information required to be entered may include: the file types that should be associated with the new malware detection scheme, identifiers associated with the detection nodes that implement the new malware detection scheme, jobs to be carried out when implementing the new malware detection scheme, and/or other items associated with applying a malware detection scheme. In some embodiments, a policy may be developed that handles files for which the file type is unknown or files for which the file type is not associated with any specific policy. Suitable malware detection schemes may be applied to such files, such as one or more of classification schemes, signature-based schemes, and behavior-based schemes.

In operation, control system 200 may be able to receive a file for malware detection and use ingest module 210 to determine what malware detection schemes should be applied to the file. File type module 220 may determine the file type and duplicate module 230 may determine whether or not the file has been previously analyzed. If the file has not been previously analyzed, policy module 240 may access a policy in ingest policy database 295. Policy module 240 may use the accessed policy to determine a set of malware detection schemes that should be applied to the file based on the file type. Policy module 240 may then enter jobs into jobs database 290 in order to implement the determined malware detection schemes. Scheduler module 250 may then schedule and monitor the application of the determined malware detection schemes to the file. Schedule module 260 may assign the jobs in jobs database 290 to one or more detection nodes.

Monitor module 270 may monitor the load applied to the detection nodes and determine if the load has surpassed a critical value or threshold. In response to passing the threshold, monitor module 270 may report this occurrence. As a result, the system may switch from operating in an active mode to operating in a passive

mode. This may be advantageous, in various embodiments, because the time it takes to complete an analysis of a file may depend on the load present on the detection nodes. For example, if the system is operating in an active mode and files are entering the system at a rate above a particular threshold, messages or other services may be disrupted causing a lower quality of service. By switching into a passive mode, the system may allow the analysis of the file to proceed without lowering the quality of service of services associated with the file being analyzed. For example, certain malware detection schemes may proceed in real-time (e.g., signature-based or classification-based analysis) while others (e.g., behavior-based analysis) may be delayed. If the detection schemes proceeding in real-time did not cause an alert, then other analyses may be delayed. If the detection schemes proceeding in real-time did cause an alert, then other jobs associated with the file may be given greater priority in the jobs queue. If the load on the detection nodes falls below a critical threshold, monitor module 270 may report this occurrence. In response, the system may enter into an active mode of operation which may allow, in some embodiments, for greater security since operations associated with the context of the file are delayed while the analysis of the file is proceeding. As another example, in a hybrid mode, files determined to be known malware may be delayed or removed from the mail stream while files determined not to be known malware may not be delayed or removed from the mail stream.

FIGURE 3 illustrates one embodiment of detection subsystem 300. Detection subsystem 300 includes detection nodes 310, 320 and 330 and network agent 340. Detection node 310 includes hypervisor 312, one or more guest operating systems (GOS) 314 and detection agent 316. Subsystem 300 may be used to apply malware detection schemes to a file (including, for example, a URL).

In some embodiments, detection nodes 310-330 may conform to an interface standard for applying malware detection. Such an interface may include standards for one or more of the following: specifying the file (including, possibly, a URL) that is to be analyzed configuration parameters for applying the detection scheme, time limit for completing the analysis, format of results, specifying the reason for indicating that an analyzed item is suspect, providing log files, and other suitable items involved with applying malware detection schemes. In some embodiments, having such an interface

may be advantageous because it may allow policies to call for the application of malware detection schemes without having to give precise parameters based on the configuration of the detection node. In this manner, in various embodiments, new detection schemes may be added to the system without needing to recode various parts of the system since the detection node applying the new malware detection scheme would conform to the interface standard. For example, to add a new malware detection scheme, the detection node applying the new malware detection scheme may be configured to conform to the interface standard by being configured to receive files for analysis in the same or similar manner as other configuration nodes applying other malware detection schemes. In addition, for example, the configuration node applying the new malware detection scheme may be configured to report the results of applying the new malware detection scheme in the same or similar manner as other configuration nodes applying other malware detection schemes. This, in some embodiments, may be advantageous in that it allows for the system to adapt to new malware detection schemes.

Detection nodes 310-330 may be implemented on a variety of types of hardware. They may be configured in a blade architecture or on physical hosts. Detection nodes 310-330 may also be configured utilizing clusters or other suitable distributed computing architectures. Detection nodes 310-330 may utilize virtualization or may themselves be virtual machines. Detection nodes 310-330 may be used to apply a variety of malware detection schemes to a file (which, in some embodiments, may include one or more URLs). In some embodiments, detection nodes 310-330 may be specialized such that each detection node may be configured to apply a type of malware detection scheme. For example, detection node 310 may be configured to apply behavior-based malware detection schemes while detection node 320 may be configured to apply metadata-based detection schemes where metadata of a file is analyzed. In yet another example, detection node 330 may be configured to apply signature-based detection schemes to files. As another example, nodes 310-330 may also apply classification-based detection schemes. As discussed above, detection nodes 310-330 may be configured to apply other forms of detection schemes that conform to an interface to facilitate the incorporation of new or different detection

schemes. In various embodiments, subsystem 300 may include a variety of detection nodes in addition to detection nodes 310-330.

Detection agent 316 may be implemented using software. In some embodiments, detection agent 316 may be installed on detection node 310. In various
5 embodiments detection agent 316 may reside on hardware separate from detection node 310 but in communication with detection node 310 and in particular hypervisor 312. Detection agent 316 may be configured to receive jobs that describe malware detection schemes that should be applied to a file. Detection agent 316 may also be
10 configured to receive the file and send it to hypervisor 312 and cause hypervisor 312 to implement the malware detection scheme from the job on the file. Detection agent 316 may also be configured to monitor the execution of the malware detection scheme and report when it is completed. Detection agent 316 may report the results of the application of the malware detection scheme to the file. In various embodiments,
15 agents such as detection agent 316 may be present in other detection nodes such as nodes 320-330 and cause those nodes to implement a variety of malware detection schemes. For example, a detection agent on detection node 320 may cause detection node 320 to implement a signature-based malware detection scheme. As another example, a detection agent on detection node 330 may cause detection node 330 to implement a metadata-based malware detection scheme on a file.

20 Hypervisor 312 may be implemented using various types of hypervisors such as those provided in virtualization solutions from VMWARE, PARALLELS, and XEN. In some embodiments, the KI-2 hypervisor from RAYTHEON may be employed. Hypervisor 312 may be used to apply malware detection schemes in a virtual environment. Hypervisor 312 may host GOS 314. In various embodiments,
25 multiple guest operating systems may be hosted by hypervisor 312.

Hypervisor 312 may cause the application of malware detection schemes within GOS 314. In some embodiments, hypervisor 312 may be able to apply a malware detection scheme to a file at a configurable observation time (including, in
30 some embodiments, in faster-than-real-time) by causing the execution to ignore wait states associated with GOS 314.

Hypervisor 312 may, in various embodiments, be able to provide near-instant reversions of GOS 314 during jobs or between jobs. This may be accomplished, for

example, by maintaining a state of GOS 314 in memory and tracking changes made to GOS 314 during the execution of the malware detection scheme. The tracked changes may also be stored in memory. When the malware detection scheme is completed, GOS 314 may be reverted by undoing the changes to GOS 314 during execution of
5 the malware detection scheme using changes stored in memory.

In some embodiments, by applying near-instant reversion and/or applying malware detection schemes at a configurable observation time, processing of files may be accomplished faster allowing the system to have a higher throughput of analyzed files. These techniques may be applied in both an active mode and a passive
10 mode.

Hypervisor 312 may also be able to provide information regarding the execution of the malware detection scheme without running a process within GOS 314 in various embodiments. Doing so, in some embodiments, may thwart attempts by malware to detect if a malware detection scheme is being applied to the malware.
15 For example, a file may contain malware that is configured to examine processes running in an operating system. Based on that examination, the malware may recognize certain processes as being associated with the application of malware detection schemes. As a result, in an attempt to remain undetected during the application of the malware detection scheme, the malware may change its behavior so
20 that it does not act maliciously or otherwise abnormally. However, the malware may change its behavior once it is in a normal environment where malware detection schemes are not being applied. In some situations, monitoring of GOS 314 may occur at the machine layer so as to avoid running processes. Hence, in various embodiments, it may be advantageous for hypervisor 312 to provide monitoring of
25 GOS 314 without running a process within GOS 314. Other suitable techniques may be applied within hypervisor 312 to hide the fact that a hypervisor is being used from an application and/or to hide the existence of an application running inside an operating system.

In various embodiments, GOS 314 may be any of a variety of virtualized
30 operating systems. Examples of such operating systems include various versions of MICROSOFT WINDOWS, APPLE MAC OS, operating systems using the LINUX kernel, operating systems using the BSD kernel, embedded operating systems, mobile

operating systems (including those used in mobile telephones or devices), or other computing environments. GOS 314 may include a variety of applications used to access different file types. Such applications may include e-mail clients, word processing applications, image viewing applications, audio applications, video applications, Internet browsing applications, e-mail applications, or other applications.

Network agent 340 may, in various embodiments, be implemented using software, hardware, or a combination of both. Network agent 340 may be configured to provide virtualized networking services to detection nodes 310-330. In some embodiments, these services include simulating a network environment and network nodes in order to facilitate the application of various malware detection schemes. For example, detection node 310 may implement a behavior-based malware detection scheme that includes accessing the file in GOS 314. Part of the malware detection scheme may include observing how accessing the file impacts a network. Network agent 340 may provide virtual networking services in order to provide the file opportunity to interact with a network. In some embodiments, the network may be simulated.

FIGURE 4 illustrates one embodiment of disposition subsystem 400. Disposition subsystem 400 includes adjudication and disposition module 410, workflow module 420, policy database 440, and results database 450. Each of modules 410-420 may communicate with one another as well as communicate with results database 450 and policy database 440 in various embodiments. Further details regarding the components of disposition subsystem 400 are given below and in the description of FIGURE 5.

Modules 410-420 and 460 may be implemented using software, hardware, or a combination of the two. Each of modules 410-420 and 460 may be implemented on the same hardware or on different hardware. For example, modules 410-420 and 460 may each be on separate pieces of hardware.

Databases 440 and 450 may each be implemented in hardware separate from each other or on the same hardware. Databases 440 and 450 may be implemented on the same or different hardware as modules 410-420. The databases may be row-oriented or column-oriented. They may include relational databases and/or object-oriented databases. Databases 440 and 450 may be indexed and accessed in a variety

of manners, including by utilizing an operating system or through one or more networks. In some embodiments, databases 440 and 450 may be implemented in one database such that separate tables within the database may be used to store the information contained in databases 440 and 450.

5 Adjudication and disposition module 410, in some embodiments, may analyze the outcome of malware detection schemes applied to a file to determine if the file is malware. Adjudication and disposition module 410 may access policy database 440 to select one or more adjudication policies in order to determine if the file is malware. Adjudication policies from policy database 440 may include various ways to
10 determine how to characterize a file. In various embodiments, possible characterizations of a file may include: (1) the file is known malware; (2) the file is known not to be malware; (3) the file is suspected malware; and (4) the file is not suspected to be malware. The adjudication policies may contain rules which indicate which one of the characterizations should be applied to a file based on the results of
15 malware detection schemes applied to the file. For example, an adjudication policy may include a weighted score that is applied to the outcomes of each of the malware detection schemes applied to the file. Based on the sum of the weighted scores, the adjudication policy may indicate the proper characterization of the file. Adjudication and disposition module 410 may determine the appropriate adjudication policy to use
20 for the file in question, retrieve the outcomes of the malware detection schemes performed on the file, and use the adjudication policy to apply weighted scores to each of those outcomes. Adjudication and disposition module 410 may also, in some embodiments, use the sum of the applied weighted scores to characterize the file according to the adjudication policy. Although weighted scoring is described, the
25 present invention contemplates using any suitable selection mechanism according to particular needs.

One or more adjudication policies may be chosen to evaluate the outcome of the malware detection schemes applied to the file based on a variety of factors. For example, the file's type may be used to determine what policy to select. The context
30 from where the file came may also be used to select an adjudication policy. Examples of context include a messaging context, a file uploading context, and a browsing context as discussed above.

Adjudication and disposition module 410 may provide entries for results database 450 after determining the characterization to use for the file. For example, if the weighted score determined by adjudication and disposition module 410 leads adjudication and disposition module 410 to characterize the file as malware according to a selected policy, adjudication and disposition module 410 may create an entry in results database 450 which include any or all of the following items: the name of the file, the type of the file, a hash value associated with the file, the individual scores applied to the file, the sum of the scores applied to the file, the adjudication policy applied to the file, and the characterization applied to the file.

In some embodiments, workflow module 420 may determine a workflow for use by a human analyst to review the file. A file may need to be reviewed by a human analyst based on the characterization of the file determined by adjudication and disposition module 410. For example, if adjudication and disposition module 410 characterizes the file as suspected malware then a human analyst review the file to determine if it is malware. As another example, if the file is determined to be malware rather than merely to be suspected malware, a human analyst may review the file in order to learn more about the malware in the file and/or to start an incident response review (e.g., to clean up the file). In such and other scenarios, workflow module 420 may determine tasks associated with reviewing the file. In some embodiments, workflow module 420 may generate a set of tasks to be performed by a human analyst reviewing the file based on a variety of factors. Such factors could include the type of file being reviewed and the characterization of the file by adjudication and disposition module 410. Other factors may include whether the system is operating in a passive or an active mode. For example, if operating in an active mode, a determination that the file is suspected malware may invoke an advanced malware analysis workflow to be followed by a human analyst. As another example, if the file is determined to be known malware, then workflows associated with cleaning up the environment(s) the file has been in as well as other incident response workflows (e.g., notifying a user or an administrator) may be determined. Other suitable workflows may be generated based on these or other suitable factors.

Workflow module 420 may, in various embodiments, automatically perform certain tasks associated with reviewing the file by a human analyst to facilitate the

review of the file by the human analyst. For example, it may be determined that in order for a file to be properly reviewed by a human analyst, all strings in the file should be extracted. Workflow module 420 may automate this procedure such that it is already performed for the human analyst when they receive the task. Workflow module 420 may also provide a priority associated with the review of the file. For example, if the file has already been determined to be malware, a lower priority may be assigned to the review of the file compared to a file that is suspected of being malware. Also, based on the characterization and/or the mode the system is operating in, entirely different workflows may be generated for the file or the file may be placed in different queues. In various embodiments, workflow module 420 may be configured to interact with an analysis console (e.g., analysis console 170) to facilitate the creation and operation of the workflows. Policies stored in policy database 440 may also contain information on workflows to be applied to files based on their characterization. Workflow module 420 may, in some embodiments, access policy database 440 in order to determine the workflow that should be applied to the file. In various embodiments, workflow module 420 may receive one or more disposition policies that may be used to determine the workflow from adjudication and disposition module 410, as further discussed below.

In some embodiments, adjudication and disposition module 410 may perform operations associated with the context of the file in response to the characterization of the file determined by adjudication and disposition module 410. As discussed above, a file may come from a variety of contexts including a messaging context, a browsing context (this context may include processing URLs), or a file uploading context. The characterization of the file may cause certain operations to be performed related to the context of the file. For example, if the file came from a messaging context (e.g., the file was an attachment to an e-mail), the characterization of the file may cause effects on the e-mail from which the file was extracted. It may be determined whether the e-mail should be delivered based on the characterization of the file. As another example, if the system were analyzing a URL, the context of where the URL was found may affect the operations performed. If the URL was found in the body of an e-mail message, for example, the message may be prevented from being delivered. On the other hand, if the URL was found in the body of an attachment to the e-mail

message, the message may be delivered but, optionally, the attachment may be stripped and/or the URL may be removed. As another example, in certain embodiments, regardless of the context in which the URL is found, if the URL is determined to be associated with malware, the message may not be delivered.

5 Adjudication and disposition module 410 provides functionality that allows for these types of operations to be performed.

In various embodiments, adjudication and disposition module 410 may take the mode that the malware detection system is operating in into account when determining actions to perform. For example, if the malware detection system is

10 operating in a passive mode and the file is characterized as malware, adjudication and disposition module 410 may cause an alert to be sent to an administrator that the file has been characterized as malware. As another example, if the file was attached to an e-mail message and the malware detection system was operating in an active mode, then characterizing the file as not being malware would cause the adjudication and

15 disposition module 410 to indicate that the e-mail should be delivered. In various embodiments, adjudication and disposition module 410 may refer to one or more disposition policies in policy database 440 to determine one or more actions to be taken as a result of the characterization.

In some embodiments, when adding a new malware detection scheme, an

20 interface may be provided to modify one or more adjudication or disposition policies of policy database 440. The interface may require information to be entered regarding how the malware detection scheme should be applied. Such information may be entered using a tabbed interface, a wizard-style interface, or other suitable interfaces for entering information. Such information may include: weighted scoring

25 algorithms to apply to responses from detection nodes, workflows that may be generated based on the response of the detection nodes, thresholds or ranges that determine how the file should be characterized based on the response of the detection nodes, actions that should occur based on the characterization of the file, and/or other items associated with a malware detection scheme. In some embodiments, the

30 interface may allow for new actions to be defined that would respond to characterizations of the file. A modular architecture may be utilized where new

actions would be able to be "plugged in" to the system so that disposition policies may be easily modified or added.

Listener module 460, in various embodiments, may be implemented using software, hardware or a combination of the two. Listener module 460 may be able to
5 determine when jobs assigned to detection nodes are complete and report the completed status. In some embodiments, the results of listener module 460 may be used by monitor 270 to determine the load of the detection nodes. In some
embodiments, listener module 460 may be contacted by a detection node once it has completed analyzing a file and receive the results of the analysis. Listener module
10 460 may determine when the job has been completed and report the results of completed jobs. In some embodiments, listener module 460 may store the results of the jobs in results database 450. Listener module 460 may increase the priority of jobs in jobs database 290 based on the reported results of completed jobs. For
example, if a file has been scheduled to have five malware detection schemes applied
15 to the file and one of the malware detection schemes has reported that there is a high likelihood that the file is malware, listener module 460 may increase the priority of the jobs associated with this file.

FIGURE 5 illustrates an example computer system 500 suitable for implementing one or more portions of particular embodiments. Although the present
20 disclosure describes and illustrates a particular computer system 500 having particular components in a particular configuration, the present disclosure contemplates any suitable computer system having any suitable components in any suitable configuration. Moreover, computer system 500 may have take any suitable physical
form, such as for example one or more integrated circuit (ICs), one or more printed
25 circuit boards (PCBs), one or more handheld or other devices (such as mobile telephones or PDAs), one or more personal computers, one or more super computers, one or more servers, and one or more distributed computing elements. Blocks 130-180 of FIGURE 1 as well as the elements discussed in FIGURES 2-4 may be implemented using all of the components, or any appropriate combination of the
30 components, of computer system 500 described below.

Computer system 500 may have one or more input devices 502 (which may include a keypad, keyboard, mouse, stylus, or other input devices), one or more output

devices 504 (which may include one or more displays, one or more speakers, one or more printers, or other output devices), one or more storage devices 506, and one or more storage medium 508. An input device 502 may be external or internal to computer system 500. An output device 504 may be external or internal to computer system 500. A storage device 506 may be external or internal to computer system 500. A storage medium 508 may be external or internal to computer system 500.

System bus 510 couples subsystems of computer system 500 to each other. Herein, reference to a bus encompasses one or more digital signal lines serving a common function. The present disclosure contemplates any suitable system bus 510 including any suitable bus structures (such as one or more memory buses, one or more peripheral buses, one or more a local buses, or a combination of the foregoing) having any suitable bus architectures. Example bus architectures include, but are not limited to, Industry Standard Architecture (ISA) bus, Enhanced ISA (EISA) bus, Micro Channel Architecture (MCA) bus, Video Electronics Standards Association local (VLB) bus, Peripheral Component Interconnect (PCI) bus, PCI-Express bus (PCI-X), and Accelerated Graphics Port (AGP) bus.

Computer system 500 includes one or more processors 512 (or central processing units (CPUs)). A processor 512 may contain a cache 514 for temporary local storage of instructions, data, or computer addresses. Processors 512 are coupled to one or more storage devices, including memory 516. Memory 516 may include random access memory (RAM) 518 and read-only memory (ROM) 520. Data and instructions may transfer bidirectionally between processors 512 and RAM 518. Data and instructions may transfer unidirectionally to processors 512 from ROM 520. RAM 518 and ROM 520 may include any suitable computer-readable storage media.

Computer system 500 includes fixed storage 522 coupled bi-directionally to processors 512. Fixed storage 522 may be coupled to processors 512 via storage control unit 507. Fixed storage 522 may provide additional data storage capacity and may include any suitable computer-readable storage media. Fixed storage 522 may store an operating system (OS) 524, one or more executables (EXECs) 526, one or more applications or programs 528, data 530 and the like. Fixed storage 522 is typically a secondary storage medium (such as a hard disk) that is slower than primary storage. In appropriate cases, the information stored by fixed storage 522 may be

incorporated as virtual memory into memory 516. In some embodiments, fixed storage 522 may include network resources, such as one or more storage area networks (SAN) or network-attached storage (NAS).

Processors 512 may be coupled to a variety of interfaces, such as, for example, graphics control 532, video interface 534, input interface 536, output interface 537, and storage interface 538, which in turn may be respectively coupled to appropriate devices. Example input or output devices include, but are not limited to, video displays, track balls, mice, keyboards, microphones, touch-sensitive displays, transducer card readers, magnetic or paper tape readers, tablets, styli, voice or handwriting recognizers, biometrics readers, or computer systems. Network interface 540 may couple processors 512 to another computer system or to network 542. Network interface 540 may include wired, wireless, or any combination of wired and wireless components. Such components may include wired network cards, wireless network cards, radios, antennas, cables, or any other appropriate components. With network interface 540, processors 512 may receive or send information from or to network 542 in the course of performing steps of particular embodiments. Particular embodiments may execute solely on processors 512. Particular embodiments may execute on processors 512 and on one or more remote processors operating together.

In a network environment, where computer system 500 is connected to network 542, computer system 500 may communicate with other devices connected to network 542. Computer system 500 may communicate with network 542 via network interface 540. For example, computer system 500 may receive information (such as a request or a response from another device) from network 542 in the form of one or more incoming packets at network interface 540 and memory 516 may store the incoming packets for subsequent processing. Computer system 500 may send information (such as a request or a response to another device) to network 542 in the form of one or more outgoing packets from network interface 540, which memory 516 may store prior to being sent. Processors 512 may access an incoming or outgoing packet in memory 516 to process it, according to particular needs.

Particular embodiments involve one or more computer-storage products that include one or more tangible, computer-readable storage media that embody software for performing one or more steps of one or more processes described or illustrated

herein. In particular embodiments, one or more portions of the media, the software, or both may be designed and manufactured specifically to perform one or more steps of one or more processes described or illustrated herein. In addition or as an alternative, in particular embodiments, one or more portions of the media, the software, or both may be generally available without design or manufacture specific to processes described or illustrated herein. Example computer-readable storage media include, but are not limited to, CDs (such as CD-ROMs), FPGAs, floppy disks, optical disks, hard disks, holographic storage devices, ICs (such as ASICs), magnetic tape, caches, PLDs, RAM devices, ROM devices, semiconductor memory devices, and other suitable computer-readable storage media. In particular embodiments, software may be machine code which a compiler may generate or one or more files containing higher-level code which a computer may execute using an interpreter.

As an example and not by way of limitation, memory 516 may include one or more tangible, computer-readable storage media embodying software and computer system 500 may provide particular functionality described or illustrated herein as a result of processors 512 executing the software. Memory 516 may store and processors 512 may execute the software. Memory 516 may read the software from the computer-readable storage media in mass storage device 516 embodying the software or from one or more other sources via network interface 540. When executing the software, processors 512 may perform one or more steps of one or more processes described or illustrated herein, which may include defining one or more data structures for storage in memory 516 and modifying one or more of the data structures as directed by one or more portions the software, according to particular needs.

In some embodiments, the described processing and memory elements (such as processors 512 and memory 516) may be distributed across multiple devices such that the operations performed utilizing these elements may also be distributed across multiple devices. For example, software operated utilizing these elements may be run across multiple computers that contain these processing and memory elements. Other variations aside from the stated example are contemplated involving the use of distributed computing.

In addition or as an alternative, computer system 500 may provide particular functionality described or illustrated herein as a result of logic hardwired or otherwise embodied in a circuit, which may operate in place of or together with software to perform one or more steps of one or more processes described or illustrated herein.

5 The present disclosure encompasses any suitable combination of hardware and software, according to particular needs.

Although the present disclosure describes or illustrates particular operations as occurring in a particular order, the present disclosure contemplates any suitable operations occurring in any suitable order. Moreover, the present disclosure
10 contemplates any suitable operations being repeated one or more times in any suitable order. Although the present disclosure describes or illustrates particular operations as occurring in sequence, the present disclosure contemplates any suitable operations occurring at substantially the same time, where appropriate. Any suitable operation or sequence of operations described or illustrated herein may be interrupted, suspended,
15 or otherwise controlled by another process, such as an operating system or kernel, where appropriate. The acts can operate in an operating system environment or as stand-alone routines occupying all or a substantial part of the system processing.

FIGURES 6-10 are flowcharts that illustrate various embodiments of the operation of a malware detection system. In general, the steps illustrated in
20 FIGURES 6-10 may be combined, modified, or deleted where appropriate, and additional steps may also be added to the example operation. Furthermore, the described steps may be performed in any suitable order. In some embodiments, the steps described below may be performed by any suitable combination of the elements discussed above with respect to FIGURES 1-5.

25 FIGURE 6 is a flowchart illustrating one embodiment of how an e-mail attachment may be processed by a malware detection system operating in an active mode. As described further below, the steps described in this example may be applied to other contexts, such as the browsing context (e.g., processing URLs) or the service context (e.g., receiving one or more files to be processed by a security agent on a
30 desktop or network boundary).

At step 602, in some embodiments, a message is received. The message may be an e-mail message that is received by an e-mail server. At step 604, it may be

determined whether the message has an attachment. In some embodiments, if the message does not have an attachment, then the message may be processed normally, such as at step 606. This type of processing may include delivering the message to the recipient. If the message does have an attachment, then the system may proceed
5 to step 608. In some embodiments, the determination at step 604 may also include determining if the body of the message includes items that may need to be analyzed, such as a URL. If such an item exists, then the system may proceed to step 608.

At step 608, in some embodiments, the processing of the message by an e-mail delivery system may be paused. This may be done because the malware detection
10 system operating in an active mode. By pausing the processing of the message, the message may be prevented from being delivered. A copy of the message as well as the attachment may be created and sent to the malware detection system. In various embodiments, the message and the attachment themselves may be sent to the malware detection system without creating a copy. These actions may be performed or
15 facilitated by an agent that communicates both with the malware detection system and the messaging system. In contexts other than e-mail, other underlying processes or services may be paused. For example, if the context is uploading files to a network location, the uploading process may be paused while the system analyzes the files as further described below.

At step 610, the file attached to the message may be analyzed to determine if it
20 has been previously analyzed by the malware detection system. This determination, in some embodiments, may be made by creating a hash of the file and comparing the hash to a database that contains the results of applying the malware detection system to various files. The step may be performed utilizing a module such as duplicate
25 module 230 of FIGURE 2 and a database such as results database 450 of FIGURE 4 in various embodiments. If it is determined that the file has been previously analyzed then the system may proceed to step 626. However, if the file has not been previously analyzed the system may proceed to step 612.

At step 612, the type of the file may be determined. This may be
30 accomplished by analyzing the extension of the file and/or by examining a header of the file. This step may be performed by items such as file type module 220 of FIGURE 2 in various embodiments. At step 613, in some embodiments, it may be

determined if the file should be analyzed. This determination may occur based on the determined file type. For example, it may be determined that plain text files are not to be analyzed. If the file should not be analyzed, then the system should proceed to step 642. If the file should be analyzed, then the system should proceed to step 614.

5 At step 614, the determined file type may be used to select a policy that contains procedures to be followed when analyzing the file. These procedures may include determining what types of malware detection schemes to apply to the file. More than one policy may be selected at this step. The policy may contain more than one malware detection scheme to apply to the file. For example, the policy may
10 indicate that the file should be accessed by several versions of an application in a virtual environment or the policies may indicate that the file should be analyzed using signature based analysis to determine if it is malware. Classification-based analysis may also be performed in various embodiments. Elements such as policy module 240 of FIGURE 2 may be utilized to perform this step in various embodiments.

15 At step 616, in some embodiments, jobs associated with malware detection schemes may be sent to detection nodes. The jobs may be determined in accordance with policies selected at step 614. The jobs may be entered into a database and a scheduler may determine when jobs are sent to the detection nodes in accordance with the malware detection schemes determined at step 614. The scheduler may operate in
20 a first-in-first-out (FIFO) mode where jobs that are first entered into the database are the first ones to be assigned to available detection nodes. Schedule module 260 of FIGURE 2 is an example of how this step may be performed in various embodiments.

 At step 618, in some embodiments, malware detection may be performed on the file. This may be performed by one or more detection nodes. A variety of
25 malware detection schemes may be applied during this step. Each of these malware detection schemes may be applied at different nodes or at the same node, and they may be applied concurrently or sequentially or a combination of the two. Malware detection schemes may include performing behavior-based analysis, signature-based analysis, or metadata-based analysis. Items such as detection subsystem 300 of
30 FIGURE 3 may, in some embodiments, be used to implement this step. At step 620, a query for the results of the detection nodes applying the malware detection schemes determined by the policy may be performed. The results may be received as each

detection node completes its job or after all of the detection nodes have completed their jobs in accordance with the set of malware detection schemes to be applied to the file. Listener module 460 of FIGURE 4, in various embodiments, may provide an example for how this step may be implemented.

5 At step 622, in some embodiments, an adjudication policy may be applied to the results determined at step 620. In some embodiments, the adjudication policy may use scores that may be based on the type of file being analyzed as well as the malware detection scheme applied to the file. The scoring may be weighted in various embodiments. At step 624, it may be determined whether or not the file is
10 suspected malware. This may be done in accordance with the adjudication policy or policies accessed at step 622. If the file is determined to be suspected malware, the system may proceed to step 632. If the file is determined not to be suspected malware the system may proceed to step 642. Adjudication and disposition module 410 of FIGURE 4, or similar structures, may be used to implement steps 622 and 624 in
15 various embodiments.

 At step 626, in some embodiments, the results of a previous malware analysis performed on the file are obtained. This may be done in response to determining that the file has been previously scanned at step 610. These results may be retrieved from a database such as results database 450. The results may include a determination of
20 whether the file was considered to be: known malware, suspected malware, not malware, or not suspected as malware. The results of the adjudication policy or policies applied to the file may also be retrieved. Performing step 626, in some embodiments, may conserve resources and/or provide faster processing of a file because a file that has been previously analyzed may not need to be reanalyzed.

25 At step 628, the results are analyzed to determine if the analysis on the file has been completed and a characterization of the file has been determined. If a characterization of the file is not available, then step 626 may be performed. In some cases, the system may continue to perform steps 626 and/or 628 until a determination that the file is either known to be malware or known not to be malware is completed.
30 If a characterization is available, then step 630 may be performed. At step 630, the characterization of the file that was previously determined is examined. If the file was previously adjudicated as known malware, then step 640 may be performed. If

the file was previously adjudicated as not being malware, then step 642 may be performed.

At step 632, in some embodiments, the file may be sent for review by human analysts. This may be done in response to determining that the file is suspected
5 malware as in step 624. Along with the file, in various embodiments, a workflow for human analysts may be invoked in accordance with the results of the application of malware detection schemes to the file, the file type, and any other suitable information. In addition to the tasks in a workflow being invoked, a set of those tasks may be automatically performed before the human analyst receives the workflow
10 package in various embodiments. Workflow module 420 of FIGURE 4 may be an example of how such functionality may be provided. At step 634, a response from the human analyst may be waited for before proceeding. At step 636, a response from the human analyst may be received and analyzed. If the response indicates that the file is malware, step 640 may be performed. If the response from the human analyst
15 indicates that the file is not malware, step 642 may be performed.

At step 640, actions may be performed based on a determination that the file is known to be malware or suspected to be malware. These actions may be determined based on one or more disposition policies. For example, such actions may include removing the file, sanitizing the file, quarantining the file, and/or sending the file for
20 review by human analysts. The actions may be done in response to obtaining the results of a previous analysis of the file, a response from human analysts, or the application of an adjudication policy based on results returned by detection nodes applying various malware detection schemes to the file. For example, in the messaging context, an indication that the attachment should be removed may be sent
25 to a mail server or agent, which may then remove the attachment from the message that is to be delivered at this step. In some embodiments, actions such as removing attachments from messages may be performed by the system and the modified message may be sent back to a mail server or agent for delivery or an edited message may be sent by the system. As another example, if the system was being used as a
30 service by another entity (such as a desktop or network security agent), a report may be sent to the entity that may include the adjudication as well as the results from the detection nodes. In some cases, a sanitized version of the file(s) may be sent to the

entity using the system as a service. Adjudication and disposition module 410 of FIGURE 4, in some embodiments, may be an example for how step 640 may be implemented.

In some embodiments, a determination may be made at step 640 that waits for
5 other files to be completely processed before performing any actions. Prioritizing the jobs associated with other files that are still being analyzed may occur at this step in various embodiments. This prioritization may occur, for example, if one file of a group of files being processed has been determined to be known malware or suspected malware. As a result of such a determination, the other files part of the
10 group may have their analysis prioritized. In the messaging context, if an e-mail attachment includes multiple files, determining what actions should be performed on one file may be based on the characterization of the other files. As a result, it may be determined that actions should not be performed until all the files have been analyzed.

In some embodiments, workflows for human analysts may be determined and
15 invoked during this step. Such workflows may include sending a file determined to be or suspected to be malware to a human analyst for further analysis. The workflows may also include actions associated with incident response. For example, if a file has been determined to be malware, several actions involving human interaction may be helpful. These may include items such as contacting relevant personnel or
20 administrators, taking precautionary measures to secure network and computing resources, or other suitable actions depending on the context associated with the file.

At step 642, the message may resume being processed. In some embodiments where multiple files are being processed as a group (i.e., if there are multiple files in an attachment), this step may include waiting for the results of the analysis of other
25 files. In some embodiments, this may occur if it is determined that the attachment does not contain malware. This step may be reached if the attachment has been determined as malware but after the attachment has been removed from the message as in step 640. Hence the processing and delivering of the message may be paused at step 608 and resumed at step 642. In some cases, a message may not be delivered at
30 all. For example, the message may contain a URL in the body of the message that was analyzed in accordance with the steps described above. The URL may be determined to be malicious and, as a result, it may be determined that the message

should not be delivered at all. This may provide enhanced messaging security, in some embodiments, since messages are not delivered until it is determined that they do not contain malware or until attachments determined to be malware are removed from the messages. In some embodiments, this step may be performed in different contexts. For example, if the system was analyzing files that were to be uploaded to a network location, this step may include allowing some or all of the files to be uploaded. If the system was invoked as a service, this step may include the entity that invoked the system as a service performing actions in response to receiving the results of the malware analysis. For example, if a desktop security agent invoked the system as a service, the agent may have received the results at step 640 and may alert the user of the desktop system of those results at step 642.

FIGURE 7 is a flowchart illustrating one embodiment of a malware detection system operating in a passive mode to detect malware in messages such as e-mail. As described further below, the steps described in this example may be applied to other contexts, such as the browsing context (e.g., processing URLs) or the service context (e.g., receiving one or more files to be processed by a security agent on a desktop or network boundary).

Steps 702, 704, and 706, in various embodiments, may be completed in the same or in a similar manner as step 602, 604, and 606 of FIGURE 6. Generally, if a determination at step 704 indicates that the message received at step 702 does not have an attachment then the message is processed normally by nodes in the messaging context at step 706. In some embodiments, a determination may be made at this step if the body of the message contains items that need to be analyzed, such as URLs. If not, then the message is processed normally by nodes in the messaging context at step 706. If the message received at step 702 does have an attachment or has other content (such as a URL) that needs to be analyzed, then the message may be duplicated (at step 708). At step 710, the original message may be processed normally by nodes in the messaging context. For example, a result of the processing at step 710 may be delivering the message to the recipient. In various embodiments, as discussed above, other actions may be performed at this step in accordance with other contexts. As an example, if the file was associated with a file uploading context, a result of the processing at step 710 may be allowing the files to be uploaded rather than delivering

a message. Steps 708 and 710 are, in various embodiments, performed when the malware detection system operates in a passive mode. As discussed further below, while malware detection proceeds, the messages (when the system is analyzing files from a messaging context) are not prevented from being delivered or otherwise
5 processed by nodes or systems of the messaging context. By duplicating the message at step 708, a copy of the message may be processed by the malware detection system while another copy may be processed as normal by the messaging context as in step 710. Ingest block 130 of FIGURE 1 may provide an example for how step 708 may be implemented in various embodiments.

10 Steps 712-724, in some embodiments, may be performed similarly as steps 610-622 in FIGURE 6. Generally, a file attached to the message duplicated at step 708 undergoes malware detection in steps 712-724. A policy selected at step 716 is used to determine various malware detection schemes that are applied at step 720 and the results of the malware detection schemes are analyzed at step 724.

15 At step 726, in some embodiments, the results of applying an adjudication policy at step 724 are analyzed to determine if the file is suspected malware. If these results indicate that the file is suspected malware, then step 732 may be performed. If these results indicate that the file is not suspected malware, then analysis for that file may be complete. In some embodiments, the file may be part of a group of files that
20 are being analyzed. If this is the case, determining that a file of the group of files is not suspected malware will not end the analysis of the group of files. Rather, this determination is stored and used to analyze the other files in the group, as described further below at step 736. Step 726 may be performed similarly to step 624 in FIGURE 6 in various embodiments.

25 Step 727, in some embodiments, may be performed in a manner similar to step 626 of FIGURE 6. In this situation, a file has been determined to have already been analyzed by the malware detection system. Hence, results from the previous analysis are obtained in this step in various embodiments. At step 729, the results are analyzed to determine if the analysis on the file has been completed and a characterization of
30 the file has been determined. If a characterization of the file is not available, then step 727 may be performed. In some cases, the system may continue to perform steps 727 and/or 729 until a determination that the file is either known to be malware or known

not to be malware is completed. If a characterization is available, then step 730 may be performed.

At step 730, in some embodiments, the characterization of the file that was previously determined is examined. If the file was previously adjudicated as known
5 malware, then step 736 may be performed. If the file was previously adjudicated as not being malware, then the analysis for that file may be complete. In some embodiments, the file may be part of a group of files that are being analyzed. If this is the case, determining that a file of the group of files is not suspected malware will not end the analysis of the group of files. Rather, this determination is stored and used to
10 analyze the other files in the group, as described further below at step 736.

Steps 732 and 733 may be accomplished in a manner similar to step 632 and step 634 at FIGURE 6. In various embodiments, the file may be reviewed by human analysts at step 732 and a response from the human analyst may be awaited at step 733. At step 734, this response may be received and a determination is made as to
15 whether the human analyst has indicated that the file is malware. If the file is determined to be malware, step 736 may be performed. If the file is determined not to be malware, the analysis of the file may be complete. As with steps 726 and 730, in various embodiments, when the file is determined not to be malware, then nothing more needs to be done to the message because the message was not delayed in its
20 processing due to the malware detection schemes since the system is operating in a passive mode. In some embodiments, the file may be part of a group of files that are being analyzed. If this is the case, determining that a file of the group of files is not suspected malware will not end the analysis of the group of files. Rather, this determination is stored and used to analyze the other files in the group, as described
25 further below at step 736.

Step 736 may be accomplished in a manner similar to step 640 of FIGURE 6. Generally, actions may be performed based on a determination that the file is known to be malware. These actions may be determined based on one or more disposition policies. For example, such actions may include removing the file, sanitizing the file,
30 quarantining the file, and/or sending the file for review by human analysts. The actions may be done in response to obtaining the results of a previous analysis of the file, a response from human analysts, or the application of an adjudication policy

based on results returned by detection nodes applying various malware detection schemes to the file. Information about the file may also be stored with the file in some embodiments. For example, message headers associated with the file may be stored with the file at step 736. As another example, timing information related to when the message was received or when the message was processed with the file. As yet another example, information as to how the file was determined to be malware may also be stored along with the file at step 736.

At step 738, a warning may be sent in response to determining that the file is determined to be malware. In various embodiments, as a result of operating in a passive mode, an indication may be sent that the file was detected as malware since the process of the message was not delayed while the malware detection analysis was being applied to the file. For example, if the file was determined to be malware, then a message may be sent to an administrator to inform the administrator that a message had been processed that included malware. Other examples of warnings or indicators that may be sent include sending a warning to recipients of the message. An example of the latter response may include an agent that is installed at the recipient's computer wherein the agent is configured to take actions to prevent damage to the computer based on the malware that may have been delivered to the computer while the message was being analyzed for malware.

FIGURE 8 is a flowchart that illustrates one embodiment of a malware detection system dynamically changing between an active and a passive mode. In some embodiments, the steps of FIGURE 8 may be performed with items such as monitor module 270 of FIGURE 2. At step 810, detection nodes employed to apply malware detection schemes to a file are monitored. In some embodiments the monitoring may occur by communicating with agents that are associated with the detection nodes, such as agent 340. Information analyzed while monitoring the detection nodes may include the number of idle nodes, the number of nodes that are actively processing jobs, the amount of time that each job will take, the number of jobs to be processed, the number of jobs that are assigned to each detection node, the amount of delay introduced to the context associated with the file (i.e., delay before an e-mail is delivered or delay before a file is allowed to be uploaded). Other types of information may also be collected. For example, information based on the type of

malware detection schemes that a detection node applies may also be monitored. Such information may include information related to behavior-based malware detection schemes, signature-based malware detection schemes, classification-based malware detection schemes, and/or metadata-based malware detection schemes.

5 At step 820, in some embodiments, a determination may be made as to whether or not a threshold has been surpassed based on the information determined at step 810. A variety of thresholds may be used alone or in combination during this step. For example, if the amount of load (current and/or predicted) to be placed on
10 detection nodes indicates that the response times to the jobs that are scheduled to be applied to detection nodes are over a particular amount, then a threshold may be surpassed. In another example, jobs may be processed by different types of detection nodes. As such, the overall load on all the detection nodes may be below a threshold for the collection of detection nodes. A particular set of detection nodes that are used to perform a particular malware detection scheme, however, may have a sufficiently
15 high load that another threshold may be surpassed. An example of this situation is when there is a large number of behavior-based malware detection schemes to be processed but only a certain set of the detection nodes may be able to process those types of jobs. As a result, the analysis of the file may take a longer time to complete because one of the jobs associated with analyzing the file includes analysis by
20 detection nodes that perform behavior-based analysis. A threshold may be surpassed in this type of situation where there is a bottleneck at one or more of the nodes that causes processing of a message to take longer than it should.

 In some embodiments, analyzing the load on the detection nodes as well as the list of jobs may cause the system to dynamically reprovision some or all of the
25 detection nodes to alleviate any bottlenecks. For example, in a situation where a surge of files entered the system, detection nodes may be modified so that the increase of jobs may be handled. A detection node that had been applying behavior-based analysis using a particular version of software may be reconfigured to perform generic behavior-based analysis or to perform signature-based or classification-based malware
30 detection. After the surge of jobs has been handled, the system may be configured to configure the detection nodes back to their original state. In this manner, surpassing

certain thresholds may be avoided and the system may be able to remain in an active mode.

Thresholds, in some embodiments, may be determined by the context that the file was a part of. For example, a required response time may be lower if the file was
5 part of a messaging environment as opposed to a file uploading context. Hence, a number of suitable considerations may be utilized to create this sophisticated set of thresholds to compare the monitoring information to.

If one or more of the thresholds are surpassed, then step 830 may be performed and the malware detection system may enter (or remain in) a passive mode.
10 If the thresholds are not surpassed, then step 840 may be performed and the malware detection system may enter (or remain in) an active mode. An active mode of operation, in various embodiments, may include pausing the processing of a file while the file is being analyzed for malware whereas a passive mode of operation, in various embodiments, may include allowing a file or a message associated with a file to be
15 processed while the analysis for malware is pending. While in a passive mode, in some embodiments, certain malware detection schemes may be applied as if the system was in the active mode. For example, when processing an e-mail message, signature-based and classification-based malware detection schemes may be applied before the message is allowed to be delivered while behavior-based detection schemes
20 may be applied after the message is allowed to be delivered

In some embodiments, when transitioning to a passive mode from an active mode, certain jobs may be placed into a backlog queue where they will be processed as appropriate. For example, behavior-based jobs may be placed in such a queue while signature-based or classification-based jobs may be prioritized. In certain
25 systems, signature-based jobs and classification-based jobs typically run faster than behavior-based jobs. If a signature-based or classification-based job results in a determination that a particular file or a particular message associated with the particular file is malware, in certain embodiments, the behavior-based job for the particular file may be moved up in priority. If the particular file or message is a part
30 of a group of files, it may also be appropriate for all jobs associated with all files and/or messages in the group to be moved up in priority. For example, in response to determining based on a signature-based job that a first file is malware, the

classification-based job and behavior-based job of the first file, as well as the jobs for any other files grouped with the first file may be moved up in priority.

After either step 830 or 840, the detection nodes may continue to be monitored (as in step 810). In some embodiments, this may provide the ability for the malware detection system to dynamically change between a passive mode and an active mode
5 depending upon the loads present on the detection nodes. By switching between these modes, the system may be able to balance the competing interests of having greater security by operating in an active mode while also maintaining an acceptable level of quality of service by entering into the passive mode.

10 FIGURE 9 is a flowchart illustrating one embodiment of dynamic scheduling in a malware detection system. In some embodiments, the steps of FIGURE 8 may be performed with items such as schedule module 260 of FIGURE 2 and/or detection agent 316 of FIGURE 3.

A malware detection system, in some embodiments, may schedule the analysis
15 of a file, such as at step 910. This may include assigning one or more detection nodes to apply one or more malware detection schemes to the file (i.e., jobs). In various embodiments, the jobs assigned to the detection nodes relating to applying malware detection schemes to the file may be scheduled based on a first-in-first-out (FIFO) approach. For example, the jobs that are first received to be scheduled may be the
20 jobs that are scheduled to be applied to available detection nodes first. Other suitable scheduling techniques may be used.

In some embodiments, scheduling the analysis of the file includes assigning jobs to detection nodes that are capable of applying particular malware detection schemes. For example, the analysis of a file may include several detection jobs such
25 as applying a signature-based detection scheme and applying a behavior-based detection scheme. In such situations, it may be the case that certain detection nodes are capable of applying signature-based detection schemes while other detection nodes are capable of applying behavior-based malware detection schemes. As a result, in various embodiments, the scheduling of the analysis of a file may include
30 maintaining separate queues for scheduling jobs for each detection node.

At step 920, in some embodiments, a result from the completion of a job at a detection node may be determined. This may be received by an agent present on the

detection node. The agent may determine the results of the detection node and send those results. At step 930, the jobs currently scheduled for the detection nodes may be reassessed based on the result determined at step 920. For example, if the result of the completion of a job at a detection node indicates that a file may be malware, a
5 malware detection system may consider other jobs associated with that file to be important. In some embodiments, other jobs associated with the file may be assessed to determine when they are scheduled to be processed by a detection node.

At step 940, in some embodiments, a job associated with the file may be prioritized for processing at a detection node. This may be done in response to the
10 assessment at 930. For example, a file may have had three jobs associated with it in order to determine if a file is malware. The first job may have been assigned to a detection node identifying metadata-based detection schemes. The second job may have been assigned to a detection node applying signature-based detection schemes. The third job may have been assigned to a detection node performing behavior-based
15 detection schemes. The first job may have been completed and the result may have been determined at step 920. In response to step 930, it may have been determined that the remaining two jobs should be prioritized higher than they currently were. The queue maintained for the detection nodes that were scheduled to process the second and third jobs may then be altered such that, for example, the detection nodes may
20 process the second and third jobs once they have completed their current analysis without regard to the pending jobs in the queue.

In some embodiments, prioritizing the jobs associated with the file at the detection nodes may include determining if other jobs scheduled to be processed by the detection nodes have been previously prioritized. If this is the case, then the jobs
25 currently being prioritized for the file may be processed after jobs that have previously been prioritized but before other jobs that have not been prioritized. In some embodiments, prioritizing the file may include assigning priority ranks to jobs associated with the file. In this manner it may be possible to prioritize a job associated with a file over a job that has been previously prioritized. As a result of the
30 prioritization, jobs associated with the file may be processed sooner than they would have otherwise. This may, in some embodiments, allow a malware detection system

to dynamically allocate resources based on results generated by detection nodes before the file has been completely analyzed.

FIGURE 10 is a flowchart illustrating one embodiment of applying a behavior-based malware detection scheme to a file. In various embodiments, some or all portions of detection subsystem 300 may be used to implement some or all of the steps of FIGURE 3.

At step 1010, a baseline image of a guest operating system may be stored at a hypervisor. A hypervisor may store the baseline image before providing the file to the guest operating system. In some embodiments, this may allow the hypervisor to maintain a pristine state of the guest operating system before performing malware detection schemes within the guest operating system. Multiple baseline images, in some situations, may be stored at this step as well. In various embodiments, this may allow one machine with a single hypervisor to run multiple guest operating systems at the same time.

At step 1020, in some embodiments, the hypervisor may receive a file to be analyzed. In certain cases, the hypervisor may receive this file from an agent coupled to the hypervisor. At step 1030, the file may be accessed within the guest operating system. The type of access performed by the guest operating system may be determined based on the type of file in various embodiments. For example, if the file is an executable file then the file may be executed at step 1030. If the file has an associated application such as a MICROSOFT WORD document or a Portable Document Format (PDF) document, then the associated application may open the file at this step. As another example, if the hypervisor receives a URL, an Internet browser (such as INTERNET EXPLORER or MOZILLA FIREFOX) may be used to access the file at this step. In some situations, the hypervisor may access the file within multiple guest operating systems that are on the same machine.

At step 1040, in some embodiments, the access to the file may be monitored. This may be facilitated by the hypervisor. In various embodiments, the hypervisor may provide monitoring of the behavior of the file while it is being accessed without running a process in the guest operating system. In some cases, the hypervisor may monitor the accessing of the file in multiple guest operating systems on the same machine. At step 1045, in some embodiments, changes to the guest operating system

occurring as a result of accessing the file at step 1030 may be tracked. For example, this may occur by maintaining the changes made to the guest operating system in memory. In various embodiments, the changes may be stored on disk instead of in memory. A suitable disk may, for example, include one or more hard drives, removable media, and/or storage arrays. In certain situations, it may be advantageous to store the changes in memory because the reversion discussed below at step 1060 may be accomplished quicker. The items stored in memory may be an optimized form of representing the changes. Other items that are tracked may include processes, crashes, network activity, and/or other suitable activities. In some embodiments, the hypervisor may allow for tracking changes in memory or on disk that occur on multiple guest operating systems on the same machine.

At step 1050, one or more results may be determined based upon the monitoring of the access of the file performed at step 1040. The results may include determining that the file behaved normally, determining what aspects of the guest operating system were affected by accessing the file, determining how the resources of the virtualized hardware were used during access of the file, and/or other items associated with accessing the file. In various embodiments, it may be determined whether the file is suspected malware or not suspected malware. At step 1060, the guest operating system may be reverted to the baseline image stored at step 1010. This may be done using the changes stored in memory at step 1045. In some embodiments, multiple guest operating systems may be reverted to the same or a different baseline image. The changes occurring on each guest operating system that may be stored in memory (or on disk, in some cases) at step 1045 may be used to revert each of the guest operating systems.

At step 1070, in some embodiments, a report may be sent based on the monitored behavior and results at steps 1040 and 1050. An agent associated with the hypervisor may gather the information and send the report. The report may include information such as the name of the file, the type of the file, the versions of applications used to access the file, the versions of the guest operating system used to access the file, the version of the hypervisor used to access the file, the changes to the guest operating system while the file was being accessed, the resources of the virtualized hardware being used while the file was being accessed, and/or other items

associated with accessing the file. This information may be used by human analysts to determine whether or not the file is malware or to gain more insight to the file.

FIGURE 11 illustrates one embodiment of system 1100 for host-level malware detection. System 1100 may include host 1110, control module 1180 and malware detection system 1190. Host 1100 may be coupled to control module 1180. Control module 1180 may be coupled to malware detection system 1190. Host 1110 may include inspector 1120, monitor 1130, configuration files 1140, working files 1150, file system 1160, memory 1170, and security module 1175. In some embodiments, inspector 1120 may analyze file system 1160 in accordance with configuration files 1140. This analysis may be activated by monitor 1130 or control module 1180. Inspector 1120 may send suspect files to control module 1180. Control module 1180 may send the suspect files to malware detection system 1190 and receive a determination from malware detection system 1190 indicating whether the suspect files are malware.

In some embodiments, host 1110 may include one or more computing devices. Such devices may include a computer, a personal computer, a workstation, a server, a kiosk computing device, a PDA, a mobile computing device, a laptop or notebook computer, a tablet computer, a mobile telephone, and/or a smartphone. Host 1110 may include any suitable combination of hardware, firmware, and software. Host 1110 may include one or more operating systems. Examples of suitable operating systems include MICROSOFT WINDOWS, MAC OS, operating systems using BSD components, operating systems using the LINUX kernel, mobile operating systems (such as WINDOWS MOBILE, SYMBIAN OS, BLACKBERRY OS, IPHONE OS, PALM OS, and PALM WEBOS), and real-time operating systems. In some embodiments, host 1110 may include some or all aspects of computer system 500 of FIGURE 5.

In some embodiments, inspector 1120 may include software and/or hardware on or coupled to host 1110. Inspector 1120 may utilize configuration files 1140 to scan file system 1160. In some embodiments, inspector 1120 may examine applications running on host 1110 (e.g., by scanning memory 1170) in order to determine files or other resources associated with the application. Inspector 1120 may utilize parallel processing while examining items on host 1110. This

examination of applications running on host 1110 may include reading a portion of data (such as a block) from file system 1160 or memory 1170 and performing several operations on the portion of data. As described further below, in certain embodiments such operations include creating hashes of the portion of data, comparing the portion
5 of data to one or more lists (e.g., as specified in configuration files 1140), and comparing the portion of data to one or more rules (e.g., as specified in configuration files 1140).

In some embodiments, monitor 1130 may include software and/or hardware on or coupled to host 1110. Monitor 1130 may utilize configuration files 1140 to
10 cause file system 1160 to be scanned or to scan one or more files on file system 1160 on-demand. Monitor 1130 may also scan and analyze processes running on host 1110 in accordance with configuration files 1140. This analysis of processes running on host 1110 may include scanning memory 1170. As described further below, in certain embodiments operations performed by monitor 1130 include creating hashes of files,
15 comparing the files to one or more lists (e.g., as specified in configuration files 1140), and comparing the files to one or more rules (e.g., as specified in configuration files 1140).

In some embodiments, inspector 1120 and monitor 1130 may be stored within file system 1160. They may be low-level software tools with the ability to process
20 large files. Control module 1180 may deploy inspector 1120 and/or monitor 1130 to host 1110. Inspector 1120 and/or monitor 1130 may also be introduced to host 1110 using external media (such as a computer-readable storage medium). Examples of such external media include a diskette, a digital disc, a portable hard drive, a flash drive, or any other suitable type of external media. In some embodiments, inspector
25 1120 and/or monitor 1130 may be executed from the external media. In some situations, inspector 1120 and/or monitor 1130 may be transferred from the external media to file system 1160 before being executed.

In some embodiments, security module 1175 may include software and/or hardware on host 1110 that detects malware such as computer viruses and spyware on
30 host 1110. Security module 1175 may employ signature-based detection, classification-based detection, and/or heuristic techniques to detect malware on host 1110. Examples of security module 1175 include MCAFEE INTERNET

SECURITY, SYMANTEC NORTON 360, KASPERSKY INTERNET SECURITY, or any other suitable type of malware-detection security module, in any suitable combination.

In some embodiments, file system 1160 and memory 1170 may include one or
5 more memory elements. Memory elements may include internal hard drives (such as magnetic, solid state, electric, and/or optical drives), ROM, RAM, and other suitable devices for storing electrical information. In some embodiments, inspector 1120, monitor 1130, configuration files 1140, working files 1150, and security module 175 may be stored in file system 1160. Processes associated with applications or files on
10 host 1110 may be run in memory 1170.

In some embodiments, control module 1180 may be implemented using software and/or hardware. For example, control module 1180 may be implemented as middleware using JAVA on a TOMCAT server. Control module 1180 may be coupled to one or more networks that allow for communication to host 1110 and/or
15 malware detection system 1190. In various embodiments, this may be done using Web service technology, such as Javascript-based Application Programming Interfaces (APIs), Simple Object Access Protocol (SOAP) Remote Procedure Calls (RPCs), and/or other suitable network based APIs. For example, control module 1180 may use one or more networks to communicate with host 1110 and may also use one
20 or more different networks to communicate with malware detection system 1190. In some embodiments, control module 1180 may store and receive information from host 1110 and/or malware detection system 1190 without using computer networks. For example, media external to control module 1180 may be used to store information (e.g., inspector 1120, monitor 1130, configuration files 1140, and/or files suspected to
25 be malware) from control module 1180 and the media may be used to provide the stored information to host 1110 and/or malware detection system 1190.

In some embodiments, control module 1180 may cause inspector 1120 and/or monitor 1130 to be deployed on host 1110. Control module 1180 may also communicate with inspector 1120 and/or monitor 1130 once they are deployed on
30 host 1110. For example, control module 1180 may send configuration information indicating how inspector 1120 and/or monitor 1130 should perform operations on host 1110. Control module 1180 may also receive files and/or information about files

from inspector 1120 and/or monitor 1130 and use malware detection system 1190 to determine if these files are malware. Listing 1 below provides examples of a communication protocol between inspector 1120 and/or monitor 1130 and control module 1180. In some embodiments, control module 1180 may be coupled to other
5 hosts that are part of the same set of networks and/or enterprise as host 1110 (e.g., other hosts on a LAN). Control module 1180 may be configured to deploy inspectors and/or monitors (such as inspector 1120 and monitor 1130) to the other hosts. Control module 1180 may be configured to compare or otherwise analyze the information received from host 1110 as well as the other hosts. By aggregating and
10 comparing this information, control module 1180 may be able to update configuration information used by inspector 1120 and monitor 1130 as further described below.

Listing 1

RegisterClient - register Client's public key with Control Module and
15 Control Module's public Key with Client.
RequestFile - provide updated files upon request to Client.
UploadHashFile - accepts signed file of suspicious hash files from Client.
UploadWantedFiles - Accepts signed package of wanted files from Client for
storage and forwarding to Control Module.
20 **QueryArchive** - Provides query view of Malware Detection System's archive table. Used to search Malware Detection System data for possible matches found during Client scan.
UploadFilesForAnalysis - Accepts signed package of suspicious files from Control Module (or other sources) along with required metadata describing the files.
25

Configuration files 1140 may be one or more files that contain information regarding how inspector 1120 and/or monitor 1130 may operate. Configuration files 1140 may be in a binary format or in a human-readable format. For example, configuration files 1140 may be in an Extensible Markup Language (XML) format
30 (see Listing 2 below). In some embodiments, configuration files 1140 may contain information including: lists of files that are known not to be malware (e.g., a National Institute of Standards and Technology (NIST) software repository list and/or custom lists of files that are known not to be malware, lists or hashes of files that are known to be malware, identifiers of system registry values known not to be malware-related,
35 identifiers of system startup settings known to not be malware-related, identifiers of

process-port associations known not to be malware-related, directories that should be scanned, directories that should not be scanned, plugin information, directories that should be used for the location of working files 1150, rules and policies (e.g., based on file type) specifying how scanning and monitoring should be accomplished on host 5 1110, instructions for determining when a file should be provided to control module 1180 for further analysis, instructions for information that should be provided to control module 1180 along with files that are suspected to be malware, and instructions for how files should be communicated to control module 1180 (i.e., network information and/or external media information). In some embodiments, the 10 form and structure of configuration files 1140 allows for both inspector 1120 and monitor 1130 to be easily configured and modular.

Listing 2

```

15 <?xral version="1 .0" encoding="UTF- 8"?>
<MHK_CONF xralns="rule.config.file.com"
xmlns :xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="rule.config.file.com config.xsd" >
  <CASE_LABEL > </CASE_LABEL>
  <WARDEN_COMM>
20   <WARDEN_IPx/WARDEN_IP>
   <WARDEN_PORT>< /WARDEN_PORT >
  </WARDEN_COMM >
  <FILE_KNOWN_GOOD>
   <LISTx/LIST>
25   <LISTx/LIST>
  </FILE_KNOWN_GOOD >
  <FILE_KNOWN_BAD >
   <LIST> </LIST>
   <LIST> </LIST>
30 </FILE_KNOWN_BAD>
  <REG_KNOWN_GOOD >
   <LISTx/LIST>
   <LISTx/LIST>
  </REG_KNOWN_GOOD >
35 <REG_KNOWN_BAD>
   <LISTx/LIST>
   <LISTx/LIST>
  </REG_KNOWN_BAD >
  <PORT_KNOWN_GOOD >
40   <LISTx/LIST>
   <LISTx/LIST>

```

```

    </PORT_KNOWN_GOOD >
    <PORT_KNOWN_BAD >
      <LISTx/LIST>
      <LISTx/LIST>
5    </PORT_KNOWN_BAD >
    <MAGIC_DB ></MAGIC_DB >
    <OUTPUT >
      <DIRx/DIR>
      <OUTFILEx/OUTFILE>
10    <LOGFILEx/LOGFILE>
      <LOGLEVEL ></LOGLEVEL >
      <HASH_IMG_FILEx/HASH_IMG_FILE >
      <REG_IMG_FILEx/REG_IMG_FILE>
    </OUTPUT >
15  <PLUGINS>
    <EXTERNAL >
      <PLUGIN_NAME x /PLUGIN_NAME >
      <EXE_NAME x /EXE_NAME >
      <PARAMETERS x /PARAMETERS >
20    <MAX_RUN_T IMEx /MAX_RUN_T IME >
    </EXTERNAL >
  </PLUGINS>
  <MHK_RULE>
    <DIRECTORY_SCAN>
25    <RECURS IVEx /RECURS IVE >
      <DETECT_ADS </DETECT_ADS>
      <EXTENS IONS ></EXTENS IONS >
      <SELECTx/SELECT>
      <PRUNE x /PRUNE >
30    <PATH></ PATH >
    </DIRECTORY_SCAN>
    <REGISTRY_SCAN>
      <RECURS IVEx /RECURS IVE >
      <SELECTx/SELECT>
35    <PRUNE x /PRUNE >
      <KEYx/KEY>
    </REGISTRY_SCAN>
    <PROCESS_SCAN>
      <SELECTx/SELECT>
40    <SELECTx/SELECT>
      <PRUNE x /PRUNE >
    </PROCESS_SCAN>
    <PORT_SCAN>
      <SELECTx/SELECT>
45    <SELECTx/SELECT>
      <PRUNE x /PRUNE >
    </PORT_SCAN>
    <SERVICE_SCAN>

```

```

    <SELECTx/SELECT>
    <SELECTx/SELECT>
    <PRUNE></PRUNE>
5  </SERVICE_SCAN>
    <STARTUP_SCANx/STARTUP_SCAN>
    <MEMORY_DUMP>
        <NAME></NAME>
        <PATH></PATH>
        <MD5_HASHx/MD5_HASH>
10    <SHA1_HASHx/SHA1_HASH>
        <OUTPUT_TEMPLATE_NAME x /OUTPUT_TEMPLATE_NAME >
    </MEMORY_DUMP >
    <MEMORY_SEARCH>
        <SELECTx/SELECT>
15    <PRUNE x /PRUNE >
        <STRINGx /STRING>
        <STRINGx /STRING>
        <CONTEXT_SIZE x /CONTEXT_SIZE >
    </MEMORY_SEARCH >
20    <RETRIEVE_FILEx/RETRIEVE_FILE>
    <RETRIEVE_FILEx/RETRIEVE_FILE>
</MHK_RULE>

```

In some embodiments, malware detection system 1190 comprises one or more
 25 components operable to analyze files and/or information about files to determine
 whether those files and/or information are, or include, malware. Malware detection
 system 1190 may include software and/or hardware as well as networking
 infrastructure if appropriate. In some embodiments, malware detection system 1190
 provides facilities for human analysts to analyze files and/or information about files to
 30 determine whether the files are malware. Malware detection system 1190 may be
 configured to receive determinations by human analysts as to whether the files are
 malware and may be operable to transmit these determinations to other suitable
 systems. In some embodiments, malware detection system 1190 may be implemented
 using some or all of system 100 of FIGURE 1, control module 200 of FIGURE 2,
 35 detection subsystem 300 of FIGURE 3, and disposition subsystem 400 of FIGURE 4.

In some embodiments, inspector 1120 may perform a scan of host 1110. The
 scan may include scanning file system 1160 and or memory 1170. The scan may
 include various aspects of host 1110, including one or more of a system registry,
 system startup information, system services, and a system network port. The scan is

performed in accordance with configuration files 1140. For example, configuration files 1140 may specify what files should be scanned. In various embodiments, this may be done based on the location of the files or based on the type of the files. This may provide an advantage in that it may optimize the scan so that it is performed
5 more quickly and so that it consumes less resources.

In some embodiments, files that are determined to be scanned are then processed by the inspector 1120. If inspector 1120 is also scanning processes loaded into memory 1170, then the files to be processed may include any files associated with the processes. The processing of the files of interest may include creating one or
10 more hashes of the files. As just two examples, the hashes may include an MD5 hash and/or a SHA1 hash. However, the present invention contemplates using any suitable types of hashing for processing the files of interest. Entropy values may also be computed for the files. "Magic number" analysis may also be performed on the files, which may include analyzing the contents of the file to determine its file type. During
15 this processing, a file may be created that contains information regarding host 1110, process information regarding processes running on host 1110, and information regarding the files being processed. Information regarding the files being processed may include one or more of the following, for example: name, size, owner, author, permissions, time stamps, hash values, entropy values, magic number results, other
20 metadata, and/or other suitable information. The file may be created in a binary format or a human-readable format, such as XML (see Listing 3 below). Such files may be stored as part of working files 1150.

Listing 3

```
25 <MHK_SCAN>
    <SYS_INFO>
        <SCAN_DATE></SCAN_DATE>
        <UPTIMEx/UPTIME>
        <HOST_NAME></HOST_NAME>
30 <USER_NAMEx/USER_NAME>
        <MAC_ADDRESS VALUE="">
            <IP_ADDRESSx / IP_ADDRESS>
        </MAC_ADDRESS>
    </SYS_INFO>
35 <TOOL_INFO>
```

```

    <TOOL_USED></TOOL_USED>
    <TOOLJVERSIONx /TOOL_VERSION>
    <CONFIG_FILE MD5="" SHA1=""x/CONFIG_FILE>
</TOOL_INFO>
5 <REGISTRY_SCAN>
    <RECURSIVE></RECURSIVE>
    <KEYx/KEY>
    <REGISTRY_DATA>
        <VALUE x /VALUE >
10 </REGISTRY_DATA>
</REGISTRY_SCAN>
<PROCESS_SCAN>
    <PROCESS_DATA>
        <PROCESS>
15 <PIDx/PID>
        <NAME x /NAME >
        <PATH></PATH>
        <PRIORITYx/PRIORITY>
        <OWNERx/OWNER>
20 <USERx/USER>
        <CREATION_TIME></CREATION_TIME>
        <USER_TIMEx/USER_TIME>
        <WORKING_SETx/WORKING_SET>
        <PEAK_WORKING_SETx/PEAK_WORKING_SET>
25 <PAGE_FILEx /PAGE_FILE >
        <PEAK_PAGE_FILEx /PEAK_PAGE_FILE>
        <MD5_HASHx/MD5_HASH>
        <SHA1_HASH ></SHA1_HASH >
        <ENTROPY_SCORE x /ENTROPY_SCORE >
30 <MAGIC_NUMBER_VALUE ></MAGIC_NUMBER_VALUE >
        <KNOWN_LISTx /KNOWN_LIST>
        <PE_CHECKSUMx/PE_CHECKSUM>
        <PDF_METADATA>
            <PLUGIN_DATA key=""></PLUGIN_DATA>
35 </PDF_METADATA>
        <O2K7_METADATA>
            <PLUGIN_DATA key="" x /PLUGIN_DATA>
        </O2K7_METADATA>
        <EXTERNAL_PLUGIN>
40 <PLUGIN_DATA key=""></PLUGIN_DATA>
        <EXTERNAL_PLUGIN>
        <DLL_LOADED x /DLL_LOADED >
    </PROCESS >
    <DLL>
45 <PATH></PATH>
        <FILE_SIZEx/FILE_SIZE>
        <MD5_HASHx/MD5_HASH>
        <SHA1_HASHx/SHA1_HASH>

```

```

5      <ENTROPY_SCORE></ENTROPY_SCORE>
      <MAGIC_NUMBER_VALUE></MAGIC_NUMBER_VALUE>
      <KNOWN_LIST></KNOWN_LIST>
      <PE_CHECKSUM></PE_CHECKSUM>
      <PDF_METADATA></PDF_METADATA>
      <O2K7_METADATA></O2K7_METADATA>
      </DLL>
      </PROCESS_DATA>
10     </PROCESS_SCAN>
      <SERVICERS_CAN>
      <SERVICE_DATA>
      <SERVICE>
      <NAME_x /NAME >
      <DISPLAY_NAME ></DISPLAY_NAME >
15     <MD5x/MD5>
      <SHA1x/SHA1>
      <KNOWN_LISTx /KNOWN_LIST>
      <PATH></PATH>
      <DEPENDENCYx/DEPENDENCY>
20     <DEPENDENCY></DEPENDENCY>
      <ACCOUNT ></ACCOUNT >
      <STATEx/STATE>
      <TYPEx/TYPE>
      <START_TYPE x /START_TYPE >
25     <ERROR_CONTROL x /ERROR_CONTROL >
      <DESCRIPTION></DESCRIPTION>
      </SERVICE>
      </SERVICE_DATA>
      </SERVICE_SCAN>
30     <STARTUP_SCAN>
      <LOCATION>
      <NAME_x /NAME >
      <PATH></PATH>
      <CONTENTS >
35     <ENTRY ></ENTRY >
      </CONTENTS >
      </LOCATION>
      </STARTUP_SCAN>
      <PORT_SCAN>
40     <PORT_DATA>
      <PORT>
      <LOCAL_PORT x /LOCAL_PORT >
      <LOCAL_IPx/LOCAL_IP>
      <STATE ></STATE >
45     <REMOTE_IPx/REMOTE_IP>
      <REMOTE_PORTx/REMOTE_PORT>
      <OWNER_NAME x /OWNER_NAME >
      <OWNER_PATH x /OWNER_PATH>

```

```

    <OWNER_MD5></OWNER_MD5>
    <OWNER_SHAL></OWNER_SHAL>
    <OWNER_KNOWN_LISTx /OWNERKNOWN_LIST>
    <KNOWN_LIST></KNOWN_LIST>
5      </PORT>
      </PORT_DATA>
    </PORT_SCAN>
    <MEMORY_SCAN>
      <MEMORY_SEARCH>
10      <STRINGx/STRING>
      <SEARCH_RESULTS>
        <PROCESS_RESULTS>
          <NAME></NAME>
          <PATH></PATH>
15      <MD5_HASHx/MD5_HASH>
          <SHAL_HASHx/SHAL_HASH>
          <KNOWN_LISTx/KNOWN_LIST>
          <CONTEXT_STRINGS>
            <STRINGx /STRING>
20      </CONTEXT_STRINGS>
          </PROCESS_RESULTS>
        </SEARCH_RESULTS>
      </MEMORY_SEARCH>
    </MEMORY_SCAN>
25    <DIRECTORY_SCAN>
      <DIRECTORYx/DIRECTORY>
      <RECURSIVE></RECURSIVE>
      <DETECT_ADSx/DETECT_ADS>
      <EXTENSIONSx /EXTENSIONS>
30    <FILE_DATA>
      <FILE>
        <FILE_NAME x /FILE_NAME>
        <FILE_LOCATIONx/FILE_LOCATION>
        <FILE_SIZEx/FILE_SIZE>
35      <IS_ADSx/IS_ADS>
        <MD5_HASHx/MD5_HASH>
        <SHAL_HASH></SHAL_HASH>
        <ENTROPY_SCORE x /ENTROPY_SCORE>
        <MAGIC_NUMBER_VALUE></MAGIC_NUMBER_VALUE>
40      <KNOWN_LISTx/KNOWN_LIST>
        <PE_CHECKSUM></PE_CHECKSUM>
        <PDF_METADATA>
          <PLUGIN_DATA key=""></PLUGIN_DATA>
        </PDF_METADATA>
45      <O2K7_METADATA>
        <PLUGIN_DATA key=""></PLUGIN_DATA>
      </O2K7_METADATA>
    <EXTERNAL_PLUGIN>

```

```

        <PLUGIN_DATA key= ""></ PLUGIN_DATA>
        <EXTERNAL_PLUGIN>
          </FILE>
          </FILE_DATA>
5      </DIRECTORY_SCAN>
      <SCAN_INFO>
        <CASE_LABEL x /CASE_LABEL >
        <SCAN_START_TIME ></SCAN_START_TIME >
        <SCAN_STOP_TIME ></SCAN_STOP_TIME >
10     <SCAN_SECONDSx/SCAN_SECONDS>
        <DIRECTORY_SCAN_INFO>
          <FILES_FOUND ></FILES_FOUND >
          <FILES_PROCESSEDx/FILES_PROCESSED>
          <FILES_RECORDEDx/FILES_RECORDED>
15     <FILES_KNOWN_GOOD x /FILES_KNOWN_GOOD >

          <FILES_KNOWN_GOOD_PERCENTx/FILES_KNOWN_GOOD_PERCENT>
          <FILES_KNOWN_BADx /FILES_KNOWN_BAD >

20     <FILES_KNOWN_BAD_PERCENT>< /FILES_KNOWN_BAD_PERCENT>
          </DIRECTORY_SCAN_INFO >
          <PROCESS_SCAN_INFO>
            <PROCESSES_FOUND x/PROCESSES_FOUND >
            <PROCESSES_KNOWN_GOOD x /PROCESSES_KNOWN_GOOD >
25     <PROCESSES_KNOWN_GOOD_PERCENTx/PROCESSES_KNOWN_GOOD_PERC
ENT>
            <PROCESSES_KNOWN_BAD x /PROCESSES_KNOWN_BAD >

30     <PROCESSES_KNOWN_BAD_PERCENTx /PROCESSES_KNOWN_BAD_PERCEN
T>
            <PROCESSES_RECORDEDx/PROCESSES_RECORDED>
            <PROCESS_DLLS_FOUND x /PROCESS_DLLS_FOUND >
            <PROCESS_DLLS_GOOD x /PROCESS_DLLS_GOOD >
35     <PROCESS_DLLS_GOOD__PERCENTx/PROCESS_DLLS_GOOD_PERCENT>
            <PROCESS_DLLS_BAD x /PROCESS_DLLS_BAD >

            <PROCESS_DLLS_BAD_PERCENTx/PROCESS_DLLS_BAD_PERCENT>
40     <PROCESS_DLLS_RECORDEDx/ PROCESS_DLLS_RECORDED>
          </PROCESS_SCAN_INFO >
          <MEMORY_SCAN_INFO>
            <MEMORY_DUMPS ></MEMORY_DUMPS >
            <STRING_OCCURRENCES ></STRING_OCCURRENCES >
45     <PROCESS_MATCHES></PROCESS_MATCHES>
            <PROCESSES_RECORDEDx/PROCESSES_RECORDED>
          </MEMORY_SCAN_INFO >
          <SERVICE_SCAN_INFO>

```

```

    <SERVICES_FOUNDx/SERVICES_FOUND>
    <SERVICES_KNOWN_GOOD></SERVICES_KNOWN_GOOD >

5   <SERVICES_KNOWN_GOOD_PERCENT></SERVICES_KNOWN_GOOD_PERCEN
    T>
    <SERVICES_KNOWN_BAD></SERVICES_KNOWN_BAD >

    <SERVICES_KNOWN_BAD_PERCENT> </SERVICES_KNOWNBAD_PERCENT >
    <SERVICES_RECORDEDx/SERVICES_RECORDED>
10  </SERVICE_SCAN_INFO>
    <PORT_SCAN_INFO>
    <PORTS_FOUND></PORTSFOUND>
    <PORTS_KNOWN_GOOD></PORTS_KNOWN_GOOD>

15  <PORTS_KMOWN_GOOD_PERCENT>< /PORTS_KNOWN_GOOD_PERCENT>
    <PORTS_KNOWN_BAD></PORTS_KNOWN_BAD >

    <PORTS_KNOWN_BAD_PERCENTx/PORTS_KNOWN_BAD_PERCENT>
    <PORTS_RECORDED ></PORTS_RECORDED >
20  </PORT_SCAN_INFO>
    </SCAN_INFO>

```

Inspector 1120 may compare the information generated by processing the files to configuration files 1140. For example, the files may be compared to a list of files known not to be malware. As another example, the files of interest may be compared to a list of files known to be malware. If a file has been determined not to be malware, inspector 1120 may take no further action with respect to that file. If a file has been determined to be malware or if the file does not appear in the list of known malware or the list of files that are known not to be malware, then the file may be sent to control module 1180. Sending files to control module 1180 may include copying the file, compressing the file into a package, securing the package using encryption or digital signatures, and then sending the file to control module 1180. Information related to files (i.e., metadata) determined to be sent to control module 1180 may also be part of the package that is sent. Configuration files 1140 may determine whether files are sent to control module 1180 based on their file type. For example, a rule may indicate that if it is determined that a scanned executable file does not appear on the list of files that are known to be malware nor does it appear on the list of files known not to be malware, then the file should be sent for further analysis. Another rule may

indicate that if the same circumstances existed for a text file, then the file should not be sent for further analysis.

After the scan has been completed, inspector 1120 may delete aspects of the scan from host 1110. This may include some or all of configuration files 1140 and/or
5 working files 1150. This may also include inspector 1120 itself. Performing such deletions may, in some embodiments, prevent the scan from being detected by malware that may be present on host 1110. As a result, if host 1110 was comprised by malware in some situations, attempts to detect the malware by the scan may not be
10 thwarted by the malware itself. Deleting such items may also prevent malware present on host 1110 from identifying the information being used to detect the malware's presence on host 1110.

In some embodiments, deleting aspects of the scan from host 1110 may be performed in a secure manner. This may include overwriting blocks (in some cases, multiple times) of file system 1160 and/or memory 1170 with data (such as binary
15 "ones" or "zeroes") rather than merely deleting these aspects by modifying a table used to organize file system 1160 and/or memory 1170 as may be the case when using conventional deletion commands provided by an operating system.

In some embodiments, scanning file system 1160 and/or memory 1170 using inspector 1120 may provide a fast, yet thorough manner of detecting malware that
20 may be attempting to mask its true nature as malware. By being able to sort through files that are both known to be malware and known not to be malware, inspector 1120 can thoroughly search and analyze files whose status as malware is unknown. Further, in conjunction with control module 1180 and malware detection system 1190, files may be quickly sent and analyzed to determine if they are malware. In
25 some embodiments, control module 1180 may leverage information from other hosts within an enterprise that host 1110 belongs to modify configuration files 1140. For example, configuration files 1140 may direct inspector 1120 to detect files that are statistically variant from other hosts within the enterprise. In another example, control module 1180 may compare files received from a variety of hosts in order to
30 determine if a file sent by inspector 1120 is a statistical variant from files on other hosts. This may be the case, in various embodiments, if the file is present on only a small number of hosts amongst a larger number of hosts within a network. Based on

this determination, control module 1180 may send the file to malware detection system 1190.

In some embodiments, statistical variance may be determined with respect to one more different criteria. As discussed above, for example, statistical variance may be determined with respect to the distribution of a file amongst various hosts in the same network or enterprise. Another example includes determining statistical variance with respect to metadata. This may occur, in some embodiments, by determining anomalies within a file's metadata when comparing it to the file's type. For example, if the file is a Portable Document Format (PDF) file, analysis of its metadata may include determining what program was used to create the file. A statistical variance may be determined through this analysis if it is determined that the tool used to create the file is not commonly used. Configuration files 1140 may include information regarding metadata based on file type to facilitate this analysis. As another example, a statistical variance with respect to metadata may be determined if there is an absence of some or all metadata of a file. For example, if the file is a PDF file but it does not have any metadata or if portions of its existing metadata (e.g., the tool used to create the file, the owner of the file, the creation date, and other suitable aspects of metadata) are missing, then a statistical variance may be determined regarding the file.

In some embodiments, statistical variance may be determined with respect to a file's entropy. For example, an entropy value of a file may be determined and analyzed in light of the file's type. If the entropy value is higher than normal in light of the file's type, a statistical variance may be determined. In some situations, high entropy values mean that there exists a high degree of randomness in the file which may mean the file has been encrypted or otherwise packed in an attempt to make reverse engineering difficult. Configuration files 1140 may include information regarding normal entropy values for various file types. An algorithm such as Shannon entropy may be used to determine the entropy value of the file. Other suitable algorithms for determining entropy may be used, including the binary entropy function along with a Markov model.

In some embodiments, monitor 1130 may perform monitoring of host 1110 to detect potential files or processes that are malware. Configuration files 1140, for

example, may direct monitor 1130 to monitor access to certain files or components (such as registry) of host 1110. Configuration files 1140 may also direct monitor 1130 to monitor the introduction of new files into host 1110. Similar to the discussion of inspector 1120 above, monitor 1130 may process files associated with any access
5 of host 1110 that configuration files 1140 indicate should be monitored. Examples of types of access that may be monitored include receiving or transmitting files through a network (i.e., downloading using a browser, an e-mail client, an instant messaging client, and a peer-to-peer client), modifying or deleting files, running or terminating processes, adding from or transferring files to storage mediums (i.e., compact discs,
10 diskettes, and removable storage drives). As with inspector 1120, monitor 1130 may compare the files to lists of files known to be malware and lists of files known not to be malware. Using these comparisons, monitor 1130 may be able to quickly and efficiently determine what files need to be sent to control module 1180 for further analysis. After the files have been sent, monitor 1130 may delete files that may have
15 been part of working files 1150 as did inspector 1120. This may include deleting files associated with the analysis of the processes running on host 1110 and the monitoring of host 1110. For example, some or all of configuration files 1140, the copies of the files generated for packaging that may be part of working files 1150, and information generated during the processing of the files may be deleted.

20 Monitor 1130 may be transparent to a user of host 1110 as it uses configuration files 1140 to direct the monitoring and analysis of files and processes on host 1110. In some embodiments, this may be advantageous because monitor 1130 may be used as a real-time file scanner that is difficult to detect by malware that may be present on host 1110.

25 In some embodiments, monitor 1130 may perform actions visible to a user of host 1110. For example, monitor 1130 may cause a form to appear on host 1110 that requests a user to enter information regarding detected activity. Such activity may include downloading a file, accessing a file, modifying the configuration of host 1110, and introducing a file to host 1110 through external media. Other suitable activities
30 may also be detected. The requested information may include reasons for performing the detected activity, and program used to perform the detected activity. Other suitable information may be requested. In some embodiments, the requested

information may be transmitted along with the a file for analysis. This may prove helpful, in some embodiments, when analyzing the file to determine if it is malware. Configuration files 1140 may include rules and/or policies that determine when information should be requested from a user. This may be done based on file type or
5 other suitable criteria. For example, if the file is determined not to correspond to either a list of files known to be malware or a list of files known not to be malware, it may be determined to request the user enter information regarding the file as described above.

In some embodiments, monitor 1130 may invoke inspector 1120 to perform a
10 scan of host 1110. Monitor 1130 may request a scan of some or all of file system 1160 and/or memory 1170. Monitor 1130 may call upon inspector 1120 as directed by configuration files 1140.

In some embodiments, inspector 1120 may reside on a bootable medium and a scan of host 1110 may be initiated by coupling the bootable medium to host 1110.
15 For example, inspector 1120 may reside in a removable storage medium that may be coupled to host 1110 using a Universal Serial Bus (USB) port. Host 1110 may then be booted and configured to run programs on the bootable medium, such as an operating system that may be present on the bootable medium. Afterwards, a scan may be initiated using inspector 1120. Performing a scan in this manner may provide
20 several advantages in various embodiments. For example, problems with scanning hosts with rootkits may be avoided. As another example, a scan may be performed without needing a network connection.

As may be apparent from the discussion above, security module 1175, in some embodiments, may perform different functions than either inspector 1120 and monitor
25 1130. Security module 1175 may examine host 1110 for malware based on signatures or other forms of detecting known malware. This type of analysis may be resource intensive. In addition, parameters and files used to detect malware by security module 1175 remain on host 1110, making the protection offered by security module
1175 susceptible to attack. In contrast, in some embodiments, the operation of system
30 1100 provide a distinct advantage in detecting malware that has thwarted attempts such as those performed by security module 1175 since it detects files that may be suspected malware by focusing on files whose presence on host 1110 is a statistical

variance when compared with other hosts. The low-level aspects of inspector 1120 and monitor 1130 as well as the use of configuration files 1140 allow them to provide speedy analysis that may be crucial in the event of a zero-day attack.

In some embodiments, the operation of control module 1180 may provide several advantages when detecting malware. For example, control module 1180 may be able to handle a large number of communication sessions with various inspectors and monitors on various hosts. The ability to interface with malware detection system 1190 provides modularity to system 1100 as control module 1180 may use a plug-in based architecture. Further, since control module 1180 receives the results of malware detection system 1190, system 1100 may adapt as new forms of malware are detected. For example, control module 1180 may incorporate the results of malware detection system 1190 and send new versions of configuration files 1140 to inspector 1120 and/or monitor 1130.

FIGURES 12 and 13 are flowcharts illustrating example operations for detecting malware, according to certain embodiments of the present invention. In general, the steps illustrated in FIGURES 12 and 13 may be combined, modified, or deleted where appropriate, and additional steps may also be added to the example operation. Furthermore, the described steps may be performed in any suitable order. In some embodiments, the steps described below may be performed by any suitable combination of the elements discussed above with respect to FIGURE 11.

FIGURE 12 is a flowchart depicting an example of scanning a file system of a host (such as file system 1160 of host 1110) for suspected malware. At step 1210, in some embodiments, one or more security credentials may be generated. For example, inspector 1120 and/or monitor 1130 may generate the security credentials. An example security credential that may be generated at step 1210 includes an asymmetric key pair. At step 1211, in some embodiments, one or more security credentials may be retrieved. Credentials retrieved at this step may include a public key associated with a control module (such as control module 1180). System information (such as hardware identifiers, network information, protocol information, or other suitable forms of system information) associated with the control module may also be part of the retrieved security credentials. Step 1211 may occur by sending a request for the credentials to control module 1180. In some embodiments,

the credentials retrieved and/or generated at steps 1210 and 1211 may be used to perform secure communication with control module 1180, as further described below. In some embodiments, some or all of the credentials discussed above may already be included with inspector 1120 and/or monitor 1130.

5 At step 1212, in some embodiments, configuration parameters may be requested. For example, inspector 1120 and/or monitor 1130 may request the configuration parameters. The request may be sent to control module 1180 and may be secured using one or more of the security credentials generated and retrieved at steps 1210-1211. For example, the request may be encrypted using the security
10 credentials. As another example, the request may be digitally signed using the credentials. At step 1213, in some embodiments, the request sent at step 1212 may be validated. Step 1213 may be performed by a control module, such as control module 1180. Validation of the request may include examining the security credentials used in transmitting the request, such as the credentials generated at step 1210 and received
15 at step 1211. For example, a digital signature may be validated. As another example, the request may be decrypted.

 At step 1214, in some embodiments, the parameters requested at step 1212 may be received. For example, inspector 1120 and/or monitor 1130 may receive the parameters requested at step 1212. As a particular example, in response to validating
20 the request at step 1213, control module 1180 may send the configuration parameters. The configuration parameters received at step 1214 may be secured. For example, the configuration parameters may be encrypted or digitally signed using the security credentials generated and retrieved at steps 1210 and 1211. At step 1215, in some
25 embodiments, the parameters received at step 1214 may be validated. For example, inspector 1120 and/or monitor 1130 may validate the parameters received at step 1214. The validation of the parameters may include examining the security credentials used in transmitting the parameters, such as the credentials generated at step 1210 and received at step 1211. For example, a digital signature may be
30 validated. As another example, the parameters received at step 1214 may be decrypted.

 In some embodiments, steps 1212-1215 may be repeated as needed to request various types of configuration parameters. For example, when steps 1212-1215 are

initially performed, a set of rules may be requested and received. Continuing the example, when steps 1212-1215 are performed again, a list of files known not to be malware may be requested and received. Further, in this example, when steps 1212-1215 are performed yet again, a list of files known to be malware may be requested and received. In some embodiments, steps 1212-1215 are not repeated as in this example. Instead, all of the desired configuration parameters are requested, transmitted, and validated once.

At step 1216, in some embodiments, a scan of the file system (e.g., file system 1160) may begin by determining files of the file system (e.g., file system 1160) that are of interest. For example, inspector 1120 and/or monitor 1130 may perform the scan of file system 1160. The files of interest may be determined by referencing the configuration parameters received and validated at steps 1214-1215. For example, the configuration parameters may indicate that files corresponding to one or more file types should be analyzed while others should not. As another example, the configuration parameters may specify that files residing in one or more directories should not be analyzed while files residing in other directories should be analyzed. In some embodiments, all files in the file system are analyzed.

At step 1217, in some embodiments, the files of interest determined at step 1216 may be processed. For example, inspector 1120 and/or monitor 1130 may process the files determined to be of interest. The processing of the files of interest may include creating one or more hashes of the files. As just two examples, the hashes may include an MD5 hash and/or a SHA1 hash. However, the present invention contemplates using any suitable types of hashing for processing the files of interest. Entropy values may also be computed for the files. "Magic number" analysis may also be performed on the files, which may include analyzing the contents of the file to determine its file type. During this processing, a file may be created that contains information regarding host 1110, process information regarding processes running on host 1110, and information regarding the files being processed. Information regarding the files (i.e., metadata) being processed may include one or more of the following, for example: name, size, owner, author, permissions, time stamps, hash values, entropy values, magic number results, and/or other suitable

information. The file may be created in a binary format or a human-readable format, such as XML.

At step 1218, in some embodiments, the files of interest are filtered. Filtering of the files of interest may be performed after the files have been processed at step 5 1217. In certain embodiments, inspector 1120 and/or monitor 1130 filters the files of interest. The filtering may be performed according to the parameters received at step 1214. For example, the files of interest may be compared to a list of files known not to be malware. As another example, the files of interest may be compared to a list of files known to be malware. In some embodiments, the processing of step 1217 may 10 be used at step 1218 to facilitate the filtering. For example, hashes computed at step 1217 may be used to compare a file of interest to lists of files known to be malware and known not to be malware. In some embodiments, the contents of the files of interest may also be analyzed at this step. The content that is analyzed may include (but not limited to) ASCII or Hex data. For example, some or all of the files of 15 interest may be analyzed to detect network addresses, such as URLs or Web proxy addresses. The detected items may be compared to the lists of files known to be malware and files known not to be malware. For example, detected URLs may be compared to lists of URLs known to be malware and lists of URLs known not to be malware.

At step 1219, in some embodiments, a determination may be made as to which 20 files should be sent for further analysis. For example, inspector 1120 and/or monitor 1130 may determine which files should be sent for further analysis. The determination of step 1219 may be performed with reference to the configuration parameters received at step 1214. The configuration parameters may include rules 25 and/or policies. The parameters may indicate that files matching the list of files known not to be malware should not be sent for further analysis while files that match the list of files known to be malware should be sent for further analysis or for appropriate responses to be taken. Files that do not match either the list of files known to be malware or the list of files known not to be malware may be sent for 30 further analysis. As another example, the configuration parameters may indicate that not all files that have corresponding entries in the list of files known to be malware should be sent. Instead, in this example, the configuration parameters may indicate

that only a subset of these files should be sent for further analysis based on file type or the type of malware to which the file corresponds. In some embodiments, files for which a match is not determined in the lists of the configuration parameters may be determined to be sent for further analysis. In some situations, files that do not match
5 any of the lists of files known to be malware or of files known not to be malware may be sent only if they correspond to one or more file types that may be specified in the parameters received at step 1214. As another example, the parameters may specify that files that do not correspond to such lists, but which have a file type that does not match the file extension of the file should be sent. This type of analysis may occur by
10 comparing the "magic number" of the file (e.g., determined at step 1217) to the name and extension of the file.

At step 1220, in some embodiments, files that are determined to be sent for further analysis may be packaged. For example, inspector 1120 and/or monitor 1130 may package the files that are determined to be sent for further analysis. The files
15 may be packaged individually, in groups of one or more files, all together, or in any other suitable manner. Files that are to be packaged may be copied to another directory on host 1110, to an external medium such as removable storage drive, or to any other suitable location. Packaging the files may include compressing the files, encrypting the files, and/or signing the files. The package may include information
20 related to the file, such as one or more of the following: name, size, owner, author, permissions, time stamps, hash values, entropy values, magic number results, and/or other suitable information. The package may be in a compressed format (e.g., a zip format). Other compression schemes such as gzip, bz2, 7-zip, rar, and/or other suitable schemes may be utilized. The package may also be signed and/or encrypted
25 using the security credentials generated and or received at steps 1210 and 1211.

At step 1221, the one or more packages created at step 1220 may be sent for analysis. This may be done using Web service technology, such as Javascript-based Application Programming Interfaces (APIs), Simple Object Access Protocol (SOAP) Remote Procedure Calls (RPCs), and/or other suitable network based APIs. For
30 example, inspector 1120 and/or monitor 1130 may send the packages created at step 1220 for further analysis. In certain embodiments, control module 1180 receives and validates the packages sent at step 1221. The validation of packages may be

according to the credentials discussed at steps 1210 and 1211. In certain embodiments, after the packages have been validated (if appropriate), control module 1180 may further analyze the files and the information about the files. This further analysis may include comparing the files and the information about the files to similar information from other hosts on the same or different networks. The further analysis may also include noting statistical variances between the files present on different hosts. Control module 1180 may alter configuration parameters that are sent to items such as inspector 1120 and/or monitor 1130 in response to this analysis. In some embodiments, control module 1180 may transmit the files sent at step 1220 so that those files may be analyzed for malware. For example, control module 1180 may transmit the files sent at step 1220 to a malware detection system, such as malware detection system 1190 of FIGURE 11. Control module 1180 may receive a response from the malware detection system (e.g., malware detection system 1190) indicating whether the transmitted files are or include malware. This information may also be used by control module 1180 to alter configuration parameters that are sent to components such as inspector 1120 and/or monitor 1130. For example, the lists of files known to be malware and known not to be malware may be modified according to the determination made by the malware detection system (e.g., malware detection system 1190) for use by inspector 1120 and/or monitor 1130 future analysis of files of hosts. As another example, this information may be used to modify rules and or policies regarding files or file types.

At step 1222, in some embodiments, a clean-up operation may be performed. For example, inspector 1120 and/or monitor 1130 may perform the clean-up operation. The clean-up operation may include deleting files associated with the scan of host 1110. For example, some or all of the configuration parameters received at step 1214, the copies of the files generated for packaging at step 1220, and information generated during the analysis that occurred at steps 1217-1219 may be deleted. In some embodiments, the component that performed the scan (e.g., inspector 1120 and/or monitor 1130) may also be deleted. The deletion operations of step 1222 may be performed in a secure manner. For example, digital zeroes or ones may be written to disk where the items that should be deleted reside. This write operation may occur multiple times to help ensure secure deletion. Any suitable

method of secure deletion may be used as part of the clean-up operation. After the clean-up operation has been performed, in some embodiments, the scan of file system 1160 of host 1110 may be complete (as in step 1223).

FIGURE 13 is a flowchart illustrating an example method for scanning processes and monitoring a host (e.g., processes running in memory 1170 of host 1110 of FIGURE 11) for suspected malware. In some embodiments, steps 1310 through steps 1315 may be performed similarly to steps 1210 through steps 1215 of FIGURE 12. Generally, security credentials may be generated by a monitor, such as monitor 1130 of FIGURE 11. The monitor may retrieve a set of security credentials from a control module, such as control module 1180 of FIGURE 11 at step 1311. The security credentials obtained at steps 1310 and 1311 may be used to secure files transferred between monitor 1130 and control module 1180. At step 1312, in some embodiments, monitor 1130 may request configuration parameters from control module 1180. The request for configuration parameters may be secured using the credentials obtained in steps 1310 and 1311. At step 1313, control module 1180 may validate the request for parameters using the security credentials obtained at step 1311. In certain embodiments, after validating the request (if appropriate), control module 1180 may send the requested parameters to monitor 1130. The requested parameters may be sent in a secure manner, using the security credentials obtained at step 1311 for example. Monitor 1130 may receive the configuration parameters at step 1314 and validate the parameters at step 1315, using the security credentials obtained at steps 1310 and 1311 for example.

At step 1316, in some embodiments, files associated with processes running on host 1110 may be processed. For example, monitor 1130 may identify files associated with each process running on host 1110. These files may include, for example, executable files, library files (such as dynamically linked libraries or "DLLs"), files being accessed by a process, and/or any other suitable files. These files may then be processed in a manner similar to the files processed at step 1217 of FIGURE 12. Generally, in certain embodiments, this processing may include calculating hash values and entropy values, as well as determining "magic number" information for the files.

At step 1317, in some embodiments, monitor 1130 may determine which processes should be monitored. This determination may be accomplished by referencing the configuration parameters received at step 1314. For example, files associated with running processes may be compared to a list of files known not to be malware. As another example, files associated with running processes may be compared to a list of files known to be malware. In some embodiments, if a file associated with a process matches an entry in the list of files known to be malware or if a file associated with a process does not match any entry in the list of files known to be malware or in the list of files known to be malware, then monitor 1130 may send the file (as well as other files or information associated with the process, if appropriate) to control module 1180 for analysis. This may occur in a manner similar to steps 1220 and 1221 of FIGURE 12. In various embodiments, all files associated with running processes may be sent for further analysis. In some embodiments, the processing of step 1316 may be used at step 1317 to facilitate the determination of which files should be monitored. For example, hashes computed at step 1316 may be used to compare the a file associated with a running process to lists of files known to be malware and known not to be malware. In some embodiments, the contents of the files as well as the portions of memory associated with the processes may also be analyzed at step 1317. For example, network addresses (such as URLs or Web proxy addresses) in the files or the memory portions associated with the running processes may be detected. Detected network addresses may be compared to the lists of files known to be malware and files known not to be malware. For example, detected URLs may be compared to lists of URLs known to be malware and lists of URLs known not to be malware.

The results of the analysis performed on the processes running on host 1110 may be compared to rules and/or policies that may be part of the configuration parameters received at step 1314. The determination of which processes should be monitored may be made in response to this comparison. For example, if all of files associated with a process are on the list of files that are known not to be malware, then monitor 1130 may determine that the process should not be monitored. As another example, if any file associated with a process is on the list of files that are known to be malware, then monitor 1130 may determine that the process should be

monitored. In certain embodiments, if monitor 1130 determines that the process should be monitored, monitor 1130 may also determine that the process should be terminated. As another example, monitor 1130 may determine that a process should be monitored if some or all of the files associated with the process are not in either the
5 list of files known to be malware or the list of files known not to be malware. In some embodiments, the rules and/or policies may indicate that all of the processes running on host 1110 should be monitored.

At step 1318, in some embodiments, monitor 1130 may begin monitoring host 1110, including the processes determined at step 1317. In monitoring the processes,
10 monitor 1130 may track activity associated with each process. The tracked activity of the processes may include one or more of the following operations: creating, adding, or modifying files; accessing or modifying libraries; accessing or modifying system configuration parameters (such as registries, startup configurations, service configurations, port/process usage, or other files or settings associated with the
15 configuration of the host); accessing network resources or network locations; and/or any other suitable operations. Monitor 1130 may also perform other forms of monitoring of host 1130. For example, monitor 1130 may detect new files that are introduced to host 1110 (e.g., when a file is downloaded or a file is added from external media).

At step 1319, in some embodiments, as a result of monitoring these and/or
20 other aspects of host 1110, monitor 1130 may detect an access by a monitored process to system resources of host 1110. This may include, for example, a modification of a configuration file of host 1110 or an introduction of a new file to host 1110. At step 1320, in some embodiments, monitor 1130 may determine whether files associated
25 with the detected access to system resources of host 1110 at step 1319 should be sent for further analysis. The configuration parameters received at step 1314 may include rules and/or policies for determining whether an access of system resources should trigger monitor 1130 to send appropriate files for further analysis. For example, monitor 1130 determines that a monitored process modifies a sensitive file (e.g., a
30 system configuration file), monitor 1130 may determine that the file (i.e., the sensitive file in this example) and other aspects of the monitored process that modified the sensitive file should be sent to control module 1180 for analysis. As another example,

monitor 1130 may determine that a monitored process opened (or attempted to open) a network connection to a location. Monitor 1130 may analyze the location and compare the location to the configuration parameters received at step 1314. This comparison may indicate that the location is hazardous or otherwise inappropriate and may trigger the sending of files associated with the process to control module 1180 for analysis. As yet another example, a monitored process may introduce a file to host 1110 (e.g., by downloading a file to the host). The introduced file may be processed as in step 1316. The introduced file may be compared to lists of files that are known to be malware as well as lists of files that are known not to be malware. If the file is known to be malware or has a particular file type, then the file as well as other files associated with the monitored process may be determined to be sent to control module 1180 for analysis.

In some embodiments, a report may be generated (e.g., by monitor 1130). The report may include one or more of the following: what action occurred, when the action occurred, what user was logged into the system when the action occurred, what process performed the action, network information regarding the host where the action occurred (such as an IP address, host name, and/or a MAC address), or any other suitable information. Any files associated with the detected activity that had not been processed at step 1316 may be processed in a similar fashion as in step 1316. In some embodiments, the information collected about the process may be stored in one or more files such as binary files or a human-readable files (e.g., XML files). Monitor 1130 may also determine at step 1320 that these files should be transmitted.

At steps 1321 and 1322, in some embodiments, the files determined at step 1320 may be packaged and transmitted for further analysis. This may be done using Web service technology, such as Javascript-based Application Programming Interfaces (APIs), Simple Object Access Protocol (SOAP) Remote Procedure Calls (RPCs), and/or other suitable network based APIs. These steps may be performed in a similar manner to steps 1220 and 1221 of FIGURE 12. Generally, the files identified at step 1320 may be compressed and packaged for sending to control module 1180. The packages may then be digitally signed or encrypted and sent to control module 1180 for further analysis. Control module 1180 may validate the packages and send the packages to a malware detection system (e.g., malware

detection system 1190) to determine if the files in the packages are or include malware.

At step 1323, in some embodiments, monitor 1130 may perform a clean-up operation. The clean-up operation may include deleting files associated with the analysis of the processes running on host 1110 and the monitoring of host 1110. For example, some or all of the configuration parameters received at step 1314, the copies of the files generated for packaging at step 1321, and information generated during the analysis that occurred at steps 1316, 1317, and 1320 may be deleted. The deletion operations of step 1323 may be performed in a secure manner. For example, digital zeroes or ones may be written to disk where the items that should be deleted reside. This write operation may occur multiple times to help ensure secure deletion. Any suitable method of secure deletion may be used as part of the clean-up operation. After the clean-up operation has been performed, in some embodiments, monitor 1130 may continue to monitor host 1110 and the method may return to step 1318.

Although several embodiments have been illustrated and described in detail, it will be recognized that modifications and substitutions are possible without departing from the spirit and scope of the appended claims.

WHAT IS CLAIMED IS:

1. A computer-implemented method, comprising:
 - accessing a set of configuration parameters;
 - accessing a set of identifiers of files known not to be malware;
 - 5 accessing a set of identifiers of files known to be malware;
 - determining that a first file in a first host of a plurality of hosts should be analyzed by comparing the first file to the set of configuration parameters;
 - generating a first hash of the first file in response to determining that a first file should be analyzed;
 - 10 determining that the first hash is not in the set of identifiers of files known not to be malware and that the first hash is not in the set of identifiers of files known to be malware;
 - determining a statistical variance with respect to the first file in response to determining that the first hash is not in the set of identifiers of files known not to be malware and is not in the set of identifiers of files known to be malware;
 - 15 sending the first file and information related to the first file to be analyzed for malware in response to determining the statistical variance; and
 - deleting the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known to be malware after
 - 20 sending the first file.

2. The method of Claim 1, wherein:
 - the set of configuration parameters comprises a set of file types; and
 - comparing the first file to the set of configuration parameters comprises
 - 25 determining that a file type associated with the first file corresponds to a file type of the set of file types.

3. The method of Claim 1, wherein determining the statistical variance comprises determining that the first file is not present in a set of hosts of the plurality
- 30 of hosts.

4. The method of Claim 1, wherein deleting the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known to be malware further comprises overwriting a set of memory blocks that store the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known to be malware.

5. The method of Claim 1, further comprising:
determining, according to a first policy, a plurality of malware detection schemes to apply to the first file based on a file type of the first file;
10 scheduling the application of the determined plurality of malware detection schemes to the first file amongst a plurality of detection nodes according to a second policy; and
in response to determining the results of applying the plurality of malware detection schemes, determining that the first file is suspected malware or determining
15 that the at least one file is malware according to a third policy.

6. The method of Claim 1, wherein sending the first file and information related to the first file comprises storing the first file and information related to the first file in a computer-readable storage medium attached to the first host.

7. The method of Claim 1, wherein sending the first file and information related to the first file comprises sending the first file to a node remote from the first host.

8. The method of Claim 1, wherein information related to the first file comprises metadata related to the first file.

9. The method of Claim 1, wherein:
identifiers of files known to be malware comprise hashes of files known to be
30 malware; and
identifiers of files known not to be malware comprise hashes of files known not to be malware.

10. The method of Claim 1, wherein determining that the first file should be analyzed comprises determining that the first file should be analyzed in response to detecting that the first file has entered the first host through a network connection.

5

11. The method of Claim 1, wherein determining that the first file should be analyzed comprises determining that the first file should be analyzed in response to detecting that the first file has entered the first host through a removable, computer-readable, storage medium.

10

12. The method of Claim 1, further comprising:

detecting a network address associated with a process of the first host in random access memory;

comparing the detected network address to the set of configuration parameters;

15 and

sending a second file associated with the process, information related to the second file, and information related to the process to be analyzed for malware in response to comparing the detected network address to the set of configuration parameters.

20

13. A computer-implemented method, comprising:

accessing a set of identifiers of files known not to be malware;

accessing a set of identifiers of files known to be malware;

generating at least one hash of at least one file associated with a first process running on a host of a plurality of hosts;

25

determining that the at least one hash is not in the set of identifiers of files known not to be malware and that the at least one hash is not in the set of identifiers of files known to be malware;

monitoring the first process in response to determining that the at least one hash is not in the set of identifiers of files known not to be malware and that the at least one hash is not in the set of identifiers of files known to be malware;

30

sending the at least one file, information related to the at least one file, and information related to the first process to be analyzed for malware in response to comparing a system call associated with the first process to a set of rules.

5

14. The method of Claim 13, further comprising:

determining, according to a first policy, a plurality of malware detection schemes to apply to the at least one file based on a file type of the at least one file;

scheduling the application of the determined plurality of malware detection schemes to the at least one file amongst a plurality of detection nodes according to a

10

second policy; and

in response to determining the results of applying the plurality of malware detection schemes, determining that the at least one file is suspected malware or determining that the at least one file is malware according to a third policy.

15

15. The method of Claim 13, further comprising:

detecting a network address associated with a second process of the host in random access memory;

comparing the detected network address to a set of configuration parameters;

and

20

sending a second file associated with the second process, information related to the second file, and information related to the second process to be analyzed for malware in response to comparing the detected network address to the set of configuration parameters.

25

16. The method of Claim 13, wherein the at least one file associated with the first process comprises at least one library module associated with the first process.

30

17. The method of Claim 13, wherein sending the at least one file and information related to the at least one file comprises storing the at least one file and information related to the at least one file in a computer-readable storage medium attached to the host.

18. The method of Claim 13, wherein sending the at least one file and information related to the at least one file comprises sending the at least one file to a node remote from the host.

5

19. The method of Claim 13, wherein information related to the at least one file comprises metadata related to the at least one file.

20. The method of Claim 13, wherein:

10 identifiers of files known to be malware comprise hashes of files known to be malware; and

identifiers of files known not to be malware comprise hashes of files known not to be malware.

15 21. The method of Claim 13, further comprising:

requesting a user of the host to enter information regarding the at least one file in response to determining that the at least one hash is not in the set of identifiers of files known not to be malware and that the at least one hash is not in the set of identifiers of files known to be malware; and

20 wherein sending the at least one file, information related to the at least one file, and information related to the first process to be analyzed for malware comprises sending information entered by the user.

22. A system for malware detection, comprising:

25 at least one computer-readable, non-transitory storage medium comprising instructions that, when executed by at least one processor, are operable to:

access a set of configuration parameters;

access a set of identifiers of files known not to be malware;

access a set of identifiers of files known to be malware;

30 determine that a first file in a first host of a plurality of hosts should be analyzed by comparing the first file to the set of configuration parameters;

generate a first hash of the first file in response to determining that a first file should be analyzed;

determine that the first hash is not in the set of identifiers of files known not to be malware and that the first hash is not in the set of identifiers of files known to be malware;

determine a statistical variance with respect to the first file in response to determining that the first hash is not in the set of identifiers of files known not to be malware and is not in the set of identifiers of files known to be malware;

send the first file and information related to the first file to be analyzed for malware in response to determining the statistical variance; and

delete the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known to be malware after sending the first file.

23. The system of Claim 22, wherein:

the set of configuration parameters comprises a set of file types; and

comparing the first file to the set of configuration parameters comprises determining that a file type associated with the first file corresponds to a file type of the set of file types.

20

24. The system of Claim 22, wherein determining the statistical variance comprises determining that the first file is not present in a set of hosts of the plurality of hosts.

25. The system of Claim 22, wherein deleting the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known to be malware further comprises overwriting a set of memory blocks that store the set of configuration parameters, the set of identifiers of files known not to be malware, and the set of identifiers of files known to be malware.

30

26. The system of Claim 22, wherein the at least one computer-readable, non-transitory storage medium further comprises instructions that, when executed by at least one processor, are operable to:

determine, according to a first policy, a plurality of malware detection
5 schemes to apply to the first file based on a file type of the first file;

schedule the application of the determined plurality of malware detection schemes to the first file amongst a plurality of detection nodes according to a second policy; and

in response to determining the results of applying the plurality of malware
10 detection schemes, determine that the first file is suspected malware or determining that the at least one file is malware according to a third policy.

27. The system of Claim 22, wherein sending the first file and information related to the first file comprises storing the first file and information related to the
15 first file in a computer-readable storage medium attached to the first host.

28. The system of Claim 22, wherein sending the first file and information related to the first file comprises sending the first file to a node remote from the first
20 host.

29. The system of Claim 22, wherein information related to the first file comprises metadata related to the first file.

30. The system of Claim 22, wherein:
25 identifiers of files known to be malware comprise hashes of files known to be malware; and

identifiers of files known not to be malware comprise hashes of files known not to be malware.

31. The system of Claim 22, wherein determining that the first file should
30 be analyzed comprises determining that the first file should be analyzed in response to detecting that the first file has entered the first host through a network connection.

32. The system of Claim 22, wherein determining that the first file should be analyzed comprises determining that the first file should be analyzed in response to detecting that the first file has entered the first host through a removable, computer-
5 readable, storage medium.

33. The system of Claim 22, wherein the at least one computer-readable, non-transitory storage medium further comprises instructions that, when executed by at least one processor, are operable to:

10 detect a network address associated with a process of the first host in random access memory;

compare the detected network address to the set of configuration parameters;

and

15 send a second file associated with the process, information related to the second file, and information related to the process to be analyzed for malware in response to comparing the detected network address to the set of configuration parameters.

34. A system for malware detection, comprising:

20 at least one computer-readable, non-transitory storage medium comprising instructions that, when executed by at least one processor, are operable to:

access a set of identifiers of files known not to be malware;

access a set of identifiers of files known to be malware;

25 generate at least one hash of at least one file associated with a first process running on a host of a plurality of hosts;

determine that the at least one hash is not in the set of identifiers of files known not to be malware and that the at least one hash is not in the set of identifiers of files known to be malware;

30 monitor the first process in response to determining that the at least one hash is not in the set of identifiers of files known not to be malware and that the at least one hash is not in the set of identifiers of files known to be malware;

send the at least one file, information related to the at least one file, and information related to the first process to be analyzed for malware in response to comparing a system call associated with the first process to a set of rules.

5 35. The system of Claim 34, wherein the at least one computer-readable, non-transitory storage medium further comprises instructions that, when executed by at least one processor, are operable to:

 determine, according to a first policy, a plurality of malware detection schemes to apply to the at least one file based on a file type of the at least one file;

10 schedule the application of the determined plurality of malware detection schemes to the at least one file amongst a plurality of detection nodes according to a second policy; and

 in response to determining the results of applying the plurality of malware detection schemes, determine that the at least one file is suspected malware or
15 determining that the at least one file is malware according to a third policy.

 36. The system of Claim 34, wherein the at least one computer-readable, non-transitory storage medium further comprises instructions that, when executed by at least one processor, are operable to:

20 detect a network address associated with a second process of the host in random access memory;

 compare the detected network address to a set of configuration parameters; and

 send a second file associated with the second process, information related to
25 the second file, and information related to the second process to be analyzed for malware in response to comparing the detected network address to the set of configuration parameters.

 37. The system of Claim 34, wherein the at least one file associated with
30 the first process comprises at least one library module associated with the first process.

38. The system of Claim 34, wherein sending the at least one file and information related to the at least one file comprises storing the at least one file and information related to the at least one file in a computer-readable storage medium attached to the host.

5

39. The system of Claim 34, wherein sending the at least one file and information related to the at least one file comprises sending the at least one file to a node remote from the host.

10

40. The system of Claim 34, wherein information related to the at least one file comprises metadata related to the at least one file.

41. The system of Claim 34, wherein:
identifiers of files known to be malware comprise hashes of files known to be
malware; and
identifiers of files known not to be malware comprise hashes of files known
not to be malware.

15

42. The system of Claim 34, wherein the at least one computer-readable,
non-transitory storage medium further comprises instructions that, when executed by
at least one processor, are operable to:

20

request a user of the host to enter information regarding the at least one file in
response to determining that the at least one hash is not in the set of identifiers of files
known not to be malware and that the at least one hash is not in the set of identifiers
of files known to be malware; and

25

wherein sending the at least one file, information related to the at least one
file, and information related to the first process to be analyzed for malware comprises
sending information entered by the user.

1/8

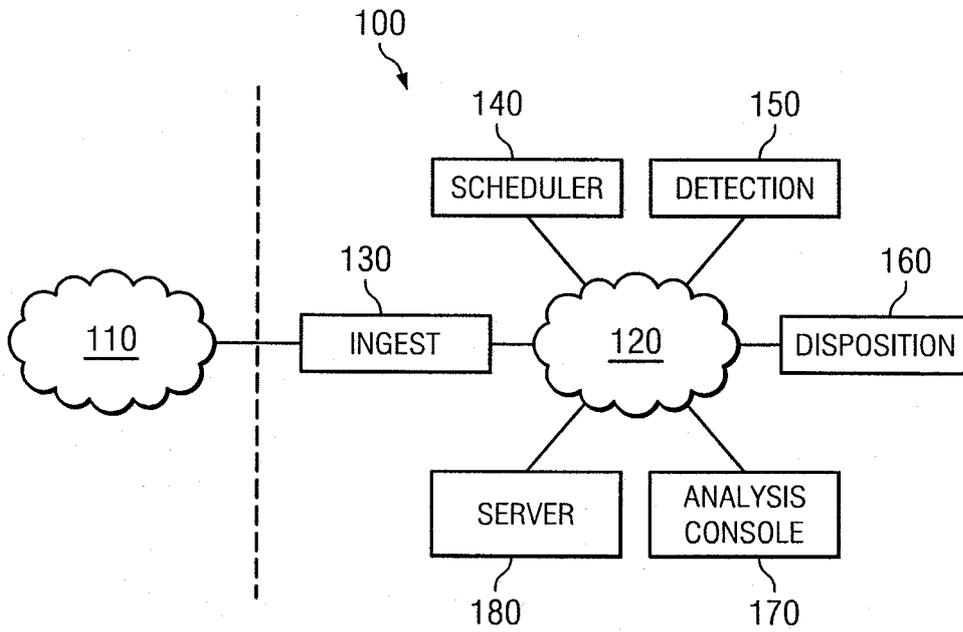


FIG. 1

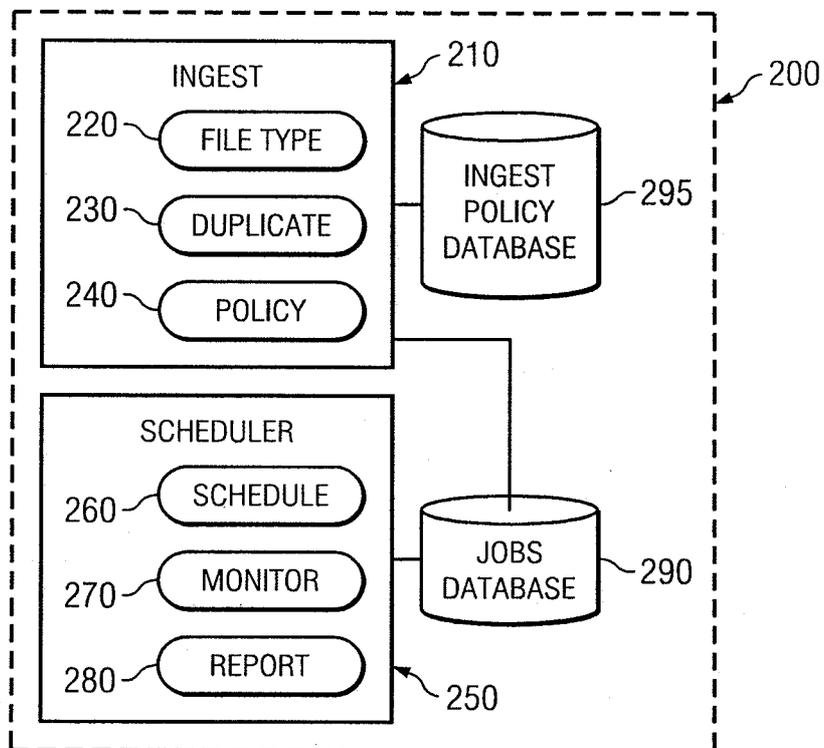


FIG. 2

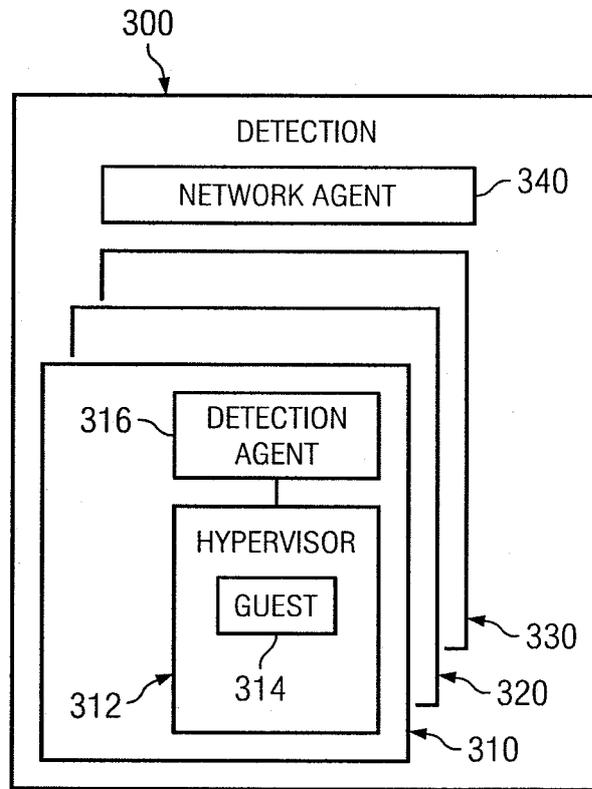


FIG. 3

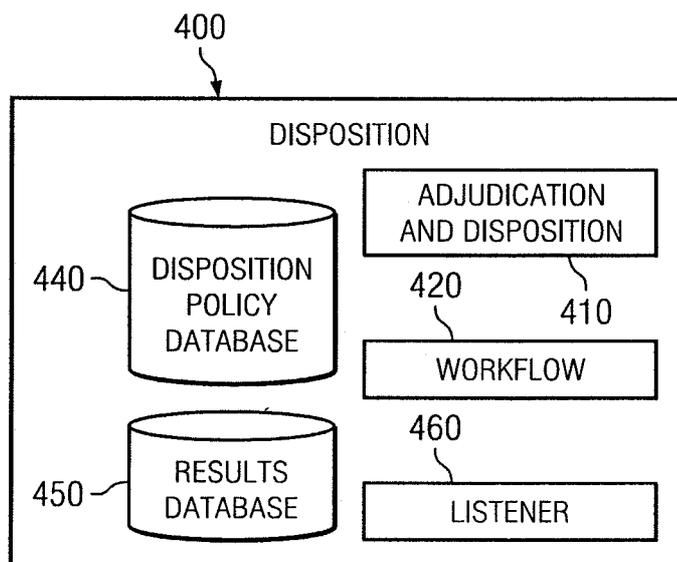


FIG. 4

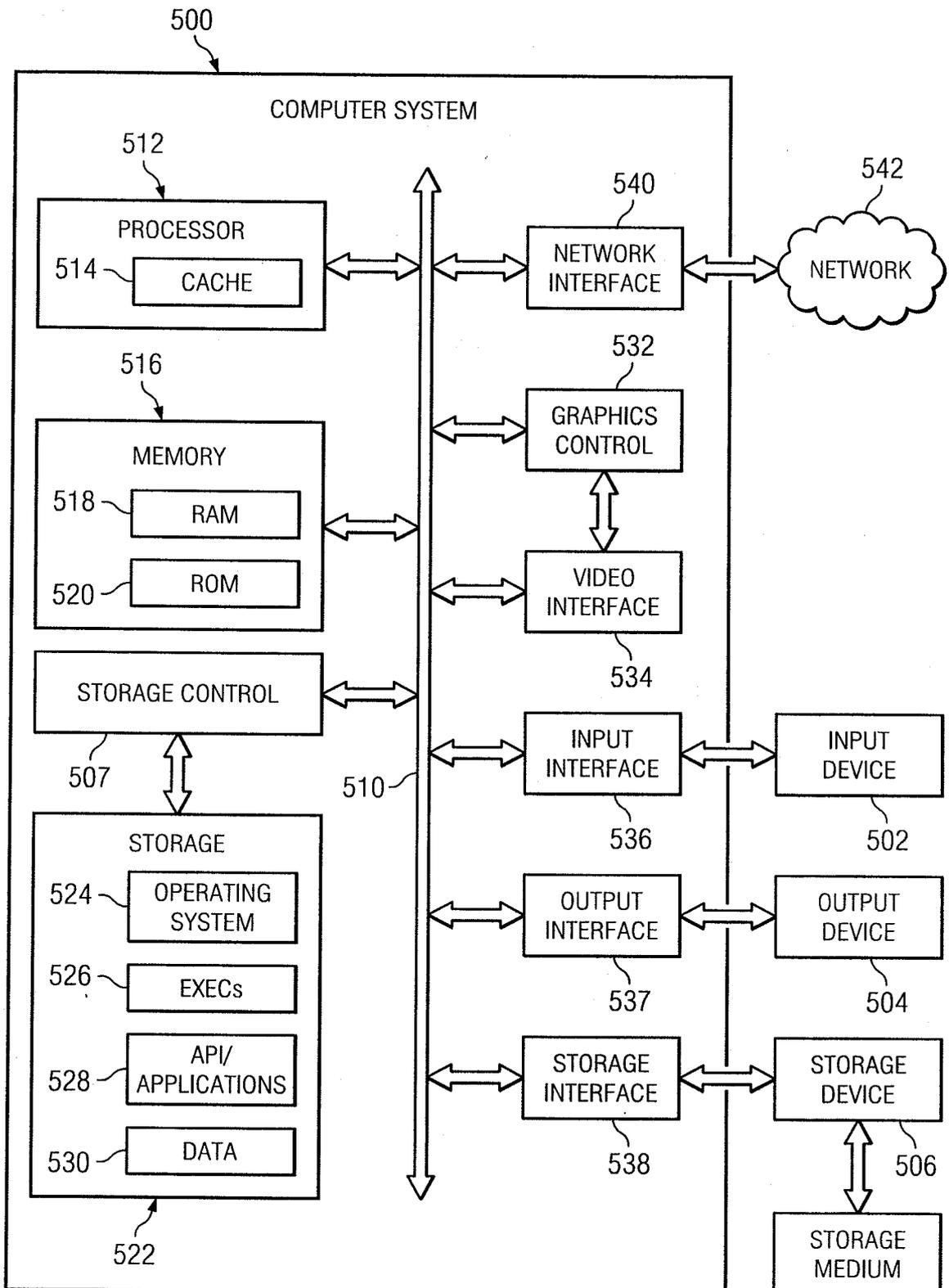


FIG. 5

508

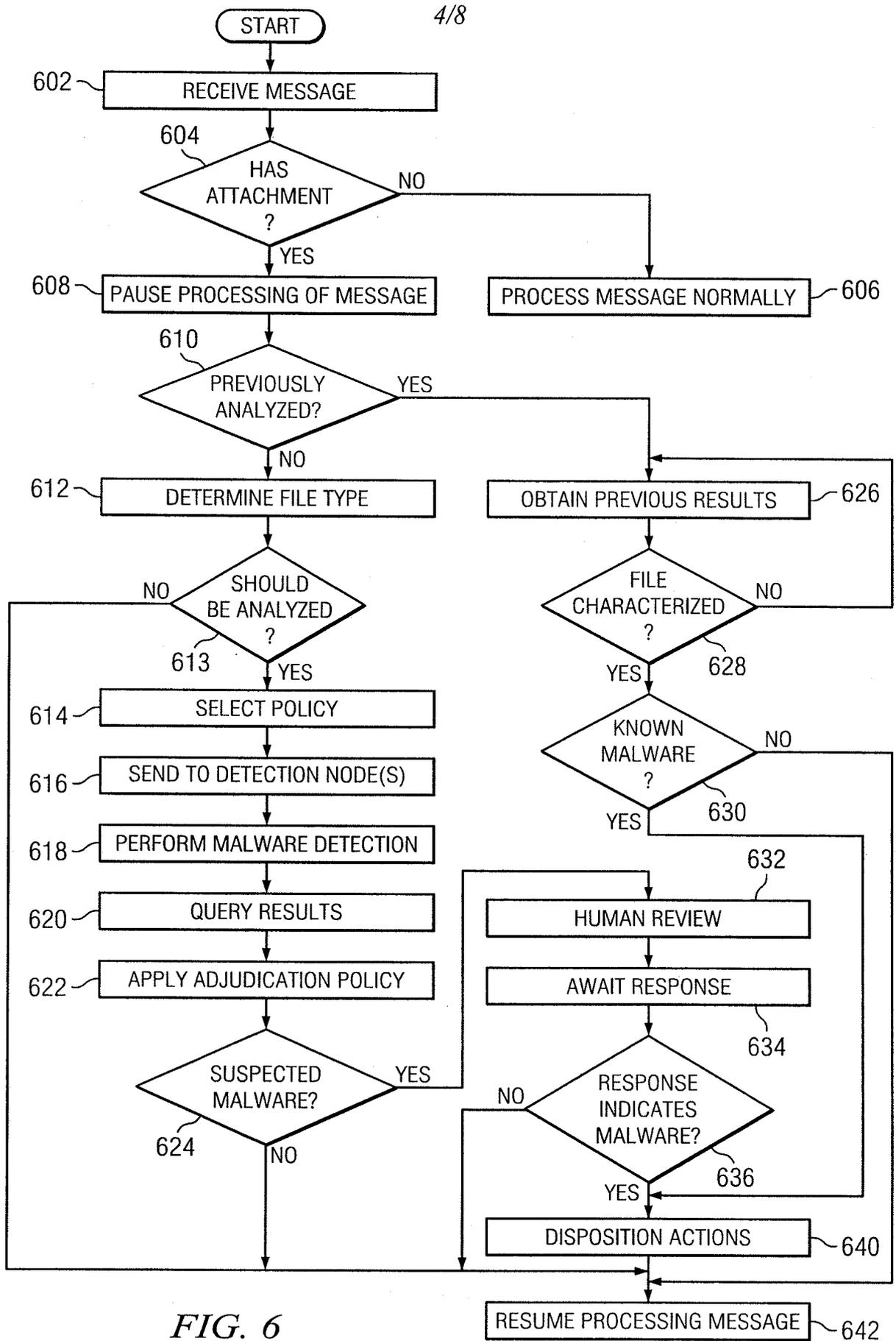


FIG. 6

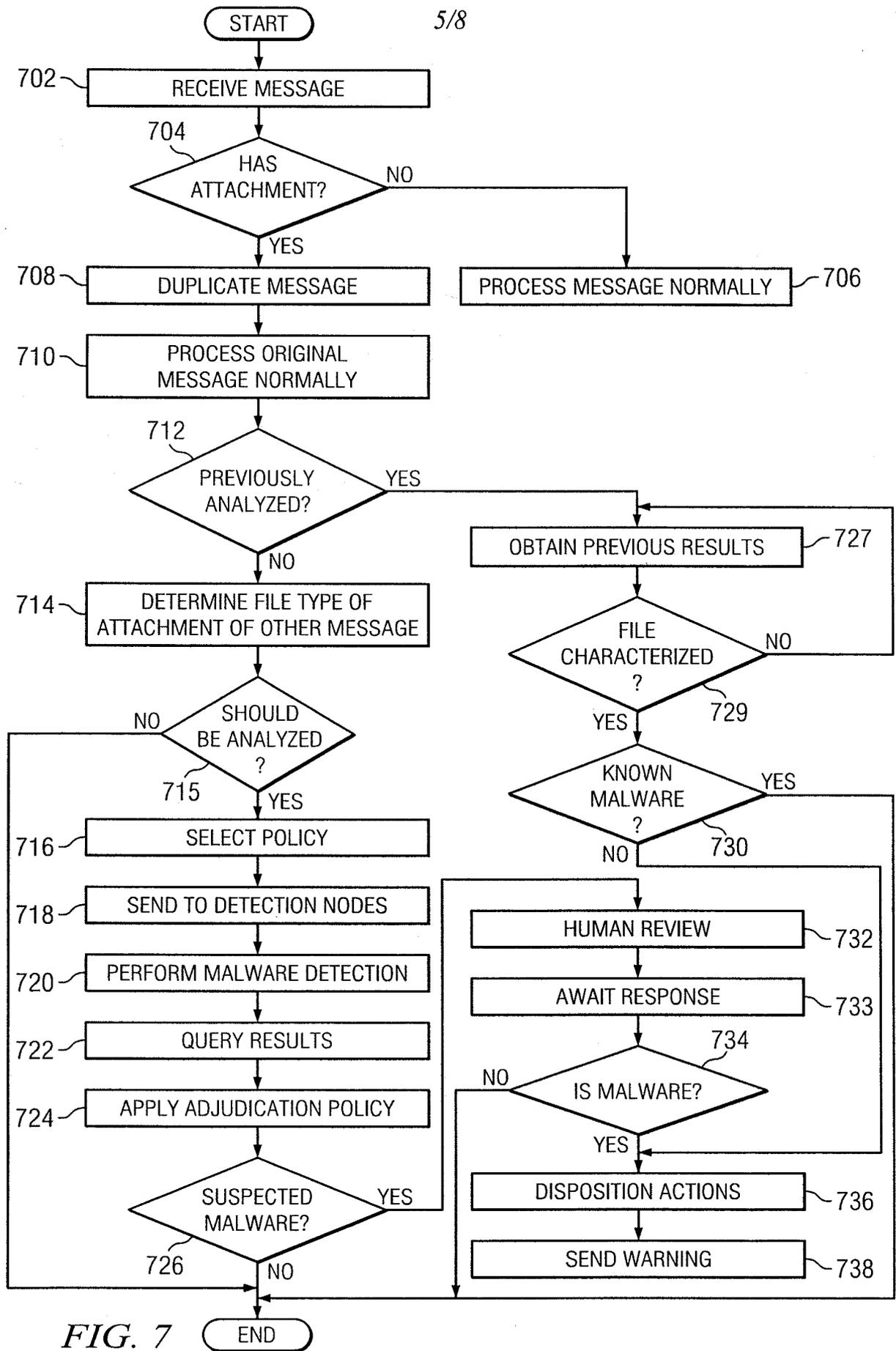


FIG. 7

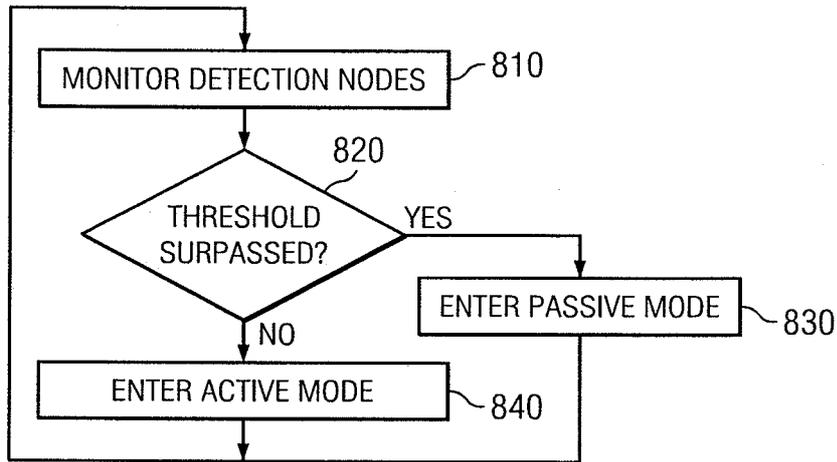


FIG. 8

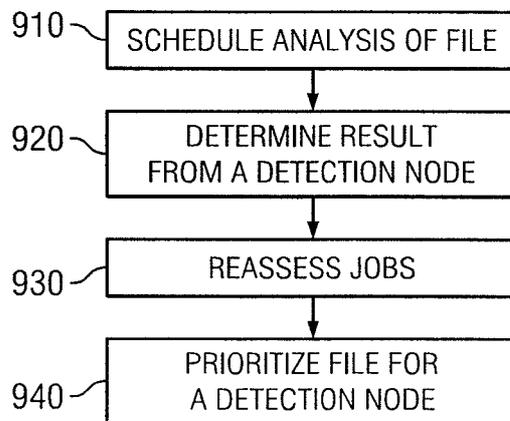


FIG. 9

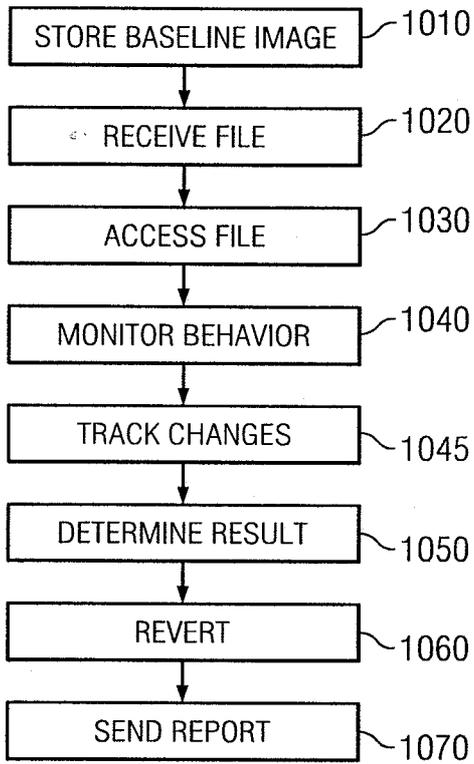


FIG. 10

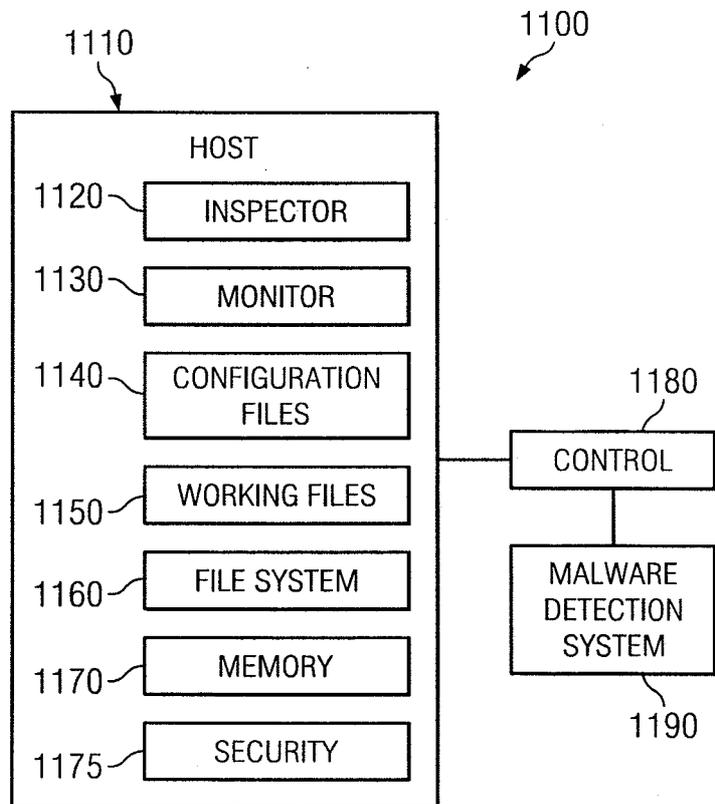


FIG. 11

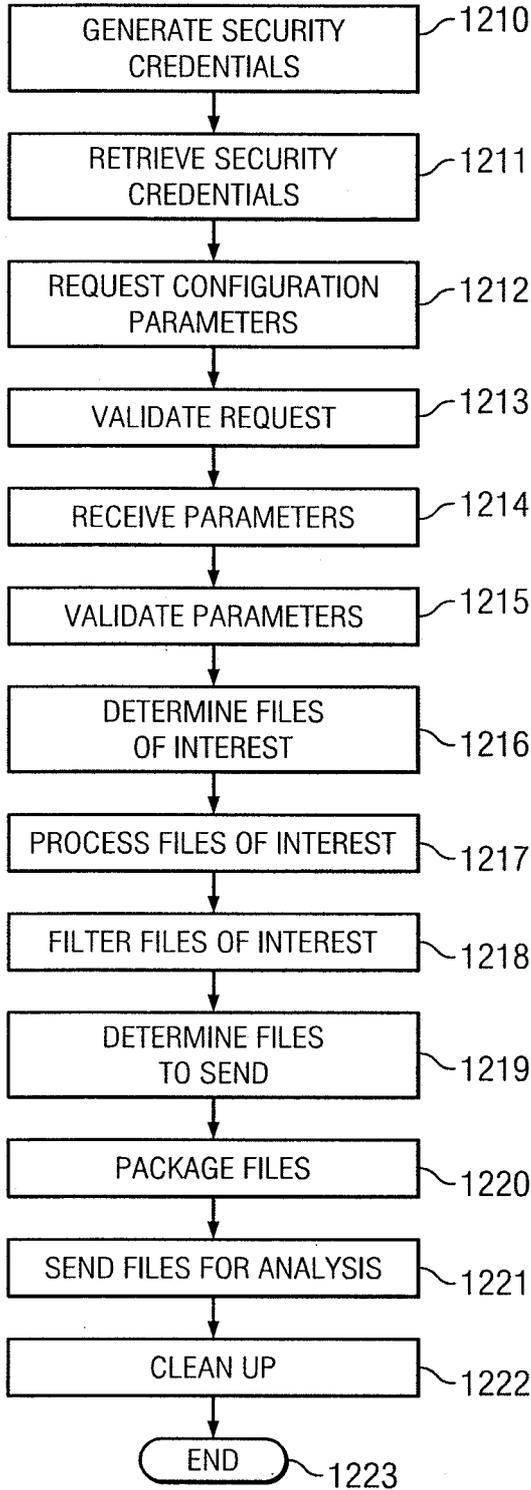


FIG. 12

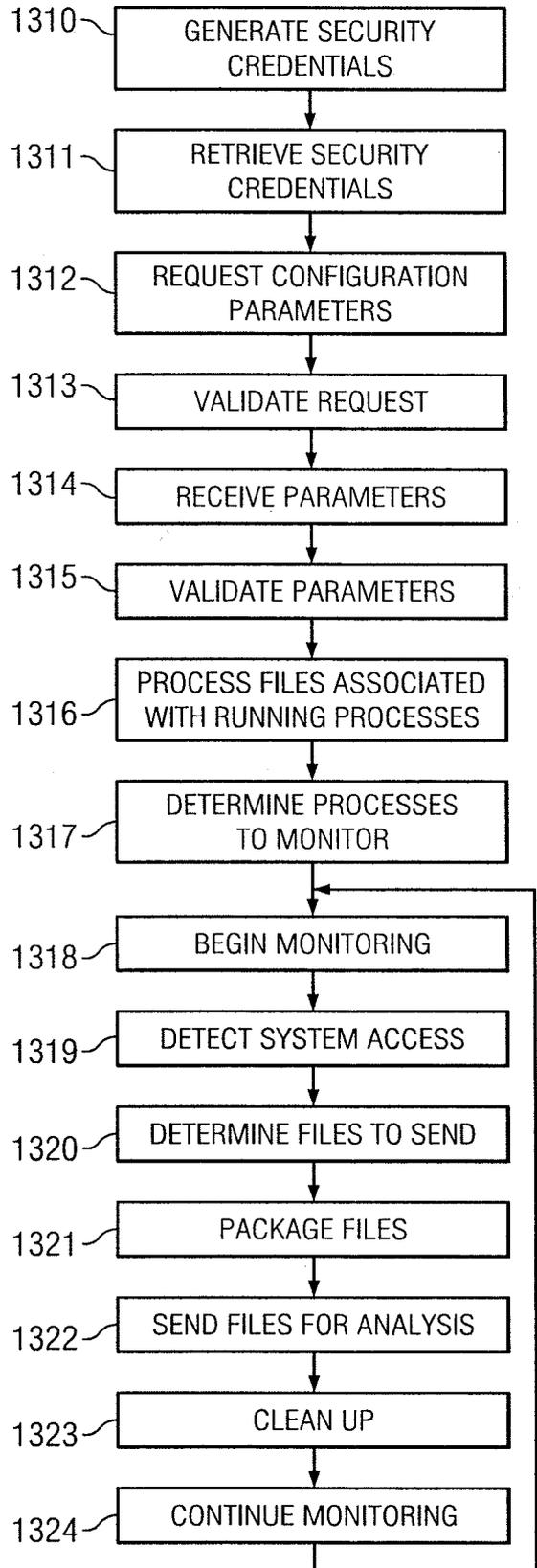


FIG. 13

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2011/025683

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/00
ADD.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6 092 194 A (TOUBOUL SHLOMO [I L]) 18 July 2000 (2000-07-18) * abstract	1-42
Y	US 7 392 544 B1 (PAVLYUSHCHIK MI KHAI L A [RU]) 24 June 2008 (2008-06-24) * abstract	1-42
Y	US 2005/021994 A1 (BARTON CHRISTOPHER ANDREW [GB] ET AL) 27 January 2005 (2005-01-27) * abstract	1-42
Y	EP 1 657 662 A2 (MICROSOFT CORP [US]) 17 May 2006 (2006-05-17) * abstract	1-42
	----- -/- .	

Further documents are listed in the continuation of Box C.

See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 29 March 2011	Date of mailing of the international search report 08/04/2011
--	--

Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Authorized officer Mezbdı , Stephan
--	--

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2011/025683

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 7 640 589 B1 (MASHEVSKY YURI V [RU] ET AL) 29 December 2009 (2009-12-29) * abstract -----	1-42
Y	W0 2005/114414 A1 (COMPUTER ASS THINK INC [US] ; GASSOWAY PAUL A [US]) 1 December 2005 (2005-12-01) * abstract -----	1-42
Y	US 2006/015940 A1 (ZAMIR SHAY [IL] ET AL) 19 January 2006 (2006-01-19) * abstract -----	1-42
Y	US 2003/120947 A1 (MOORE ROBERT EDWARD [GB] ET AL) 26 June 2003 (2003-06-26) * abstract -----	1-12
Y	W0 2008/091785 A2 (MANDIANT INC [US]; MCMILLAN CHAD [US]; GARMAN JASON [US]) 31 July 2008 (2008-07-31) * abstract -----	1-42
Y	US 5 907 834 A (KEPHART JEFFREY OWEN [US] ET AL) 25 May 1999 (1999-05-25) * abstract -----	1-42
Y	US 2006/015630 A1 (STOLFO SALVATORE J [US] ET AL) 19 January 2006 (2006-01-19) * abstract -----	1-42

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No PCT/US2011/025683
--

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6092194	A	18-07-2000	CA 2275771 AI 22-05-1998
			EP 0965094 A2 22-12-1999
			JP 3952315 B2 01-08-2007
			JP 2002514326 T 14-05-2002
			US 6804780 BI 12-10-2004

US 7392544	BI	24-06 -2008	US 7725941 BI 25-05 -2010

US 2005021994	AI	27-01 -2005	NONE

EP 1657662	A2	17-05 -2006	CN 1766778 A 03-05 -2006
			JP 2006127497 A 18-05 -2006
			KR 20060051379 A 19-05 -2006
			US 2006230452 AI 12-10 -2006
			US 2006095971 AI 04-05 -2006

US 7640589	BI	29-12 -2009	CN 101901314 A 01-12 -2010
			EP 2278516 AI 26-01 -2011

Wo 2005114414	AI	01-12 -2005	EP 1743228 AI 17-01 -2007

US 2006015940	AI	19-01 -2006	EP 1782198 A2 09-05 -2007
			wo 2006006144 A2 19-01 -2006

US 2003120947	AI	26-06 -2003	US 7383581 BI 03-06 -2008

Wo 2008091785	A2	31-07 -2008	US 2008184367 AI 31-07 -2008

US 5907834	A	25-05 -1999	US 5675711 A 07-10 -1997

US 2006015630	AI	19-01 -2006	NONE
