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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 February 2012 (02.02.2012)

PCT

(10) International Publication Number
WO 2012/015489 A1

(51) International Patent Classification:
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/US2011/024869

(22) International Filing Date:
15 February 2011 (15.02.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/844,964 28 July 2010 (28.07.2010) US

(71) Applicant *(for all designated States except US)*: **MCAFEE, INC.** [US/US]; 5000 Headquarters Drive, Plano, Texas 75024 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BHARGAVA, Rishi** [IN/US]; 19973 Pear Tree Ct., Cupertino, California 95014 (US). **REESE JR., David P.** [US/US]; 212 Red Oak Dr. E., Apt. G, Sunnyvale, California 94086 (US).

(74) Agent: **FRAME, Thomas J.**; Patent Capital Group, 6119 McCommas Blvd., Dallas, Texas 75214 (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, 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(ii))*

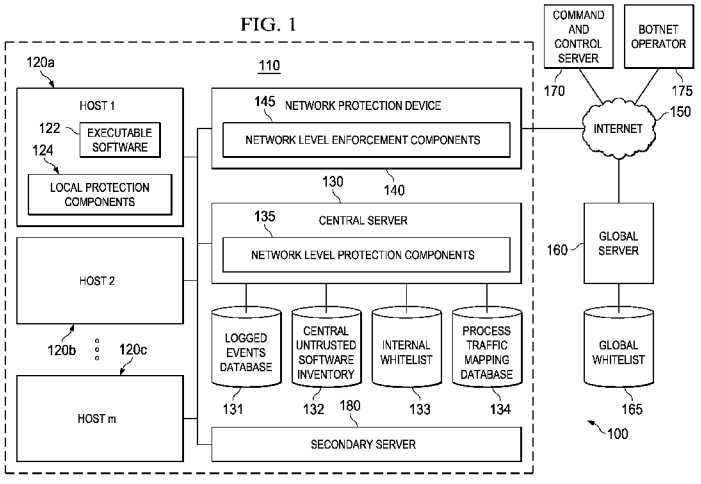
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

(54) Title: SYSTEM AND METHOD FOR NETWORK LEVEL PROTECTION AGAINST MALICIOUS SOFTWARE

FIG. 1



(57) Abstract: A method in one example implementation includes receiving information related to a network access attempt on a first computing device with the information identifying a software program file associated with the network access attempt. The method also includes evaluating a first criterion to determine whether network traffic associated with the software program file is permitted and then creating a restriction rule to block the network traffic if the network traffic is not permitted. The first criterion includes a trust status of the software program file. In specific embodiments, the method includes pushing the restriction rule to a network protection device that intercepts the network traffic associated with the software program file and applies the restriction rule to the network traffic. In more specific embodiments, the method includes searching a whitelist identifying trustworthy software program files to determine the trust status of the software program file.

WO 2012/015489 A1

SYSTEM AND METHOD FOR NETWORK LEVEL PROTECTION AGAINST MALICIOUS SOFTWARE

RELATED U.S. APPLICATION INFORMATION

[0001] This application is related to co-pending U.S. Patent Application Serial No. 12/844,892, filed July 28, 2010, entitled "SYSTEM AND METHOD FOR LOCAL PROTECTION AGAINST MALICIOUS SOFTWARE," Inventors Rishi Bhargava, et al. (Attorney Docket No. 04796.1052). The disclosure of this application is considered part of and is incorporated by reference herein in its entirety.

TECHNICAL FIELD

[0002] This disclosure relates in general to the field of network security and, more particularly, to network level protection against malicious software.

BACKGROUND

[0003] The field of network security has become increasingly important in today's society. The Internet has enabled interconnection of different computer networks all over the world. The ability to effectively protect and maintain stable computers and systems, however, presents a significant obstacle for component manufacturers, system designers, and network operators. This obstacle is made even more complicated due to the continually-evolving array of tactics exploited by malicious operators. Of particular concern more recently are botnets, which may be used for a wide variety of malicious purposes. Once a malicious software program file (*e.g.*, a bot) has infected a host computer, a malicious operator may issue commands from a "command and control server" to control the bot. Bots can be instructed to

perform any number of malicious actions such as, for example, sending out spam or malicious emails from the host computer, stealing sensitive information from a business or individual associated with the host computer, propagating the botnet to other host computers, and/or assisting with distributed denial of service attacks. In addition, the malicious operator can sell or otherwise give access to the botnets to other malicious operators through the command and control servers, thereby escalating the exploitation of the host computers. Consequently, botnets provide a powerful way for malicious operators to access other computers and to manipulate those computers for any number of malicious purposes. Security professionals need to develop innovative tools to combat such tactics that allow malicious operators to exploit computers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] To provide a more complete understanding of the present disclosure and features and advantages thereof, reference is made to the following description, taken in conjunction with the accompanying figures, wherein like reference numerals represent like parts, in which:

[0005] FIGURE 1 is a pictorial representation of an exemplary network environment in which embodiments of a system for network level protection against malicious software may be implemented in accordance with the present disclosure;

[0006] FIGURE 2 is a block diagram of an example server in which components of the system may be implemented in accordance with embodiments of the present disclosure;

[0007] FIGURE 3 is a schematic diagram of an example computing device in which components of the system may be implemented in accordance with embodiments of the present disclosure;

[0008] FIGURE 4 is a block diagram of an example network protection device in the network environment in which the system may be implemented in accordance with embodiments of the present disclosure;

[0009] FIGURE 5 is a simplified flowchart illustrating a series of example steps associated with embodiments of the system in accordance with the present disclosure;

[0010] FIGURE 6 is a simplified flowchart illustrating a series of example steps of a trust determination flow associated with FIGURE 5 in accordance with embodiments of the present disclosure;

[0011] FIGURE 7 is a simplified flowchart illustrating a series of example steps of another embodiment of a trust determination flow associated with FIGURE 5 in accordance with embodiments of the present disclosure;

[0012] FIGURE 8 is a simplified flowchart illustrating a series of example steps associated with other embodiments of the system in accordance with the present disclosure;

[0013] FIGURE 9 is a simplified flowchart illustrating an example step of a trust determination flow associated with FIGURE 8 in accordance with embodiments of the present disclosure;

[0014] FIGURE 10 is a simplified flowchart illustrating a series of example steps of another trust determination flow associated with FIGURE 8 in accordance with embodiments of the present disclosure;

[0015] FIGURE 11 is a simplified flowchart illustrating another series of example steps associated with the system in accordance with embodiments of the present disclosure; and

[0016] FIGURE 12 is a simplified flowchart illustrating yet another series of example steps associated with the system in accordance with embodiments of the present disclosure.

DETAILED DESCRIPTION OF EXAMPLE EMBODIMENTS

OVERVIEW

[0017] A method in one example implementation includes receiving information related to a network access attempt on a first computing device with the information identifying a software program file associated with the network access attempt. The method

further includes evaluating a first criterion to determine whether network traffic associated with the software program file is permitted and creating a restriction rule to block the network traffic if the network traffic is not permitted. The first criterion includes a trust status of the software program file. In specific embodiments, the method includes pushing the restriction rule to a network protection device that intercepts the network traffic associated with the software program file and that applies the restriction rule to the network traffic. In more specific embodiments, the method includes searching a whitelist identifying trustworthy software program files to determine the trust status of the software program file, where the trust status of the software program file is defined as untrusted if the software program file is not included in the whitelist and the network traffic is not permitted if the trust status of the software program file is defined as untrusted. In yet other specific embodiments, event data related to the network traffic may be logged if the trust status of the software program file associated with the network traffic is defined as untrusted, and such logging may occur instead of blocking the network traffic or may occur in addition to blocking the network traffic.

EXAMPLE EMBODIMENTS

[0018] FIGURE 1 is a pictorial representation of an exemplary network environment 100 in which embodiments of a system for network level protection against malicious software may be implemented. Network environment 100 may include a local network 110 with electronic connection to a wide area network (WAN) such as Internet 150. The Internet 150 provides access to many other networks, computing devices, and web services. For example, a global server 160 may provide a database 165 containing global whitelists indicating software program files that have been evaluated and determined to be free of malicious code. In addition, malicious users such as a botnet operator 175 may also have access to the Internet 150 along with a command and control server 170, which may be manipulated by botnet operator 175 to send out and subsequently control malicious software (*e.g.*, a bot) that attempts to infect networks, such as local network 110. Local network 110 may include hosts

120a, 120b, and 120c operably connected to a central server 130, a secondary server 180, and a network protection device 140. Host 120a may include executable software 122 and local protection components 124. For ease of reference, only host 120a is shown with such components, however, it will be apparent that any other hosts within local network 110 could also be configured with similar components as shown in host 120a of FIGURE 1. Central protection components 135 may be provided in central server 130, and network level enforcement components 145 may be provided in network protection device 140. Central server 130 may also have access to a logged events database 131, a central untrusted software inventory 132, an internal whitelist 133, and a process traffic mapping database 134.

[0019] In example embodiments, local protection components 124 on host 120a, central protection components 135 on central server 130, and network level enforcement components 145 on network protection device 140 may cooperate to provide a system for network level protection against network traffic associated with malicious software. Network traffic, as used herein in this Specification, is intended to mean data in a network such as, for example, electronic packets being sent from a host to any network or other computer (*i.e.*, outbound network traffic), and electronic packets being sent to the host from any network or other computer (*i.e.*, inbound network traffic). Network traffic may be blocked or otherwise restricted by network protection device 140 if network protection device 140 includes an applicable restriction rule associated with an untrusted program file.

[0020] In accordance with embodiments of this disclosure, network protection device 140 may receive restriction rules created for network traffic when untrusted program files are discovered. A trust status (*i.e.*, trusted or untrusted) of program files may be determined in a batch mode or may be determined in real-time during a network access attempt. A network access attempt as used herein in this Specification is intended to include any inbound or outbound network access attempt on a host (*e.g.*, accepting a connection request, making a connection request, receiving electronic data from a network, sending electronic data to a network). In both batch mode and real-time mode, program files are evaluated to determine whether the trust status of each program file is defined as trusted or untrusted, using one or

more trust evaluation techniques (*e.g.*, whitelist comparisons, program file change comparisons, blacklist comparisons, etc.). Policies may also be used when creating restriction rules. Such policies may include, for example, only allowing access to a specified subnet of network addresses, blocking all inbound and outbound network traffic, blocking only inbound or outbound network traffic, blocking all local network traffic and allowing Internet traffic, and the like. Any network traffic associated with untrusted program files may also be logged and aggregated for reporting.

[0021] For purposes of illustrating the techniques of the system for network level protection against malicious software, it is important to understand the activities occurring within a given network. The following foundational information may be viewed as a basis from which the present disclosure may be properly explained. Such information is offered earnestly for purposes of explanation only and, accordingly, should not be construed in any way to limit the broad scope of the present disclosure and its potential applications. In addition, it will be appreciated that the broad scope of this disclosure intends for references to "program file", "software program file", and "executable software" to encompass any software file comprising instructions that can be understood and processed on a computer such as, for example, executable files, library modules, object files, other executable modules, script files, interpreter files, and the like.

[0022] Typical network environments used in organizations and by individuals include the ability to communicate electronically with other networks using, for example, the Internet to access web pages hosted on servers connected to the Internet, to send or receive electronic mail (*i.e.*, email) messages, or to exchange files with end users or servers connected to the Internet. Malicious users are continuously developing new tactics using the Internet to spread malware and to gain access to confidential information.

[0023] Tactics that represent an increasing threat to computer security often include botnets. Botnets use a client-server architecture where a type of malicious software (*i.e.*, a bot) is placed on a host computer and communicates with a command and control server, which may be controlled by a malicious user (*e.g.*, a botnet operator). The bot may receive

commands from the command and control server to perform particular malicious activities and, accordingly, may execute such commands. The bot may also send any results or pilfered information back to the command and control server. In addition to receiving commands to perform malicious activities, bots also typically include one or more propagation vectors that enable it to spread within an organization's network or across other networks to other organizations or individuals. Common propagation vectors include exploiting known vulnerabilities on hosts within the local network and sending malicious emails having a malicious program attached or providing malicious links within the emails. Bots may also infect host computers through, for example, drive-by downloads, viruses, worms, Trojan horses, etc.

[0024] Botnets provide a powerful way for botnet operators to compromise computer systems by employing a variety of attacks. Once a bot has infected a host computer, the command and control server can issue commands to the bot to perform various types of attacks. Commonly, botnets have been used to send bulk email and to perform distributed denial of service attacks. More recently, however, botnets have been used to perform more targeted attacks against businesses and individuals to obtain confidential data or other sensitive information such as intellectual property and financial data.

[0025] Existing firewall and network intrusion prevention technologies are generally deficient in recognizing and containing botnets. Bots are often designed to initiate communication with the command and control server and to masquerade as normal web browser traffic. Bots may be crafted with a command and control protocol that makes the bot appear to be making normal network connections to a web server. For example, a bot may use a port typically used to communicate with a web server. Such bots, therefore, may not be detected by existing technologies without performing more detailed packet inspection of the web traffic. Moreover, once a bot is discovered, the botnet operator may simply find another way to masquerade network traffic by the bot to continue to present as normal web traffic. More recently, botnet operators have crafted bots to use encryption protocols such as, for example, secure socket layer (SSL), thereby encrypting malicious network traffic. Such encrypted traffic may use a Hypertext Transfer Protocol Secure (HTTPS) port such that only the

endpoints involved in the encrypted session can decrypt the data. Thus, existing firewalls and other network intrusion prevention technologies are unable to perform any meaningful inspection of the web traffic. Consequently, bots continue to infect host computers within networks.

[0026] Other software security technology focused on preventing unauthorized program files from executing on a host computer may have undesirable side effects for end users or employees of a business or other organizational entity. Network or Information Technology (IT) administrators may be charged with crafting extensive policies relevant to all facets of the business entity to enable employees to obtain software and other electronic data from desirable and trusted network resources. Without extensive policies in place, employees may be prevented from downloading software and other electronic data from network resources that are not specifically authorized, even if such software and other data are legitimate and necessary business activities. In addition, such systems may be so restrictive that if unauthorized software is found on a host computer, any host computer activities may be suspended pending network administrator intervention. For businesses, this type of system may interfere with legitimate and necessary business activities, resulting in worker downtime, lost revenue, significant Information Technology (IT) overhead, and the like.

[0027] A system and method for network level protection against malicious software, as outlined in FIGURE 1, can reduce the propagation and malicious activities of botnets from infected networks, while allowing legitimate activities to continue within an infected network with less IT overhead being required. In accordance with one example embodiment, a system is provided to determine which software program files associated with a network access attempt on a host, such as host 120a, are a risk (*i.e.*, untrusted). If each of the associated program files is not a risk (*i.e.*, trusted), then network traffic associated with software processes mapped to those program files may be allowed. However, if one or more of the program files is untrusted, then restriction rules may be created to enable network protection device 140 to block or otherwise restrict network traffic associated with software processes mapped to the untrusted program files, and/or to block or otherwise restrict other network traffic in

accordance with any applicable policies. Such blocking or other restriction may prevent propagation and malicious activities of a possible bot. In one example embodiment, when an untrusted program file is discovered, inbound and outbound network traffic of host 120a may be selectively permitted access to only a specified network subnet (*e.g.*, a subnet known to be safe and required for necessary business activities, etc.), based on policy considerations. Thus, the ability of a bot to respond to commands from the command and control server and to propagate may be significantly diminished, without interrupting or preventing necessary business activities. As a result, the system as implemented in FIGURE 1 provides better protection to networks with an infected host and to other networks the infected host attempts to access.

[0028] Turning to the infrastructure of FIGURE 1, local network 110 is representative of an example architecture in which the system for network level protection may be implemented. Local network 110 may be configured in a variety of forms including, but not limited to, one or more local area networks (LANs), any other suitable network or any combinations thereof. The Internet 150 is representative of a wide area network (WAN) to which local network 110 may be suitably connected. Connection between local network 110 and Internet 150 may be provided through network protection device 140, which could be a common network security device (*e.g.*, a firewall, a router, a gateway, a managed switch, etc.). In example embodiments, all network traffic to or from hosts 120 is routed through network protection device 140. An Internet Service Provider (ISP) or an Internet Server with dedicated bandwidth may provide connection to Internet 150 using any appropriate medium such as, for example, digital subscriber lines (DSL), telephone lines, T1 lines, T3 lines, wireless, satellite, fiber optics, cable, Ethernet, etc. or any combination thereof. Additional gateways, switches, routers, and the like may be used to facilitate electronic communication between hosts 120, central server 130, network protection device 140, and the Internet 150. The ISP or Internet Server may be configured to allow hosts 120 to communicate with other nodes on the Internet using Transmission Control Protocol/Internet Protocol (TCP/IP) and a mail server (not shown)

may allow hosts 120 to send and receive email messages using Simple Mail Transfer Protocol (SMTP).

[0029] In example embodiments, local network 110 represents a network environment of an organization (*e.g.*, a business, a school, a government entity, a family, etc.), with hosts 120a, 120b, and 120c representing end user computers operated by employees or other individuals associated with the organization. The end user computers may include computing devices such as desktops, laptops, mobile or handheld computing devices (*e.g.*, personal digital assistants (PDAs) or mobile phones), or any other computing device capable of executing software processes associated with network access to local network 110. Connection between hosts 120a, 120b, and 120c, central server 130, secondary server 180, network protection device 145, and any additional components in local network 110 may include any appropriate medium such as, for example, cable, Ethernet, wireless (*e.g.*, WiFi, 3G, 4G, etc.), ATM, fiber optics, etc. It should be noted that the network configurations and interconnections shown and described herein are for illustrative purposes only. FIGURE 1 is intended as an example and should not be construed to imply architectural limitations in the present disclosure.

[0030] In the example embodiment shown in FIGURE 1, command and control server 170 and botnet operator 175 are operably coupled to the Internet 150. In one example, command and control server 170 may be a web server controlled or used by botnet operator 175 to issue commands to distributed bots. In another example, command and control server 170 could be maliciously installed and hidden on a large corporate, educational, or government site. Botnet operator 175 may remotely access command and control server 170 through, for example, the Internet 150 to issue instructions for controlling distributed bots on infected host computers such as hosts 120a, 120b, or 120c. Numerous botnet operators and command and control servers controlling millions of bots may be operably connected to the Internet 150. In one example, once a bot has infected one of hosts 120a, 120b, or 120c, botnet operator 175 may begin issuing commands through command and control server 170 to propagate the bot throughout local network 110 and/or other networks. In addition, botnet operator 175 may

also issue instructions for the bot to undertake malicious activities from the infected host 120a, 120b, or 120c such as spam email, theft of confidential information, distributed denial of service attacks, etc.

[0031] FIGURE 1 also shows a global server 160 connected to Internet 150. While numerous servers may be connected to Internet 150, global server 160 represents a service providing one or more databases containing information related to software program files evaluated for risk. For example, software program files evaluated and determined to be untrustworthy (*e.g.*, containing malicious code such as viruses, worms, and the like, etc.) may be included in a so-called "blacklist". Software program files evaluated and determined to be trustworthy (*e.g.*, uncontaminated, free of malicious code, etc.) may be included in a so-called "whitelist". Although whitelists and blacklists may be implemented separately, it is also possible for them to be combined in a database with each software program file being identified as either a whitelist file or a blacklist file.

[0032] Whitelists and blacklists may be implemented using checksums where a unique checksum for each program file is stored, which can be readily compared to a computed checksum of a program file sought to be evaluated. A checksum can be a mathematical value or hash sum (*e.g.*, a fixed string of numerical digits) derived by applying an algorithm to a software program file. If the algorithm is applied to a second software program file that is identical to the first software program file, then the checksums should match. However, if the second software program file is different (*e.g.*, it has been altered in some way, it is a different version of the first software program file, it is a wholly different type of software, etc.) then the checksums are very unlikely to match.

[0033] Databases such as global whitelist 165 in FIGURE 1 may be provided by independent third parties and may be regularly updated to provide a comprehensive listing of trustworthy software program files available to consumers. Similarly, blacklists (not shown) may be provided by independent third parties and may be regularly updated to provide a comprehensive listing of untrusted, malicious software program files. Global whitelists and blacklists may be external to local network 110 and may be accessible through other networks

such as Internet 150, or through any other suitable connection that permits electronic communication between local network 110 and global whitelist 165. Examples of such global whitelists and blacklists include Artemis databases provided by McAfee, Inc. of Santa Clara, California, and SignaCert® databases provided by SignaCert, Inc. of Portland, Oregon.

[0034] FIGURE 1 also includes an internal whitelist 133 shown in local network 110. Internal whitelist 133 may also contain information related to software program files evaluated for risk and may identify such software program files using checksums. Software program files identified in internal whitelist 133 may be inclusive of software program files from one or more global whitelists and/or may be customized to provide selected software program files. In particular, software program files developed internally within the organization, but not necessarily available to the general public, may be identified in internal whitelist 133. Additionally, an internal blacklist could also be provided to identify particular software program files evaluated and determined to be untrustworthy.

[0035] In local network 110 shown in FIGURE 1, host 120a may be configured with executable software 122 and local protection components 124. Executable software 122 may include all software program files (*e.g.*, executable files, library modules, object files, other executable modules, script files, interpreter files, etc.) on host 120a. If host 120a has been infected by a bot, then the bot could be stored as a program file in executable software 122 of infected host 120a. Local protection components 124 of host 120a may provide different functionality depending upon the embodiment. In some embodiments of the system, local protection components 124 may regularly provide an inventory of executable software 122 or an inventory of changes and additions to executable software 122 on host 120a. In real-time embodiments of the system, local protection components 124 may intercept a network access attempt and provide information related to the network access attempt to central server 130. Local protection components 124 may also hold the network access attempt for a predetermined amount of time or until receiving a signal from central server 130 allowing the network access attempt to be released.

[0036] Central server 130 in local network 110 may include central protection components 135 for determining a trust status of software program files on host 120a, for creating restriction and logging rules for network traffic associated with untrusted software program files, for pushing restriction and logging rules to network protection device 140, and for updating logged events database 131 with entries related to network traffic associated with untrusted program files. In some embodiments central protection components may also update central untrusted software inventory 132 with entries identifying untrusted software program files. Central server 130 may also include or have access to process traffic mapping database 134, which could map software processes to software program files, including information such as program file paths, addresses (*e.g.*, Internet Protocol (IP) addresses), and/or port numbers. Logged events database 131, central untrusted software inventory 132, internal whitelist 133, and process traffic mapping database 134 may be provided in any network and device accessible to central server 130. As will be further described herein, central untrusted software inventory 132 may be omitted in some embodiments of the system. Network protection device 140 may include network level enforcement components 145 for intercepting network traffic (*e.g.*, electronic packets inbound to host 120a or outbound from host 120a, etc.) and enforcing any applicable restriction and logging rules to the intercepted packets.

[0037] Turning to FIGURE 2, FIGURE 2 shows a schematic diagram of a central server 200 and associated memory components 231, 232, 233, and 234, which is a more detailed example of central server 130 and associated memory components 131, 132, 133, and 134 shown in FIGURE 1. In the example embodiment shown in FIGURE 2, central protection components of central server 200 may include an administrative protection module 220, a policy module 230, a policy database 235, a software trust determination module 240, and a rule making module 250. In some embodiments, central protection components may also include a central trusted cache 245 for storing trusted program file cache entries from software trust determination module 240. Rule making module 250 may be configured in a batch processing mode embodiment, or in a real-time (or substantially real-time) processing

embodiment. In both batch processing and real-time processing embodiments of rule making module 250, network restriction rules and/or logging rules may be created when untrusted program files are discovered. Central protection components may also include a process traffic mapping module 260 for updating process traffic mapping database 234 with information related to a software process such as program file information, source address and port number, destination address and port number, and the like.

[0038] Central server 200 may also include or have access to appropriate hardware and memory elements such as, for example, a logged events database 231 and an internal whitelist 233. In some embodiments, central server 200 may also include or have access to a central untrusted software inventory 232, and in other embodiments central untrusted software inventory 232 may not be a required component of the system. Other hardware elements including a processor 280 and a memory element 290 may also be included in central server 200. Finally, a management console 210 may be suitably connected to central server 200 for authorized persons to deploy, configure, and maintain the system through an administrative component such as administrative protection module 220.

[0039] In embodiments using central trusted cache 245, the cache may be implemented in hardware as a block of memory for temporary storage of entries (*e.g.*, checksums) identifying program files that have been previously determined to have a trusted status, such as those program files found during searches of global and/or internal whitelists. Central trusted cache 245 can provide quick and transparent access to data indicating program files previously evaluated for a trust status. Thus, if a requested program file is found in central trusted cache 245 then a search of global and/or internal whitelists, or any other trust evaluation, may not need to be performed. In addition, embodiments using central trusted cache 245 may not need to maintain central untrusted software inventory 232.

[0040] Turning to FIGURE 3, FIGURE 3 shows a schematic diagram of one embodiment of a computing device or host 300 and a block diagram of central server 200, in which the system for network level protection against malicious software may be implemented. Host 300 is a more detailed example of host 120a of FIGURE 1. Host 300 includes a set of executable

software 340 including, for example, program files 1 through n. In some embodiments such as, for example, the real-time processing embodiment, local protection components in host 300 may include an event feed 330 with a bidirectional flow to central server 200 for intercepting a network access attempt on host 300 and sending host event information (*i.e.*, information related to the network access attempt and its associated program files) to central server 200. Event feed 330 may also be configured to hold the network access attempt for a predetermined time or until a signal is received from central server 200, in order to allow any resulting network restriction and/or logging rules to be pushed to network protection device 400 before releasing the network access attempt.

[0041] In various embodiments of the system such as the batch processing embodiment or certain real-time processing embodiments, local protection components may include software program inventory feed 335 with a data flow to central server 200 for pushing an inventory of executable software 340 or an inventory of new and/or changed program files in executable software 340 to central server 200. Event feed 330 and software program inventory feed 335 may reside in the user space of host 300. Also shown in user space of host 300 is an example executing software process 345, which corresponds to one or more of the program files of executable software 340. For ease of reference, executable software 340 is shown in user space on host 300. However, executable software 340 may be stored in a memory element such as disk space of host 300.

[0042] Host 300 may also include hardware components such as a network interface card (NIC) device 370, a processor 380, and a memory element 390. Transmission protocols, such as Transmission Control Protocol/Internet Protocol (TCP/IP) 350 and other protocols 360, may reside in a kernel space of host 300.

[0043] Not shown in FIGURES 2 and 3 is additional hardware that may be suitably coupled to processors 280 and 380 in the form of memory management units (MMU), additional symmetric multiprocessing (SMP) elements, physical memory, Ethernet, peripheral component interconnect (PCI) bus and corresponding bridges, small computer system interface (SCSI)/integrated drive electronics (IDE) elements, etc. In addition, suitable modems and/or

additional network adapters may also be included for allowing network access. Central server 200 and host 300 may include any additional hardware and software necessary to properly perform their intended functions. Furthermore, any suitable operating systems will also be configured in central server 200 and host 300 to appropriately manage the operation of hardware components therein. It will be appreciated that the hardware configurations may vary and the depicted examples are not meant to imply architectural limitations.

[0044] Turning to FIGURE 4, FIGURE 4 shows a schematic diagram of a network protection device 400, which may be used in the system for network level protection against malicious software. Network protection device 400 is an example of network protection device 140 of FIGURE 1. Network level enforcement components of network protection device 400 may include a filter/firewall 410, a network rules element 420, and an event log 430. Network protection device 400 may also include hardware components such as a processor 480 and a memory element 490. In example embodiments, network protection device 140 may be a common network security device (*e.g.*, a firewall, a router, a web gateway, a mail gateway, a managed switch, etc.) and may include other suitable hardware and software components as required to perform its intended functions.

[0045] In network protection device 400, filter/firewall 410 may intercept network traffic (*e.g.*, electronic packets outbound from local network 110, inbound to local network 110, or within local network 110) and may query network rules element 420 to determine whether any restriction and/or logging rules apply to the particular intercepted packets of network traffic. If an applicable restriction rule is found, then it is applied to the packets, which may be blocked, rerouted, selectively allowed, and the like. Event log 430 may be provided in network protection device 400 for logging network traffic event data. Network traffic event data may include information related to particular packets received by network protection device 400 such as, for example, source address and port number, destination address and port number, date and time stamp, and/or rule ID (*i.e.*, identifier indicating a restriction or logging rule applied to the intercepted packets). Such logging may occur if network rules element 420 has a logging rule corresponding to the particular packets received by network protection device 400

or if logging is performed by default such as, for example, logging a network traffic event when a restriction rule is applied to packets corresponding to the network traffic event.

[0046] In various example embodiments, trust determination, logging, and rule-making activities may be provided by administrative protection module 220, policy module 230, software trust determination module 240, rule making module 250, and process traffic mapping module 260 of central server 200 and by software program inventory feed 335 and/or event feed 330 of host 120a. Information related to the trust determination, logging, and rule making activities can be suitably rendered, or sent to a specific location (*e.g.*, central server 200, network rules element 420, etc.) or simply stored or archived (*e.g.*, logged events database 231, central untrusted software inventory 232, policy database 235, process traffic mapping database 234, central trusted cache 245, etc.), and/or properly displayed in any appropriate format (*e.g.*, through management console 210, etc.). Security technology that relates to one or more of such trust determination, logging, and rule-making activities can include elements of McAfee® software (*e.g.*, ePolicy Orchestrator, Application Control, and/or Change Control) or any other similar software. Thus, any such components may be included within the broad scope of the terms 'administrative protection module', 'policy module', 'software trust determination module', 'rule making module', 'process traffic mapping module', 'software program inventory feed', and 'event feed' as used herein in this Specification. Logged events database 231, central untrusted software inventory 232, internal whitelist 233, process traffic mapping database 234, policy database 235, central trusted cache 245, network rules element 420, and event log 430 may include information related to the trust determination, logging, and rule making for electronic data (*e.g.*, trust determinations for program files, network access attempts, destination address and port numbers of software processes, source address and port numbers of software processes, network restriction rules, logging rules, etc.), and these elements can readily cooperate, coordinate, or otherwise interact with the modules and components of host 300, central server 200, and network protection device 400.

[0047] FIGURES 5-12 include flowcharts of example embodiments of flows associated with various embodiments of the system and method for network level protection

against malicious software. For ease of reference, FIGURES 5-12 will be described herein with references to certain components in network environment 100 of FIGURE 1 and to server 200, host 300, network protection device 400, and their associated components, elements, and modules, although various other devices, components, elements, modules and the like may be used to implement the network level embodiment of the system and method shown and described herein.

[0048] Turning to FIGURE 5, batch processing flow 500 illustrates a flow for batch processing embodiments of the system for network level protection that may be implemented at least in part as rule making module 250 of central server 200. In embodiments of the system implementing batch processing flow 500, neither central trusted cache 245 of central server 200, nor central untrusted software inventory 232 may be necessary. Batch processing flow 500 may run at any suitable predefined intervals of time (*e.g.*, hourly, every half-hour, etc.). Flow may begin at step 520 where an inventory of executable software is received from a host, such as host 300. Central server 200 may poll host 300 for the inventory, which may be provided by software program inventory feed 335. Software program inventory feed 335 may be configured to provide central server 200 with a complete updated software inventory on host 300 or with an inventory of software program file additions and/or changes on host 300. Software program inventory feed 335 could be implemented with existing security software such as, for example, Policy Auditor software or Application Control software, both manufactured by McAfee, Inc. of Santa Clara, California.

[0049] After receiving the software inventory, flow passes to step 525 where a first program file is retrieved from the software inventory. Flow then passes to step 530 to determine a trust status (*i.e.*, trusted or untrusted) of the program file. Trust status may be determined by software trust determination module 240 of central server 200 using one or more software trust determination techniques (*e.g.*, evaluating internal whitelists, evaluating external whitelists, evaluating state change of program files, evaluating blacklists, etc.), which will be shown and described further herein with reference to FIGURES 6 and 7.

[0050] After the program file trust status is determined in step 530, flow passes to step 535 where a query is made as to whether the program file is trusted. If the program file is trusted, then flow passes to step 580 to bypass creating network restriction or logging rules and to continue looping through the software inventory to evaluate each program file. However, if the program file trust status is untrusted in step 535, then flow passes to step 540 where process traffic mapping information is obtained for the untrusted program file from process traffic mapping database 234. The process traffic mapping information may include, for example, source address and destination port number mapped to the program file, which can be used to create rules to log and/or restrict network traffic.

[0051] After the process traffic mapping information is retrieved for the program file, a query is made as to whether logging is enabled in step 545. If the query in step 545 indicates that logging is enabled, then flow passes to step 550 where a logging rule may be created and then pushed to network protection device 400 to be stored in network rules element 420. In one example, the source address and destination port number from the process traffic mapping information retrieved in step 540 could be used to create a rule to log particular network event data. In this example, the logging rule could require network traffic event data related to electronic packets intercepted by network protection device 400 to be stored in event log 430 when the intercepted packets have a source address and a destination port matching the process traffic mapping information. In some embodiments, a rule ID identifying the logging rule may be stored in process mapping database 234 and mapped to the untrusted program file.

[0052] After the logging rule has been created and pushed to network protection device 400 in step 550, or if logging is not enabled in step 545, then flow passes to step 555 where a query is made as to whether enforcement is enabled. If enforcement is not enabled, then flow passes to step 580 to bypass creating a network restriction rule for the untrusted program file and to continue looping through the software inventory to evaluate the remaining program files in the software inventory. If enforcement is enabled in step 555, however, then policy database 235 may be queried in step 560 to determine whether any configured policy

overrides the untrusted status of the program file to allow network traffic associated with the program file. In example embodiments, policy module 230 of central server 200 may allow a network administrator or other authorized user to craft policy configurations through management console 210, and to store such policies in policy database 235. Policy database 235 may then be queried in step 560 for any policies relevant to the untrusted program file.

[0053] If a policy is found in policy database 235 that overrides the untrusted status of the program file, then flow passes to step 580 to bypass creating a network restriction rule for the untrusted program file and to continue looping through the software inventory to evaluate the remaining program files in the software inventory. However, if a policy does not override the untrusted status of the program file (*i.e.*, a policy requires some type of restriction rule or no policy is applicable), then flow passes to step 570 where one or more network restriction rules can be created using process traffic mapping information and/or any applicable policies.

[0054] Policies may be used to create various types of restriction rules, and these policy configurations may be implemented as desired by particular network owners. In some example embodiments, policy configurations may include one or more broad-based restrictions such as blocking all inbound and outbound network traffic, blocking all inbound network traffic and allowing outbound network traffic, or allowing inbound network traffic and blocking outbound network traffic. More specific strategies may also be employed, such as blocking outbound network traffic to the local network but allowing outbound network traffic to the Internet, or allowing inbound network traffic from a specified subnet of source addresses and/or allowing outbound network traffic to a specified subnet of destination addresses. Finally, even more granular strategies may be used such as blocking specific inbound services and/or specific outbound services on a port (*e.g.*, domain name service (DNS), simple mail transfer protocol (SMTP), Internet Relay Chat (IRC), etc.). These example policy configurations are for illustrative purposes, and are intended to include any other policy configurations to restrict inbound, outbound, and/or local network traffic or any combination thereof. Such policies may be implemented in the system for network level protection by routing network

traffic through network protection device 400, which applies network restriction rules created using the configured policies.

[0055] Particular policy configurations may be balanced between competing interests such as the need to prevent the propagation and potentially malicious activities of untrusted software and the need to conduct necessary business activities. For example, in a network having a host subnet and a server subnet, a policy may be configured to allow network traffic associated with untrusted program files to access only the server subnet but not the host subnet. This may be desirable because it may prevent the propagation of malicious software to other hosts within the network, while allowing each host uninterrupted access to a secured server subnet. Another policy may block network traffic associated with untrusted program files from accessing the Internet except for a known subnet hosting job critical services. Thus, many different blocking options may be employed by crafting policies allowing selective network access.

[0056] In embodiments of the system and method for network level protection, network level specific policies may also be crafted for untrusted program files. For example, a policy may be crafted to redirect network traffic associated with an untrusted program file to another server, such as secondary server 180. In one example, potentially malicious network traffic associated with an untrusted program file could be forced through additional firewalls, filters, antispam/antivirus gateways, proxies, and the like on secondary server 180. In another example, secondary server 180 may be configured to respond with one or more predefined commands upon receiving a network connection. Some bots are designed to self-destruct upon receiving particular commands and secondary server 180 could be configured to respond to a network connection with such commands, thereby causing a bot that has been redirected to secondary server 180 to be destroyed.

[0057] Another network level specific policy includes switching virtual local area network (VLAN) membership to another VLAN port. In this example, VLAN membership could be switched for the port associated with an untrusted program file. Although switching VLAN membership to another port would effectively move all of the network traffic on that port

rather than individual streams, the alternate VLAN could be configured to force the network traffic through additional firewalls, filters, antispam and antivirus gateways, proxies, and the like. This type of restriction could be particularly useful if network protection device 400 is configured as a layer 2 managed switch.

[0058] In another example embodiment, network protection device 400 may be adapted to do a deeper packet inspection to determine whether multiple conversation streams are being transferred over a single port and to identify the stream associated with the untrusted program file to which the applicable network restriction and logging rules apply. Thus, in this embodiment, a policy could be configured so that network restriction or logging rules are crafted to selectively block and/or log the particular stream associated with the untrusted program file, while allowing other streams to continue connections over the same port.

[0059] Turning back to step 570 of FIGURE 5, a network restriction rule is created using process traffic mapping information and/or any applicable policies and then pushed to network protection device 400. For example, if a policy is configured requiring all inbound and outbound network traffic associated with an untrusted program file to be blocked, then the source address and destination port from the process traffic mapping information retrieved in step 540 could be used to create one or more restriction rules to block any inbound or outbound packets having a matching source address and destination port. After the network restriction rule has been created in step 570, it is pushed to network protection device 400 and stored in network rules element 420. Flow then passes to step 580 where a determination is made as to whether the program file being evaluated is the last program file in the software inventory. If it is the last program file, then flow 500 ends. However, if it is not the last program file, then the next program file in the software inventory is retrieved in step 590 and flow loops back to step 530 to determine the trust status of the newly retrieved program file and to create any appropriate network restriction or logging rules for the program file.

[0060] Turning to FIGURES 6-7, example flows illustrate alternative embodiments for determining a trust status of program files in the batch processing embodiment of the system

for network level protection. The example steps of FIGURES 6 or7 may be implemented at least in part as software trust determination module 240 of central server 200 and may correspond to step 530 of the batch processing embodiment of FIGURE 5. FIGURE 6 illustrates a trust determination flow 600 using whitelist evaluations, which is performed for each program file in the software inventory pushed to central server 200 from host 300. Flow may begin at step 610 where the current program file is evaluated to determine if it is identified on at least one global whitelist, such as global whitelist 165 shown in FIGURE 1, and/or any other whitelist outside of local network 110. As previously described herein, checksums may be used in whitelists to identify program files. If the program file is found on any of the global or other external whitelists in step 610, then a trust status of the program file is defined as trusted, the flow ends, and the trust status is returned to step 530 of FIGURE 5. If the program file is not found on the global or other external whitelists in step 610, however, then flow moves to step 620 where one or more internal whitelists, such as internal whitelist 133, may be searched. Organizations may employ multiple whitelists (*e.g.*, an organization-wide whitelist, an enterprise level whitelist, etc.). If the program file is found on any internal whitelist, then the trust status of the program file is defined as trusted, the flow ends, and the trust status is returned to step 530 of FIGURE 5.

[0061] If the program file is not found on any internal or external whitelist in steps 610 and 620, however, then the program file has an untrusted status. Flow may then move to step 630 where the program file may be evaluated to determine whether any predefined condition exists that allows the program file to be promoted from the untrusted status to a trusted status. Such predefined conditions may include heuristic considerations such as, for example, software owned by an administrator, file access controls, file attributes (*e.g.*, creation time, modification time, etc.), and the like. In one example, an untrusted program file owned by an administrator could be promoted to a trusted status and, therefore, flow could end and the trust status could be returned to step 530 of FIGURE 5. If the program file does not satisfy any predefined condition in step 630, however, then the untrusted status persists such that the

program file is defined as untrusted, the flow ends, and the trust status is returned to step 530 of FIGURE 5.

[0062] Trust determination flow 600 may also include additional logic (not shown) to evaluate blacklists in addition to whitelists. Blacklists identify software program files known to be malicious. Blacklists may be provided by numerous sources including Artemis and Anti-Virus databases provided by McAfee, Inc., and locally maintained blacklists within a local network. In this embodiment, if the program file is found on any internal or external blacklist, then the program file is defined as untrusted.

[0063] Turning to FIGURE 7, FIGURE 7 illustrates an alternative embodiment of a trust determination flow 700, corresponding to step 530 of FIGURE 5, which may be performed for each program file in the software inventory. Flow may begin at step 720 where the current program file is evaluated to determine whether it has changed. If a current state of the program file has not changed from a previous state, then a trust status of the program file is defined as trusted, the flow ends, and the trust status is returned to step 530 of FIGURE 5. If the current state of the program file has changed from the previous state, however, then the program file has an untrusted status. Existing change tracking products (*e.g.*, McAfee® Change Control software, McAfee® Application Control software, McAfee® ePolicy Orchestrator software, McAfee® Policy Auditor software, Tripwire® software manufactured by Tripwire, Inc. of Portland, Oregon, etc.) may be used to examine change data of the program files to determine whether a change has occurred. In one example, change records may be aggregated centrally at central server 200, and McAfee® ePO software may make the determination whether the program file has changed.

[0064] Referring again to step 720 of FIGURE 7, if the program file is determined to have changed and, therefore, to have an untrusted status, flow may then move to step 730 where the program file is evaluated to determine whether any predefined condition exists to allow the program file to be promoted from an untrusted status to a trusted status, as previously described herein with reference to FIGURE 6. If one or more predefined conditions exist, then the program file may be promoted to a trusted status. If no predefined condition

applies to the program file, however, then the untrusted status persists and the program file may be defined as untrusted. After the trust status determination is made in step 730, then the flow ends and the trust status of the program file is returned to step 530 of FIGURE 5.

[0065] Alternative implementations to enumerate program files and to determine a trust status will be readily apparent. Embodiments previously shown and described herein refer to enumerating an inventory of executable software on each host in a network, such as host 300, pushing the software inventory to central server 200 and determining the trust status associated with each program file in the inventory. In alternative embodiments, however, the trust determination of software program files could be locally performed by each host and resulting information could be pushed to another location (*e.g.*, central server 200) and/or maintained locally on a local untrusted software inventory.

[0066] Locally determining a trust status of software program files could be performed by whitelist evaluations, blacklist evaluations, state change evaluations, or any other suitable trust evaluation technique. In such embodiments an inventory of executable software may be enumerated by, for example, McAfee® software (*e.g.*, Policy Auditor, Application Control, or Change Control). When performing whitelist evaluations as shown in FIGURE 6, the host could access internal whitelists on a local network and/or external whitelists accessible through another network, such as the Internet. Program file state change evaluations could also be performed locally by evaluating a current and previous state of the program file for changes, as shown in FIGURE 7. For example, McAfee® Change Control software allows a sequence of change records to be locally maintained on hosts, which could be evaluated to determine if any of the program files on a particular host had changed. In addition, the broad scope of this disclosure permits any number of other techniques to be used to determine the trust status of the software program files including, for example, performing both whitelist evaluations and state change evaluations of software program files and/or performing blacklist evaluations of software program files.

[0067] The batch processing embodiments shown and described with reference to FIGURES 5-7, may be implemented so that software inventories can be evaluated at predefined

time intervals (*e.g.*, hourly, every half hour, etc.), and restriction rules can be created to block network traffic associated with untrusted program files. Thus, the configuration of the network protection device 400 is open by default (*i.e.*, network traffic is allowed unless a rule proscribes it). This open by default approach, however, may create windows of time in which a bot, which has recently infected a host, has not yet been defined as untrusted, and could, therefore, be propagating or performing malicious activities by accessing local or remote networks because no restriction rules have been created to block network traffic associated with the bot. For some businesses, however, this approach may be desirable because the predefined time interval may be insignificant or because alternative approaches may hinder legitimate and necessary business activities.

[0068] Other businesses may prefer tighter control and a closed by default approach could be implemented in an alternative embodiment. In the closed by default approach, network protection device 400 could be configured to block all network traffic unless specifically permitted by a rule. All electronic packets intercepted by network protection device 400 could be evaluated to determine whether network rules element 420 contains a rule specifically permitting the intercepted packets to be transmitted (*e.g.*, packets having an allowed source address and port number, packets having an allowed destination address and port number, etc.). In such an embodiment, permission rules, rather than restriction rules, could be created and pushed to network protection device 400 whenever new program files are determined to have a trusted status.

[0069] Turning to FIGURE 8, a real-time processing flow 800 illustrates a flow for real-time embodiments of the system for network level protection that may be implemented at least in part as rule making module 250 of FIGURE 2. Flow may begin at step 810 where central server 200 receives host event information (*e.g.*, information related to a network access attempt associated with software process 345 on host 300). In example embodiments, event feed 330 generates host event information and pushes it to central server 200 whenever a software process, such as software process 345 on host 300, attempts network access. The network access attempt may be intercepted using any suitable techniques including those

described with reference to the various embodiments in copending U.S. Patent Application Serial No. 12/844,892, entitled *System and Method for Local Protection Against Malicious Software*, filed July 28, 2010, and previously incorporated herein by reference.

[0070] Once the network access attempt has been intercepted, a process traffic mapping element provided, for example, in the operating system kernel of host 300 may be queried to determine which program files (*e.g.*, executable files, library modules, object files, other executable modules, script files, interpreter files, etc.) correspond to the network access attempt associated with software process 345. In this example, the network access attempt is mapped to executing software process 345, which could be mapped to an executable file and one or more library modules loaded into executing software process 345. Thus, the host event information that is pushed to central server 200 may include program file paths for the one or more identified program files, the associated program file hashes, a source address and port, and/or a destination address and port of the network access attempt.

[0071] After the host event information is received by central server 200 in step 810, flow passes to step 820 to get a trust status of the program files associated with the network access attempt. Determining the trust status of the program files can be accomplished using various techniques, which will be shown and described in more detail in a first embodiment in FIGURE 9 (*i.e.*, determining trust statuses using central untrusted software inventory 232) and in another embodiment in FIGURE 10 (*i.e.*, determining trust statuses in real-time using central trusted cache 245).

[0072] After obtaining the trust status for each of the program files associated with the network access attempt in step 820, flow passes to step 830, which may be implemented, at least in part, as process traffic mapping module 260 of central server 200. In step 830, if any of the program files have an untrusted status, then the host event information may be used to populate process traffic mapping database 234. For example, detailed port and address information and program file path information associated with the program files may be added to process traffic mapping database 234. Flow then passes to step 835 and a query is made as to whether all program files associated with the network access attempt have a trusted status,

and if so, flow ends without creating logging or restriction rules for network traffic associated with the program files. However, if any of the program files has an untrusted status, then flow passes to step 845 to determine whether logging is enabled.

[0073] If logging is enabled, then flow passes to step 850 where a logging rule may be created and pushed to network protection device 400 to be stored in network rules element 420. In one example, the source address and port number and the destination address and port number from the host event information could be used to create a rule to log particular network event data. In this example, the logging rule could require network traffic event data related to electronic packets intercepted by network protection device 400 to be stored in event log 430 when the intercepted packets have a source address, a source port, a destination address, and a destination port matching the host event information. In some embodiments, a rule ID identifying the logging rule may be stored in process traffic mapping database 234 and mapped to the program files associated with the network access attempt.

[0074] After the logging rule has been created and pushed to network protection device 400 in step 850, or if logging was not enabled in step 845, flow passes to step 855 to determine whether enforcement is enabled. If enforcement is not enabled, then the flow ends without creating restriction rules for network traffic associated with the one or more untrusted program files. If enforcement is enabled in step 855, however, then policy database 235 may be queried in step 860 to determine whether any configured policy overrides the untrusted status of the program files to allow network traffic associated with the program files. If such a policy is found, then flow ends without creating a restriction rule. However, if a policy does not override the untrusted status of the program file (*i.e.*, a policy requires some type of restriction rule or no policy is applicable), then flow passes to step 870 where one or more network restriction rules can be created using host event information and/or any applicable policies and then pushed to network protection device 400. In one example, the host event information could be used to create a restriction rule to block any inbound, outbound, and/or local packets having a source address and port and destination address and port matching the source address and port and destination address and port from the host event information. The use of

policies to create restriction rules and examples of such policies, including network level specific policies, have been previously described herein with reference to the batch processing flow of FIGURE 5.

[0075] After the restriction rule has been created using host event information and/or any applicable policies, the restriction rule is pushed to network protection device 400 and stored in network rules element 420 and then the flow ends. In this real-time embodiment, a time-delay may be configured in host 300 after the network access attempt has been intercepted in order to allow real-time processing flow 800 sufficient time to create any necessary rules and push such rules to network protection device 400. In another embodiment, the network access attempt may be held on host 300 until central server 200 acknowledges to host 300 that it has updated process traffic mapping database 234 with mapping information for the network access attempt and/or that it has pushed any resulting logging or restriction rules to network protection device 400. This acknowledgement could be accomplished, for example, via a signal over bidirectional data flow to event feed 330 on host 300.

[0076] Turning to FIGURES 9-10, example flows illustrate alternative embodiments for determining a trust status of program files in the real-time processing embodiment of the system for network level protection. The example steps of FIGURES 9 or 10 may be implemented at least in part as software trust determination module 240 of central server 200 and may correspond to step 820 of the real-time processing embodiment of FIGURE 8.

[0077] FIGURE 9 illustrates a trust determination flow 900, utilizing central untrusted software inventory 232. In this embodiment, central untrusted software inventory 232 may be created and maintained, for example, as described with reference to FIGURES 4-6 of copending U.S. Patent Application Serial No. 12/844,892, previously incorporated herein by reference. In step 930, central untrusted software inventory 232 is searched for each of the program files associated with the network access attempt. If a program file is identified on central untrusted software inventory 232, then a trust status of the identified program file is untrusted. However, if the program file is not identified on central untrusted software inventory 232, then the trust status of the program file is trusted. After the program files have been evaluated, the

trust statuses of the program files are returned to step 820 of FIGURE 8, where logging and restriction rules may be created as previously described herein.

[0078] Turning to FIGURE 10, FIGURE 10 illustrates an alternative trust determination flow 1000 corresponding to step 820 of FIGURE 8, for performing a real-time trust determination in the real-time embodiment of the system for network level protection. Central untrusted software inventory 232 may be omitted from this embodiment and, alternatively, central trusted cache 245 may be used to store identifications (*e.g.*, checksums) of trusted program files for quicker retrieval and overall processing during the trust status determination.

[0079] Beginning in step 1010, a query is made as to whether all program files associated with the network access attempt are found in central trusted cache 245. If all of the program files are found in central trusted cache 245, then all of the program files have a trusted status. Consequently, flow ends and the trust statuses are returned to step 820 of FIGURE 8. If any program file is not found in central trusted cache 245, however, then flow passes to step 1020 where one or more global or other external whitelists, such as global whitelist 165 shown in FIGURE 1, and/or one or more internal whitelists, such as internal whitelist 133 shown in FIGURE 1, are searched for each of the program files not found in central trusted cache 245. If a program file is identified on at least one of the whitelists, then the trust status of the identified program file is defined as trusted, but if a program file is not identified on any of the whitelists, then the trust status of the program file is defined as untrusted. After the whitelists have been searched, flow passes to step 1030 where central trusted cache 245 is updated with checksums of the program files having a trusted status (*i.e.*, identified on one of the whitelists). The trust statuses of the program files can then be returned to step 820 of FIGURE 8.

[0080] Real-time trust determination flow 1000 may also include additional logic to evaluate a program file not found in any whitelist and consequently having an untrusted status to determine whether a predefined condition exists that allows the untrusted program file to be promoted to a trusted status. Such predefined conditions may include heuristic considerations, which have been previously shown and described herein with reference to FIGURE 6. If one or more predefined conditions exist, then the program file may be promoted

to a trusted status. If no predefined condition applies to the program file, however, then the untrusted status persists and the program file may be defined as untrusted. Real-time trust determination flow 1000 may also include additional logic (not shown) to evaluate blacklists in addition to whitelists, also previously described herein.

[0081] Turning to FIGURE 11, FIGURE 11 illustrates a network enforcement flow 1100, of network protection device 400, which may be used with both the batch processing embodiments (FIGURES 5-7) and the real-time embodiments (FIGURES 8-10) of the system for network level protection. Flow may begin at step 1110 where filter/firewall 410 of network protection device 400 intercepts network traffic associated with a software process, such as executing software process 345 on host 300. In example embodiments, TCP/IP network traffic may be in the form of multiple electronic packets. Flow then passes to step 1120 where a query is made as to whether a logging rule applies to the intercepted packets, which may be determined by searching network rules element 420. If a logging rule applies, event log 430 is updated with event data related to the network traffic in step 1130. Once event log 430 is updated, or if no logging rule applies as determined in step 1120, then flow passes to step 1140 where a query is made as to whether a network restriction rule applies to the intercepted packets. This query may be made by searching network rules element 420. If a network restriction rule does not exist for the intercepted packets, then flow passes to step 1190 and the packets are allowed to pass through the filter/firewall 410 to their destination address. If a network restriction rule applies to the intercepted packets, however, then flow passes to step 1180. In step 1180, the network restriction rule is applied to the packets and the packets are restricted accordingly (*e.g.*, packets are blocked, packets are redirected to another server, VLAN membership is switched, etc.).

[0082] FIGURE 12 illustrates a flow 1200 for logging network access attempts, which may be implemented in the batch processing embodiments (FIGURES 5-7) and in the real-time embodiments (FIGURES 8-10) of the system for network level protection. Flow may begin at step 1210 where log records from event log 430 of network protection device 400 are retrieved. In example embodiments, only new log records (*e.g.*, log records added to event log

430 since a last predetermined date and time, log records added to event log 430 since the last update to logged events database 231, etc.) are retrieved. Alternatively, all log records could be retrieved and processed or all log records could be retrieved and flow 1200 could include additional processing to determine which of the log records are new.

[0083] After the log records are retrieved from network protection device 400, flow passes to step 1220 where information may be retrieved from process traffic mapping database 234. Process traffic mapping database 234 may provide user-understandable information, such as program file paths and host identification, corresponding to the network traffic event data in the log record. For example, a destination address and port and a source address and port from a log record may be mapped to an untrusted program file path in process traffic mapping database 234. In another example, a rule ID could be used from a log record to find a mapping to an untrusted program file path in process traffic mapping database 234 or, alternatively, in some other separate database or record.

[0084] After the user-understandable information is retrieved from process traffic mapping database 234, flow passes to step 1230 where logged events database 231 can be updated with network traffic event data from the log records and any corresponding process traffic mapping information. Examples of possible network traffic event data and corresponding process traffic mapping information stored in logged events database 231 include data associated with intercepted packets such as program file paths, identification of the hosts, a date and time stamp, source address and port numbers, destination address and port numbers, and the like.

[0085] With both the batch processing embodiments and the real-time embodiments, flow 1200 may be configured to run at any predetermined intervals of time (*e.g.*, weekly, daily, hourly, etc.). Flow 1200 may, in some embodiments, be implemented as part of administrative protection module 220. Alternatively, in the batch processing embodiments, flow 1200 may be implemented as part of flow 500 shown in FIGURE 5. Administrative protection module 220 may provide access to the data in logged events database 231 through, for example, management console 210 or any other reporting mechanisms. The data in logged events

database 231 can provide users with information regarding network traffic intercepted by network protection device 400 and associated with untrusted program files on hosts within the network. As previously discussed herein, such network traffic may or may not be blocked or otherwise restricted depending upon whether enforcement is enabled and a corresponding restriction rule has been created and pushed to network protection device 400. Thus, logging event data related to such network traffic and the subsequent processing of that event data in flow 1200 may occur regardless of whether the network traffic is blocked or otherwise restricted.

[0086] Software for achieving the operations outlined herein can be provided at various locations (*e.g.*, the corporate IT headquarters, end user computers, distributed servers in the cloud, etc.). In other embodiments, this software could be received or downloaded from a web server (*e.g.*, in the context of purchasing individual end-user licenses for separate networks, devices, servers, etc.) in order to provide this system for network level protection against malicious software. In one example implementation, this software is resident in one or more computers sought to be protected from a security attack (or protected from unwanted or unauthorized manipulations of data).

[0087] In other examples, the software of the system for network level protection against malicious software could involve a proprietary element (*e.g.*, as part of a network security solution with McAfee® Application Control software, McAfee® Change Control software, McAfee® ePolicy Orchestrator software, McAfee® Policy Auditor software, McAfee® Artemis Technology software, McAfee® Host Intrusion Prevention software, McAfee® VirusScan software, etc.), which could be provided in (or be proximate to) these identified elements, or be provided in any other device, server, network appliance, console, firewall, switch, router, information technology (IT) device, distributed server, etc., or be provided as a complementary solution (*e.g.*, in conjunction with a firewall), or provisioned somewhere in the network.

[0088] In an example local network 110 as shown in FIGURE 1, hosts 120, central server 130, network protection device 140, and secondary server 180 are computers that facilitate protecting computer networks against malicious software. As used herein in this

Specification, the terms 'computer' and 'computing device' are meant to encompass any personal computers, network appliances, routers, switches, gateways, processors, servers, load balancers, firewalls, or any other suitable device, component, element, or object operable to affect or process electronic information in a network environment. Moreover, this computer or computing device may include any suitable hardware, software, components, modules, interfaces, or objects that facilitate the operations thereof. This may be inclusive of appropriate algorithms and communication protocols that allow for the effective protection and communication of data.

[0089] In certain example implementations, the activities involved in network level protection against malicious software outlined herein may be implemented in software. This could be inclusive of software provided in central servers 130 (*e.g.*, central protection components 135), hosts 120 (*e.g.*, local protection components 124), network protection device (*e.g.*, network level enforcement components 145), and/or secondary server 180. These components, elements and/or modules can cooperate with each other in order to perform activities to provide network level protection against malicious software such as botnets, as discussed herein. In other embodiments, these features may be provided external to these elements, included in other devices to achieve these intended functionalities, or consolidated in any appropriate manner. For example, the protection activities could be further localized in hosts 120 or further centralized in central server 130, and some of the illustrated processors may be removed, or otherwise consolidated to accommodate the particular system configuration. In a general sense, the arrangement depicted in FIGURES 1 may be more logical in its representations, whereas a physical architecture may include various permutations/combinations/hybrids of these components, elements, and modules.

[0090] All of these elements (hosts 120, central server 130, network protection device 140, and/or secondary server 180) include software (or reciprocating software) that can coordinate, manage, or otherwise cooperate in order to achieve the protection activities, including trust determination, logging, enforcement, intercepting, as outlined herein. In addition, one or all of these elements may include any suitable algorithms, hardware, software,

components, modules, interfaces, or objects that facilitate the operations thereof. In the implementations involving software, such a configuration may be inclusive of logic encoded in one or more tangible media (*e.g.*, embedded logic provided in an application specific integrated circuit (ASIC), digital signal processor (DSP) instructions, software (potentially inclusive of object code and source code) to be executed by a processor, or other similar machine, etc.), with the tangible media being inclusive of non-transitory media. In some of these instances, one or more memory elements (as shown in various FIGURES including FIGURES 1, 2, 3 and 4) can store data used for the operations described herein. This includes the memory elements being able to store software, logic, code, or processor instructions that are executed to carry out the activities described in this Specification. A processor can execute any type of instructions associated with the data to achieve the operations detailed herein in this Specification. In one example, the processors (as shown in FIGURES 2, 3, and 4) could transform an element or an article (*e.g.*, data) from one state or thing to another state or thing. In another example, the activities outlined herein may be implemented with fixed logic or programmable logic (*e.g.*, software/computer instructions executed by a processor) and the elements identified herein could be some type of a programmable processor, programmable digital logic (*e.g.*, a field programmable gate array (FPGA), an erasable programmable read only memory (EPROM), an electrically erasable programmable read only memory (EEPROM)) or an ASIC that includes digital logic, software, code, electronic instructions, or any suitable combination thereof.

[0091] Any of these elements (*e.g.*, a computer, a server, a network protection device, a firewall, distributed server, etc.) can include memory elements for storing information to be used in achieving the protection activities as outlined herein. These devices may further keep information in any suitable memory element (*e.g.*, random access memory (RAM), ROM, EPROM, EEPROM, ASIC, etc.), software, hardware, or in any other suitable component, device, element, or object where appropriate and based on particular needs. Any of the memory items discussed herein (*e.g.*, logged events database, central untrusted software inventory, local untrusted software inventory, internal whitelist, policy database, process traffic mapping database, central trusted cache, network rules element, event log, etc.) should be construed as

being encompassed within the broad term 'memory element.' Similarly, any of the potential processing elements, modules, and machines described in this Specification should be construed as being encompassed within the broad term 'processor.' Each of the computers, hosts, network protection devices, servers, distributed servers, etc. may also include suitable interfaces for receiving, transmitting, and/or otherwise communicating data or information in a network environment.

[0092] Note that with the examples provided herein, interaction may be described in terms of two, three, four, or more network components. However, this has been done for purposes of clarity and example only. It should be appreciated that the system for network level protection against malicious software can be consolidated in any suitable manner. Along similar design alternatives, any of the illustrated computers, modules, components, and elements of the FIGURES may be combined in various possible configurations, all of which are clearly within the broad scope of this Specification. In certain cases, it may be easier to describe one or more of the functionalities of a given set of flows by only referencing a limited number of components or network elements. Therefore, it should also be appreciated that the system of FIGURE 1 (and its teachings) are readily scalable. The system can accommodate a large number of components, as well as more complicated or sophisticated arrangements and configurations. Accordingly, the examples provided should not limit the scope or inhibit the broad teachings of the system for network level protection as potentially applied to various other architectures.

[0093] It is also important to note that the operations described with reference to the preceding FIGURES illustrate only some of the possible scenarios that may be executed by, or within, the system for network level protection against malicious software. Some of these operations may be deleted or removed where appropriate, or these operations may be modified or changed considerably without departing from the scope of the discussed concepts. In addition, the timing of these operations may be altered considerably and still achieve the results taught in this disclosure. For example, trust determination by searching whitelists may be performed by searching internal whitelists prior to searching external or global whitelists.

The preceding operational flows have been offered for purposes of example and discussion. Substantial flexibility is provided by the system in that any suitable arrangements, chronologies, configurations, and timing mechanisms may be provided without departing from the teachings of the discussed concepts.

WHAT IS CLAIMED IS:

1. A method, comprising:

receiving information related to a network access attempt on a first computing device, the information identifying a software program file associated with the network access attempt;

evaluating a first criterion to determine whether network traffic associated with the software program file is permitted; and

creating a restriction rule to block the network traffic if the network traffic is not permitted, wherein the first criterion includes a trust status of the software program file.

2. The method of Claim 1, further comprising pushing the restriction rule to a network protection device, wherein the network protection device intercepts the network traffic associated with the software program file and applies the restriction rule to the network traffic.

3. The method of Claim 2, wherein the network traffic intercepted by the network protection device includes outbound electronic packets from the first computing device, the electronic packets having a destination address corresponding to a second computing device.

4. The method of Claim 2, wherein the network traffic intercepted by the network protection device includes inbound electronic packets received from a second computing device and having a destination address corresponding to the first computing device.

5. The method of Claim 1, further comprising:

searching a whitelist identifying trustworthy software program files to determine the trust status of the software program file, wherein the trust status of the software program file is defined as untrusted if the software program file is not included in the whitelist.

6. The method of Claim 5, wherein the network traffic is not permitted if the trust status of the software program file is defined as untrusted.

7. The method of Claim 1, wherein the software program file is an executable file and the network access attempt is associated with an executing software process initiated by the executable file on the first computing device.

8. The method of Claim 1, wherein the software program file is a library module loaded into an executing software process on the first computing device and the network access attempt is associated with the executing software process.

9. The method of Claim 1, further comprising:

evaluating a second criterion to determine whether the second criterion overrides the first criterion, the second criterion including a network access policy for one or more software program files having a trust status defined as untrusted.

10. The method of Claim 9, wherein a network protection device intercepts electronic packets of the network traffic associated with the software program file and applies a restriction rule to the electronic packets, wherein the restriction rule is created using the network access policy selected from a group of network access policies, the group consisting of:

blocking all inbound electronic packets to the first computing device and all outbound electronic packets from the first computing device;

blocking all inbound electronic packets to the first computing device;

blocking all outbound electronic packets from the first computing device;

blocking all outbound electronic packets from the first computing device to a specified network subnet;

blocking all outbound electronic packets from the first computing device to the Internet and allowing all outbound electronic packets from the first computing device to a specified subnet;

blocking all outbound electronic packets from the first computing device to domain name system (DNS) servers;

blocking all outbound electronic packets from the first computing device using simple mail transfer protocol (SMTP);

blocking all outbound electronic packets from the first computing device to an Internet Relay Chat (IRC) server;

blocking all outbound electronic packets having a source address, a source port, a destination address, and a destination port matching a source address, a source port, a destination address, and a destination port of the network access attempt; and

blocking all inbound electronic packets having a source address, a source port, a destination address, and a destination port matching a source address, a source port, a destination address, and a destination port of the network access attempt.

11. Logic encoded in one or more tangible media that includes code for execution and when executed by one or more processors is operable to perform operations comprising:

receiving information related to a network access attempt on a first computing device, the information identifying a software program file associated with the network access attempt;

evaluating a first criterion to determine whether network traffic associated with the software program file is permitted; and

creating a restriction rule to block the network traffic if the network traffic is not permitted, wherein the first criterion includes a trust status of the software program file.

12. The logic of Claim 11, the one or more processors being operable to perform further operations comprising:

pushing the restriction rule to a network protection device, wherein the network protection device intercepts the network traffic associated with the software program file and applies the restriction rule to the network traffic.

13. The logic of Claim 11, the one or more processors being operable to perform further operations comprising:

searching a whitelist identifying trustworthy software program files to determine the trust status of the software program file, wherein the trust status of the software program file is defined as untrusted if the software program file is not included in the whitelist.

14. The logic of Claim 11, the one or more processors being operable to perform further operations comprising:

evaluating a second criterion to determine whether the second criterion overrides the first criterion, wherein the second criterion includes a network access policy for one or more software program files having a trust status defined as untrusted.

15. An apparatus, comprising:

a protection module;

one or more processors operable to execute instructions associated with the protection module, the one or more processors being operable to perform further operations comprising:

receiving information related to a network access attempt on a first computing device, the information identifying a software program file associated with the network access attempt;

evaluating a first criterion to determine whether network traffic associated with the software program file is permitted; and

creating a restriction rule to block the network traffic if the network traffic is not permitted, wherein the first criterion includes a trust status of the software program file.

16. The apparatus of Claim 15, wherein the one or more processors are operable to perform further instructions, including:

pushing the restriction rule to a network protection device, wherein the network protection device intercepts network traffic associated with the software program file and applies the restriction rule to the network traffic.

17. The apparatus of Claim 15, wherein the one or more processors are operable to perform further instructions, including:

searching a whitelist identifying trustworthy software program files to determine the trust status of the software program file, wherein the trust status of the software program file is defined as untrusted if the software program file is not included in the whitelist.

18. A method, comprising:

receiving information related to a network access attempt on a first computing device, the information identifying a software program file associated with the network access attempt;

evaluating a first criterion to determine whether network traffic associated with the software program file is permitted; and

creating a logging rule to log event data related to the network traffic if the network traffic is not permitted, wherein the first criterion includes a trust status of the software program file.

19. The method of Claim 18, further comprising pushing the logging rule to a network protection device, wherein the network protection device intercepts the network traffic associated with the software program file and logs event data related to the network traffic.

20. The method of Claim 19, further comprising:

receiving the event data logged by the network protection device;

using selected data from the event data to search a process traffic mapping element for the software program file associated with the network traffic; and

updating a logged events memory element to include an entry identifying the software program file.

21. The method of Claim 20, wherein the selected data used to search the process traffic mapping element includes a source address and port number and a destination address and port number.

22. The method of Claim 20 wherein the selected data used to search the process traffic mapping element includes a rule identifier corresponding to the logging rule.

23. The method of Claim 18, further comprising:

searching a whitelist identifying trustworthy software program files to determine whether the software program file is identified in the whitelist, wherein the trust status of the software program file is defined as untrusted if the software program file is not identified in the whitelist.

24. The method of Claim 18, wherein the software program file is an executable file and the network access attempt is associated with an executing software process initiated by the executable file on the computing device.

25. The method of Claim 18, wherein the software program file is a library module loaded into an executing software process and the network access attempt is associated with the executing software process.

26. Logic encoded in one or more tangible media that includes code for execution and when executed by one or more processors is operable to perform the operations comprising:

receiving information related to a network access attempt on a first computing device, the information identifying a software program file associated with the network access attempt;

evaluating a first criterion to determine whether network traffic associated with the software program file is permitted; and

creating a logging rule to log event data related to the network traffic if the network traffic is not permitted, wherein the first criterion includes a trust status of the software program file.

27. The logic of Claim 26, the one or more processors operable to perform further operations comprising:

pushing the logging rule to a network protection device, wherein the network protection device intercepts the network traffic associated with the software program file and logs event data related to the network traffic.

28. The logic of Claim 27, the one or more processors operable to perform further operations comprising:

receiving the event data logged by the network protection device;

using selected data from the event data to search a process traffic mapping element for the software program file associated with the network traffic; and

updating a logged events memory element to include an entry identifying the software program file.

29. The logic of Claim 26, the one or more processors operable to perform further operations comprising:

searching a whitelist identifying trustworthy software program files to determine whether the software program file is identified in the whitelist, wherein the trust status of the software program file is defined as untrusted if the software program file is not identified in the whitelist.

30. An apparatus, comprising:

a protection module; and

one or more processors operable to execute instructions associated with the protection module, including:

receiving information related to a network access attempt on a first computing device, the information identifying a software program file associated with the network access attempt;

evaluating a first criterion to determine whether network traffic associated with the software program file is permitted; and

creating a logging rule to log event data related to the network traffic if the network traffic is not permitted, wherein the first criterion includes a trust status of the software program file.

31. The apparatus of Claim 30, the one or more processors operable to execute further instructions comprising:

pushing the logging rule to a network protection device, wherein the network protection device intercepts the network traffic associated with the software program file and logs event data related to the network traffic.

32. The apparatus of Claim 31, the one or more processors operable to execute further instructions comprising:

receiving the event data logged by the network protection device;

using selected data from the event data to search a process traffic mapping element for the software program file associated with the network traffic; and

updating a logged events memory element to include an entry identifying the software program file.

33. The apparatus of Claim 30, the one or more processors operable to execute further instructions comprising:

searching a whitelist identifying trustworthy software program files to determine whether the software program file is identified in the whitelist, wherein the trust status of the software program file is defined as untrusted if the software program file is not identified in the whitelist.

FIG. 1

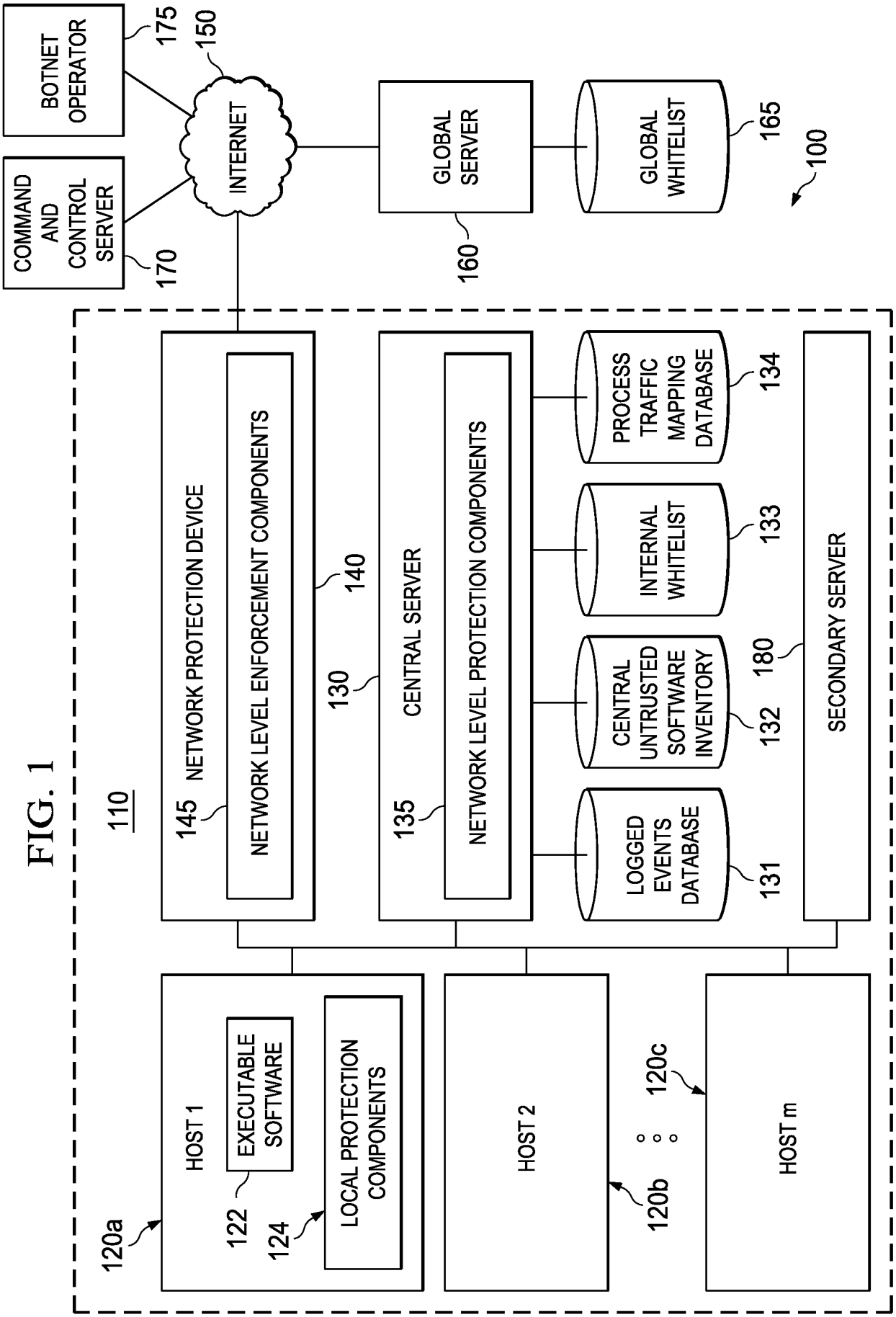
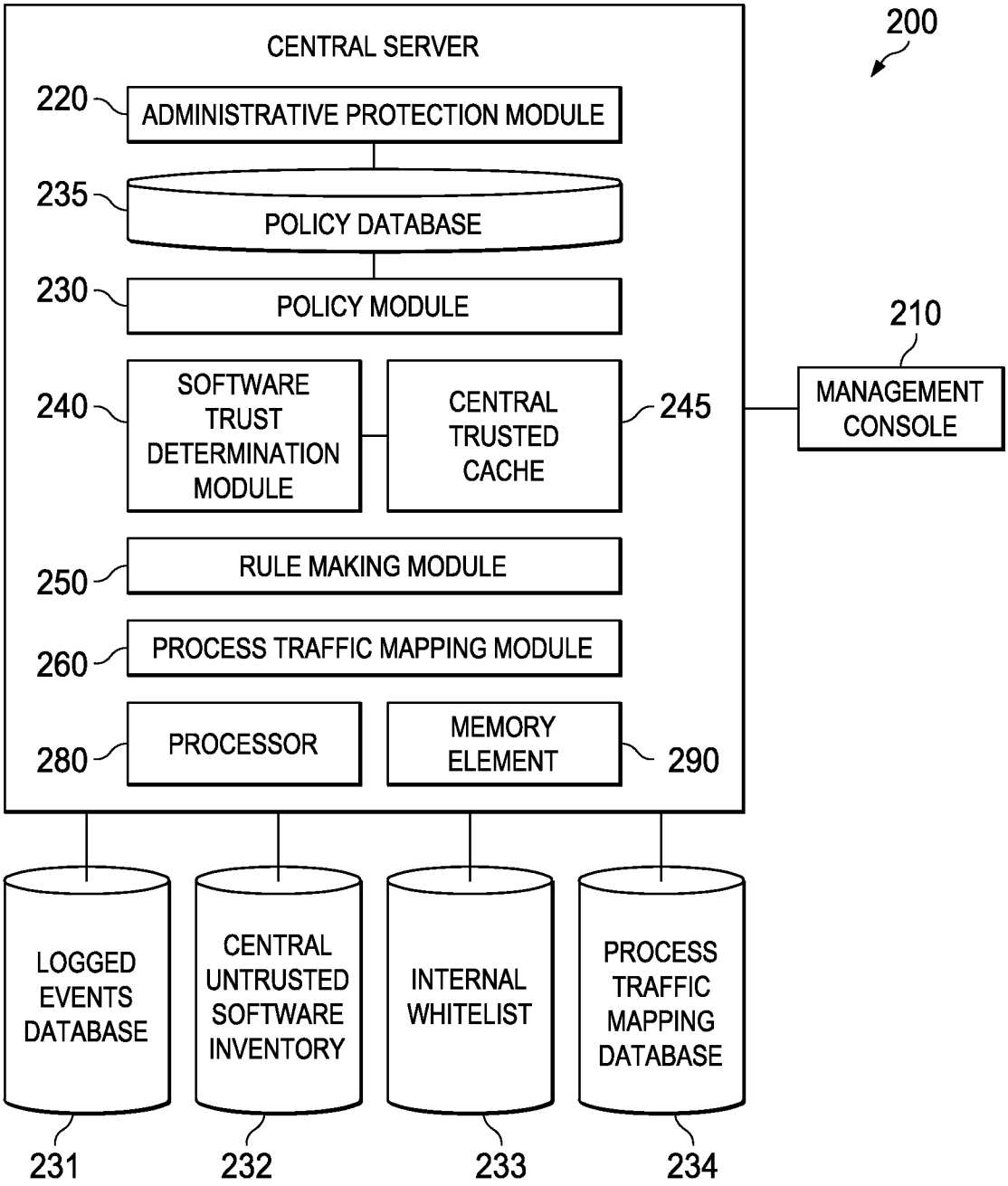

BOTNET OPERATOR
175
150
INTERNET
COMMAND AND CONTROL SERVER
170
GLOBAL SERVER
160
GLOBAL WHITELIST
165
100
110
NETWORK PROTECTION DEVICE
145
NETWORK LEVEL ENFORCEMENT COMPONENTS
140
CENTRAL SERVER
130
NETWORK LEVEL PROTECTION COMPONENTS
135
PROCESS TRAFFIC MAPPING DATABASE
134
INTERNAL WHITELIST
133
CENTRAL UNTRUSTED SOFTWARE INVENTORY
132
180
SECONDARY SERVER
LOGGED EVENTS DATABASE
131
HOST 1
EXECUTABLE SOFTWARE
122
LOCAL PROTECTION COMPONENTS
124
120a
HOST 2
120b
120c
HOST m

FIG. 2


200
CENTRAL SERVER
220
ADMINISTRATIVE PROTECTION MODULE
235
POLICY DATABASE
230
POLICY MODULE
240
SOFTWARE TRUST DETERMINATION MODULE
CENTRAL TRUSTED CACHE
245
210
MANAGEMENT CONSOLE
250
RULE MAKING MODULE
260
PROCESS TRAFFIC MAPPING MODULE
280
PROCESSOR
MEMORY ELEMENT
290
LOGGED EVENTS DATABASE
CENTRAL UNTRUSTED SOFTWARE INVENTORY
INTERNAL WHITELIST
PROCESS TRAFFIC MAPPING DATABASE
231
232
233
234

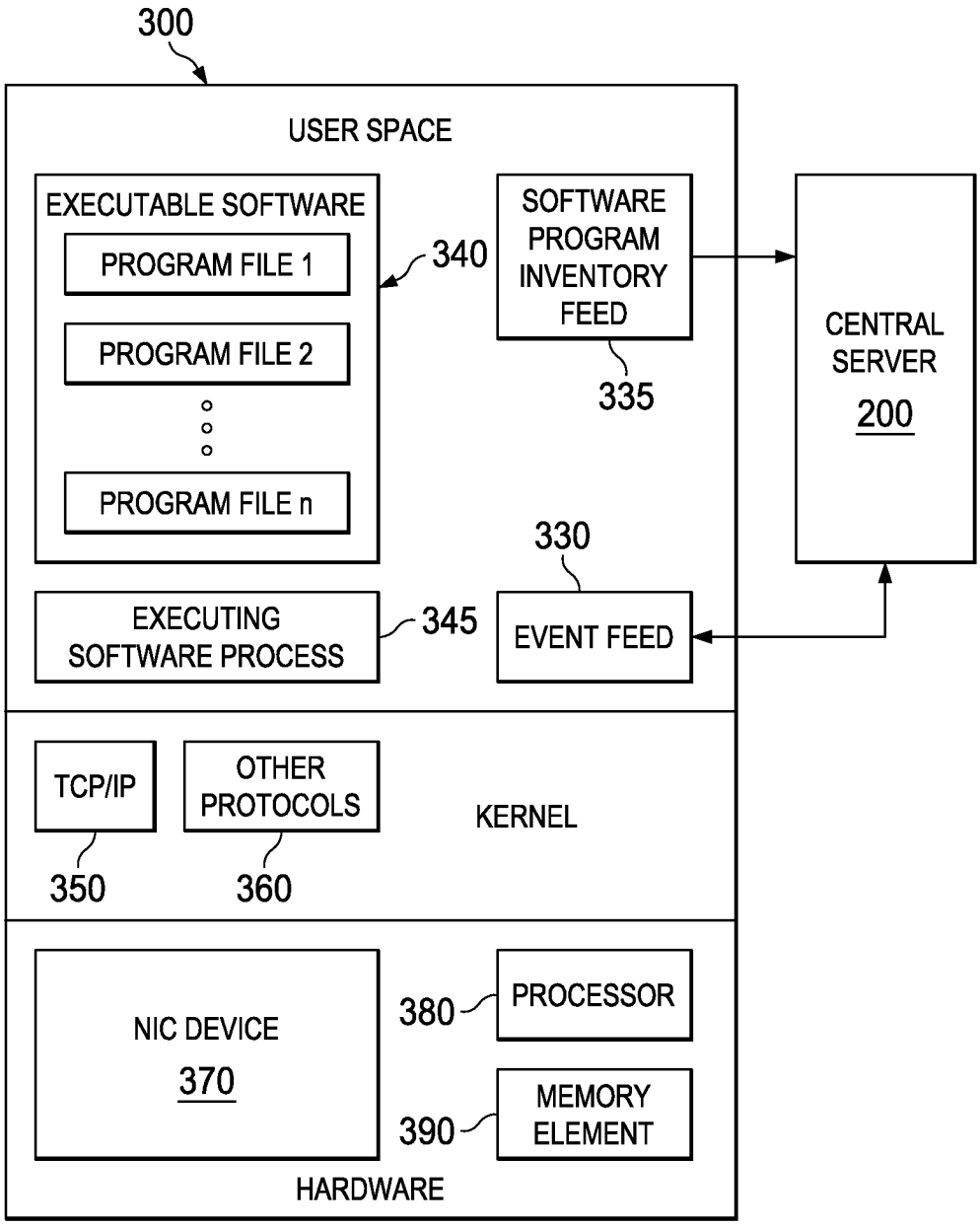

300
USER SPACE
EXECUTABLE SOFTWARE
PROGRAM FILE 1
PROGRAM FILE 2
PROGRAM FILE n
340
SOFTWARE PROGRAM INVENTORY FEED
335
CENTRAL SERVER
200
330
EXECUTING SOFTWARE PROCESS
345
EVENT FEED
TCP/IP
OTHER PROTOCOLS
KERNEL
350
360
NIC DEVICE
370
380
PROCESSOR
390
MEMORY ELEMENT
HARDWARE

FIG. 3

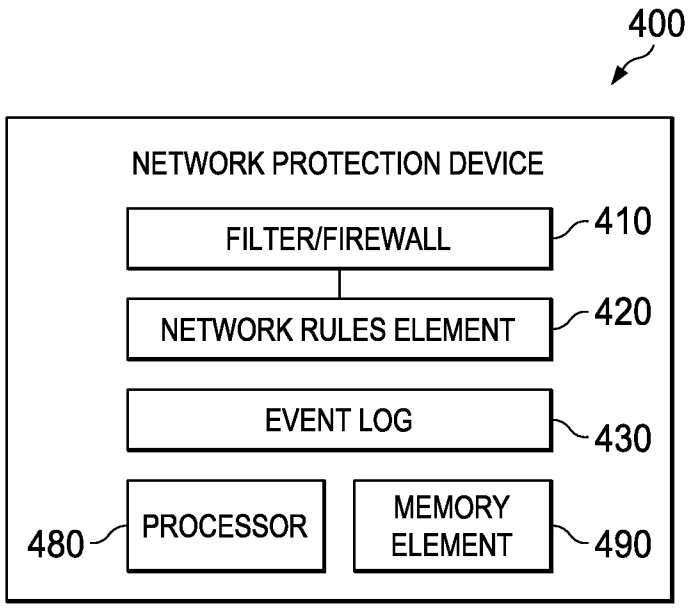

400
NETWORK PROTECTION DEVICE
FILTER/FIREWALL
410
NETWORK RULES ELEMENT
420
EVENT LOG
430
480
PROCESSOR
MEMORY ELEMENT
490

FIG. 4

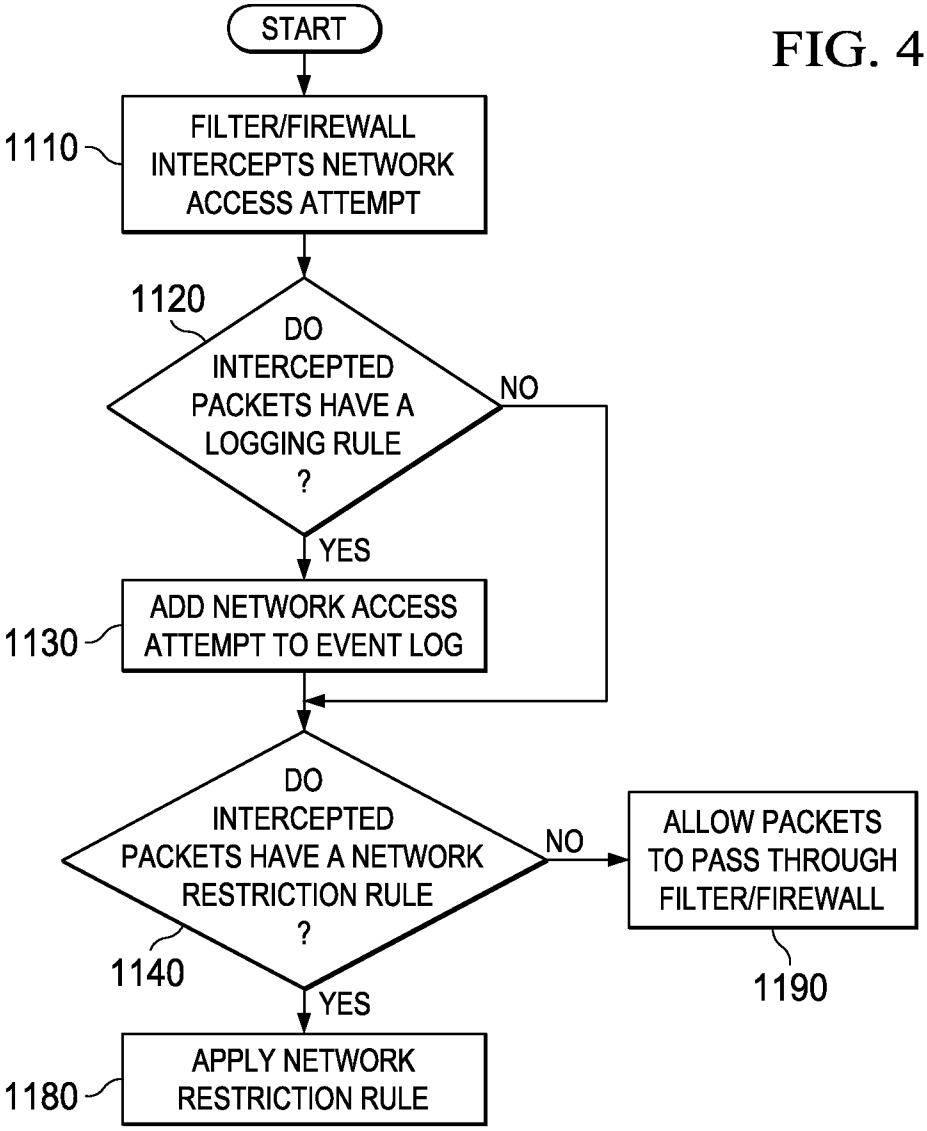

START
1110
FILTER/FIREWALL INTERCEPTS NETWORK ACCESS ATTEMPT
1120
DO INTERCEPTED PACKETS HAVE A LOGGING RULE ?
NO
YES
1130
ADD NETWORK ACCESS ATTEMPT TO EVENT LOG
DO INTERCEPTED PACKETS HAVE A NETWORK RESTRICTION RULE ?
NO
ALLOW PACKETS TO PASS THROUGH FILTER/FIREWALL
1190
1140
YES
1180
APPLY NETWORK RESTRICTION RULE

FIG. 11

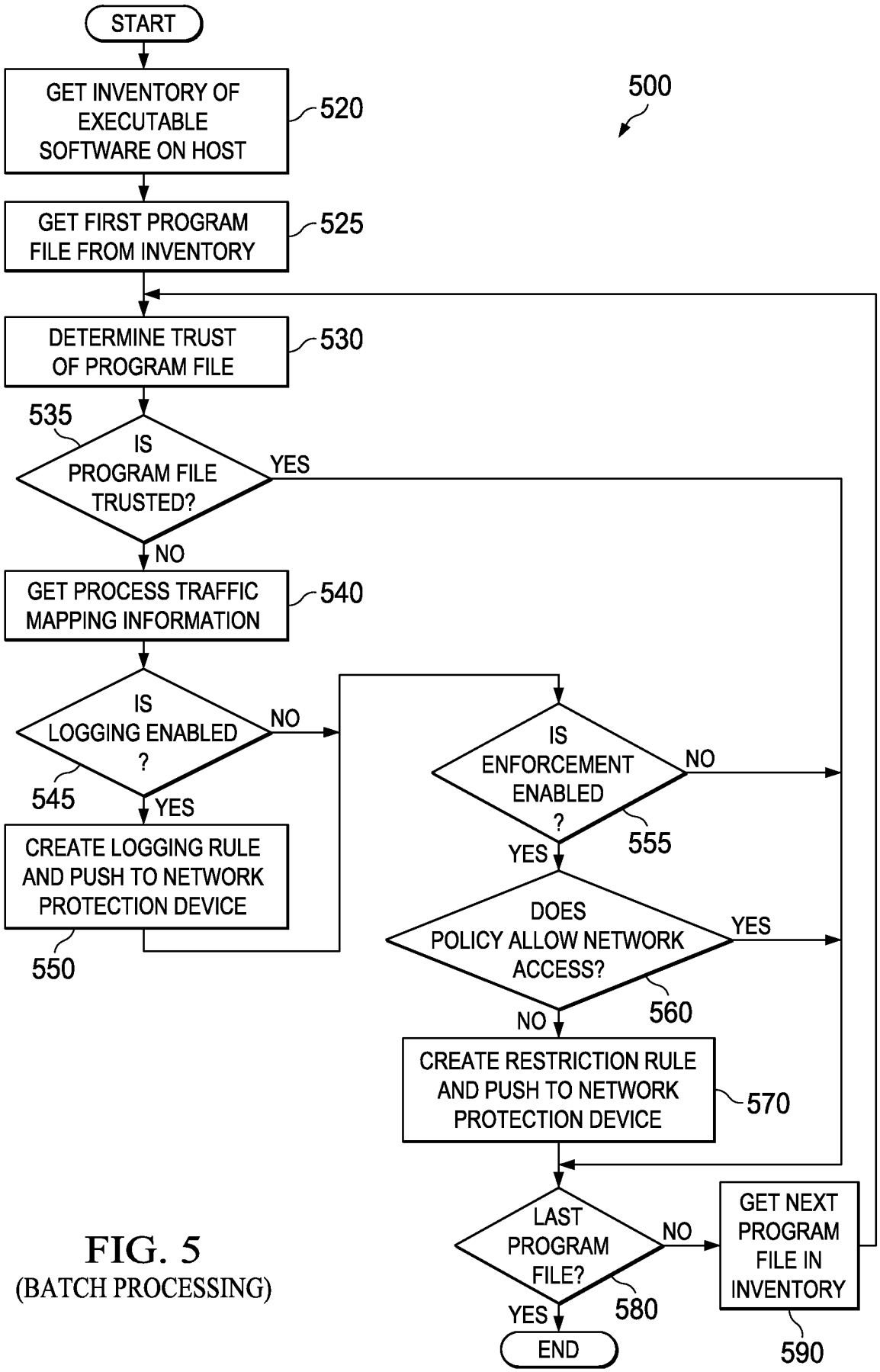


FIG. 5
(BATCH PROCESSING)

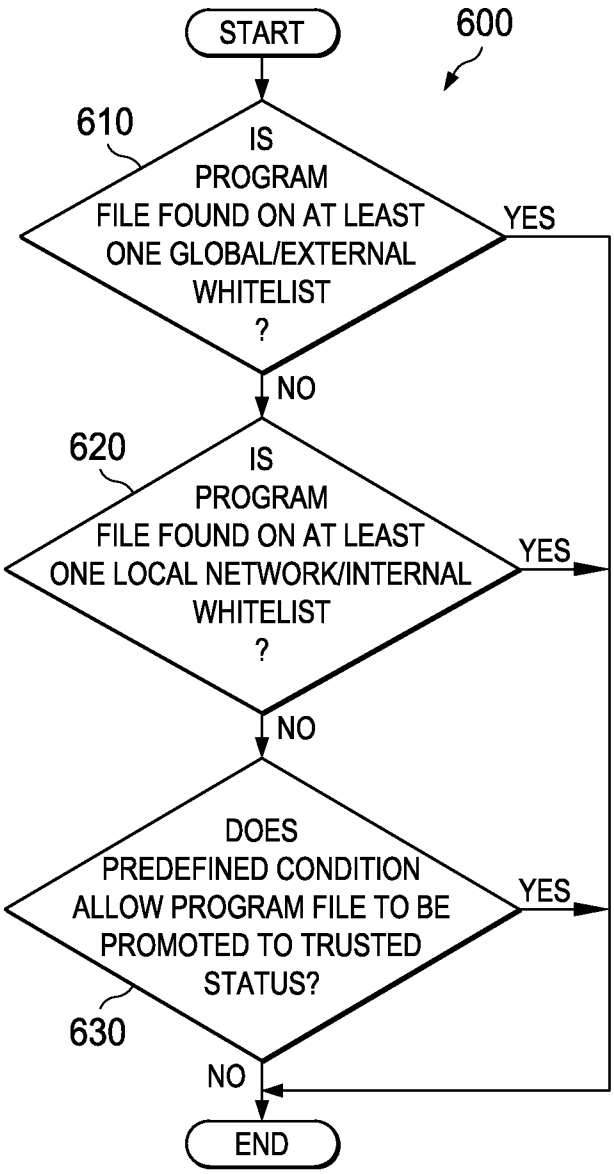

600
START
610
IS PROGRAM FILE FOUND ON AT LEAST ONE GLOBAL/EXTERNAL WHITELIST ?
YES
NO
620
IS PROGRAM FILE FOUND ON AT LEAST ONE LOCAL NETWORK/INTERNAL WHITELIST ?
YES
NO
DOES PREDEFINED CONDITION ALLOW PROGRAM FILE TO BE PROMOTED TO TRUSTED STATUS?
630
YES
NO
END

FIG. 6

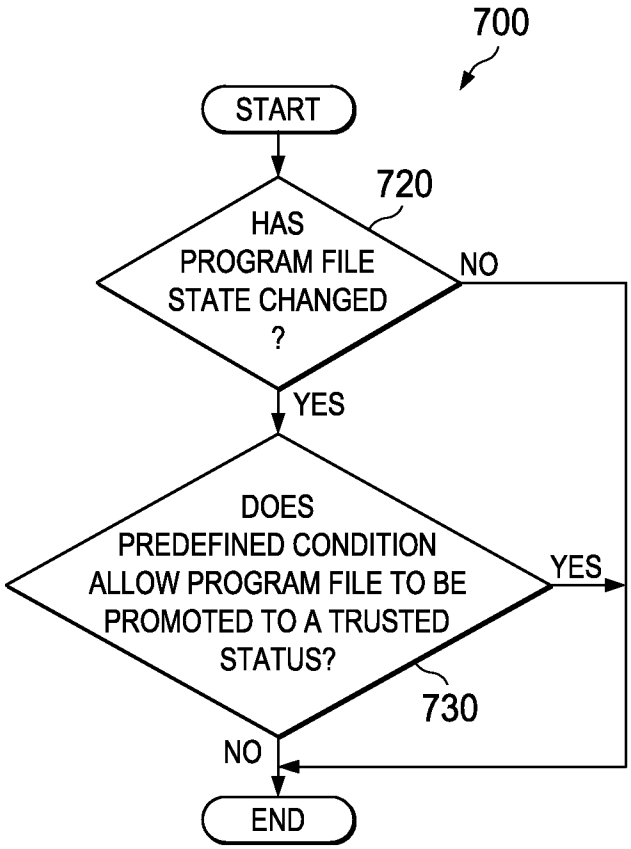

700
START
720
HAS PROGRAM FILE STATE CHANGED ?
NO
YES
DOES PREDEFINED CONDITION ALLOW PROGRAM FILE TO BE PROMOTED TO A TRUSTED STATUS?
730
YES
NO
END

FIG. 7

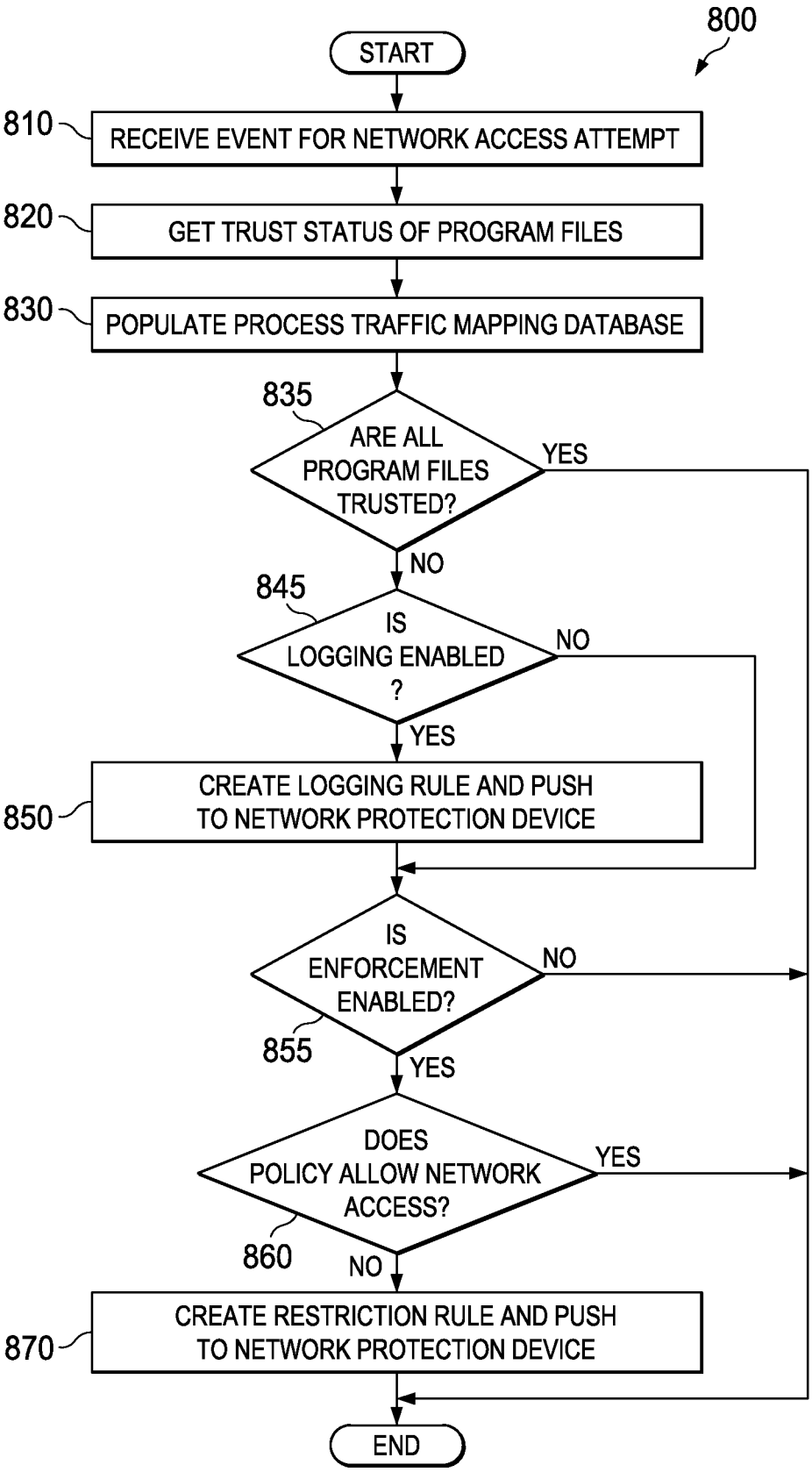


FIG. 8
(REAL-TIME)

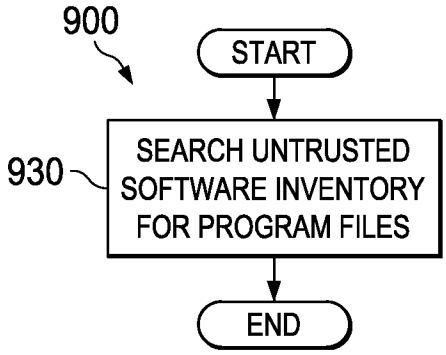


FIG. 9

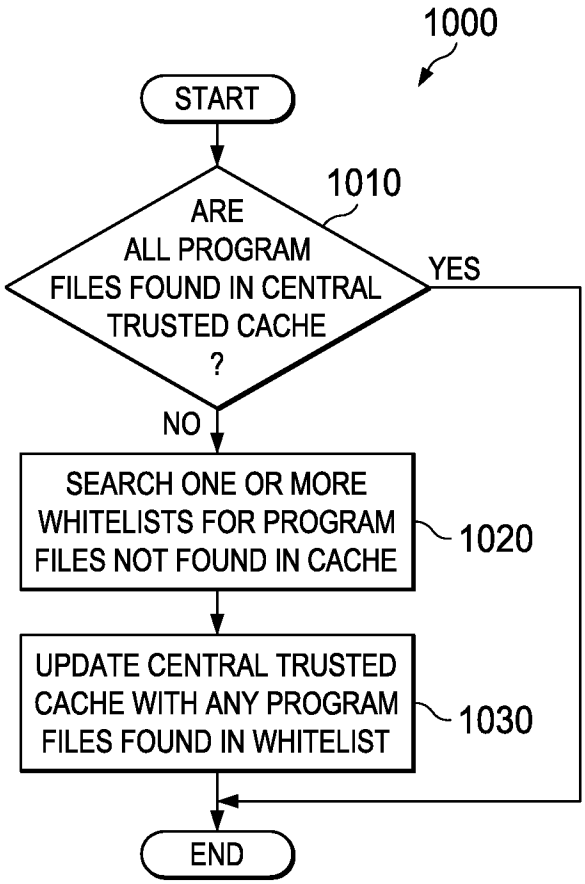


FIG. 10

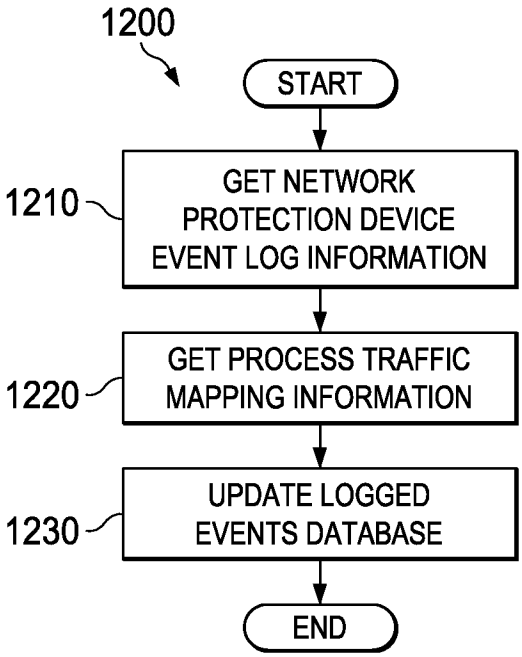


FIG. 12

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2011/024869

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/06
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)
H04L

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2006/124832 A1 (WEBSENSE INC [US]) 23 November 2006 (2006-11-23) paragraphs [0006], [0007], [0056]	1-33
X	US 7 069 330 B1 (MCARDLE MARK J [CA] ET AL) 27 June 2006 (2006-06-27) column 4, lines 60-65; claims 1,7,12	1-33

☐ 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	Date of mailing of the international search report
7 July 2011	14/07/2011

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

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2011/024869

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2006124832 A1	23-11-2006	AU 2006247382 A1	23-11-2006
		CA 2608077 A1	23-11-2006
		EP 1886243 A1	13-02-2008
		JP 2008546060 A	18-12-2008
US 7069330 B1	27-06-2006	NONE	