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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
06 October 2022 (06.10.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/208045 A1

(51) International Patent Classification:
H04L 9/08 (2006.01) *G06F 21/62* (2013.01)
G06F 21/60 (2013.01)

(21) International Application Number:
PCT/GB2022/050393

(22) International Filing Date:
14 February 2022 (14.02.2022)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/168,654 31 March 2021 (31.03.2021) US
17/467,733 07 September 2021 (07.09.2021) US

(71) Applicant: **SOPHOS LIMITED** [GB/GB]; The Pentagon, Abingdon Science Park, Abingdon Oxfordshire OX14 3YP (GB).

(72) Inventors: **LOMAN, Mark Willem**; c/o SurfRight B.V., Lansinkesweg 4, 7553 AE Hengelo (NL). **ENGELS, Lute Edwin**; c/o SurfRight B.V., Lansinkesweg 4, 7553 AE Hengelo (NL). **TIJINK, Ronny Henk Gert**; c/o SurfRight B.V., Lansinkesweg 4, 7553 AE Hengelo (NL). **VAN HILLO, Victor Marinus Johann Simon**; c/o SurfRight B.V., Lansinkesweg 4, 7553 AE Hengelo (NL). **VERMANING, Alexander**; c/o SurfRight B.V., Lansinkesweg 4, 7553 AE Hengelo (NL). **HARMSEN, Jeroen**; c/o SurfRight B.V., Lansinkesweg 4, 7553 AE Hengelo (NL).

(74) Agent: **WITHERS & ROGERS LLP**; 2 London Bridge, London Greater London SE1 9RA (GB).

(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, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK,

(54) Title: ENCRYPTED CACHE PROTECTION

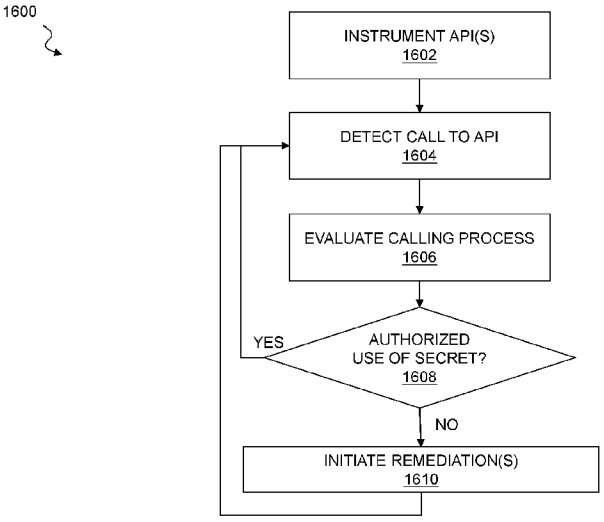


FIG. 16

(57) Abstract: Secrets such as secure session cookies for a web browser can be protected on a compute instance with multiple layers of encryption, such as by encrypting key material that in turn controls cryptographic access to the secret. A compute instance can be instrumented to detect when a process attempts to decrypt this key material so that the process requesting decryption can be compared to authorized or legitimate users of the secret.

[Continued on next page]

WO 2022/208045 A1

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, KM, 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))*

Published:

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

ENCRYPTED CACHE PROTECTION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Patent Application No. 17/467,733 filed on September 7, 2021, and U.S. Provisional Patent Application No. 63/168,654 filed on March 31, 2021, where the entire content of each of the foregoing is hereby incorporated by reference.

BACKGROUND

[0002] Computing devices often contain valuable information such as the credentials stored in a browser's cache, or corresponding cookies and other tokens used to automatically log in to websites. Malicious actors frequently target these potentially valuable assets in an attempt to lift login capabilities or control the cryptographic tools protecting them. There remains a need for improved techniques to detect and prevent attempted access to local secrets such as encrypted password and cookie caches.

SUMMARY

[0003] Secrets such as secure session cookies can be protected on a compute instance with multiple layers of encryption, such as by encrypting key material that in turn controls cryptographic access to the secret. A compute instance can be instrumented to detect when a process attempts to decrypt this key material so that the process requesting decryption can be compared to authorized or legitimate users of the secret.

[0004] In an aspect, a method disclosed herein may include: instrumenting an application programming interface on a compute instance to detect access to a decryption service used by an operating system of the compute instance; detecting a call from a first process executing on the compute instance to the application programming interface to unprotect a key used to cryptographically secure a secret on the compute instance; comparing first process information for the first process to second process information for one or more other processes associated with the secret; in response to determining that the first process is an authorized user of the secret, permitting the first process to decrypt the key for use in accessing the secret; and, in response to determining that the first process is not an authorized user of the secret, preventing the first process from decrypting the key for use in accessing the secret.

[0005] In an aspect, a compute instance disclosed herein may include: a memory in a user space of an operating system, the memory storing a first key encrypted with a master key

derived from user credentials for the compute instance and a secret encrypted with the first key; an application programming interface configured to provide programmatic access to cryptographic tools of the operating system based on the master key; and a security function hooked to the application programming interface, the security function configured to detect a request to decrypt the first key with the application programming interface, and to determine whether a process requesting decryption of the first key is an authorized user of the secret.

[0006] In an aspect, a computer program product disclosed herein may include computer executable code embodied in a computer readable medium that, when executing on one or more computing devices, performs the steps of: instrumenting a data protection application programming interface for an operating system on an endpoint to detect access to a decryption service used by the operating system to encrypt and decrypt data blobs using a master key derived from user credentials for the endpoint; detecting a call from a process executing on the endpoint to the data protection application programming interface to unprotect a symmetric key used to cryptographically secure a web browser session cookie stored by a web browser application on the endpoint; comparing first process information for the process to second process information for the web browser application that stored the web browser session cookie; and, in response to determining that the process is not associated with the web browser application, preventing the process from accessing the web browser session cookie with the key and initiating a remediation of the endpoint.

[0007] Implementations of any one or more of the aforementioned method, compute instance, and/or computer program product may include one or more of the following features. Techniques may further include, when the first process is not an authorized user, initiating a remediation. The remediation may include terminating the first process. The remediation may include quarantining the compute instance. The remediation may include performing a malware scan. The remediation may include generating a beacon identifying the first process. The remediation may include performing a root cause analysis. The first process information may include one or more of a process name, a process identifier, an application name, and a path. One or more of the other processes may include at least one process associated with a web browser application. One or more of the other processes may include a process for an authorized application that stored the secret. One or more of the other processes may include a process for an application including one or more of a privacy cleaner and a backup utility. The secret may include a web browser session cookie. The secret may include logon credentials stored in an

application cache and encrypted with the key. The secret may include one or more of a token, a cookie, a credential, and a cryptographic key. The decryption service may encrypt and decrypt using a master key derived from user credentials for the compute instance. The application programming interface may be a data protection application programming interface for the operating system. The application programming interface may access decryption resources in a kernel of the operating system. The secret may be a web browser session cookie and the security function may be configured to detect whether the process is associated with a web browser that stored the web browser session cookie.

BRIEF DESCRIPTION OF THE FIGURES

[0008] The foregoing and other objects, features and advantages of the devices, systems, and methods described herein will be apparent from the following description of particular embodiments thereof, as illustrated in the accompanying drawings. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the devices, systems, and methods described herein.

[0009] Fig. 1 illustrates an environment for threat management.

[0010] Fig. 2 illustrates a computer system.

[0011] Fig. 3 illustrates a threat management system.

[0012] Fig. 4 illustrates a system for behavioral tracking, coloring, and generation of indications of compromise (IOCs).

[0013] Fig. 5 illustrates a system for encryption management.

[0014] Fig. 6 illustrates a threat management system using heartbeats.

[0015] Fig. 7 shows an architecture for endpoint protection in an enterprise network security system.

[0016] Fig. 8 illustrates a system for forensic analysis for computer processes.

[0017] Fig. 9 is a flowchart of a method for forensic analysis for computer processes.

[0018] Fig. 10 illustrates an event graph.

[0019] Fig. 11 shows an architecture for instrumenting an endpoint.

[0020] Fig. 12 shows a process for accessing encrypted information on an endpoint.

[0021] Fig. 13 illustrates malware attempting to access a decrypted key.

[0022] Fig. 14 illustrates malware attempting to access a decrypted key.

[0023] Fig. 15 illustrates malware attempting to access a decrypted key.

[0024] Fig. 16 shows a method for detecting access to an encrypted secret stored on a compute instance.

DETAILED DESCRIPTION

[0025] Embodiments will now be described with reference to the accompanying figures, in which preferred embodiments are shown. The foregoing may, however, be embodied in many different forms and should not be construed as limited to the illustrated embodiments set forth herein.

[0026] All documents mentioned herein are hereby incorporated by reference in their entirety. References to items in the singular should be understood to include items in the plural, and vice versa, unless explicitly stated otherwise or clear from the context. Grammatical conjunctions are intended to express any and all disjunctive and conjunctive combinations of conjoined clauses, sentences, words, and the like, unless otherwise stated or clear from the context. Thus, the term "or" should generally be understood to mean "and/or" and so forth.

[0027] Recitation of ranges of values herein are not intended to be limiting, referring instead individually to any and all values falling within the range, unless otherwise indicated herein, and each separate value within such a range is incorporated into the specification as if it were individually recited herein. The words "about," "approximately," or the like, when accompanying a numerical value, are to be construed as indicating a deviation as would be appreciated by one of ordinary skill in the art to operate satisfactorily for an intended purpose. Ranges of values and/or numeric values are provided herein as examples only, and do not constitute a limitation on the scope of the described embodiments. The use of any and all examples, or exemplary language ("e.g.," "such as," or the like) provided herein, is intended merely to better illuminate the embodiments and does not pose a limitation on the scope of the embodiments or the claims. No language in the specification should be construed as indicating any unclaimed element as essential to the practice of the embodiments.

[0028] In the following description, it is understood that terms such as "first," "second," "third," "above," "below," and the like, are words of convenience and are not to be construed as implying a chronological order or otherwise limiting any corresponding element unless expressly state otherwise.

[0029] Fig. 1 illustrates an environment for threat management. Specifically, Fig.1 depicts a block diagram of a threat management system providing protection to an enterprise

against a plurality of threats – a context in which the following techniques may usefully be deployed. One aspect relates to corporate policy management and implementation through a unified threat management facility 100. As will be explained in more detail below, a threat management facility 100 may be used to protect computer assets from many threats, both computer-generated threats and user-generated threats. The threat management facility 100 may be multi-dimensional in that it may be designed to protect corporate assets from a variety of threats and it may be adapted to learn about threats in one dimension (e.g., worm detection) and apply the knowledge in another dimension (e.g., spam detection). Policy management is one of the dimensions for which the threat management facility can provide a control capability. A corporation or other entity may institute a policy that prevents certain people (e.g., employees, groups of employees, types of employees, guest of the corporation, etc.) from accessing certain types of computer programs. For example, the corporation may elect to prevent its accounting department from using a particular version of an instant messaging service or all such services. In this example, the policy management facility 112 may be used to update the policies of all corporate computing assets with a proper policy control facility or it may update a select few. By using the threat management facility 100 to facilitate the setting, updating and control of such policies the corporation only needs to be concerned with keeping the threat management facility 100 up to date on such policies. The threat management facility 100 can take care of updating all of the other corporate computing assets.

[0030] It should be understood that the threat management facility 100 may provide multiple services, and policy management may be offered as one of the services. We will now turn to a description of certain capabilities and components of the threat management system 100.

[0031] Over recent years, malware has become a major problem across the Internet 154. From both a technical perspective and a user perspective, the categorization of a specific threat type, whether as virus, worm, spam, phishing exploration, spyware, adware, or the like, is becoming reduced in significance. The threat, no matter how it is categorized, may need to be stopped at various points of a networked computing environment, such as one of an enterprise facility 102, including at one or more laptops, desktops, servers, gateways, communication ports, handheld or mobile devices, firewalls, and the like. Similarly, there may be less and less benefit to the user in having different solutions for known and unknown threats. As such, a consolidated threat management facility 100 may need to apply a similar set of technologies and

capabilities for all threats. In certain embodiments, the threat management facility 100 may provide a single agent on the desktop, and a single scan of any suspect file. This approach may eliminate the inevitable overlaps and gaps in protection caused by treating viruses and spyware as separate problems, while simultaneously simplifying administration and minimizing desktop load. As the number and range of types of threats has increased, so may have the level of connectivity available to all IT users. This may have led to a rapid increase in the speed at which threats may move. Today, an unprotected PC connected to the Internet 154 may be infected quickly (perhaps within 10 minutes) which may require acceleration for the delivery of threat protection. Where once monthly updates may have been sufficient, the threat management facility 100 may automatically and seamlessly update its product set against spam and virus threats quickly, for instance, every five minutes, every minute, continuously, or the like. Analysis and testing may be increasingly automated, and also may be performed more frequently; for instance, it may be completed in 15 minutes, and may do so without compromising quality. The threat management facility 100 may also extend techniques that may have been developed for virus and malware protection and provide them to enterprise facility 102 network administrators to better control their environments. In addition to stopping malicious code, the threat management facility 100 may provide policy management that may be able to control legitimate applications, such as VoIP, instant messaging, peer-to-peer file-sharing, and the like, that may undermine productivity and network performance within the enterprise facility 102.

[0032] The threat management facility 100 may provide an enterprise facility 102 protection from computer-based malware, including viruses, spyware, adware, Trojans, intrusion, spam, policy abuse, uncontrolled access, and the like, where the enterprise facility 102 may be any entity with a networked computer-based infrastructure. In an embodiment, Fig. 1 may depict a block diagram of the threat management facility 100 providing protection to an enterprise against a plurality of threats. The enterprise facility 102 may be corporate, commercial, educational, governmental, or the like, and the enterprise facility's 102 computer network may be distributed amongst a plurality of facilities, and in a plurality of geographical locations, and may include administration 134, a firewall 138A, an appliance 140A, server 142A, network devices 148A–B, clients 144A–D, such as protected by computer security facilities 152, and the like. It will be understood that any reference herein to client facilities may include the clients 144A–D shown in Fig. 1 and vice-versa. The threat management facility 100

may include a plurality of functions, such as security management facility 122, policy management facility 112, update facility 120, definitions facility 114, network access rules facility 124, remedial action facility 128, detection techniques facility 130, testing facility 118, threat research facility 132, and the like. In embodiments, the threat protection provided by the threat management facility 100 may extend beyond the network boundaries of the enterprise facility 102 to include clients 144D (or client facilities) that have moved into network connectivity not directly associated or controlled by the enterprise facility 102. Threats to client facilities may come from a plurality of sources, such as from network threats 104, physical proximity threats 110, secondary location threats 108, and the like. Clients 144A–D may be protected from threats even when the client 144A–D is not located in association with the enterprise 102, such as when a client 144E–F moves in and out of the enterprise facility 102, for example when interfacing with an unprotected server 142C through the Internet 154, when a client 144F is moving into a secondary location threat 108 such as interfacing with components 140B, 142B, 148C, 148D that are not protected, and the like. In embodiments, the threat management facility 100 may provide an enterprise facility 102 protection from a plurality of threats to multiplatform computer resources in a plurality of locations and network configurations, with an integrated system approach. It should be understood that an enterprise model is applicable to organizations and users of any size or type. For example, an enterprise may be or may include a group or association of endpoints, networks, users, and the like within or outside of one or more protected locations. It should be understood that an enterprise may include one or more offices or business locations, or one or more homes, where each location, or portions of each location, or a collection of locations may be treated as a client facility.

[0033] In embodiments, the threat management facility 100 may be provided as a stand-alone solution. In other embodiments, the threat management facility 100 may be integrated into a third-party product. An application programming interface (e.g., a source code interface) may be provided such that the threat management facility 100 may be integrated. For instance, the threat management facility 100 may be stand-alone in that it provides direct threat protection to an enterprise or computer resource, where protection is subscribed to directly 100. Alternatively, the threat management facility 100 may offer protection indirectly, through a third-party product, where an enterprise may subscribe to services through the third-party product, and threat protection to the enterprise may be provided by the threat management facility 100 through the third-party product.

[0034] The security management facility 122 may include a plurality of elements that provide protection from malware to enterprise facility 102 computer resources, including endpoint security and control, email security and control, web security and control, reputation-based filtering, control of unauthorized users, control of guest and non-compliant computers, and the like. The security management facility 122 may be a software application that may provide malicious code and malicious application protection to a client facility computing resource. The security management facility 122 may have the ability to scan the client facility files for malicious code, remove or quarantine certain applications and files, prevent certain actions, perform remedial actions and perform other security measures. In embodiments, scanning the client facility may include scanning some or all of the files stored to the client facility on a periodic basis, scanning an application when the application is executed, scanning files as the files are transmitted to or from the client facility, or the like. The scanning of the applications and files may be performed to detect known malicious code or known unwanted applications. In an embodiment, new malicious code and unwanted applications may be continually developed and distributed, and updates to the known code database may be provided on a periodic basis, on a demand basis, on an alert basis, or the like.

[0035] The security management facility 122 may provide email security and control, where security management may help to eliminate spam, viruses, spyware and phishing, control of email content, and the like. The security management facility's 122 email security and control may protect against inbound and outbound threats, protect email infrastructure, prevent data leakage, provide spam filtering, and the like. In an embodiment, security management facility 122 may provide for web security and control, where security management may help to detect or block viruses, spyware, malware, unwanted applications, help control web browsing, and the like, which may provide comprehensive web access control enabling safe, productive web browsing. Web security and control may provide Internet use policies, reporting on suspect devices, security and content filtering, active monitoring of network traffic, URI filtering, and the like. In an embodiment, the security management facility 122 may provide for network access control, which may provide control over network connections. Network control may stop unauthorized, guest, or non-compliant systems from accessing networks, and may control network traffic that may not be bypassed from the client level. In addition, network access control may control access to virtual private networks (VPN), where VPNs may be a communications network tunneled through another network, establishing a logical connection

acting as a virtual network. In embodiments, a VPN may be treated in the same manner as a physical network.

[0036] The security management facility 122 may provide host intrusion prevention through behavioral based protection, which may guard against unknown threats by analyzing behavior before software code executes. Behavioral based protection may monitor code when it runs and intervene if the code is deemed to be suspicious or malicious. Advantages of behavioral based protection over runtime protection may include code being prevented from running. Whereas runtime protection may only interrupt code that has already partly executed, behavioral protection can identify malicious code at the gateway or on the file servers and delete the code before it can reach endpoint computers and the like.

[0037] The security management facility 122 may provide reputation filtering, which may target or identify sources of known malware. For instance, reputation filtering may include lists of URIs of known sources of malware or known suspicious IP addresses, or domains, say for spam, that when detected may invoke an action by the threat management facility 100, such as dropping them immediately. By dropping the source before any interaction can initiate, potential threat sources may be thwarted before any exchange of data can be made.

[0038] In embodiments, information may be sent from the enterprise back to a third party, a vendor, or the like, which may lead to improved performance of the threat management facility 100. For example, the types, times, and number of virus interactions that a client experiences may provide useful information for the preventions of future virus threats. This type of feedback may be useful for any aspect of threat detection. Feedback of information may also be associated with behaviors of individuals within the enterprise, such as being associated with most common violations of policy, network access, unauthorized application loading, unauthorized external device use, and the like. In embodiments, this type of information feedback may enable the evaluation or profiling of client actions that are violations of policy that may provide a predictive model for the improvement of enterprise policies.

[0039] The security management facility 122 may support overall security of the enterprise facility 102 network or set of enterprise facility 102 networks, e.g., by providing updates of malicious code information to the enterprise facility 102 network and associated client facilities. The updates may include a planned update, an update in reaction to a threat notice, an update in reaction to a request for an update, an update based on a search of known malicious code information, or the like. The administration facility 134 may provide control

over the security management facility 122 when updates are performed. The updates may be automatically transmitted without an administration facility's 134 direct control, manually transmitted by the administration facility 134, or otherwise distributed. The security management facility 122 may manage the receipt of malicious code descriptions from a provider, distribution of the malicious code descriptions to enterprise facility 102 networks, distribution of the malicious code descriptions to client facilities, and so forth.

[0040] The threat management facility 100 may provide a policy management facility 112 that may be able to block non-malicious applications, such as VoIP, instant messaging, peer-to-peer file-sharing, and the like, that may undermine productivity and network performance within the enterprise facility 102. The policy management facility 112 may be a set of rules or policies that may indicate enterprise facility 102 access permissions for the client facility, such as access permissions associated with the network, applications, external computer devices, and the like. The policy management facility 112 may include a database, a text file, a combination of databases and text files, or the like. In an embodiment, a policy database may be a block list, a black list, an allowed list, a white list, or the like that may provide a list of enterprise facility 102 external network locations/applications that may or may not be accessed by the client facility. The policy management facility 112 may include rules that may be interpreted with respect to an enterprise facility 102 network access request to determine if the request should be allowed. The rules may provide a generic rule for the type of access that may be granted. The rules may be related to the policies of an enterprise facility 102 for access rights for the enterprise facility's 102 client facility. For example, there may be a rule that does not permit access to sporting websites. When a website is requested by the client facility, a security facility may access the rules within a policy facility to determine if the requested access is related to a sporting website. In an embodiment, the security facility may analyze the requested website to determine if the website matches with any of the policy facility rules.

[0041] The policy management facility 112 may be similar to the security management facility 122 but with the addition of enterprise facility 102 wide access rules and policies that may be distributed to maintain control of client facility access to enterprise facility 102 network resources. The policies may be defined for application type, subset of application capabilities, organization hierarchy, computer facility type, user type, network location, time of day, connection type, or the like. Policies may be maintained by the administration facility 134, through the threat management facility 100, in association with a third party, or the like. For

example, a policy may restrict IM activity to only support personnel for communicating with customers. This may allow communication for departments requiring access, but may maintain the network bandwidth for other activities by restricting the use of IM to only the personnel that need access to instant messaging (IM) in support of the enterprise facility 102. In an embodiment, the policy management facility 112 may be a stand-alone application, may be part of the network server facility 142, may be part of the enterprise facility 102 network, may be part of the client facility, or the like.

[0042] The threat management facility 100 may provide configuration management, which may be similar to policy management, but may specifically examine the configuration set of applications, operating systems, hardware, and the like, and manage changes to their configurations. Assessment of a configuration may be made against a standard configuration policy, detection of configuration changes, remediation of improper configuration, application of new configurations, and the like. An enterprise may keep a set of standard configuration rules and policies which may represent the desired state of the device. For example, a client firewall may be running and installed, but in the disabled state, where remediation may be to enable the firewall. In another example, the enterprise may set a rule that disallows the use of USB disks, and sends a configuration change to all clients, which turns off USB drive access via a registry.

[0043] The threat management facility 100 may also provide for the removal of applications that potentially interfere with the operation of the threat management facility 100, such as competitor products that may also be attempting similar threat management functions. The removal of such products may be initiated automatically whenever such products are detected. In the case where such applications are services are provided indirectly through a third-party product, the application may be suspended until action is taken to remove or disable the third-party product's protection facility.

[0044] Threat management against a quickly evolving malware environment may require timely updates, and thus an update management facility 120 may be provided by the threat management facility 100. In addition, a policy management facility 112 may also require update management (e.g., as provided by the update facility 120 herein described). The update management for the security facility 122 and policy management facility 112 may be provided directly by the threat management facility 100, such as by a hosted system or in conjunction with the administration facility 134. In embodiments, the threat management facility 100 may provide for patch management, where a patch may be an update to an operating system, an

application, a system tool, or the like, where one of the reasons for the patch is to reduce vulnerability to threats.

[0045] The security facility 122 and policy management facility 112 may push information to the enterprise facility 102 network and/or client facility. The enterprise facility 102 network and/or client facility may also or instead pull information from the security facility 122 and policy management facility 112 network server facilities 142, or there may be a combination of pushing and pulling of information between the security facility 122 and the policy management facility 112 network servers 142, enterprise facility 102 network, and client facilities, or the like. For example, the enterprise facility 102 network and/or client facility may pull information from the security facility 122 and policy management facility 112 network server facility 142 may request the information using the security facility 122 and policy management facility 112 update module; the request may be based on a certain time period, by a certain time, by a date, on demand, or the like. In another example, the security facility 122 and policy management facility 112 network servers 142 may push the information to the enterprise facility's 102 network and/or client facility by providing notification that there are updates available for download and then transmitting the information. The combination of the security management 122 network server facility 142 and security update module may function substantially the same as the policy management facility 112 network server and policy update module by providing information to the enterprise facility 102 network and the client facility in a push or pull method. In an embodiment, the policy management facility 112 and the security facility 122 management update modules may work in concert to provide information to the enterprise facility's 102 network and/or client facility for control of application execution. In an embodiment, the policy update module and security update module may be combined into a single update module.

[0046] As threats are identified and characterized, the threat management facility 100 may create definition updates that may be used to allow the threat management facility 100 to detect and remediate the latest malicious software, unwanted applications, configuration and policy changes, and the like. The threat definition facility 114 may contain threat identification updates, also referred to as definition files. A definition file may be a virus identity file that may include definitions of known or potential malicious code. The virus identity (IDE) definition files may provide information that may identify malicious code within files, applications, or the like. The definition files may be accessed by security management facility 122 when scanning

files or applications within the client facility for the determination of malicious code that may be within the file or application. The definition files may contain a number of commands, definitions, or instructions, to be parsed and acted upon, or the like. In embodiments, the client facility may be updated with new definition files periodically to provide the client facility with the most recent malicious code definitions; the updating may be performed on a set time period, may be updated on demand from the client facility, may be updated on demand from the network, may be updated on a received malicious code alert, or the like. In an embodiment, the client facility may request an update to the definition files from an update facility 120 within the network, may request updated definition files from a computing facility external to the network, updated definition files may be provided to the client facility 114 from within the network, definition files may be provided to the client facility from an external computing facility from an external network, or the like.

[0047] A definition management facility 114 may provide timely updates of definition files information to the network, client facilities, and the like. New and altered malicious code and malicious applications may be continually created and distributed to networks worldwide. The definition files that maintain the definitions of the malicious code and malicious application information for the protection of the networks and client facilities may need continual updating to provide continual defense of the network and client facility from the malicious code and malicious applications. The definition files management may provide for automatic and manual methods of updating the definition files. In embodiments, the network may receive definition files and distribute the definition files to the network client facilities, the client facilities may receive the definition files directly, or the network and client facilities may both receive the definition files, or the like. In an embodiment, the definition files may be updated on a fixed periodic basis, on demand by the network and/or the client facility, as a result of an alert of a new malicious code or malicious application, or the like. In an embodiment, the definition files may be released as a supplemental file to an existing definition files to provide for rapid updating of the definition files.

[0048] In a similar manner, the security management facility 122 may be used to scan an outgoing file and verify that the outgoing file is permitted to be transmitted per the enterprise facility 102 rules and policies. By checking outgoing files, the security management facility 122 may be able discover malicious code infected files that were not detected as incoming files as a result of the client facility having been updated with either new definition files or policy

management facility 112 information. The definition files may discover the malicious code infected file by having received updates of developing malicious code from the administration facility 134, updates from a definition files provider, or the like. The policy management facility 112 may discover the malicious code infected file by having received new updates from the administration facility 134, from a rules provider, or the like.

[0049] The threat management facility 100 may provide controlled access to the enterprise facility 102 networks. For instance, a manager of the enterprise facility 102 may want to restrict access to certain applications, networks, files, printers, servers, databases, or the like. In addition, the manager of the enterprise facility 102 may want to restrict user access based on certain criteria, such as the user's location, usage history, need to know, job position, connection type, time of day, method of authentication, client-system configuration, or the like. Network access rules may be developed for the enterprise facility 102, or pre-packaged by a supplier, and managed by the threat management facility 100 in conjunction with the administration facility 134.

[0050] A network access rules facility 124 may be responsible for determining if a client facility application should be granted access to a requested network location. The network location may be on the same network as the facility or may be on another network. In an embodiment, the network access rules facility 124 may verify access rights for client facilities from within the network or may verify access rights of computer facilities from external networks. When network access for a client facility is denied, the network access rules facility 124 may send an information file to the client facility containing. For example, the information sent by the network access rules facility 124 may be a data file. The data file may contain a number of commands, definitions, instructions, or the like to be parsed and acted upon through the remedial action facility 128, or the like. The information sent by the network access facility rules facility 124 may be a command or command file that the remedial action facility 128 may access and take action upon.

[0051] The network access rules facility 124 may include databases such as a block list, a black list, an allowed list, a white list, an unacceptable network site database, an acceptable network site database, a network site reputation database, or the like of network access locations that may or may not be accessed by the client facility. Additionally, the network access rules facility 124 may incorporate rule evaluation; the rule evaluation may parse network access requests and apply the parsed information to network access rules. The network access rule

facility 124 may have a generic set of rules that may be in support of an enterprise facility's 102 network access policies, such as denying access to certain types of websites, controlling instant messenger accesses, or the like. Rule evaluation may include regular expression rule evaluation, or other rule evaluation method for interpreting the network access request and comparing the interpretation to the established rules for network access. In an embodiment, the network access rules facility 124 may receive a rules evaluation request from the network access control and may return the rules evaluation to the network access control.

[0052] Similar to the threat definitions facility 114, the network access rule facility 124 may provide updated rules and policies to the enterprise facility 102. The network access rules facility 124 may be maintained by the network administration facility 134, using network access rules facility 124 management. In an embodiment, the network administration facility 134 may be able to maintain a set of access rules manually by adding rules, changing rules, deleting rules, or the like. Additionally, the administration facility 134 may retrieve predefined rule sets from a remote provider of a set of rules to be applied to an entire enterprise facility 102. The network administration facility 134 may be able to modify the predefined rules as needed for a particular enterprise facility 102 using the network access rules management facility 124.

[0053] When a threat or policy violation is detected by the threat management facility 100, the threat management facility 100 may perform or initiate a remedial action facility 128. Remedial action may take a plurality of forms, such as terminating or modifying an ongoing process or interaction, sending a warning to a client or administration facility 134 of an ongoing process or interaction, executing a program or application to remediate against a threat or violation, record interactions for subsequent evaluation, or the like. Remedial action may be associated with an application that responds to information that a client facility network access request has been denied. In an embodiment, when the data file is received, remedial action may parse the data file, interpret the various aspects of the data file, and act on the parsed data file information to determine actions to be taken on an application requesting access to a denied network location. In an embodiment, when the data file is received, remedial action may access the threat definitions to parse the data file and determine an action to be taken on an application requesting access to a denied network location. In an embodiment, the information received from the facility may be a command or a command file. The remedial action facility may carry out any commands that are received or parsed from a data file from the facility without performing any interpretation of the commands. In an embodiment, the remedial action facility

may interact with the received information and may perform various actions on a client requesting access to a denied network location. The action may be one or more of continuing to block all requests to a denied network location, a malicious code scan on the application, a malicious code scan on the client facility, quarantine of the application, terminating the application, isolation of the application, isolation of the client facility to a location within the network that restricts network access, blocking a network access port from a client facility, reporting the application to an administration facility 134, or the like.

[0054] Remedial action may be provided as a result of a detection of a threat or violation. The detection techniques facility 130 may include monitoring the enterprise facility 102 network or endpoint devices, such as by monitoring streaming data through the gateway, across the network, through routers and hubs, and the like. The detection techniques facility 130 may include monitoring activity and stored files on computing facilities, such as on server facilities 142, desktop computers, laptop computers, other mobile computing devices, and the like. Detection techniques, such as scanning a computer's stored files, may provide the capability of checking files for stored threats, either in the active or passive state. Detection techniques, such as streaming file management, may provide the capability of checking files received at the network, gateway facility, client facility, and the like. This may provide the capability of not allowing a streaming file or portions of the streaming file containing malicious code from entering the client facility, gateway facility, or network. In an embodiment, the streaming file may be broken into blocks of information, and a plurality of virus identities may be used to check each of the blocks of information for malicious code. In an embodiment, any blocks that are not determined to be clear of malicious code may not be delivered to the client facility, gateway facility, or network.

[0055] Verifying that the threat management facility 100 is detecting threats and violations to established policy, may require the ability to test the system, either at the system level or for a particular computing component. The testing facility 118 may allow the administration facility 134 to coordinate the testing of the security configurations of client facility computing facilities on a network. The administration facility 134 may be able to send test files to a set of client facility computing facilities to test the ability of the client facility to determine acceptability of the test file. After the test file has been transmitted, a recording facility may record the actions taken by the client facility in reaction to the test file. The recording facility may aggregate the testing information from the client facility and report the

testing information to the administration facility 134. The administration facility 134 may be able to determine the level of preparedness of the client facility computing facilities by the reported information. Remedial action may be taken for any of the client facility computing facilities as determined by the administration facility 134; remedial action may be taken by the administration facility 134 or by the user of the client facility.

[0056] The threat research facility 132 may provide a continuously ongoing effort to maintain the threat protection capabilities of the threat management facility 100 in light of continuous generation of new or evolved forms of malware. Threat research may include researchers and analysts working on known and emerging malware, such as viruses, rootkits a spyware, as well as other computer threats such as phishing, spam, scams, and the like. In embodiments, through threat research, the threat management facility 100 may be able to provide swift, global responses to the latest threats.

[0057] The threat management facility 100 may provide threat protection to the enterprise facility 102, where the enterprise facility 102 may include a plurality of networked components, such as client facility, server facility 142, administration facility 134, firewall 138, gateway, hubs, and routers 148, threat management appliance 140, desktop users, mobile users, and the like. In embodiments, it may be the endpoint computer security facility 152, located on a computer's desktop, which may provide threat protection to a user, and associated enterprise facility 102. In embodiments, the term endpoint may refer to a computer system that may source data, receive data, evaluate data, buffer data, or the like (such as a user's desktop computer as an endpoint computer), a firewall as a data evaluation endpoint computer system, a laptop as a mobile endpoint computer, a personal digital assistant or tablet as a hand-held endpoint computer, a mobile phone as an endpoint computer, or the like. In embodiments, endpoint may refer to a source or destination for data, including such components where the destination is characterized by an evaluation point for data, and where the data may be sent to a subsequent destination after evaluation. The endpoint computer security facility 152 may be an application loaded onto the computer platform or computer support component, where the application may accommodate the plurality of computer platforms and/or functional requirements of the component. For instance, a client facility computer may be one of a plurality of computer platforms, such as Windows, Macintosh, Linux, and the like, where the endpoint computer security facility 152 may be adapted to the specific platform, while maintaining a uniform product and product services across platforms. Additionally, components may have different

functions to serve within the enterprise facility's 102 networked computer-based infrastructure. For instance, computer support components provided as hubs and routers 148, server facility 142, firewalls 138, and the like, may require unique security application software to protect their portion of the system infrastructure, while providing an element in an integrated threat management system that extends out beyond the threat management facility 100 to incorporate all computer resources under its protection.

[0058] The enterprise facility 102 may include a plurality of client facility computing platforms on which the endpoint computer security facility 152 is adapted. A client facility computing platform may be a computer system that is able to access a service on another computer, such as a server facility 142, via a network. This client facility server facility 142 model may apply to a plurality of networked applications, such as a client facility connecting to an enterprise facility 102 application server facility 142, a web browser client facility connecting to a web server facility 142, an e-mail client facility retrieving e-mail from an Internet 154 service provider's mail storage servers 142, and the like. In embodiments, traditional large client facility applications may be switched to websites, which may increase the browser's role as a client facility. Clients 144 may be classified as a function of the extent to which they perform their own processing. For instance, client facilities are sometimes classified as a fat client facility or thin client facility. The fat client facility, also known as a thick client facility or rich client facility, may be a client facility that performs the bulk of data processing operations itself, and does not necessarily rely on the server facility 142. The fat client facility may be most common in the form of a personal computer, where the personal computer may operate independent of any server facility 142. Programming environments for fat clients 144 may include CURI, Delphi, Droplets, Java, win32, X11, and the like. Thin clients 144 may offer minimal processing capabilities, for instance, the thin client facility may primarily provide a graphical user interface provided by an application server facility 142, which may perform the bulk of any required data processing. Programming environments for thin clients 144 may include JavaScript/AJAX, ASP, JSP, Ruby on Rails, Python's Django, PHP, and the like. The client facility may also be a mix of the two, such as processing data locally, but relying on a server facility 142 for data storage. As a result, this hybrid client facility may provide benefits from both the fat client facility type, such as multimedia support and high performance, and the thin client facility type, such as high manageability and flexibility. In embodiments, the threat management facility 100, and associated endpoint computer security facility 152, may provide

seamless threat protection to the plurality of clients 144, and client facility types, across the enterprise facility 102.

[0059] The enterprise facility 102 may include a plurality of server facilities 142, such as application servers, communications servers, file servers, database servers, proxy servers, mail servers, fax servers, game servers, web servers, and the like. A server facility 142, which may also be referred to as a server facility 142 application, server facility 142 operating system, server facility 142 computer, or the like, may be an application program or operating system that accepts client facility connections in order to service requests from clients 144. The server facility 142 application may run on the same computer as the client facility using it, or the server facility 142 and the client facility may be running on different computers and communicating across the network. Server facility 142 applications may be divided among server facility 142 computers, with the dividing depending upon the workload. For instance, under light load conditions all server facility 142 applications may run on a single computer and under heavy load conditions a single server facility 142 application may run on multiple computers. In embodiments, the threat management facility 100 may provide threat protection to server facilities 142 within the enterprise facility 102 as load conditions and application changes are made.

[0060] A server facility 142 may also be an appliance facility 140, where the appliance facility 140 provides specific services onto the network. Though the appliance facility 140 is a server facility 142 computer, that may be loaded with a server facility 142 operating system and server facility 142 application, the enterprise facility 102 user may not need to configure it, as the configuration may have been performed by a third party. In an embodiment, an enterprise facility 102 appliance may be a server facility 142 appliance that has been configured and adapted for use with the threat management facility 100, and located within the facilities of the enterprise facility 102. The enterprise facility's 102 threat management appliance may enable the enterprise facility 102 to administer an on-site local managed threat protection configuration, where the administration facility 134 may access the threat resources through an interface, such as a web portal. In an alternate embodiment, the enterprise facility 102 may be managed remotely from a third party, vendor, or the like, without an appliance facility 140 located within the enterprise facility 102. In this instance, the appliance functionality may be a shared hardware product between pluralities of enterprises 102. In embodiments, the appliance facility 140 may be located at the enterprise facility 102, where the enterprise facility 102 maintains a degree of

control. In embodiments, a hosted service may be provided, where the appliance 140 may still be an on-site black box to the enterprise facility 102, physically placed there because of infrastructure requirements, but managed by a third party, vendor, or the like.

[0061] Simple server facility 142 appliances may also be utilized across the enterprise facility's 102 network infrastructure, such as switches, routers, wireless routers, hubs and routers, gateways, print servers, net modems, and the like. These simple server facility appliances may not require configuration by the enterprise facility 102, but may require protection from threats via an endpoint computer security facility 152. These appliances may provide interconnection services within the enterprise facility 102 network, and therefore may advance the spread of a threat if not properly protected.

[0062] A client facility may be protected from threats from within the enterprise facility 102 network using a personal firewall, which may be a hardware firewall, software firewall, or combination of these, that controls network traffic to and from a client. The personal firewall may permit or deny communications based on a security policy. Personal firewalls may be designed for use by end-users, which may result in protection for only the computer on which it's installed. Personal firewalls may be able to control network traffic by providing prompts each time a connection is attempted and adapting security policy accordingly. Personal firewalls may also provide some level of intrusion detection, which may allow the software to terminate or block connectivity where it suspects an intrusion is being attempted. Other features that may be provided by a personal firewall may include alerts about outgoing connection attempts, control of program access to networks, hiding the client from port scans by not responding to unsolicited network traffic, monitoring of applications that may be listening for incoming connections, monitoring and regulation of incoming and outgoing network traffic, prevention of unwanted network traffic from installed applications, reporting applications that make connection attempts, reporting destination servers with which applications may be attempting communications, and the like. In embodiments, the personal firewall may be provided by the threat management facility 100.

[0063] Another important component that may be protected by an endpoint computer security facility 152 is a network firewall facility 138, which may be a hardware or software device that may be configured to permit, deny, or proxy data through a computer network that has different levels of trust in its source of data. For instance, an internal enterprise facility 102 network may have a high level of trust, because the source of all data has been sourced from

within the enterprise facility 102. An example of a low level of trust is the Internet 154, because the source of data may be unknown. A zone with an intermediate trust level, situated between the Internet 154 and a trusted internal network, may be referred to as a "perimeter network." Since firewall facilities 138 represent boundaries between threat levels, the endpoint computer security facility 152 associated with the firewall facility 138 may provide resources that may control the flow of threats at this enterprise facility 102 network entry point. Firewall facilities 138, and associated endpoint computer security facility 152, may also be associated with a network node that may be equipped for interfacing between networks that use different protocols. In embodiments, the endpoint computer security facility 152 may provide threat protection in a plurality of network infrastructure locations, such as at the enterprise facility 102 network entry point, i.e., the firewall facility 138 or gateway; at the server facility 142; at distribution points within the network, i.e., the hubs and routers 148; at the desktop of client facility computers; and the like. In embodiments, the most effective location for threat detection may be at the user's computer desktop endpoint computer security facility 152.

[0064] The interface between the threat management facility 100 and the enterprise facility 102, and through the appliance facility 140 to embedded endpoint computer security facilities, may include a set of tools that may be the same for all enterprise implementations, but allow each enterprise to implement different controls. In embodiments, these controls may include both automatic actions and managed actions. Automatic actions may include downloads of the endpoint computer security facility 152 to components of the enterprise facility 102, downloads of updates to existing endpoint computer security facilities of the enterprise facility 102, uploaded network interaction requests from enterprise facility 102 components to the threat management facility 100, and the like. In embodiments, automatic interactions between the enterprise facility 102 and the threat management facility 100 may be configured by the threat management facility 100 and an administration facility 134 in the enterprise facility 102. The administration facility 134 may configure policy rules that determine interactions, such as developing rules for accessing applications, as in who is authorized and when applications may be used; establishing rules for ethical behavior and activities; rules governing the use of entertainment software such as games, or personal use software such as IM and VoIP; rules for determining access to enterprise facility 102 computing resources, including authentication, levels of access, risk assessment, and usage history tracking; rules for when an action is not allowed, such as whether an action is completely deigned or just modified in its execution; and

the like. The administration facility 134 may also establish license management, which in turn may further determine interactions associated with a licensed application. In embodiments, interactions between the threat management facility 100 and the enterprise facility 102 may provide threat protection to the enterprise facility 102 by managing the flow of network data into and out of the enterprise facility 102 through automatic actions that may be configured by the threat management facility 100 or the administration facility 134.

[0065] Client facilities within the enterprise facility 102 may be connected to the enterprise facility 102 network by way of wired network facilities 148A or wireless network facilities 148B. Client facilities connected to the enterprise facility 102 network via a wired facility 148A or wireless facility 148B may receive similar protection, as both connection types are ultimately connected to the same enterprise facility 102 network, with the same endpoint computer security facility 152, and the same threat protected enterprise facility 102 environment. Mobile wireless facility clients 144B–F, because of their ability to connect to any wireless 148B,D network access point, may connect to the Internet 154 outside the enterprise facility 102, and therefore outside the threat-protected environment of the enterprise facility 102. In this instance the mobile client facility (e.g., the clients 144 B–F), if not for the presence of the endpoint computer security facility 152 may experience a malware attack or perform actions counter to enterprise facility 102 established policies. In addition, there may be a plurality of ways for the threat management facility 100 to protect the out-of-enterprise facility 102 mobile client facility (e.g., the clients 144 D–F) that has an embedded endpoint computer security facility 152, such as by providing URI filtering in personal routers, using a web appliance as a DNS proxy, or the like. Mobile client facilities that are components of the enterprise facility 102 but temporarily outside connectivity with the enterprise facility 102 network may be provided with the same threat protection and policy control as client facilities inside the enterprise facility 102. In addition, mobile the client facilities may receive the same interactions to and from the threat management facility 100 as client facilities inside the enterprise facility 102, where the mobile client facilities may be considered a virtual extension of the enterprise facility 102, receiving all the same services via their embedded endpoint computer security facility 152.

[0066] Interactions between the threat management facility 100 and the components of the enterprise facility 102, including mobile client facility extensions of the enterprise facility 102, may ultimately be connected through the Internet 154. Threat management facility 100 downloads and upgrades to the enterprise facility 102 may be passed from the firewalled

networks of the threat management facility 100 through to the endpoint computer security facility 152 equipped components of the enterprise facility 102. In turn the endpoint computer security facility 152 components of the enterprise facility 102 may upload policy and access requests back across the Internet 154 and through to the threat management facility 100. The Internet 154 however, is also the path through which threats may be transmitted from their source. These network threats 104 may include threats from a plurality of sources, including without limitation, websites, e-mail, IM, VoIP, application software, and the like. These threats may attempt to attack a mobile enterprise client facility (e.g., the clients 144B–F) equipped with an endpoint computer security facility 152, but in embodiments, as long as the mobile client facility is embedded with an endpoint computer security facility 152, as described above, threats may have no better success than if the mobile client facility were inside the enterprise facility 102.

[0067] However, if the mobile client facility were to attempt to connect into an unprotected connection point, such as at a secondary location 108 that is not a part of the enterprise facility 102, the mobile client facility may be required to request network interactions through the threat management facility 100, where contacting the threat management facility 100 may be performed prior to any other network action. In embodiments, the client facility's 144 endpoint computer security facility 152 may manage actions in unprotected network environments such as when the client facility (e.g., client 144F) is in a secondary location 108 or connecting wirelessly to a non-enterprise facility 102 wireless Internet connection, where the endpoint computer security facility 152 may dictate what actions are allowed, blocked, modified, or the like. For instance, if the client facility's 144 endpoint computer security facility 152 is unable to establish a secured connection to the threat management facility 100, the endpoint computer security facility 152 may inform the user of such and recommend that the connection not be made. In the instance when the user chooses to connect despite the recommendation, the endpoint computer security facility 152 may perform specific actions during or after the unprotected connection is made, including running scans during the connection period, running scans after the connection is terminated, storing interactions for subsequent threat and policy evaluation, contacting the threat management facility 100 upon first instance of a secured connection for further actions and or scanning, restricting access to network and local resources, or the like. In embodiments, the endpoint computer security facility

152 may perform specific actions to remediate possible threat incursions or policy violations during or after the unprotected connection.

[0068] The secondary location 108 may have no endpoint computer security facilities 152 as a part of its computer components, such as its firewalls 138B, servers 142B, clients 144G, hubs and routers 148C–D, and the like. As a result, the computer components of the secondary location 108 may be open to threat attacks, and become potential sources of threats, as well as any mobile enterprise facility clients 144B–F that may be connected to the secondary location's 108 network. In this instance, these computer components may now unknowingly spread a threat to other components connected to the network.

[0069] Some threats may not come directly from the Internet 154, such as from non-enterprise facility controlled mobile devices that are physically brought into the enterprise facility 102 and connected to the enterprise facility 102 client facilities. The connection may be made from direct connection with the enterprise facility's 102 client facility, such as through a USB port, or in physical proximity with the enterprise facility's 102 client facility such that a wireless facility connection can be established, such as through a Bluetooth connection. These physical proximity threats 110 may be another mobile computing device, a portable memory storage device, a mobile communications device, or the like, such as CDs and DVDs, memory sticks, flash drives, external hard drives, cell phones, PDAs, MP3 players, digital cameras, point-to-point devices, digital picture frames, digital pens, navigation devices, tablets, appliances, and the like. A physical proximity threat 110 may have been previously infiltrated by network threats while connected to an unprotected network connection outside the enterprise facility 102, and when connected to the enterprise facility 102 client facility, pose a threat. Because of their mobile nature, physical proximity threats 110 may infiltrate computing resources in any location, such as being physically brought into the enterprise facility 102 site, connected to an enterprise facility 102 client facility while that client facility is mobile, plugged into an unprotected client facility at a secondary location 108, and the like. A mobile device, once connected to an unprotected computer resource, may become a physical proximity threat 110. In embodiments, the endpoint computer security facility 152 may provide enterprise facility 102 computing resources with threat protection against physical proximity threats 110, for instance, through scanning the device prior to allowing data transfers, through security validation certificates, through establishing a safe zone within the enterprise facility 102 computing resource to transfer data into for evaluation, and the like.

[0070] Having provided an overall context for threat detection, the description now turns to a brief discussion of an example of a computer system that may be used for any of the entities and facilities described above.

[0071] Fig. 2 illustrates a computer system. In general, the computer system 200 may include a computing device 210 connected to a network 202, e.g., through an external device 204. The computing device 210 may be or include any type of network endpoint or endpoints as described herein, e.g., with reference to Fig. 1 above. For example, the computing device 210 may include a desktop computer workstation. The computing device 210 may also or instead be any suitable device that has processes and communicates over a network 202, including without limitation a laptop computer, a desktop computer, a personal digital assistant, a tablet, a mobile phone, a television, a set top box, a wearable computer (e.g., watch, jewelry, or clothing), a home device (e.g., a thermostat or a home appliance controller), just as some examples. The computing device 210 may also or instead include a server, or it may be disposed on a server.

[0072] The computing device 210 may be used for any of the entities described in the threat management environment described above with reference to Fig. 1. For example, the computing device 210 may be a server, a client an enterprise facility, a threat management facility, or any of the other facilities or computing devices described therein. In certain aspects, the computing device 210 may be implemented using hardware (e.g., in a desktop computer), software (e.g., in a virtual machine or the like), or a combination of software and hardware (e.g., with programs executing on the desktop computer), and the computing device 210 may be a standalone device, a device integrated into another entity or device, a platform distributed across multiple entities, or a virtualized device executing in a virtualization environment.

[0073] The network 202 may include any network described above, e.g., data network(s) or internetwork(s) suitable for communicating data and control information among participants in the computer system 200. This may include public networks such as the Internet, private networks, and telecommunications networks such as the Public Switched Telephone Network or cellular networks using third generation cellular technology (e.g., 3G or IMT-2000), fourth generation cellular technology (e.g., 4G, LTE. MT-Advanced, E-UTRA, etc.) or WiMax-Advanced (IEEE 802.16m)) and/or other technologies, as well as any of a variety of corporate area, metropolitan area, campus or other local area networks or enterprise networks, along with any switches, routers, hubs, gateways, and the like that might be used to carry data among

participants in the computer system 200. The network 202 may also include a combination of data networks and need not be limited to a strictly public or private network.

[0074] The external device 204 may be any computer or other remote resource that connects to the computing device 210 through the network 202. This may include threat management resources such as any of those contemplated above, gateways or other network devices, remote servers or the like containing content requested by the computing device 210, a network storage device or resource, a device hosting malicious content, or any other resource or device that might connect to the computing device 210 through the network 202.

[0075] The computing device 210 may include a processor 212, a memory 214, a network interface 216, a data store 218, and one or more input/output devices 220. The computing device 210 may further include or be in communication with peripherals 222 and other external input/output devices 224.

[0076] The processor 212 may be any as described herein, and in general be capable of processing instructions for execution within the computing device 210 or computer system 200. The processor 212 may include a single-threaded processor or a multi-threaded processor. The processor 212 may be capable of processing instructions stored in the memory 214 or on the data store 218.

[0077] The memory 214 may store information within the computing device 210 or computer system 200. The memory 214 may include any volatile or non-volatile memory or other computer-readable medium, including without limitation a Random-Access Memory (RAM), a flash memory, a Read Only Memory (ROM), a Programmable Read-only Memory (PROM), an Erasable PROM (EPROM), registers, and so forth. The memory 214 may store program instructions, program data, executables, and other software and data useful for controlling operation of the computing device 200 and configuring the computing device 200 to perform functions for a user. The memory 214 may include a number of different stages and types for different aspects of operation of the computing device 210. For example, a processor may include on-board memory and/or cache for faster access to certain data or instructions, and a separate, main memory or the like may be included to expand memory capacity as desired.

[0078] The memory 214 may, in general, include a non-volatile computer readable medium containing computer code that, when executed by the computing device 200 creates an execution environment for a computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a

combination of the foregoing, and/or code that performs some or all of the steps set forth in the various flow charts and other algorithmic descriptions set forth herein. While a single memory 214 is depicted, it will be understood that any number of memories may be usefully incorporated into the computing device 210. For example, a first memory may provide non-volatile storage such as a disk drive for permanent or long-term storage of files and code even when the computing device 210 is powered down. A second memory such as a random-access memory may provide volatile (but higher speed) memory for storing instructions and data for executing processes. A third memory may be used to improve performance by providing even higher speed memory physically adjacent to the processor 212 for registers, caching and so forth.

[0079] The network interface 216 may include any hardware and/or software for connecting the computing device 210 in a communicating relationship with other resources through the network 202. This may include remote resources accessible through the Internet, as well as local resources available using short range communications protocols using, e.g., physical connections (e.g., Ethernet), radio frequency communications (e.g., WiFi), optical communications, (e.g., fiber optics, infrared, or the like), ultrasonic communications, or any combination of these or other media that might be used to carry data between the computing device 210 and other devices. The network interface 216 may, for example, include a router, a modem, a network card, an infrared transceiver, a radio frequency (RF) transceiver, a near field communications interface, a radio-frequency identification (RFID) tag reader, or any other data reading or writing resource or the like.

[0080] More generally, the network interface 216 may include any combination of hardware and software suitable for coupling the components of the computing device 210 to other computing or communications resources. By way of example and not limitation, this may include electronics for a wired or wireless Ethernet connection operating according to the IEEE 802.11 standard (or any variation thereof), or any other short or long range wireless networking components or the like. This may include hardware for short range data communications such as Bluetooth or an infrared transceiver, which may be used to couple to other local devices, or to connect to a local area network or the like that is in turn coupled to a data network 202 such as the Internet. This may also or instead include hardware/software for a WiMax connection or a cellular network connection (using, e.g., CDMA, GSM, LTE, or any other suitable protocol or combination of protocols). The network interface 216 may be included as part of the input/output devices 220 or vice-versa.

[0081] The data store 218 may be any internal memory store providing a computer-readable medium such as a disk drive, an optical drive, a magnetic drive, a flash drive, or other device capable of providing mass storage for the computing device 210. The data store 218 may store computer readable instructions, data structures, program modules, and other data for the computing device 210 or computer system 200 in a non-volatile form for subsequent retrieval and use. For example, the data store 218 may store without limitation one or more of the operating system, application programs, program data, databases, files, and other program modules or other software objects and the like.

[0082] The input/output interface 220 may support input from and output to other devices that might couple to the computing device 210. This may, for example, include serial ports (e.g., RS-232 ports), universal serial bus (USB) ports, optical ports, Ethernet ports, telephone ports, audio jacks, component audio/video inputs, HDMI ports, and so forth, any of which might be used to form wired connections to other local devices. This may also or instead include an infrared interface, RF interface, magnetic card reader, or other input/output system for coupling in a communicating relationship with other local devices. It will be understood that, while the network interface 216 for network communications is described separately from the input/output interface 220 for local device communications, these two interfaces may be the same, or may share functionality, such as where a USB port is used to attach to a WiFi accessory, or where an Ethernet connection is used to couple to a local network attached storage.

[0083] A peripheral 222 may include any device used to provide information to or receive information from the computing device 200. This may include human input/output (I/O) devices such as a keyboard, a mouse, a mouse pad, a track ball, a joystick, a microphone, a foot pedal, a camera, a touch screen, a scanner, or other device that might be employed by the user 230 to provide input to the computing device 210. This may also or instead include a display, a speaker, a printer, a projector, a headset, or any other audiovisual device for presenting information to a user. The peripheral 222 may also or instead include a digital signal processing device, an actuator, or other device to support control or communication to other devices or components. Other I/O devices suitable for use as a peripheral 222 include haptic devices, three-dimensional rendering systems, augmented-reality displays, magnetic card readers, and so forth. In one aspect, the peripheral 222 may serve as the network interface 216, such as with a USB device configured to provide communications via short range (e.g., Bluetooth, WiFi, Infrared, RF, or the like) or long range (e.g., cellular data or WiMax) communications protocols. In

another aspect, the peripheral 222 may provide a device to augment operation of the computing device 210, such as a global positioning system (GPS) device, a security dongle, or the like. In another aspect, the peripheral may be a storage device such as a flash card, USB drive, or other solid-state device, or an optical drive, a magnetic drive, a disk drive, or other device or combination of devices suitable for bulk storage. More generally, any device or combination of devices suitable for use with the computing device 200 may be used as a peripheral 222 as contemplated herein.

[0084] Other hardware 226 may be incorporated into the computing device 200 such as a co-processor, a digital signal processing system, a math co-processor, a graphics engine, a video driver, and so forth. The other hardware 226 may also or instead include expanded input/output ports, extra memory, additional drives (e.g., a DVD drive or other accessory), and so forth.

[0085] A bus 232 or combination of busses may serve as an electromechanical platform for interconnecting components of the computing device 200 such as the processor 212, memory 214, network interface 216, other hardware 226, data store 218, and input/output interface. As shown in the figure, each of the components of the computing device 210 may be interconnected using a system bus 232 or other communication mechanism for communicating information.

[0086] Methods and systems described herein can be realized using the processor 212 of the computer system 200 to execute one or more sequences of instructions contained in the memory 214 to perform predetermined tasks. In embodiments, the computing device 200 may be deployed as a number of parallel processors synchronized to execute code together for improved performance, or the computing device 200 may be realized in a virtualized environment where software on a hypervisor or other virtualization management facility emulates components of the computing device 200 as appropriate to reproduce some or all of the functions of a hardware instantiation of the computing device 200.

[0087] Fig. 3 illustrates a threat management system according to some implementations. In general, the system 300 may include an endpoint 302, a firewall 304, a server 306 and a threat management facility 308 coupled to one another directly or indirectly through a data network 305, all as generally described above. Each of the entities depicted in Fig. 3 may, for example, be implemented on one or more computing devices such as the computing device described above with reference to Fig.2. A number of systems may be distributed across these various components to support threat detection, such as a coloring system 310, a key management system 312 and a heartbeat system 314 (or otherwise an

endpoint health system), each of which may include software components executing on any of the foregoing system components, and each of which may communicate with the threat management facility 308 and an endpoint threat detection agent 320 executing on the endpoint 302 to support improved threat detection and remediation.

[0088] The coloring system 310 may be used to label or 'color' software objects for improved tracking and detection of potentially harmful activity. The coloring system 310 may, for example, label files, executables, processes, network communications, data sources and so forth with any suitable label. A variety of techniques may be used to select static and/or dynamic labels for any of these various software objects, and to manage the mechanics of applying and propagating coloring information as appropriate. For example, a process may inherit a color from an application that launches the process. Similarly, a file may inherit a color from a process when it is created or opened by a process, and/or a process may inherit a color from a file that the process has opened. More generally, any type of labeling, as well as rules for propagating, inheriting, changing, or otherwise manipulating such labels, may be used by the coloring system 310 as contemplated herein. A suitable coloring system is described in greater detail below with reference to Fig. 4.

[0089] The key management system 312 may support management of keys for the endpoint 302 in order to selectively permit or prevent access to content on the endpoint 302 on a file-specific basis, a process-specific basis, an application-specific basis, a user-specific basis, or any other suitable basis in order to prevent data leakage, and in order to support more fine-grained and immediate control over access to content on the endpoint 302 when a security compromise is detected. Thus, for example, if a particular process executing on the endpoint is compromised, or potentially compromised or otherwise under suspicion, access by that process may be blocked (e.g., with access to keys revoked) in order to prevent, e.g., data leakage or other malicious activity. A suitable key management system useful in this context is described in greater detail below with reference to Fig. 5.

[0090] The heartbeat system 314 may be used to provide periodic or aperiodic information from the endpoint 302 or other system components about system health, security, status, and so forth. The heartbeat system 314 or otherwise an endpoint health system may thus in general include a health status report system for the endpoint 302, such as through the use of a heartbeat system or the like. A heartbeat may be encrypted or plaintext, or some combination of these, and may be communicated unidirectionally (e.g., from the endpoint 302 to the threat

management facility 308) or bidirectionally (e.g., between the endpoint 302 and the server 306, or any other pair of system components) on any useful schedule. A suitable heartbeat system that can be used as part of the endpoint health system is described in greater detail below with reference to Fig. 6.

[0091] In general, these various monitoring and management systems may cooperate to provide improved threat detection and response. For example, the coloring system 310 may be used to evaluate when a particular process is potentially opening inappropriate files, and a potential threat may be confirmed based on an interrupted heartbeat from the heartbeat system 314. The key management system 312 may then be deployed to revoke access by the process to certain resources (e.g., keys or file) so that no further files can be opened, deleted, or otherwise modified. More generally, the cooperation of these systems enables a wide variety of reactive measures that can improve detection and remediation of potential threats to an endpoint.

[0092] Fig. 4 illustrates a system for behavioral tracking, coloring, and generation of indications of compromise (IOCs). In general, the system 400 may include a number of entities participating in a threat management process such as any of the entities and threat management processes described herein. The threat management process may for example employ techniques such as behavioral tracking, encryption, endpoint recording, reputation-based threat detection, behavioral-based threat detection, signature-based threat detection, and combinations of the foregoing, or any other suitable techniques for detecting threats to endpoints in an enterprise.

[0093] In general, the system 400 may include a number of endpoints 402, 412 and a threat management facility 404 in an enterprise 410, such as any of the enterprises described herein. An external analysis facility 406 may analyze threat data and provide rules and the like for use by the threat management facility 404 and endpoints 402, 412 in managing threats to the enterprise 410. The threat management facility 404 may reside locally (e.g., a part of, embedded within, or locally coupled to the endpoint 402), a virtual appliance (e.g., which could be run by a protected set of systems on their own network system(s)), a private cloud, a public cloud, and so forth. The analysis facility 406 may store locally-derived threat information. The analysis facility 406 may also or instead receive threat information from a third-party source 416 such as MITRE Corporation or any other public, private, educational, or other organization that gathers information on network threats and provides analysis and threat detection information for use by others. Each of these components may be configured with suitable programming to participate in the various threat detection and management techniques contemplated herein. The threat

management facility 404 may monitor any stream of data from an endpoint 402 exclusively or use the full context of intelligence from the stream of all protected endpoints 402, 412 or some combination of these.

[0094] The endpoint 402 may be any of the endpoints described herein, or any other device or network asset that might join or participate in the enterprise 410 or otherwise operate on an enterprise network. This may, for example, include a server, a client such as a desktop computer or a mobile computing device (e.g., a laptop computer, a wearable device, a tablet, and the like), a cellular phone, a smart phone, or other computing device suitable for participating in the enterprise 410.

[0095] In general, the endpoint 402 may include any number of computing objects such as an object 418 labeled with a descriptor 420. While the term object has a number of specific meanings in the art, and in particular in object-oriented programming, it will be understood that the term 'object' as used herein is intended to be significantly broader, and may include any data, process, file or combination of these including without limitation any process, application, executable, script, dynamic linked library, file, data, database, data source, data structure, function, resource locator (e.g., uniform resource locator (URL) or other uniform resource identifier (URI)), or the like that might be manipulated by one of the computing devices described herein.

[0096] An object 418 may also or instead include a remote resource, such as a resource identified in a URL. That is, while the objects 418 in Fig. 4 are depicted as residing on the endpoint 402, an object 418 may also reside elsewhere in the system 400, while still being labeled with a descriptor 420 and tracked by the monitor 421 of the endpoint 402. The object 418 may be an item that is performing an action or causing an event, or the object 418 may be an item that is receiving the action or result of an event (i.e., the item in the system 400 being acted upon).

[0097] Where the object 418 is data or includes data, the object 418 may be encrypted or otherwise protected, or the object 418 may be unencrypted or otherwise unprotected. The object 418 may be a process or other computing object that performs an action, which may include a single event or a collection or sequence of events taken by a process. The object 418 may also or instead include an item such as a file or lines of code that are executable to perform such actions. The object 418 may also or instead include a computing component upon which an action is taken, e.g., a system setting (e.g., a registry key or the like), a data file, a URL, or the like. The

object 418 may exhibit a behavior such as an interaction with another object or component of the system 400.

[0098] In one aspect, objects 418 may be described in terms of persistence. The object 418 may, for example, be a part of a process, and remain persistent as long as that process is alive. The object 418 may instead be persistent across an endpoint 402 and remain persistent as long as an endpoint 402 is active or alive. The object 418 may instead be a global object having persistence outside of an endpoint 402, such as a URL or a data store. In other words, the object 418 may be a persistent object with persistence outside of the endpoint.

[0099] Although many if not most objects 418 will typically be benign objects forming a part of a normal, operating endpoint, an object 418 may contain software associated with an advanced persistent threat (APT) or other malware that resides partially or entirely on the endpoint 402. The associated software may have reached the endpoint 402 in a variety of ways and may have been placed manually or automatically on the endpoint 402 by a malicious source. It will be understood that the associated software may take any number of forms and have any number of components. For example, the associated software may include an executable file that can execute independently, or the associated software may be a macro, plug-in, or the like that executes within another application. Similarly, the associated software may manifest as one or more processes or threads executing on the endpoint 402. Further, the associated software may install from a file on the endpoint 402 (or a file remote from the endpoint 402), and the associated software may create one or more files such as data files or the like while executing. Associated software should be understood to generally include all such files and processes except where a specific file or process is more specifically noted.

[00100] A threat such as an APT may also take the form of an attack where no altered or additional software is directly added or modified on the endpoint 402. Instead, an adversary may reuse existing software on the system 400 to perform the attacks. It is for this reason that simply scanning for associated software may be insufficient for the detection of APTs and it may be preferable to detect APTs based on the behavior of the software and associated objects 418 that are used by, for, and with that software.

[00101] An object coloring system 414 may apply descriptors 420 to objects 418 on the endpoint 402. This may be performed continuously by a background process on the endpoint 402, or it may occur whenever an object 418 is involved in an action, such as when a process makes a call to an application programming interface (API) or takes some other action, or when

a URL is used to initiate a network request, or when a read or a write is performed on data in a file. This may also or instead include a combination of these approaches as well as other approaches, such as by pre-labeling a file or application when it is moved to the endpoint 402, or when the endpoint 402 is started up or instantiated. In general, the object coloring system 414 may add, remove, or change a color at any location and at any moment that can be practicably instrumented on a computer system.

[00102] As noted above, the term 'object' as used herein is intended to include a wide range of computing objects and as such, the manner in which particular objects 418 are labeled or 'colored' with descriptors 420 may vary significantly. Any object 418 that is performing an action may be colored at the time of and/or with a label corresponding to the action, or likewise any object 418 that is the target of the action may be colored at the time that it is used and/or with a label corresponding to a process or the like using the object 418. Furthermore, the operating system runtime representation of the object 418 may be colored, or the persistent object outside of the operating system may be colored (as is the case for a File Handle or File Object within the operating system or the actual file as stored in a file system), such as within an encryption header or other header applied to the file, or as part of a directory attribute or any other persistent location within the file or file system. A former coloring may be ephemerally tracked while the operating system maintains the representation and the latter may persist long after any reboots of the same operating system and likewise have meaning when read or used by other endpoints 402. For processes, each file handle may be supplemented with a pointer or other mechanism for locating a descriptor 420 for a particular object 420 that is a process. More specifically, each object 418 may be colored in any manner suitable for appending information to that object 418 so that the corresponding descriptor 420 can be retrieved and, where appropriate, updated.

[00103] The coloring system 414 may apply any suitable rules for adding and changing descriptors 420 for objects 418. For example, when a process with a certain descriptor accesses data with a different descriptor, the descriptor for the process may be updated to correspond to the data, or the descriptor for the data may be updated to correspond to the process, or some combination of these. Any action by or upon an object 418 may trigger a coloring rule so that descriptors 420 can be revised at any relevant time(s) during processing.

[00104] In one aspect, colors will not explicitly indicate a compromised security state or other good/bad types of distinctions (although they may be adapted to this use). Instead, colors

may record some known information or understanding about an object 418, such as a source, a purpose, and so forth. In this context, colors will not be used to label actual or potential security compromises, but to identify inconsistencies among interacting objects 418, and to restrict or control access and use accordingly. For example, where an endpoint uses file-system-based encryption as described herein, a process that is colored as exposed to external resources (e.g., the Internet) may be prohibited from accessing cleartext data for protected files. Colors can also be used in other contexts such as intrusion prevention, routing rules, and detection of odd or questionable behavior.

[00105] In one aspect, colors may be implemented as flags associated with objects 418 that provide a short hand cache of potentially relevant information. While this information could also be obtained for an object 418 through a careful inspection of related activity logs or other data recording activities, the use of a cache of flags for coloring information makes the coloring information directly available and immediately actionable, as distinguished from post hoc forensic activities that are otherwise supported by data logging.

[00106] In one aspect, colors as contemplated herein may fall into two different categories: static colors and dynamic colors. Static colors may be explicitly applied based on, e.g., a controlling application. For example, a static color may specify a status of an application or data, or an associated type of application (e.g., productivity, mail client, messaging, browser, word processing, financial, spreadsheet, etc.). In this context, a process will generally inherit static colors from a source executable, and will permit inferences for appropriate behavior and related processes. Dynamic colors may be assigned based on direct observation of executing processes, and may not be inherited or transferred among processes (although the presence of a dynamic color may be used to draw another coloring inference upon interaction with another process). Thus, the inheritance of colors may depend in part upon the type of color that is applied, or upon explicit inheritance rules provided for a particular color.

[00107] A descriptor 420 may take a variety of forms, and may in general include any information selected for relevance to threat detection. This may, for example, be a simple categorization of data or processes such as trusted or untrusted. For example, in one embodiment described herein, data and processes are labeled as either ‘IN’ (e.g., trusted) or ‘OUT’ (e.g., untrusted). The specific content of the label is unimportant, and this may be a binary flag, text string, encrypted data or other human-readable and/or machine-readable identifier, provided that the descriptor 420 can facilitate discrimination among labeled files – in this example, between

trusted objects 418 and untrusted objects 418 so that, e.g., trusted data can be selectively decrypted or encrypted for use with trusted processes. Similarly, data may be labeled as corporate data or private data, with similar type-dependent processing provided. For example, private data may be encrypted with a key exclusively controlled by the data owner, while corporate data may be encrypted using a remotely managed key ring for an enterprise operated by the corporation.

[00108] In another aspect, the descriptor 420 may provide a multi-tiered or hierarchical description of the object 418 including any information useful for characterizing the object 418 in a threat management context. For example, in one useful configuration the descriptor 420 may include a type or category, static threat detection attributes, and an explicit identification. The type or category for the object 418 may be any category or the like that characterizes a general nature or use of the object 418 as inferred from behavior and other characteristics. This may, for example, include categories such as 'game,' 'financial,' 'application,' 'electronic mail,' 'image,' 'video,' 'browser,' 'antivirus,' and so forth. The category may be more granular, or may include hierarchical categories such as 'application:spreadsheet,' 'application:word_processing,' and so forth. Such colors may be directly inferred from a single action, a sequence of actions, or a combination of actions and other colors, including, e.g., colors of processes and files related to a particular action, or other objects 418 that provide context for a particular action or group of actions. One or more colors may also or instead be explicitly provided by a user or a process, or otherwise automatically or manually attributed to computer objects as contemplated herein.

[00109] The static threat detection attributes may be any readily ascertainable characteristics of the object 418 useful in threat detection. This may, for example, include an antivirus signature, a hash, a file size, file privileges, a process user, a path or directory, declarations of permissions, an access (e.g., a resource access, or an API access), and so forth. Static threat detection attributes may also include attributes that are derived by or supplied from other sources. For example, static threat detection attributes may include a reputation for an object 418, which may be expressed in any suitable or useful level of granularity such as with discrete categories (trusted/untrusted/unknown) or with a numerical score or other quantitative indicator. The explicit identification may, in general, be what an object 418 calls itself, e.g., a file name or process name.

[00110] Some actions may transfer colors from a subject of the action to the target of the action. For example, when a process creates sub-processes, the sub-processes may inherit the colors of its parent(s). By way of another example, when a process is initially loaded from an executable, it may inherit the color(s) stored in the file system for or with the executable.

[00111] In general, the descriptor 420 may be provided in any suitable format. The descriptor 420 may for example be formed as a vector of binary flags or other attributes that form the 'color' or description of an object 418. The descriptor 420 may also, where appropriate, include scalar quantities for certain properties. For example, it may be relevant how many times a system file was accessed, how many file handles a process has open, how many times a remote resource was requested or how long a remote resource is connected, and this information may be suitably included in the descriptor 420 for use in coloring objects with the coloring system 414 and applying rules for IOC detection by the IOC monitor 421.

[00112] An indication of compromise (IOC) monitor 421 may be provided to instrument the endpoint 402 so that any observable actions by or involving various objects 418 can be detected. As with the coloring system 414, it will be understood that the types of observable actions will vary significantly, and the manner in which the endpoint 402 is instrumented to detect such actions will depend on the particular type of object 418. For example, for files or the like, an API for a file system may be used to detect reads, writes, and other access (e.g., open, read, write, move, copy, delete, etc.), and may be configured to report to or otherwise initiate monitoring of the action taken with the file through the file system. As another example, kernel objects may be instrumented at the corresponding object handle or in some other manner. As a further example, a kernel driver may be used for intercepting a process startup. While a wide variety of objects are contemplated herein, one of ordinary skill in the art may create suitable instrumentation for any computing object so that it may be monitored by the IOC monitor 421.

[00113] It will be noted that suitable instrumentation may be used for a variety of functions and circumstances. For example, instrumentation may usefully track requests for network access or other actions back to a particular application or process, or data payloads back to a particular file or data location. One of ordinary skill in the art can readily implement suitable traces and/or logging for any such information that might be useful in a particular IOC monitoring operation.

[00114] In general, the IOC monitor 421 applies rules to determine when there is an IOC 422 suitable for reporting to a threat management facility 404. It will be understood that an endpoint 402 may, in suitable circumstances and with appropriate information, take immediate local action to remediate a threat. However, the monitor 421 may advantageously accumulate a sequence of actions, and still more advantageously may identify inconsistencies or unexpected behavior within a group of actions with improved sensitivity by comparing descriptors 420 for various objects 418 involved in relevant actions and events. In this manner, rules may be applied based upon the descriptors 420 that better discriminate malicious activity while reducing the quantity and frequency of information that must be communicated to a remote threat management facility 404. At the same time, all of the relevant information provided by the descriptors 420 can be sent in an IOC 422 when communicating a potential issue to the threat management facility 404. For example, during the course of execution, a specific process (as evidenced by its observed actions) may be assigned color descriptors indicating that it is a browser process. Further, the specific process may be assigned an attribute indicating that it has exposed itself to external URLs or other external data. Subsequently, the same process may be observed to be taking an action suitable for an internal or system process, such as opening up shared memory to another process that has coloring descriptions indicating that it is a system process. When this last action is observed, an inconsistency in the various color descriptors between the subject of the action—the externally exposed browser process—and the target of the action may result in a well-defined IOC, which may be directly processed with immediate local action taken. The IOC may also or instead be reported externally as appropriate.

[00115] Thus, an endpoint 402 in an enterprise 410 may be instrumented with a coloring system 414 and monitor 421 to better detect potentially malicious activity using descriptors 420 that have been selected for relevance to threat detection along with a corresponding set of rules developed for the particular descriptors 420 that are being used to label or color various objects 418. By way of example, the object 418 may be a web browser that starts off being colored as a ‘browser’ and an ‘internet facing’ application. Based on this descriptor 420, a range of behaviors or actions may be considered normal, such as accessing remote network resources. However, if an object 418 colored with this descriptor 420 attempted to elevate privileges for a process, or to access a registry or system files, then this inconsistency in action may trigger a rule violation and result in an IOC 422.

[00116] In general, any action or series of actions that cumulatively invoke a particular reporting or action rule may be combined into an IOC 422 and communicated to the threat management facility 404. For example, an IOC 422 may include a malicious or strange behavior, or an indication of a malicious or strange behavior. The IOC 422 may be a normalized IOC that expresses one or more actions in a platform independent manner. That is, the IOC 422 may express a malicious behavior or suspected malicious behavior without reference to platform-specific information such as details of an operating system (e.g., iOS, MacOS, Windows, Android, Linux, and so forth), hardware, applications, naming conventions, and so forth. Thus, a normalized IOC may be suitable for identifying a particular threat across multiple platforms, and may include platform independent processes, actions, or behaviors, or may express such process, actions, or behaviors in a platform independent manner. The normalized IOC may be generated from the IOC 422, e.g., it may be a converted version of the IOC 422 suitable for use with multiple platforms, or it may simply be any IOC 422 that has been created in a platform independent form. Process colorization (i.e., using the coloring system 414) as described herein may be used to create a normalized IOC.

[00117] In general, a threat management facility 404 for the enterprise 410 may include an IOC collector 426 that receives the IOC 422 from the endpoint 402 and determines an appropriate action. This may include any suitable remedial action, or where one or more IOCs 422 are inconclusive, continued monitoring or increased monitoring as appropriate.

[00118] The threat management facility 404 may provide a variety of threat management or monitoring tools 424, any of which may be deployed in response to IOCs 422 collected by the IOC collector 426. These tools 424 may include without limitation a scanning engine, whitelisting/blacklisting, reputation analysis, web filtering, an emulator, protection architecture, live protection, runtime detection, APT detection, network antivirus products, IOC detection, access logs, a heartbeat, a sandbox, or quarantine system, and so forth.

[00119] The analysis facility 406 may provide a remote processing resource for analyzing malicious activities and creating rules 434 suitable for detecting IOCs 422 based on objects 420 and descriptors 420. It is generally contemplated that suitable attributes of certain descriptors 418 and one or more rules 434 may be developed together so that objects 418 can be appropriately labeled with descriptors 420 that permit invocation of rules 434 and creation of IOCs 422 at appropriate times. The analysis facility 406 may include a variety of analysis tools 428 including, without limitation, tools for regular expression, whitelisting/blacklisting, crowd

sourcing, identifiers, and so forth. The analysis tools 428 may also or instead include information and tools such as URL look-ups, genotypes, identities, file look-up, reputations, and so forth. The analysis facility 406 may also provide numerous related functions such as an interface for receiving information on new, unknown files or processes, and for testing of such code or content in a sandbox on the analysis facility 406.

[00120] The analysis facility 406 may also or instead include a compromise detector 430, where the compromise detector 430 is configured to receive new threat information for analysis and creation of new rules and descriptors as appropriate, as well as corresponding remedial actions. The compromise detector 430 may include any tools described herein or otherwise known in the art for detecting compromises or evaluating new threats in an enterprise 410.

[00121] In general, a rule 434 may be manually created with corresponding human-readable semantics, e.g., where a process is labeled as a browser process or other category or type that can be interpreted by a human. It should, however, be appreciated that the compromise detector 430 may also be configured to automatically generate descriptors 420 and rules 434 suitable for distribution to a threat management facility 404 and an endpoint 402. In this latter mode, the meaning of a particular descriptor 420 may not have a readily expressible human-readable meaning. Thus, it will be understood that attributes selected for relevance to threat detection may include conventional attributes, as well as attributes without conventional labels or meaning except in the context of a particular, computer-generated rule for threat detection.

[00122] In general, the analysis facility 406 may be within an enterprise 410, or the analysis facility 406 may be external to the enterprise 410 and administered, for example, by a trusted third party. Further, a third-party source 416 may provide additional threat data 438 or analyses for use by the analysis facility 406 and the threat management facility 404. The third-party resource 416 may be a data resource that provides threat data 438 and analyses, where the threat data 438 is any data that is useful in detecting, monitoring, or analyzing threats. For example, the threat data 438 may include a database of threats, signatures, and the like. By way of example, the third-party resource 416 may be a resource provided by The MITRE Corporation.

[00123] The system 400 may include a reputation engine 440 storing a plurality of reputations 442. The reputation engine 440 may include a reputation management system for the generation, analysis, identification, editing, storing, etc., of reputations 442. The reputation

engine 440 may include reputation-based filtering, which may be similar to the reputation filtering discussed above with reference to Fig. 1. The reputation engine 440 may be located on the threat management facility 404 or the endpoint 402 as shown in Fig. 4, or the reputation engine 440 may be located elsewhere in the system 400. The reputation engine 440 may receive an IOC 422 or a stream of IOCs 422, and may generate or utilize reputations 442 for the IOCs 422. The reputation engine 440 may also or instead receive actions, behaviors, events, interactions, and so forth, and may generate or utilize reputations 442 for any of the foregoing. The reputation engine 440 may generate or revise a reputation 442 based on behaviors, actions, events, interactions, IOCs 422, other reputations 442, a history of events, data, rules, state of encryption, colors, and so forth. The reputation engine 440 may utilize a third-party resource, e.g., for the third-party resource's reputation data.

[00124] The reputations 442 may include reputations for any of the objects 418 as described herein. In general, the reputations 442 may relate to the trustworthiness of the objects 418 or an attribute thereof (e.g., the source of the object 418, a behavior of the object 418, another object interacting with the object 418, and so forth). The reputations 442 may include lists of known sources of malware or known suspicious objects 418. The reputations 442 may also or instead include lists of known safe or trusted resources or objects 418. The reputations 442 may be stored in a reputations database included on the reputation engine 440 or located elsewhere in the system 400. The reputations 442 may be expressed in any suitable or useful level of granularity such as with discrete categories (e.g., trusted, untrusted, unknown, malicious, safe, etc.) or with a numerical score or other quantitative indicator. The reputations 442 may also be scaled.

[00125] In general, in the system 400 of Fig. 4, a malicious activity on the endpoint 402 may be detected by the IOC monitor 421, and a corresponding IOC 422 may be transmitted to the threat management facility 404 for remedial action as appropriate. The threat management facility 404 may further communicate one or more IOCs 422 to the analysis facility 406 for additional analyses and/or resolution of inconclusive results. Other details and variations are provided below. While the use of coloring and IOCs as contemplated herein can improve threat detection and remediation in a number of ways, the system 400 can be further improved with granular control over access to endpoint data using an encryption system. A system for key-based management of processes and files on an endpoint is now discussed in greater detail.

[00126] Fig. 5 illustrates a system for encryption management. Generally, the system 500 may include endpoints 502, an administration host 504, and a threat management facility 506, which may include policy manager 508 and key manager 510. The system 500 may provide for the management of users 512, policies 514, keys 516 (e.g., disposed on key rings 518), and endpoints 502 (e.g., from the administration host 504). The system 500 may utilize various storage and processing resources, which may be local, remote, virtual, disposed in a cloud, or the like.

[00127] The endpoints 502 may be any of the endpoints as described herein, e.g., with reference to the other figures. The endpoints 502 may also or instead include other end user devices and other devices to be managed. The endpoints 502 may include a web browser for use by the users 512, with supporting cryptographic functions implemented using cryptographic libraries in the web browser. The endpoints 502 may communicate with the other components of the system 500 using any suitable communication interface, which may include Secure Socket Layer (SSL) encryption, Hypertext Transfer Protocol Secure (HTTPS), and so forth for additional security.

[00128] The endpoints 502 may include objects as described herein. For example, the endpoints 502 may include processes 520 and files 522. The processes 520 may be labeled (e.g., by a coloring system using descriptors as described above) in such a manner that the process is 'IN,' where the process 520 is in compliance with policies 514 administered for the endpoint 502 from a remote threat management facility 506, or the process is 'OUT,' where the process 520 is out of compliance with a policy (or a number of policies) in the policies 514 for an enterprise. This may provide IN processes 520A and OUT processes 520B as shown in Fig. 5. The files 522 may be similarly labeled by a coloring system with descriptors that identify each file 522 as IN, where the file 522 complies with the policies 514 and is accordingly encrypted using, e.g., a remotely managed key ring 518, or the file is OUT, where the file 522 does not conform to the policies 514 and is accordingly not encrypted using the remotely managed key ring 518. This may provide IN files 522A and OUT files 522B as shown in Fig. 5. One skilled in the art will recognize that other objects of the endpoint 502 or other components of the system 500 may be labeled in a similar manner where they are either IN or OUT. By coloring objects in this manner and basing key access on the corresponding color, the "IN" software objects may operate in a protected environment that objectively appears to be in compliance with the policies 514. Other files and processes may still be used on the endpoint 502, but they will operate in an

"OUT" or unprotected environment that cannot obtain access to any of the "IN" content or functionality.

[00129] In an implementation, the system 500 may include determining whether an endpoint 502 is IN or OUT or whether a component of the endpoint 502 is IN or OUT, which may be based upon a set of rules (e.g., the rules outlined herein) or policies such as the policies 514 described herein. In some aspects, if the entire endpoint 502 is OUT – that is, out of compliance with one or more policies 514, the endpoint 502 will not have key access or access to any protected content. Conversely, if the endpoint 502 is IN, the endpoint 502 may have access to protected content. Thus, in one aspect, the notion of IN/OUT may be applied at an endpoint level, and data protection may be a consequence of endpoint protection. Endpoint protection may also or instead be applied at a more granular level, e.g., by determining whether executables, processes 520, files 522, etc., on the endpoint 502 are IN or OUT, which may be based upon rules or policies 514 as described herein.

[00130] The administration host 504 may include a web browser, which may include a cryptography library 524 and a web user interface (e.g., HTML, JavaScript, etc.). An administrator may utilize the web user interface to administer a key management system and perform administrative functions such as creating and distributing keys 516, establishing security policies, creating key hierarchies and rules, and so forth. The endpoint 502 may also include a cryptographic library 524 implementing cryptographic protocols for using key material in the key ring 518 to encrypt and decrypt data as needed.

[00131] The threat management facility 506 may include any of the threat management facilities or similar systems described herein. In general, the threat management facility 506 may include a policy manager 508 and key manager 510. Alternatively, one or more of the policy manager 508 and key manager 510 may be located elsewhere on a network.

[00132] The policy manager 508 may implement one or more policies 514, and maintain, distribute, and monitor the policies for devices in an enterprise. The policies 514 may include any policies 514 relating to secure operation of endpoints 502 in an enterprise. This may, for example, include hardware configuration policies, software configuration policies, communication policies, update policies, or any other policies relating to, e.g., the configuration of an endpoint 502, communications by an endpoint 502, software executing on an endpoint 502 and so forth. Policies 514 may include usage criteria based on, e.g., signatures, indications of compromise, reputation, user identity, and so forth. With respect to the key management system

contemplated herein, the policies 514 may include a cryptographic protocol design, key servers, user procedures, and other relevant protocols, or these cryptographic protocols may be provided elsewhere for use by the policy manager 508. The policies 514 may also include any rules for compliance including those mentioned above or any other suitable rules or algorithms that can be applied to determine whether objects and components are 'IN' or 'OUT' as contemplated herein.

[00133] The key manager 510 may be part of the threat management facility 506, or it may be remotely managed elsewhere, e.g., in a remote cloud resource or the like. The key manager 510 may also or instead be disposed on the administration host 504 and one or more endpoints 502 in a manner independent of the threat management facility 506. In this manner, all cryptographic operations may be isolated from the threat management facility 506 and instead may be performed by a web browser or the like executing on the administration host 504 or an endpoint 502. The key manager 510 may manage the keys 516, including managing the generation, exchange, storage, use, and replacement of keys 516. The key manager 510 may include a key ring 518, where the keys 516 are disposed on the key ring 518 using one root key 526. The key manager 510 may also or instead include a variety of key management and other secure processes, including without limitation, administrator registration, establishing trust to endpoints 502, key distribution to endpoints 502, policy deployment, endpoint status reporting, and local key backup.

[00134] The users 512 may have full access to encrypted data. Alternatively, the users 512 may have limited access to encrypted data, or no access to encrypted data. Access may be limited to users 512 using endpoints 502 that are deemed 'IN' by the system, as well as to processes 520 that are IN, as further described herein.

[00135] The keys 516 may include cryptographic keys in a cryptosystem, i.e., decryption keys. In one aspect, the keys 516 may be disposed on one key ring 518 using one root key 526. In general, the keys 516 may be created and managed using, e.g., symmetric key technology, asymmetric key technology, or any other key technology or combination of key technologies suitable for securing data in an enterprise including, for example the Data Encryption Standard (DES), Triple DES, Advanced Encryption Standard (AES), elliptic curve cryptography (ECC), and so forth. The cryptosystem may also or instead include any suitable public key infrastructure or the like supporting the distribution and use of keys for encryption, digital signatures, and so forth.

[00136] The key ring 518 may facilitate simplified management of the system 500. For example, by reducing the data protection system down to a single key ring 518, the system can eliminate or reduce the overhead for management of keys 516. In one aspect, all of the data on a key ring 518 is protected by one root key 526. By reducing the data protection system down to a single key ring 518 protected by one root key 526, all privileged users 512 on uncompromised platforms can have access to all protected data. In this embodiment, data is either 'IN' (i.e., encrypted), or it is 'OUT' (i.e., not encrypted). In one aspect, the default system does not include any additional level of granularity of access control.

[00137] The cryptography library 524 may be disposed on the administration host 504 as shown in Fig. 5. The cryptography library 524 may also be disposed on the endpoint 502, e.g., in a web browser, or it may be disposed on another component of the system 500, or any combination of these. The cryptographic library 524 may be installed by an administrator. In general, key material 530 from the key ring 518 may be stored in a cache 532 on the endpoint 502 within any suitable memory on the endpoint 502 for use in encryption and decryption as contemplated herein. As noted above, an enterprise that systematically uses coloring and indications of compromise can be improved through the use of a synchronized or integrated key management system as contemplated herein. This system may be still further improved with the addition of a heartbeat system that communicates heartbeats from an endpoint containing health and status information about the endpoint. A suitable heartbeat system is now described in greater detail.

[00138] Fig. 6 illustrates a threat management system using heartbeats. In general, a system 600 may include an endpoint 602, a gateway 604, a threat management system 606, and an enterprise management system 608 that manages an enterprise including the endpoint 602, the gateway 604, and one or more additional endpoints 610. Each of these components may be configured with suitable programming to participate in the detection and remediation of an advanced persistent threat (APT) or other malware threat as contemplated herein. Although the term "gateway" is used for the device between an endpoint and an external network, it will be appreciated that this device may also or instead include a switch, router, firewall, and/or other network elements, any of which may be included in the "gateway" as that term is used herein.

[00139] The endpoint 602 may be any of the endpoints described herein, or any other device or network asset that might join or participate in an enterprise network. The endpoint 602 may contain a threat 612 such as an advanced persistent threat, virus, or similar malware that

resides on the endpoint 602. The threat 612 may have reached the endpoint 602 in a variety of ways, and may have been placed manually or automatically on the endpoint 602 by a malicious source. It will be understood that the threat 612 may take any number of forms and have any number of components. For example, the threat 612 may include an executable file that can execute independently, or the threat 612 may be a macro, plug-in, or the like that executes within another application. Similarly, the threat 612 may manifest as one or more processes or threads executing on the endpoint 602. The threat 612 may install from a file on the endpoint 602 or a file remote from the endpoint 602, and the threat 612 may create one or more other files such as data files or the like while executing. Advanced persistent threats can be particularly difficult to detect and remediate, and the systems and methods contemplated herein can advantageously provide improved sensitivity to such threats, as well as enabling improved remediation strategies. However, the systems and methods contemplated herein may also or instead be used to detect and remediate other types of malware threats. As such, in this context references to a particular type of threat (e.g., an advanced persistent threat) should be understood to generally include any type of malware or other threat to an endpoint or enterprise unless a more specific threat or threat type is explicitly provided or otherwise clear from the context.

[00140] The threat 612 may be analyzed by one or more threat countermeasures on the endpoint 602 such as a whitelisting filter 614 that approves each item of code before executing on the endpoint 602 and prevents execution of non-whitelisted code. The endpoint 602 may also include an antivirus engine 616 or other malware detection software that uses any of a variety of techniques to identify malicious code by reputation or other characteristics. A runtime detection engine 618 may also monitor executing code to identify possible threats. More generally, any of a variety of threat detection techniques may be applied to the threat 612 before and during execution. In general, a threat 612 may evade these and other security measures and begin executing as a process 620 on the endpoint 602.

[00141] Network traffic 622 from the process 620 may be monitored and logged by a traffic monitor 624 on the endpoint 602. The traffic monitor 624 may, for example, log a time and a source of each network request from the endpoint 602. Where the endpoint 602 is within an enterprise network, the network traffic 622 may pass through the gateway 604 in transit to a data network such as the Internet. While the gateway 604 may be logically or physically positioned between the endpoint 602 and an external data network, it will be understood that other configurations are possible. For example, where the endpoint 602 is associated with an

enterprise network but operating remotely, the endpoint 602 may form a VPN or other secure tunnel or the like to the gateway 604 for use of a threat management system 606, enterprise management system 608, and any other enterprise resources.

[00142] The endpoint 602 may use a heartbeat 626 to periodically and securely communicate status to the gateway 604. The heartbeat 626 may be created by a health monitor 628 within the endpoint 602, and may be transmitted to a remote health monitor 630, for example, at the gateway 604. The health monitor 628 may monitor system health in a variety of ways, such as by checking the status of individual software items executing on the endpoint 602, checking that antivirus and other security software is up to date (e.g., with current virus definition files and so forth) and running correctly, checking the integrity of cryptographic key stores, checking for compliance with enterprise security policies, and checking any other hardware or software components of the endpoint 602 as necessary or helpful for health monitoring. The health monitor 628 may thus condition the issuance of a heartbeat 626 on a satisfactory status of the endpoint 602 according to any suitable criteria, enterprise polices, and other evaluation techniques. The remote health monitor 630 may also or instead be provided at the threat management facility 650, for example as part of the threat management system 606 or the enterprise management system 608.

[00143] The heartbeat 626 may be secured in any suitable manner so that the health monitor 630 can reliably confirm the source of the heartbeat 626 and the status of the endpoint 602. To this end, the heartbeat 626 may be cryptographically signed or secured using a private key so that the monitor 630 can authenticate the origin of the heartbeat 626 using a corresponding public key. In one aspect, the heartbeat 626 may include a combination of plaintext information and encrypted information, such as where the status information for the endpoint is provided in plaintext while a digital signature for authentication is cryptographically secured. In another aspect, all of the information in the heartbeat 626 may be encrypted.

[00144] In one aspect, a key vault 632 may be provided on the endpoint to support cryptographic functions associated with a secure heartbeat. An obfuscated key vault 632 may support numerous useful functions, including without limitation, private key decryption, asymmetric signing, and validation with a chain of trust to a specific root validation certificate. A variety of suitable key management and cryptographic systems are known in the art and may be usefully employed to a support the use of a secure heartbeat as contemplated herein. The system may support a secure heartbeat in numerous ways. For example, the system may ensure

that signing and decryption keys can only be used in authorized ways and inside an intended Access Control mechanism. The system may use "anti-lifting" techniques to ensure that a signing key can only be used when the endpoint is healthy. The system may ensure that attacking software cannot, without first reverse-engineering the key vault 632, extract the original key material. The system may also usefully ensure that an attacker cannot undetectably replace the public keys in a root certificate store, either directly or indirectly, such as in an attack that tries to cause the code to validate against a different set of root keys without directly replacing any keys in the root store.

[00145] A robust heartbeat 626 may usefully provide defensive mechanisms against reverse engineering of obfuscated content (e.g., the private key material stored in key vault 632, the code used to validate the correct running of the remainder of the systems as part of the heartbeat 626 code itself) and any anti-lifting protections to prevent malware from directly using the endpoint 602 (or the health monitor 628 on the endpoint 602) to continue to send out signed heartbeat packets (e.g. stating that "all is well" with the endpoint) after security mechanisms have been impaired, disabled, or otherwise compromised in any way. Lifting in this manner by malicious code can be materially mitigated by providing statistical validation (e.g., with checksums of code) of call stacks, calling processes, and core processes. Likewise, statistical checks as well as checksum integrations into the cryptographic calculations may protect against code changes in the heartbeat 626 code itself.

[00146] A variety of useful techniques may be employed to improve security of the key vault 632 and the heartbeat 626. For example, the system may use domain shifting so that original key material is inferred based on hardware and software properties readily available to the key vault 632, and to ensure that key material uses non-standard or varying algorithms. Software properties may, for example, include readily determined system values such as hashes of nearby code. In another aspect, the keys may be domain shifted in a manner unique to the endpoint 602 so that the manner of statistical validation of call stacks and core software is unique to the endpoint 602. Further the key vault may be provisioned so that a public key stored in the key vault 632 is signed with a certificate (or into a certificate chain) that can be externally validated by a network appliance or other trusted third party or directly by the health monitor 628 or remote health monitor 630.

[00147] The heartbeat 626 may encode any useful status information, and may be transmitted from the endpoint 602 on any desired schedule including any periodic, aperiodic,

random, deterministic, or other schedule. Configured in this manner, the heartbeat 626 can provide secure, tamper-resistant instrumentation for status of the endpoint 602, and in particular an indication that the endpoint 602 is online and uncompromised. A delay or disappearance of the heartbeat 626 from the endpoint 602 may indicate that the endpoint 602 has been compromised; however, this may also simply indicate that the endpoint 602 has been powered off or intentionally disconnected from the network. Thus, other criteria may be used in addition to the disappearance or interruption of the heartbeat 626 to more accurately detect malicious software. Some such techniques are described below, but it will be understood that this may include any supplemental information that might tend to make an attack on the endpoint 602 more or less likely. For example, if the heartbeat 626 is interrupted but the endpoint 602 is still sourcing network traffic, then an inference might suitably be made that the endpoint 602 is compromised.

[00148] The threat management system 606 may, in general, be any of the threat management systems described herein. The enterprise management system 608 generally provides tools and interfaces for administration of the enterprise and various endpoints 610 and other resources or assets attached thereto. It will be understood that the functions of the threat management system 606 and the enterprise management system 608 may vary, and general threat management and administration functions may be distributed in a variety of ways between and among these and other components. This is generally indicated in Fig. 6 as a threat management facility 650 that includes the threat management system 606 and the enterprise management system 608. It will be understood that either or both of these systems may be administered by third parties on behalf of the enterprise, or managed completely within the enterprise, or some combination of these, all without departing from the scope of this disclosure. It will similarly be understood that a reference herein to a threat management facility 650 is not intended to imply any particular combination of functions or components, and shall only be understood to include such functions or components as explicitly stated in a particular context, or as necessary to provide countermeasures for malware (e.g., advanced persistent threats) as contemplated herein. It also should be understood that the heartbeat may be monitored and/or managed by the threat management system 606, the enterprise management system 608, or another component of the threat management facility 650.

[00149] The system 600 may include a certificate authority 660 or similar trust authority or the like (shown as a “trusted third party” in the figure). In order to provide a meaningfully

secure heartbeat 626, the heartbeat 626 may be secured with reference to a trusted authority such as a certificate authority 660 that can issue cryptographic certificates allowing other entities to rely on assertions about identity (e.g., by enabling verification with a trusted third party), and to enable cryptographically secure communications. The cryptographic techniques for creating and using such certificates and relationships are well known, and are not repeated here. The certificate authority 660 may be administered by the enterprise management system 608 or some other internal resource of the enterprise, or the certificate authority 660 may be administered by a trusted third party such as any of a variety of commercially available certificate authorities or the like. Thus, the certificate authority 660, or some other similar cloud service or the like, may operate as a security broker to register, e.g., endpoints 602, 610, the gateway 604, the threat management facility 650, and so forth, and provide cryptographic material for each of the other trusting entities to securely communicate with one another.

[00150] Once registered with the certificate authority 660 in this fashion, the heartbeat may be used to establish trust between the endpoint 602 and other entities, and to validate the source of the heartbeat 626 when it is received. More generally, a heartbeat 626 secured in this manner may provide an encrypted channel between network entities such as an endpoint 602 and the gateway 604 (or a firewall or the like). The nature of the communication may provide a technique for validating the source, as well as obfuscating the contents with encryption. Thus when, for example, the endpoint 602 provides information about a good/healthy state or a bad/compromised state, the recipient may rely on this state information and act accordingly.

[00151] Fig. 7 shows an architecture for endpoint protection in an enterprise network security system. In general, an endpoint may include a processing environment 702, a file system 706 (such as a data storage system or the like), a threat monitor 720 and a key wrapper 730.

[00152] The processing environment 702 may, for example, be any environment such as an operating system or the like suitable for executing one or more processes 704.

[00153] Each process 704 may be an instance of a software application, computer program, portion of a computer program or other code executing within the processing environment 702. A process 704 may execute, e.g., on a processor, group of processors, or other processing circuitry or platform for executing computer-executable code. A process 704 may include executable computer code, as well as an allocation of memory, file descriptors or handles for data sources and sinks, security attributes such as an owner and any associated

permissions, and a context including the content of physical memory used by the process 704. A process 704 may be or may include one or more threads. More generally, a process 704 may include any code executing on an endpoint such as any of the endpoints described herein.

[00154] The file system 706 may include a data storage system or the like, e.g., where a data store including one or more files (e.g., the files 708 shown in the figure) is included as part of the data storage system. The file system 706 may be generally associated with an operating system that provides the processing environment 702, and serves as an intermediary between processes 704 executing in the processing environment 702 and one or more files 708 accessible to the endpoint. The file system 706 may provide a directory structure or other construct to facilitate organization of the files 708, and the file system 706 generally supports file functions such as creating, deleting, opening, closing, reading, writing, and so forth.

[00155] An extension 710 may be included in the file system 706 by modifying the operating system kernel. While other programming techniques may be employed to perform the functions of an extension 710 as contemplated herein, direct modifications to or additions to the operating system permit the extension 710 to operate transparently to the processing environment 702 and the processes 704 without requiring any modifications or adaptations. The extension 710 may, for example, be implemented as a file system filter (in a MICROSOFT WINDOWS environment) or a mount point to a directory (in an APPLE iOS environment). The extension 710 to the files system as contemplated herein performs two concurrent functions. First, the extension 710 communicates with a threat monitor 720 in order to receive updates on the security status and exposure status of the processes 704 or the endpoint. Second the extension 710 communicates with a key wrapper 730 that provides key material for encrypting and decrypting data in the files 708. Finally, the extension 710 operates to conditionally provide encryption and decryption of the files 708 for the processes 704 based on a current security or exposure state, as described in greater detail below.

[00156] The threat monitor 720 may include any suitable threat monitoring, malware detection, antivirus program or the like suitable for monitoring and reporting on a security state of an endpoint or individual processes 704 executing thereon. This may include local threat monitoring using, e.g., behavioral analysis or static analysis. The threat monitor 720 may also or instead use reputation to evaluate the security state of processes 704 based on the processes 704 themselves, source files or executable code for the processes 704, or network activity initiated by the processes 704. For example, if a process 704 requests data from a remote URL that is

known to have a bad reputation, this information may be used to infer a compromised security state of the endpoint. While a threat monitor 720 may operate locally, the threat monitor 720 may also or instead use remote resources such as a gateway carrying traffic to and from the endpoint, or a remote threat management facility that provides reputation information, malware signatures, policy information and the like for the endpoint and other devices within an enterprise such as the enterprise described above.

[00157] The threat monitor 720 may also or instead monitor the health of one or more of the system, an endpoint, a process 704, and so forth. The health monitoring may be used to provide periodic or aperiodic information from one or more system components about system health, security, status, and so forth. Implementations may include using the health monitoring for controlling access, e.g., to files 708, to keys 734, to key material for encrypting and decrypting individual files 708, and so forth.

[00158] In general, the threat monitor 720 provides monitoring of a security state and an exposure state of the endpoint. The security state may, for example, be 'compromised', 'secure', or some other state or combination of states. This may be based on detections of known malware, suspicious activity, policy violations and so forth. The exposure state may be 'exposed' or 'unexposed', reflecting whether or not a particular process 704 or file 708 has been exposed to potentially unsafe content. Thus, exposure may not necessarily represent a specific threat, but the potential for exposure to unsafe content. This may be tracked in a variety of ways, such as by using the coloring system described above with reference to Fig. 5.

[00159] The key wrapper 730 may contain a key ring 732 with one or more keys 734 for encrypting and decrypting files 708. The key ring 732 may be cryptographically protected within the key wrapper 730 in order to prevent malicious access thereto, and the key wrapper 730 may communicate with the extension 710 to provide keys 734 for accessing the files 708 at appropriate times, depending, for example, on whether processes 704 are secure or exposed. In one aspect, the files 708 are stored in a non-volatile memory such as a disk drive, or in a random-access memory that provides a cache for the disk drive, and the key wrapper 730 may be stored in a separate physical memory such as a volatile memory accessible to the operating system and the extension 710 but not to processes 704 executing in the user space of the processing environment 702.

[00160] In one aspect, every document or file on the endpoint may have a separate key. This may be, for example, a unique, symmetric key that can be used for encryption and

decryption of the corresponding file. The key wrapper 730 may control access to the key material for encrypting and decrypting individual files, and may be used by the extension 710 to control access by individual processes 704 executing on the endpoint. As described herein, the extension 710 may generally control access to files 708 based on an exposure state, a security state, or other context such as the user of a calling process or the like. In the event of a severe compromise, or a detection of a compromise independent of particular processes, a key shredding procedure may be invoked to destroy the entire key wrapper 730 immediately and prevent any further access to the files 708. In such circumstances, the keys can only be recovered by the endpoint when a remediation is confirmed. Alternatively, the files may be accessed directly and decrypted from a secure, remote resource that can access the keys 734.

[00161] Fig. 8 illustrates a system for forensic analysis for computer processes. The system 800 may include an endpoint 810 containing a data recorder 820, a monitoring facility 830, and any number of objects 812 and events 814. An analysis facility 840 may be coupled in a communicating relationship with the endpoint 810 over a data network 850 such as any of the networks described above. It will be appreciated that, while illustrated as components of the endpoint 810, certain components of the system 800 such as the data recorder 820 and the monitoring facility 830 and the analysis facility may also or instead be realized as remote services instantiated on a virtual appliance, a public or private cloud, or the like, any of which may be coupled to the endpoint 810 through the data network 850 or another communication channel (not shown). Each of the components of the system 800 may be configured with suitable programming and configuration to participate in the various forensic techniques, threat detection techniques, and security management techniques contemplated herein.

[00162] The endpoint 810 may be any of the endpoints described herein, e.g., a computing device in an enterprise network, or any other device or network asset that might join or participate in an enterprise or otherwise operate on an enterprise network. This may, for example, include a server, a client device such as a desktop computer or a mobile computing device (e.g., a laptop computer or a tablet), a cellular phone, a smart phone, or other computing device suitable for participating in the system 800 or in an enterprise.

[00163] In general, the endpoint 810 may include any number of computing objects 812, which may for example, be processes executed by one or more processors or other processing circuitry, files or data stored in memory, or any other computing objects described herein. While the term object has a number of specific meanings in the art, and in particular in

object-oriented programming, it will be understood that the term 'object' as used herein is intended to be significantly broader, and may include any data, process, file or combination of these including without limitation any process, application, executable, script, dynamic linked library (DLL), file, data, database, data source, data structure, function, resource locator (e.g., uniform resource locator (URL) or other uniform resource identifier (URI)), or the like that might be resident on the endpoint 810 and manipulated by the endpoint 810 or another component of the system 800 or other systems described elsewhere herein. The object 812 may also or instead include a remote resource, such as a resource identified in a URL. That is, while the object 812 in the figure is depicted as residing on the endpoint 810, an object 812 may also reside elsewhere in the system 800, for example with a link, pointer, or reference.

[00164] The object 812 may be an item that is performing an action or causing an event 814, or the object 812 may be an item that is receiving the action or is the result of an event 814 (e.g., the object 812 may be an item in the system 800 being acted upon by an event 814 or another object 812). In general, an event 814 as contemplated herein may be any data flow, execution flow, control flow, network flow, or other similar action or event that might causally relate objects 812 to one another. Where the object 812 is data or includes data, the object 812 may be encrypted or otherwise protected, or the object 812 may be unencrypted or otherwise unprotected. The object 812 may be a process or other computing object that performs an action, which may include a single event 814 or a collection or sequence of events 814 taken by a process. The object 812 may also or instead include an item such as a file or lines of code that are executable to perform such actions. The object 812 may also or instead include a computing component upon which an action is taken, e.g., a system setting (e.g., a registry key or the like), a data file, a URL, and so forth. The object 812 may exhibit a behavior such as an interaction with another object or a component of the system 800.

[00165] Objects 812 may be described in terms of persistence. The object 812 may, for example, be a part of a process, and remain persistent as long as that process is alive. The object 812 may instead be persistent across an endpoint 810 and remain persistent as long as an endpoint 810 is active or alive. The object 812 may instead be a global object having persistence outside of an endpoint 810, such as a URL or a data store. In other words, the object 812 may be a persistent object with persistence outside of the endpoint 810.

[00166] Although many if not most objects 812 will typically be benign objects forming a normal part of the computing environment for an operating endpoint 810, an object 812 may

contain software associated with an advanced persistent threat (APT) or other malware that resides partially or entirely on the endpoint 810. This associated software may have reached the endpoint 810 in a variety of ways, and may have been placed manually or automatically on the endpoint 810 by a malicious source. It will be understood that the associated software may take any number of forms and have any number of components. For example, the associated software may include an executable file that can execute independently, or the associated software may be a macro, plug-in, or the like that executes within another application. Similarly, the associated software may manifest as one or more processes or threads executing on the endpoint 810. Further, the associated software may install from a file on the endpoint 810 (or a file remote from the endpoint 810), and the associated software may create one or more files such as data files or the like while executing. Associated software should be understood to generally include all such files and processes except where a specific file or process is more specifically noted.

[00167] An event 814 may include an action, a behavior, an interaction, and so forth. The event 814 may be generated by or otherwise related to an object 812. For example, the event 814 may be associated with a file and include an action such as a read, a write, an open, a move, a copy, a delete, and so forth. The event 814 may also or instead include an inter-process communication, e.g., a create, a handle, a debug, a remote injection, and so forth. The event 814 may also or instead include accessing an Internet Protocol (IP) address or URL.

[00168] The data recorder 820 may monitor and record activity related to the objects 812 and events 814 occurring on the endpoint 810. The activity of the endpoint 810 may be stored in a data log 822 or the like on the data recorder 820, which may be stored locally on the endpoint 810 (as depicted) or remotely at a threat management resource, or some combination of these, such as where the data log 822 is periodically transmitted to a remote facility for archiving or analysis. The data recorder 820 may continuously record any activity occurring on the endpoint 810 for predetermined periods of time before overwriting previously recorded data. Thus, the data log 822 may include a continuous data feed of events 814. When an event 814 is detected that is a beacon or trigger event (such as a file detection, a malicious traffic detection, or the like), the data log 822 may be saved and transmitted to an analysis facility 840 or the like for analysis, e.g., to determine a root cause of the beacon or trigger event. The data log 822 may be used to create an event graph or other snapshot of the activity on the endpoint 810, e.g., for a period of time surrounding a beacon or trigger event. The beacon or trigger event may be

detected locally by the monitoring facility 830, or remotely by a remote threat management facility or the like, or some combination of these.

[00169] While illustrated on the endpoint 810, it will be understood that the data recorder 820 may also or instead be implemented at a remote location such as a threat management facility or other enterprise network security resource. The data recorder 820 may be provisioned on the same or a different device than a data store in which data is stored. The data recorder 820 may be configured to record data as efficiently as possible so as to minimize impact on the endpoint 810.

[00170] The monitoring facility 830 may work in conjunction with the data recorder 820 to instrument the endpoint 810 so that any observable events 814 by or involving various objects 812 can be monitored and recorded. It will be appreciated that various filtering rules and techniques may be used to synopsize, summarize, filter, compress or otherwise process information captured by the data recorder 820 to help ensure that relevant information is captured while maintaining practical limits on the amount of information that is gathered.

[00171] A security product 832 may execute on the endpoint 810 to detect a security event on the endpoint 810, which may act as the beacon or trigger event for the system 800. The security product 832 may use techniques such as signature-based and behavioral-based malware detection including without limitation one or more of host intrusion prevention, malicious traffic detection, URL blocking, file-based detection, and so forth.

[00172] The beacon or trigger event on the endpoint 810 may be a fully qualified (e.g., definitive) detection of a compromise or other malicious activity. In another aspect, the beacon or trigger event on the endpoint 810 may be a suspicious behavior that is suspicious but not confirmed as malicious. For example, the beacon or trigger event on the endpoint 810 may signal an unusual behavior that is known to commonly appear concurrently with the detection of malware. In an aspect, when the beacon or trigger event is a suspicious behavior, the data log 822 may be analyzed differently than when the beacon or trigger event is a confirmed malicious behavior. For example, the data log 822 may be sent to a different component of the system 800 through the network, e.g., to a different analysis facility 840.

[00173] The monitoring facility 830 may be disposed remotely from the endpoint 810 or analysis facility 840. The monitoring facility 830 may be included on one or more of the endpoint 810 or analysis facility 840. In an aspect, the monitoring facility 830 and the analysis facility 840 included in the same component.

[00174] The analysis facility 840 may analyze the data log 822, e.g., as part of a root cause analysis and to identify objects 812 compromised by the root cause. To this end, the analysis facility 840 may utilize one or more rules 842 for applying to the data included in the data log 822 to determine a root cause of a beacon or trigger event such as a suspected or actual security compromise on the endpoint 810. The analysis facility 840 may reside locally on the endpoint 810 (e.g., be a part of, embedded within, or locally coupled to the endpoint 810). The analysis facility 840 may be an external facility, or it may reside in a virtual appliance (e.g., which could be run by a protected set of systems on their own network systems), a private cloud, a public cloud, and so forth. The analysis facility 840 may store locally-derived threat information for use in subsequent identification, remediation, or other similar activity. The analysis facility 840 may also or instead receive threat information from a third-party source such as any public, private, educational, or other organization that gathers information on network threats and provides analysis and threat detection information for use by others. This third-party information may, for example, be used to improve detection rules or other forensic analysis that might be performed on information in the data log 822.

[00175] The analysis facility 840 may create an event graph. In general, the event graph may represent information in the data log 822 in a graph where objects 812 are nodes and events 814 are edges connecting the nodes to one another based on causal or other relationships as generally contemplated herein. The event graph may be used by the analysis facility 840 or other component(s) of the system 800 as part of a root cause analysis and to identify objects 812 compromised by the root cause. The event graph may also or instead be displayed to a user of the system 800 or endpoint 810, e.g., using an interactive user interface or the like.

[00176] The system 800 may advantageously use the data log 822 to configure and initialize an analysis in a sandboxed or otherwise isolated environment where the execution of the recorded activity related to a detected security event is allowed to run. That is, rather than uploading a complete image of an endpoint 810 using conventional techniques, the data log 822 may include only a series of events/processes related to the detected event that may be uploaded for execution/analysis. The analysis may thus include executing this series of events/processes in the same order to determine a threat level for the endpoint 810.

[00177] The data log 822 may include data from a single endpoint 810, or from a number of endpoints 810, for example where one endpoint 810 accesses a service or a file on another endpoint. This advantageously facilitates tracking or detection of potentially malicious

activity that spans multiple devices, particularly where the behavior on a single endpoint does not appear malicious. Thus, the monitoring facility 830 may monitor activity from an endpoint 810 exclusively, or use the full context of activity from all protected endpoints 810, or some combination of these. Similarly, the event graph generated from the data log 822 may include activity from one endpoint 810 exclusively, or use the full context of activity from all protected endpoints 810, or some combination of these. Data logs 822 and event graphs may also or instead be stored for future analyses, e.g., for comparing to future data logs and event graphs.

[00178] Fig. 9 is a flowchart of a method for forensic analysis for computer processes. The method 900 may be implemented by any of the systems described above or otherwise herein. The method 900 may be used as part of a root cause analysis, e.g., for determining a root cause of malware on an endpoint, and for identifying computing objects affected by malware, e.g., computing objects causally related to the root cause.

[00179] As shown in step 902, the method 900 may include monitoring events on a device, such as a first endpoint. The events may be any as described herein, e.g., events associated with computing objects on the endpoint. The computing objects may, for example include a data file, a process, an application, a registry entry, a network address, a peripheral device, or any of the other computing objects described herein. For example, in an aspect, the computing objects may include one or more network addresses specified at any suitable level of abstraction or according to any suitable protocol such as a uniform resource locator (URL), an Internet Protocol (IP) address, and a domain name, and may include any or a portion of associated path information or the like that might be associated therewith. The computing objects may also or instead include a peripheral device such as a universal serial bus (USB) memory, a camera, a printer, a memory card, a removable bulk storage device, a keyboard, a printer, a scanner, a cellular phone, or any other input or output device that might usefully be connected to an endpoint, a server, a mobile device, and so forth. Events may include information or messages from a threat management facility, firewall, network device, and so on, for example, that may be resident on or in communication with an endpoint. For example, a threat management facility may identify a potential or actual threat, and this may be treated as an event.

[00180] In an aspect, monitoring events on a first endpoint may include instrumenting a first endpoint to monitor a number of causal relationships among a number of computing objects. For example, a monitoring facility or other monitoring component (e.g., a component

disposed on the first endpoint or otherwise in communication with the first endpoint), may be configured to detect computing objects and to monitor events on the first endpoint that associate the computing objects in a number of causal relationships. Thus, a processor and a memory disposed on the endpoint may be configured to monitor events on the endpoint. A remote server may also or instead be configured to monitor events on the endpoint, for example, to create a data log as contemplated herein.

[00181] Implementations may also or instead include monitoring events on multiple endpoints, e.g., endpoints included in an enterprise network or the like. Thus, in an aspect, the one or more computing objects include at least one or more computing object(s) on a device other than the first endpoint, such as a second endpoint in the enterprise network. The device may also or instead include a server configured to provide remote resources to other endpoints, network devices, firewalls, gateways, routers, wireless access points, mobile devices, and so forth.

[00182] The causal relationships monitored by the system may include dependencies that form a link or an association between computing objects or events. Useful causal relationships may include a data flow, e.g., linking computing objects based on the flow of data from one computing object to another computing object. The causal relationships may also or instead include a control flow. For example, a first computer program may generate a first event that triggers a second computer program to trigger a second event, thereby creating a causal relationship between the first computer program and the second computer program (and possibly a causal relationship between the first event and the second event). In yet another aspect, the causal relationships may include a network flow. For example, a computing object may access a URL or other remote resource or location and receive data. In this example, there may be a causal relationship between one or more of the computing object, the URL, and the data. It will be understood that the term "causal relationship" and the like is intended to cover a wide range of relationships between computing objects that might be formed by events, and unless explicitly stated to the contrary or otherwise clear from the text, the causal relationships may include anything that can link or associate multiple computing objects (of the same type or different types), e.g., in a directional manner, directly or indirectly.

[00183] As shown in step 904, the method 900 may include recording events such as any of the events described above that occur on the endpoint. Thus, each event detected during monitoring may be recorded, e.g., by a data recorder or other component, to provide a data log

including a sequence of events causally relating the number of computing objects. As described above, the data recorder may be configured to record events that occur on the endpoint, or events that occur on a plurality of endpoints. The data recorder may be locally disposed on the endpoint or otherwise in communication with the endpoint. The data recorder may also or instead be associated with a monitoring facility or an analysis facility such as any of those described above. The data recorder may record a sequence of events causally relating a number of computing objects on one or more endpoints in a data log or the like disposed in a memory.

[00184] A number of events within the sequence of events may be preserved for a predetermined time window. For example, in an aspect, a data recorder or the like may record all activity on an endpoint in a rolling buffer that overwrites data that is older than the predetermined time window. This may be true regardless of the types of computing objects associated with the sequence of events. In another aspect, the predetermined time window may have a different duration for different types of computing objects (e.g., for at least two types of computing objects). By way of example, when the computing objects include one or more network addresses, the sequence of events may be preserved for a longer predetermined time window relative to a sequence of events associated with data files, or vice-versa. Similarly, when the computing objects include one or more peripheral devices such as USB memories, the sequence of events may be preserved for longer predetermined time window relative to a sequence of events associated with applications, or vice-versa. In implementations, the predetermined time window for which the sequence of events is preserved may be based on the likelihood of a security event originating from a certain type of computing object. For example, the reputation of a computing object (e.g., an application) or a machine state may be used for determining the duration of the predetermined time window for which the sequence of events is preserved. Further, the predetermined time window for which the sequence of events is preserved may be determined by a color of a computing object or event, e.g., as described in U.S. Pat. App. No. 14/485,759 filed on September 14, 2014, which is incorporated by reference herein in its entirety. In an aspect, the time window for which the sequence of events is preserved may be variable or adjustable. For example, a user or administrator using a user interface or the like may adjust the time window for which the sequence of events is preserved, e.g., based on computing object type or otherwise. For example, one or more first event types may be recorded with a first time window and one or more second event types may be recorded with a second time window.

[00185] In an aspect, the data recorder or the like may record only certain activity on an endpoint, e.g., activity associated with predetermined computing objects. The activity may be preserved for a predetermined amount of time dependent upon the specific computing object to which the activity is associated. In this manner, and by way of example, the data recorder or the like may include a record of data for one week for applications, for three months for files, for two weeks for registry entries, and so forth. It will be understood that these timeframes are provided by way of example and not of limitation.

[00186] In general, data may be continuously recorded, periodically recorded, or some combination of these. Furthermore, data may be cached, stored, deleted, or transmitted to a remote processing facility in any suitable manner consistent with appropriate use of local and remote resources, and the utility or potential utility of information that is being recorded. In one aspect, data may be periodically deleted or otherwise removed from the data recorder, such as after a security event has been detected and addressed as described below. A new data log may then be created for recording subsequent events on the one or more endpoints.

[00187] As shown in step 906, the method 900 may include evaluating one or more events that occur on the endpoint. The evaluation of the one or more events may include the application of one or more security rules to determine whether the one or more events indicate or suggest a security event such as a security compromise event, a data exposure, a malware detection, or the like. Thus, the evaluation of the one or more events may lead to the detection of a security event. While illustrated as a separate step, this step 906 may be performed concurrently with or in sequence with the monitoring step 902 discussed above.

[00188] The security event may be any beacon or trigger event, such as any of those discussed herein. The security event may include an event that is related to network security, computer security, data security, data leakage, data exposure, or any other actual or potential security issue. The security event may also or instead include other events of interest that are not directly related to computer/network security where, for example, they are useful for otherwise auditing or monitoring machines or characterizing device behavior. Thus, the security event may be any event general related to operation of a computer, and does not necessarily include an actual security compromise event. However, in implementations, the security event may include an actual compromise to a network, an endpoint, or a computer system such as the detection of malware or any other threat detection. For example, the security event may be a security compromise event related to a specific threat, e.g., an event related to computer-based malware

including without limitation a virus, spyware, adware, a Trojan, an intrusion, an advanced persistent threat, spam, a policy abuse, an uncontrolled access, and so forth.

[00189] Detecting the security event may include detecting a security compromise by applying a static analysis to software objects on the first endpoint. For example, each software object may be individually analyzed for its compliance with a security policy or the like using signatures or other objective characteristics. It will be understood that while static analysis provides one useful form of evaluation for compliance with the security policy or the like, other techniques may also or instead be employed, e.g., a behavioral analysis, a sandbox execution, network traffic analysis, and so forth.

[00190] Detecting the security event may also or instead include detecting a security compromise by applying dynamic or behavioral analysis to code executing on the first endpoint, or to specific computing objects (e.g., processes) on the endpoint. For example, events that can warrant triggering the detection of the security event may include a process that loads a particular file that is known to be malicious, or a process that accesses a known malicious IP address, and the like.

[00191] In an aspect, detecting the security event may include detecting a hardware change or other state changes. Detecting the security event may also or instead include detecting a potential data leakage.

[00192] As discussed herein, a security policy may be used to detect a security event. This may include, for example, whitelists or blacklists of known computing objects and events, or reputations and signatures thereof. For example, a security policy may include rules that allow computing objects and events that are provided by a known, trusted source (e.g., a trusted user, endpoint, network, company, vendor, and so forth). The rules may be more complex, for example, where originating from a trusted source is only one factor in determining whether to whitelist computing objects and events. In general, the security policy may include any suitable rules, logic, prioritizing, etc., as desired to detect a security event.

[00193] Although referred to herein in terms of 'security,' one skilled in the art will recognize that a security policy may also or instead include other types of policies. For example, a security policy may include a corporate or network policy having a list of approved computing objects and events, where computing objects and events outside of this list may not necessarily be security risks, but are otherwise unwanted in the network. Thus, the security policy may

intend to detect malware and the like, while also detecting other types of unwanted computing objects and events that do not qualify as malware.

[00194] More generally, any technique or combination of techniques suitable for evaluating endpoint activity for the detection of actual or potential security compromises may be used to detect security events as contemplated herein.

[00195] As shown in step 908, if a security event is not detected, the method 900 may return to step 902 where monitoring can continue. As further shown in step 908, if a security event is detected, a root cause analysis or the like may be performed to identify a source of the security event as further described below. That is, detecting a security event associated with one of the number of computing objects may trigger further analysis of other causally related computing objects on an endpoint (or in certain cases, remote from an endpoint) to identify a cause of the security event, as distinguished from the symptom that generated the beacon or trigger for the analysis.

[00196] As shown in step 910, the method 900 may include generating an event graph. The event graph may be generated in response to detecting the security event, e.g., using the data log from the data recorder. The event graph may be generated at the same time as or as part of creating the data log. The event graph may include the sequence of events causally relating the number of computing objects, and more specifically, the sequence of events and computer objects causally associated with the object(s) that triggered the detected security event.

[00197] As discussed herein, the event graph may be generated based on a data log of events and computer objects stored by a data recorder during operation of the endpoint. In particular the data recorder may provide a dump of logged activities, which may be causally associated into a graph for analysis, navigation, display and so forth. Any useful portion of the data log may be used. For example, the data recorder may provide event data for a window of time before, after or surrounding the detected security event. The data log may be filtered, e.g., when the data is written to the data log (for example, by aging events as described above) or when the event graph is generated, or some combination of these. A variety of filtering techniques may be usefully employed. For example, certain types of objects or events may be removed from an event graph for specific trigger events, or certain groups of events may be condensed into a single event, such as all normal activity that occurs when a user logs into an endpoint. Similarly, computing objects that are too remote, either within the event graph or timewise, may be pruned and removed, particularly if they have a known, low diagnostic

significance. Thus, the event graph may be filtered and condensed in a variety of manners to obtain a useful snapshot of events optimized for root cause analytics. Filtering of the data may be dependent upon the type of security event that is detected. Filtering of the data may adjust the level of detail included in the event graph based on memory limits, user parameters, security event type, or any other object metrics or inputs. In an aspect, the data is filtered based on reputation or the like, e.g., of computing objects included therein. For example, if an application has a good reputation, the application may not include a high level of detail associated therewith in a filtered version of the data log.

[00198] In one aspect, the event graph may be generated based on a data log from a number of different endpoints and thus may represent a causal chain of data from various different endpoints. This approach advantageously permits an analysis using data that spans multiple endpoints or other network devices within a single data structure or package, thus permitting identification of a root cause even when an attack employs a complex, multi-hop approach to network assets that might otherwise evade detection. Event graphs may also or instead be generated separately for different endpoints and presented to a user or analytical system as separate, discrete entities. Event graphs for endpoints may be compared with one another, e.g., as part of the root cause analysis. For example, by analyzing and comparing similar event graphs or event graphs sharing similar computing objects or events, a heuristic approach may be developed for identifying suspicious events and computing objects for one or more endpoints. Similarly, event graphs for different endpoints in the same network enterprise may be compared or combined, e.g., where two or more endpoints have been exposed to a security event or threat. For example, event graphs for similar time periods of two or more endpoints may be ascertained and analyzed.

[00199] In an aspect, cross-correlating between different data logs or event graphs may be utilized in a root cause analysis. For example, if the same security event or root cause is identified on different endpoints, the endpoints may be flagged for review or remediation. This type of analysis may be used on different endpoints throughout a network.

[00200] Implementations may include a number of different event graphs stored in a data store that can be used together to detect, prevent, or determine the root causes for suspicious activity or other activity of interest, e.g., a security event. As discussed herein, the event graphs may be filtered before being stored in the data store, which can remove system activity that is not of interest in such analyses. The event graphs may be searchable, e.g., for

analysis of event graphs including similar computing objects or events. The event graphs may also or instead be linked to one another, e.g., event graphs including similar computing objects or events. The event graphs may be presented to a user on a user interface or the like, e.g., an interactive user interface that allows a user to see similar or related event graphs, search the event graphs, link between event graphs, and so forth.

[00201] An event graph may use a conventional structure of nodes (computing objects) and events (edges) to represent causal relationships among computing objects. This permits the use of a wide range of graph-based techniques to assist in analysis of the context leading up to a detected event. At the same time, numerous other data structures, computer representations, and visual representations of such interrelated objects and events are also known in the art, any of which may be employed as an event graph as contemplated herein, provided that enough descriptive data about the context of an endpoint is captured to facilitate the various types of analysis and response contemplated herein.

[00202] As shown in step 912, the method 900 may include, in response to detecting the security event, traversing the event graph based on the sequence of events in a reverse order from the one of the computing objects associated with the security event to one or more preceding ones of the computing objects. In general, the reverse order is a causally reverse order. For example, where a network flow, data flow or control flow has a direction from one computing object to another computing object, the reverse order will follow this flow or causal link from the receiving computing object backward toward the source computing object. However, this may also or instead include a chronological flow, such as in a complex event graph where the time of receipt for two different inputs from two different sources is relevant. In general, a review of each of the preceding computing objects may be conducted by working backward from the computing object associated with the security event, e.g., to determine a root cause of the security event. In an aspect, this may include a static analysis of each of the preceding computing objects, or a dynamic analysis of object and event interactions, or some combination of these.

[00203] As shown in step 914, the method 900 may include applying one or more rules to the computing objects preceding the security event. For example, the method 900 may include applying a cause identification rule to the preceding ones of the computing objects and the causal relationships while traversing the event graph in order to identify one of the computing objects as a cause of the security event. In general, the root cause analysis may attempt to

identify a pattern in the event graph using cause identification rules to identify one of the computing objects (or a group of the computing objects and events) as a root cause of the security event.

[00204] The cause identification rule may associate the cause with one or more common malware entry points. For example, common entry points include a word processing application, an electronic mail application, a spreadsheet application, a browser, or a universal serial bus (USB) drive that is attached to an endpoint, and any of these computing objects, when encountered in an event graph, may be identified as a root cause. For example, when traversing the event graph in a reverse order from the security event, if the analysis identifies an electronic mail application that opened an attachment, this may be identified as the root cause because this is often a source of compromised security on an endpoint. Similarly, when traversing the event graph in a reverse order from the security event, if the analysis identifies a USB drive, or an unsecure or unencrypted USB drive, from which a file was opened, this may be identified as a likely cause of the security event. In one aspect, multiple candidate root causes may be identified using the cause identification rules, and a final selection may be based on other contextual information such as reputation, source, etc.

[00205] Security events may also or instead be caused by a certain combination of events or combinations of events and computing objects. For example, in an aspect, the cause identification rule may associate the cause of the security event with a combination that includes a first process invoking a second process and providing data to the second process. As used herein, invoking may be interpreted broadly, e.g., where any two processes share data through an intermediate file, or narrowly, e.g., where a first process specifically spawns the second process as a child process. More generally, invoking a process as used herein is intended to broadly include any causal relationship between to processes including, e.g., spawning a process, hijacking a process (e.g., seizing control of an existing process through thread injection, process hollowing, and the like), remotely launching a process over a network, instrumenting a service in the operating system, and the like. A cause identification rule may specify a particular type of invocation relationship between two processes, or multiple types of invocation, or any relationship between two processes. Providing data from a first process to a second process may include creating a file for use by the second process. For example, the cause of a security event may include a first process that writes a file and then takes control of a second process that reads data from the file so that the first process and the second process share data through the file.

[00206] Another example of a security event may include a known non-malicious application (e.g., a commonplace word processing application) launching a command line script, which may be identified as a cause of a security event. The activity underlying events that are generated may not necessarily be malicious, but they could lead to security events or other events of interest to be further analyzed. Thus, in one aspect, a cause identification rule may flag this behavior as a root cause of a security event, or as an event that is otherwise of diagnostic interest.

[00207] As shown in step 916, the method 900 may include traversing the event graph forward from an identified or presumed cause of the security event to identify one or more other ones of the computing objects affected by the cause. In this manner, an analysis of each of the computing objects in the event graph may be conducted by working forward from the root cause to other causally dependent computing objects that might be compromised or otherwise affected by the root cause. This may include labeling or otherwise identifying the potentially compromised objects, e.g., for remediation or further analysis. A pruning step may also be employed, e.g., where any computing objects that are not causally dependent on the root cause in some way are removed from the event graph.

[00208] As shown in step 918, the method 900 may include remediating one or more computing objects affected by the cause of the security event. Remediation may include deleting computing objects from the endpoint, or otherwise remediating the endpoint(s) using computer security techniques such as any described herein. In another aspect, the identification of the root cause may be used to create new detection rules capable of detecting a security event at a point in time (or causation) closer to the root cause within the event graph. Other remediation steps may include forwarding the event graph, or a filtered and pruned event graph, to a remote facility for analysis. This data may usefully provide a map for identifying sources of malware, or for ensuring thorough remediation by identifying all of the potentially compromised computing objects that should be examined after the compromise has been addressed.

[00209] Fig. 10 illustrates a graphical depiction of a portion of an example event graph 1000. The event graph 1000 may include a sequence of computing objects causally related by a number of events, and which provide a description of computing activity on one or more endpoints. The event graph 1000 may be generated, for example, when a security event 1002 is detected on an endpoint, a gateway, or a communications server, and may be based on an identification of a malicious action in real-time or in near real-time, or on a data log or similar

records obtained by an event data recorder during operation of the endpoint, gateway, or communications server. The event graph 1000 may be used to determine a root cause 1004 of the security event 1002 as generally described above. The event graph 1000 may also or instead be continuously generated to serve as, or be a part of, the data log obtained by the data recorder. In any case, an event graph 1000, or a portion of an event graph 1000 in a window before or around the time of a security event, may be obtained and analyzed after a security event 1002 occurs determine its root cause 1004. The event graph 1000 depicted in the figure is provided by way of example only, and it will be understood that many other forms and contents for event graphs 1000 are also or instead possible. It also will be understood that the figure illustrates a graphical depiction of an event graph 1000, which may be stored in a database or other suitable data structure.

[00210] By way of example, the event graph 1000 depicted in the figure begins with a computing object that is a USB device 1012, which may be connected to an endpoint. Where the USB device 1012 includes a directory or file system, the USB device 1012 may be mounted or accessed by a file system on an endpoint to read contents. The USB device 1012 may be detected 1013 and contents of the USB device 1012 may be opened 1014, e.g., by a user of the endpoint. The USB device 1012 may include one or more files and application, e.g., a first file 1016, a second file 1018, and a first application 1020. The first file 1016 may be associated with a first event 1022 and the second file may be associated with a second event 1024. The first application 1020 may access one or more files on the endpoint, e.g., the third file 1026 shown in the figure. The first application 1020 may also or instead perform one or more actions 1028, such as accessing a URL 1030. Accessing the URL 1030 may download or run a second application 1032 on the endpoint, which in turn accesses one or more files (e.g., the fourth file 1034 shown in the figure) or is associated with other events (e.g., the third event 1036 shown in the figure).

[00211] In the example provided by the event graph 1000 depicted in the figure, the detected security event 1002 may include the action 1028 associated with the first application 1020, e.g., accessing the URL 1030. By way of example, the URL 1030 may be a known malicious URL or a URL or network address otherwise associated with malware. The URL 1030 may also or instead include a blacklisted network address that although not associated with malware may be prohibited by a security policy of the endpoint or enterprise network in which the endpoint is a participant. The URL 1030 may have a determined reputation or an unknown

reputation. Thus, accessing the URL 1030 can be detected through known computing security techniques.

[00212] In response to detecting the security event 1002, the event graph 1000 may be traversed in a reverse order from a computing object associated with the security event 1002 based on the sequence of events included in the event graph 1000. For example, traversing backward from the action 1028 leads to at least the first application 1020 and the USB device 1012. As part of a root cause analysis, one or more cause identification rules may be applied to one or more of the preceding computing objects having a causal relationship with the detected security event 1002, or to each computing object having a causal relationship to another computing object in the sequence of events preceding the detected security event 1002. For example, other computing objects and events may be tangentially associated with causally related computing objects when traversing the event graph 1000 in a reverse order—such as the first file 1016, the second file 1018, the third file 1026, the first event 1022, and the second event 1024 depicted in the figure. In an aspect, the one or more cause identification rules are applied to computing objects preceding the detected security event 1002 until a cause of the security event 1002 is identified.

[00213] In the example shown in the figure, the USB device 1012 may be identified as the root cause 1004 of the security event 1002. In other words, the USB device 1012 was the source of the application (the first application 1020) that initiated the security event 1002 (the action 1028 of accessing the potentially malicious or otherwise unwanted URL 1030).

[00214] The event graph 1000 may similarly be traversed going forward from one or more of the root cause 1004 or the security event 1002 to identify one or more other computing objects affected by the root cause 1004 or the security event 1002. For example, the first file 1016 and the second 1018 potentially may be corrupted because the USB device 1012 included malicious content. Similarly, any related actions performed after the security event 1002 such as any performed by the second application 1032 may be corrupted. Further testing or remediation techniques may be applied to any of the computing objects affected by the root cause 1004 or the security event 1002.

[00215] The event graph 1000 may include one or more computing objects or events that are not located on a path between the security event 1002 and the root cause 1004. These computing objects or events may be filtered or 'pruned' from the event graph 1000 when performing a root cause analysis or an analysis to identify other computing objects affected by

the root cause 1004 or the security event 1002. For example, computing objects or events that may be pruned from the event graph 1000 may include a USB drive and the USB device being detected 1013.

[00216] It will be appreciated that the event graph 1000 depicted in Fig. 10 is an abstracted, simplified version of actual nodes and events on an endpoint for demonstration. Numerous other nodes and edges will be present in a working computing environment. For example, when a USB device is coupled to an endpoint, the new hardware will first be detected, and then the endpoint may search for suitable drivers and, where appropriate, present a user inquiry of how the new hardware should be handled. A user may then apply a file system to view contents of the USB device and select a file to open or execute as desired, or an autorun.exe or similar file may be present on the USB device that begins to execute automatically when the USB device is inserted. All of these operations may require multiple operating system calls, file system accesses, hardware abstraction layer interaction, and so forth, all of which may be discretely represented within the event graph 1000, or abstracted up to a single event or object as appropriate. Thus, it will be appreciated that the event graph 1000 depicted in the drawing is intended to serve as an illustrative example only, and not to express or imply a particular level of abstraction that is necessary or useful for root cause identification as contemplated herein.

[00217] The event graph 1000 may be created or analyzed using rules that define one or more relationships between events and computing objects. The C Language Integrated Production System (CLIPS) is a public domain software tool intended for building expert systems, and may be suitably adapted for analysis of a graph such as the event graph 1000 to identify patterns and otherwise apply rules for analysis thereof. While other tools and programming environments may also or instead be employed, CLIPS can support a forward and reverse chaining inference engine suitable for a large amount of input data with a relatively small set of inference rules. Using CLIPS, a feed of new data can trigger a new inference, which may be suitable for dynamic solutions to root cause investigations.

[00218] An event graph such as the event graph 1000 shown in the figure may include any number of nodes and edges, where computing objects are represented by nodes and events are represented by edges that mark the causal or otherwise directional relationships between computing objects such as data flows, control flows, network flows and so forth. While processes or files are common forms of nodes that might appear in such a graph, any other

computing object such as an IP address, a registry key, a domain name, a uniform resource locator, a command line input or other object may also or instead be designated to be a node in an event graph as contemplated herein. Similarly, while an edge may be formed by an IP connection, a communication, a file read, a file write, a process invocation (parent, child, etc.), a process path, a thread injection, a registry write, a domain name service query, a uniform resource locator access and so forth other edges may be designated. As described above, when a security event is detected, the source of the security event may serve as a starting point within the event graph 1000, which may then be traversed backward to identify a root cause using any number of suitable cause identification rules. The event graph 1000 may then usefully be traversed forward from that root cause to identify other computing objects that are potentially tainted by the root cause so that a more complete remediation can be performed. The event graph 1000 may include events associated with one or more different endpoints, gateways, or communications servers.

[00219] Fig. 11 shows an architecture for instrumenting an endpoint. In general, an endpoint 1100 may include a memory having a user mode 1102 providing general memory for use by user applications and processes, and a kernel mode 1104 providing protected memory exclusively for use by an operating system. Instrumentation for monitoring and remediation may, for example, be hooked into existing operating code, e.g., by injection of dynamic linked libraries into operating system code early in the boot process, preferably before the kernel libraries (e.g., kernel32.dll, for a contemporary Windows operating system) are loaded. These dynamic linked libraries, labeled hmpalert.dll in Fig. 11, may provide an interface for programmatic access, or hooks, to code for accessing the operating system cryptography services (such as crypt32.dll, a library that implements various certificate and cryptographic functions in the Windows cryptographic application programming interface) so that execution and input/output for these kernel functions can be monitored during operation of the endpoint.

[00220] Fig. 12 shows a process for accessing encrypted information on an endpoint. In general, secrets 1202 such as cookies, tokens, credentials, authentication material, encryption keys, and so forth, may be associated with a session, a user, a group, an application, or other entity or object, and may be stored in an encrypted form on an endpoint. For example, a web browser application may store cookies so that a user can interact with a secure, remote resource without manually re-authenticating. These secrets 1202 may be encrypted using a key such as a cryptographic key for the Advanced Encryption Standard ("AES"), a symmetric block cipher

(including three distinct block ciphers, AES-128, AES-192 and AES-256, with different key lengths) that uses the same key for encryption and decryption.

[00221] While this layer of encryption provides some security, the underlying secrets 1202 may still be exposed to an adversary who can obtain or control use of the AES key. In order to further protect against such exploits, the AES key may itself be encrypted. For example, the Windows operating system provides a cryptographic application programming interface as a built-in component, the Data Protection Application Programming Interface (DPAPI), that uses a main cryptographic key or master key 1204 derived from a user's password. The DPAPI is stateless and does not store any persistent data itself. Instead, the DPAPI simply receives plaintext and returns ciphertext or vice versa, while relying on the operating system to protect the master key and control underlying cryptographic functions through the programming interface, e.g., through a local security authentication server known as lsass.exe in the Windows operating system.

[00222] In this environment, access to a stored secret, such as a cookie or session token, involves several steps. First, a request for the stored secret is received. Then a DPAPI blob—an opaque binary structure containing encrypted data—with the encrypted AES key 1206 is retrieved and transmitted to the DPAPI (or any similar operating system cryptographic resource) for decryption with the user's master key 1204. The decrypted AES key 1208 may then be used to decrypt the cookie for use in authentication, e.g., in order to access resources without further authentication. This multi-layered technique permits seamless access by a user without generally exposing secrets or encryption keys. However, these layers of protection do not inherently prevent programmatic requests for use of the AES key, thus opening the door to so-called pass-the-cookie attacks in which cookies are maliciously obtained by presenting improper requests to the data protection application programming interface. For example, Mimikatz is an open-source application that allows users to view authentication credentials on a Windows machine, and may be used to unprotect and steal protected cookies or login credentials stored by a web browser. ChromePass is a password recovery utility that allows users to similarly view in plaintext any usernames and passwords stored by the Chrome web browser. While these tools typically require the user to be logged in to the compute instance storing the secret, or otherwise require a user to provide authentication credentials to the operating system in order to obtain the master key used by the operating system's cryptographic infrastructure, they illustrate existing tools that

are available for requesting access to secrets that include a layer of operating system cryptographic protection.

[00223] As a significant advantage, the following techniques can prevent an attack aimed at improperly decrypting a local secret before the attack can present a request to the operating system's decryption resources or otherwise gain access to the key that cryptographically secures the secret. It will be understood that while much of the description herein focuses on interfaces and other resources of the Windows operating system, the principles described herein may also or instead be applied to other operating systems and that references to Windows-specific computing objects should be understood to be general in nature. For example, unless otherwise indicated, an AES key should be understood to refer to any symmetric key used to encrypt local secrets, a DPAPI Master Key should be understood to refer to any key used by an operating system to encrypt/decrypt a data blob, the Data Protection Application Programming Interface should be understood to refer to any programmatic interface for accessing cryptographic tools or libraries of an operating system, and so forth. Furthermore, while symmetric keys are commonly used for the local encryption/decryption operations described herein, asymmetric keys may also or instead be used.

[00224] Fig. 13 illustrates malware attempting to access a decrypted key. A web browser or similar application can securely store a cookie associated for example with an authenticated user or session for a connection with a remote, secure resource using the techniques above and may initiate re-use of the token by requesting decryption of a key through a cryptographic resource of the operating system. One technique for improperly accessing the key is to simply request access to a cookie store for the web browser, or more specifically to request decryption of the key used to encrypt the session token (which implicitly accesses the DPAPI or similar programming interface of the operating system). By hooking the functional interface to these operating system functions, e.g., the crypt32.dll linked library of the Windows operating system, to detect requests for decryption, a new request can be assessed for possible malicious activity.

[00225] In one aspect, parameters of the request or other context for the request may be evaluated for malicious activity. For example, while the browser application that uses the cookie store may properly access the cookie store to recover session tokens and other information, other applications typically cannot. Thus, a process (or application) requesting the access may be examined to determine if the process is a proper user of the cookie store (or other secure cache

or repository). If access to the cookies is requested by a process other than the web browser (or some other authorized user such as a malware scanner, backup utility, or privacy cleaner), this diagnostic pair (the requesting application and the data requested) may be used to evaluate for the presence of malicious activity. If the requesting application is not the owner of the data requested or an authorized user of the data requested, then the request may be denied. While the requesting application may be identified directly based on the process name, other information may also or instead be used to identify the application, such as the code signing certificate on the requesting application, the source path for the requesting process, or an application that is explicitly associated with the requesting process. More generally, by identifying a secret that should be protected and the application(s) that are permitted to access the secret, a corresponding detection rule can be created around context observable when the instrumented programming interface (e.g., for a decryption by the operating system) is accessed.

[00226] Other remedial action may also or instead be undertaken. For example, the requesting process may be scanned for malware and/or terminated. The endpoint executing the process may be scanned and/or quarantined by preventing new network connections or terminating existing network connections. Additionally or alternatively, a beacon or similar alert may be generated based on the event and transmitted to a threat management facility for automated action, or to an administrator for manual review and intervention. Additionally or alternatively, a root cause analysis may be performed to locate earlier events associated with the malicious activity, and/or to identify other processes or computing objects that might be affected.

[00227] Fig. 14 illustrates malware attempting to access a decrypted key. In a system that checks the name of a requesting process or application, e.g., as described above, a secret such as an encrypted web session token may nonetheless be vulnerable to an attack where the code of a process that would be permitted to access the secret is maliciously modified with shellcode or the like injected into the process that hijacks flow control for the process. While various malware tools exist to detect shellcode creation/deployment, such as tools to prevent access to operating system API's from dynamic memory, a shellcode exploit directed at decryption of the AES key may provide additional clues to the presence of malware that may be detected and used to prevent theft of local secrets. For example, when the call to the operating system decryption resource is made, the specific calling origin in memory, or the calling code from the process may be checked, or a signature may be checked for the executable, or behavior

of the process may be examined (e.g., to determine whether the request is associated with access by a user to the corresponding remote resource), and so forth.

[00228] Fig. 15 illustrates malware attempting to access a decrypted key. One exemplary class of exploit uses dynamic linked library preloading to substitute malicious libraries for legitimate ones. Where the operating system interface (e.g., ntdll.dll, for Windows) or cryptographic library is hooked (e.g., via hmpalert.dll), the calling library may be checked before access is granted to the data protection application programming interface. For example, the name of the DLL, the path for the DLL, a signature for the DLL (either a cryptographic signature associated with the DLL, or a hash or other signature useful for identifying software), or any other contextual data or the like, may be used to evaluate the calling library. If the module name or other DLL information for the calling process does not match an authorized user of the AES key, the request may be blocked and remedial action such as any of the remedial action described herein may be initiated.

[00229] Fig. 16 shows a method for detecting access to an encrypted secret stored on a compute instance. In general, a data protection application programming interface or the like may be instrumented to detect when an operating system's cryptographic resources are being used to decrypt a stored secret such as a session cookie so that a security evaluation can be performed to determine whether the requesting process is an authorized user of the secret.

[00230] As shown in step 1602, the method 1600 may begin with instrumenting the appropriate application programming interface. This may, for example, include instrumenting a data protection application programming interface for an operating system on an endpoint to detect access to a decryption service used by the operating system to encrypt and decrypt data blobs with a master key derived from user credentials for the endpoint. More generally, this may include instrumenting an application programming interface on a compute instance to detect access to a decryption service used by an operating system of the compute instance, or to otherwise detect access to operating system cryptographic tools used to protect secrets on the compute instance. It will be appreciated that the programming hooks for detecting calls to an operating system API are preferably installed as early in the startup of a process as possible in order to reduce opportunities for malicious intervention as each process launches.

[00231] In one aspect, the application programming interface is a data protection application programming interface for the operating system, or some other interface for accessing decryption resources of the operating system. The decryption services themselves may

execute in the kernel space, the user space, or some combination of these. For an operating system such as Windows, the decryption service may encrypt and decrypt using a master key derived from user credentials for the compute instance.

[00232] As shown in step 1604, the method 1600 may include detecting a call from a first process executing on the compute instance to the application programming interface to unprotect a key used to cryptographically secure a secret on the compute instance. For example, this may include detecting a call from a process executing on the endpoint to the data protection application programming interface to unprotect a symmetric key used to cryptographically secure a web browser session cookie stored by a web browser application on the endpoint. More generally, the secret may include logon credentials stored in an application cache and encrypted with the key, web browser session cookie, a cryptographic key, or some other cookie, token, credential, or other protected item of information.

As shown in step 1606, the method 1600 may include evaluating the calling process, for example, by comparing first process information for the first process to second process information for one or more other processes associated with the secret. For a web browser storing a web browser session cookie, this may include comparing first process information for the process to second process information for the web browser application that stored the web browser session cookie. The process information may include any process information useful for identifying the process or a source thereof. For example, the process information may include one or more of a process name, a process identifier, an application name, one or more registry keys for the process, and a path such as a path that stores executable code for the process or a path that is used by the process for other storage and retrieval during execution. In one aspect, the process information may include a hash used to identify the process, or a digital signature or the like for verifying an identity or source of the process. In another aspect, the process information may include a DLL, a module, a calling function, a calling location, or any other information that may be used to identify the calling process or application, or to otherwise identify or verify the source of the call.

[00233] In general, the process may be any process able to call the instrumented programming interface. The one or more other processes (to which the process is compared) may include any process that might usefully and legitimately store or access a secret on the compute instance. For example, the one or more other processes may include a process associated with a web browser application, a process for a privacy cleaner, a process for a

backup utility, or any process associated with an authorized application that created or stored the secret.

As shown in step 1608, the method may include determining whether the calling process is an authorized user of the secret and, where appropriate, taking responsive action. For example, in response to determining that the first process is an authorized user of the secret, this may include permitting the first process to decrypt the key for use in accessing the secret and returning to step 1604 where monitoring can continue for additional calls to the application programming interface. In response to determining that the first process is not an authorized user of the secret, this may include preventing the first process from decrypting the key for use in accessing the secret and proceeding to step 1610 for possible remediation(s). In the web browser example above, this may include, in response to determining that the process is not associated with the web browser application, preventing the process from accessing the web browser session cookie with the key and initiating a remediation of the endpoint.

As shown in step 1610, the method 1600 may include, when the first process is not an authorized user, initiating a remediation. This may include any of the remediations described herein, or any other remediation or combination of remediations suitable for responding to a suspected unauthorized access to a secret stored on a compute instance. For example, the remediation may include terminating the calling process. The remediation may also or instead include quarantining the compute instance. In another aspect, the remediation may include performing a malware scan, either of the process specifically, or a directory where the code for the process is located, or the entire compute instance. In another aspect, the remediation may include generating a beacon identifying the calling process, which may be transmitted to a threat management facility for storage and response, and/or to an administrator for manual intervention, and/or to a user of the compute instance, e.g., as a display and request for intervention through a user interface of the compute instance. In another aspect, the remediation may include performing a root cause analysis to identify a source of the calling process, or any relevant preceding events. The root cause analysis may be useful in a variety of ways, including identifying a source of the malicious action, identifying other affected computing objects on the compute instance, and creating improved tools for earlier detection based on preceding events. Upon initiation or completion of any such remediation, the method 1600 may return to step 1604 where the application programming interface may be monitored for additional calls.

[00234] According to the foregoing, there is described herein a compute instance comprising a memory in a user space of an operating system, the memory storing a first key encrypted with a master key derived from user credentials for the compute instance and a secret encrypted with the first key; an application programming interface configured to provide programmatic access to cryptographic tools of the operating system based on the master key; and a security function hooked to the application programming interface, the security function configured to detect a request to decrypt the first key with the application programming interface, and to determine whether a process requesting decryption of the first key is an authorized user of the secret. The secret may a web browser session cookie and the security function may be configured to detect whether the process is associated with a web browser that stored the web browser session cookie. More generally, the secret may include any of the secrets described herein.

[00235] Embodiments disclosed herein may include computer program products comprising computer-executable code or computer-usable code that, when executing on one or more computing devices, performs any and/or all of the steps thereof. The code may be stored in a non-transitory fashion in a computer memory, which may be a memory from which the program executes (such as random-access memory associated with a processor), or a storage device such as a disk drive, flash memory or any other optical, electromagnetic, magnetic, infrared, or other device or combination of devices. In another aspect, any of the systems and methods described above may be embodied in any suitable transmission or propagation medium carrying computer-executable code and/or any inputs or outputs from same.

[00236] The elements described and depicted herein, including in flow charts and block diagrams throughout the figures, imply logical boundaries between the elements. However, according to software or hardware engineering practices, the depicted elements and the functions thereof may be implemented on machines through computer executable media having a processor capable of executing program instructions stored thereon as a monolithic software structure, as standalone software modules, or as modules that employ external routines, code, services, and so forth, or any combination of these, and all such implementations may be within the scope of the present disclosure. Examples of such machines may include, but may not be limited to, personal digital assistants, laptops, personal computers, mobile phones, other handheld computing devices, medical equipment, wired or wireless communication devices, transducers, chips, calculators, satellites, tablet PCs, electronic books, gadgets, electronic

devices, devices having artificial intelligence, computing devices, networking equipment, servers, routers, and the like. Furthermore, the elements depicted in the flow chart and block diagrams or any other logical component may be implemented on a machine capable of executing program instructions. Thus, while the foregoing drawings and descriptions set forth functional aspects of the disclosed systems, no particular arrangement of software for implementing these functional aspects should be inferred from these descriptions unless explicitly stated or otherwise clear from the context. Similarly, it may be appreciated that the various steps identified and described above may be varied, and that the order of steps may be adapted to particular applications of the techniques disclosed herein. All such variations and modifications are intended to fall within the scope of this disclosure. As such, the depiction and/or description of an order for various steps should not be understood to require a particular order of execution for those steps, unless required by a particular application, or explicitly stated or otherwise clear from the context. Absent an explicit indication to the contrary, the disclosed steps may be modified, supplemented, omitted, and/or re-ordered without departing from the scope of this disclosure.

[00237] The method steps of the implementations described herein are intended to include any suitable method of causing such method steps to be performed, consistent with the patentability of the following claims, unless a different meaning is expressly provided or otherwise clear from the context. So, for example performing the step of X includes any suitable method for causing another party such as a remote user, a remote processing resource (e.g., a server or cloud computer) or a machine to perform the step of X. Similarly, performing steps X, Y and Z may include any method of directing or controlling any combination of such other individuals or resources to perform steps X, Y and Z to obtain the benefit of such steps. Thus, method steps of the implementations described herein are intended to include any suitable method of causing one or more other parties or entities to perform the steps, consistent with the patentability of the following claims, unless a different meaning is expressly provided or otherwise clear from the context. Such parties or entities need not be under the direction or control of any other party or entity and need not be located within a particular jurisdiction.

[00238] It will be appreciated that the methods and systems described above are set forth by way of example and not of limitation. Numerous variations, additions, omissions, and other modifications will be apparent to one of ordinary skill in the art. In addition, the order or presentation of method steps in the description and drawings above is not intended to require

this order of performing the recited steps unless a particular order is expressly required or otherwise clear from the context. Thus, while particular embodiments have been shown and described, it will be apparent to those skilled in the art that various changes and modifications in form and details may be made therein without departing from the spirit and scope of this disclosure and are intended to form a part of the invention as defined by the following claims, which are to be interpreted in the broadest sense allowable by law.

CLAIMS

What is claimed is:

1. A method comprising:

instrumenting an application programming interface on a compute instance to detect access to a decryption service used by an operating system of the compute instance;

detecting a call from a first process executing on the compute instance to the application programming interface to unprotect a key used to cryptographically secure a secret on the compute instance;

comparing first process information for the first process to second process information for one or more other processes associated with the secret;

in response to determining that the first process is an authorized user of the secret, permitting the first process to decrypt the key for use in accessing the secret; and

in response to determining that the first process is not an authorized user of the secret, preventing the first process from decrypting the key for use in accessing the secret.

2. The method of claim 1, further comprising, when the first process is not an authorized user, initiating a remediation.

3. The method of claim 2, wherein the remediation includes terminating the first process.

4. The method of either of claims 2 or 3, wherein the remediation includes quarantining the compute instance.

5. The method of any of claims 2 to 4, wherein the remediation includes performing a malware scan.

6. The method of any of claims 2 to 5, wherein the remediation includes generating a beacon identifying the first process.

7. The method of any of claims 2 to 6, wherein the remediation includes performing a root cause analysis.

8. The method of any preceding claim, wherein the first process information includes one or more of a process name, a process identifier, an application name, and a path.

9. The method of any preceding claim, wherein the one or more other processes include at least one process associated with a web browser application.

10. The method of any preceding claim, wherein the one or more other processes include a process for an authorized application that stored the secret.

11. The method of any preceding claim, wherein the one or more other processes include a process for an application including one or more of a privacy cleaner and a backup utility.

12. The method of any preceding claim, wherein the secret includes a web browser session cookie.

13. The method of any preceding claim, wherein the secret includes logon credentials stored in an application cache and encrypted with the key.

14. The method of any preceding claim, wherein the secret includes one or more of a token, a cookie, a credential, and a cryptographic key.

15. The method of any preceding claim, wherein the decryption service encrypts and decrypts using a master key derived from user credentials for the compute instance.

16. The method of any preceding claim, wherein the application programming interface is a data protection application programming interface for the operating system.

17. The method of any preceding claim, wherein the application programming interface accesses decryption resources in a kernel of the operating system.

18. A compute instance comprising:

a memory in a user space of an operating system, the memory storing a first key encrypted with a master key derived from user credentials for the compute instance and a secret encrypted with the first key;

an application programming interface configured to provide programmatic access to cryptographic tools of the operating system based on the master key; and

a security function hooked to the application programming interface, the security function configured to detect a request to decrypt the first key with the application programming interface, and to determine whether a process requesting decryption of the first key is an authorized user of the secret.

19. The compute instance of claim 18, wherein the secret is a web browser session cookie and wherein the security function is configured to detect whether the process is associated with a web browser that stored the web browser session cookie.

20. A computer program product comprising computer executable code embodied in a computer readable medium that, when executing on one or more computing devices, performs the steps of:

instrumenting a data protection application programming interface for an operating system on an endpoint to detect access to a decryption service used by the operating system to encrypt and decrypt data blobs using a master key derived from user credentials for the endpoint;

detecting a call from a process executing on the endpoint to the data protection application programming interface to unprotect a symmetric key used to cryptographically secure a web browser session cookie stored by a web browser application on the endpoint;

comparing first process information for the process to second process information for the web browser application that stored the web browser session cookie; and

in response to determining that the process is not associated with the web browser application, preventing the process from accessing the web browser session cookie with the key and initiating a remediation of the endpoint.

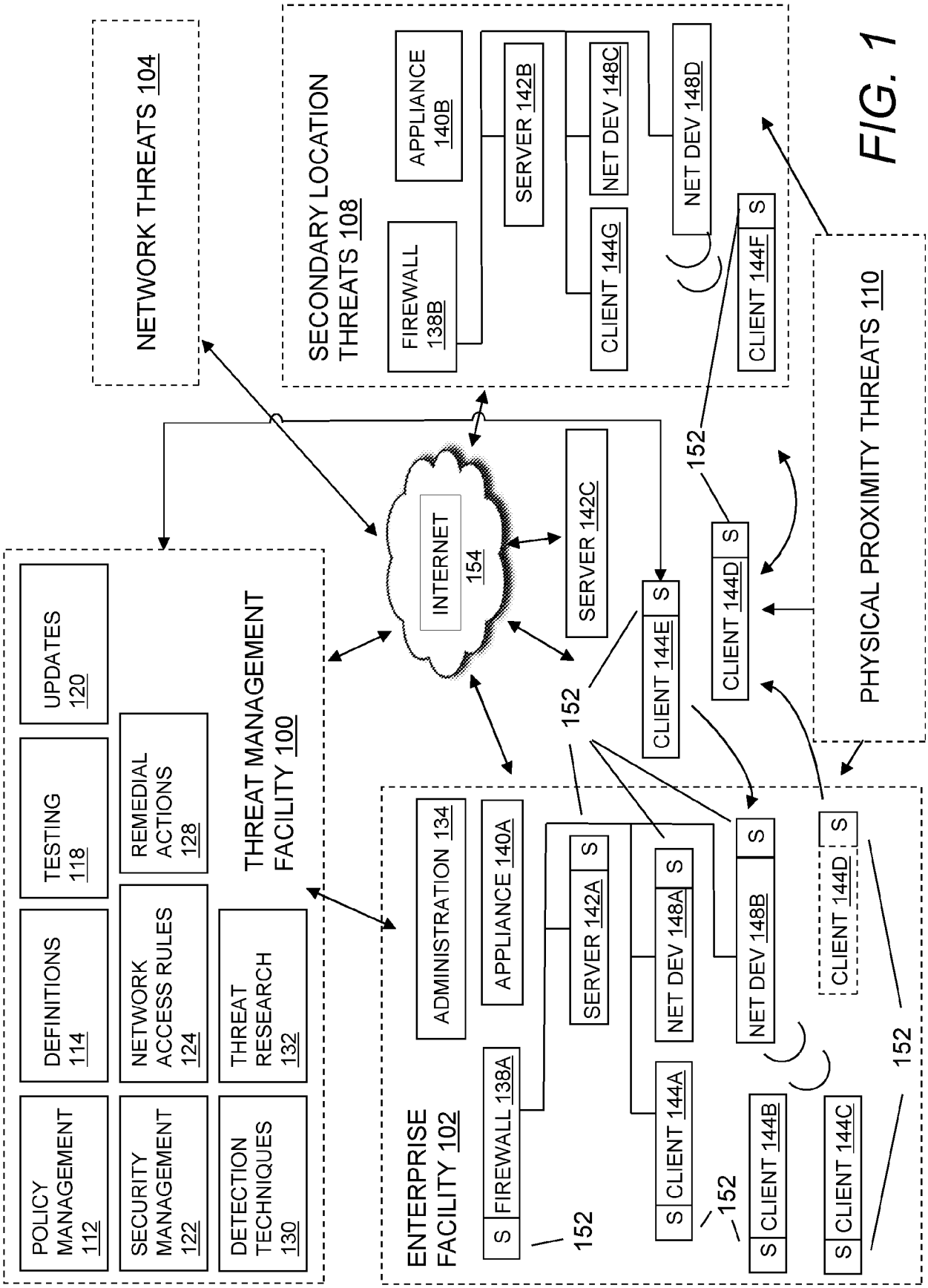
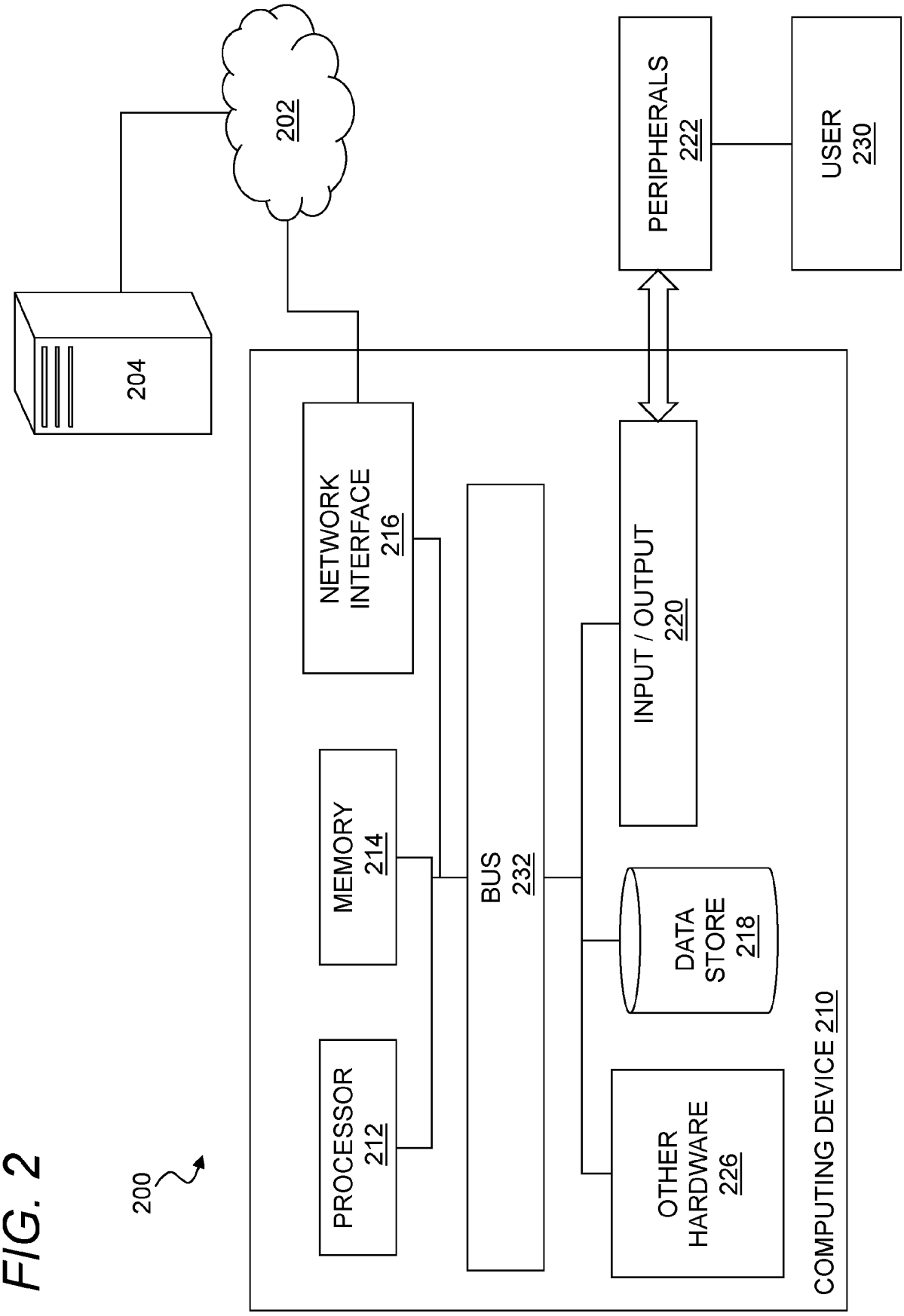

NETWORK THREATS 104
SECONDARY LOCATION THREATS 108
FIREWALL 138B
APPLIANCE 140B
SERVER 142B
NET DEV 148C
NET DEV 148D
CLIENT 144G
CLIENT 144F
S
INTERNET 154
SERVER 142C
152
CLIENT 144E
CLIENT 144D
PHYSICAL PROXIMITY THREATS 110
THREAT MANAGEMENT FACILITY 100
POLICY MANAGEMENT 112
DEFINITIONS 114
TESTING 118
UPDATES 120
SECURITY MANAGEMENT 122
NETWORK ACCESS RULES 124
REMEDIAL ACTIONS 128
DETECTION TECHNIQUES 130
THREAT RESEARCH 132
ENTERPRISE FACILITY 102
ADMINISTRATION 134
APPLIANCE 140A
SERVER 142A
NET DEV 148A
NET DEV 148B
CLIENT 144D
S FIREWALL 138A
S CLIENT 144A
S CLIENT 144B
S CLIENT 144C

FIG. 1

FIG. 2


200
202
204
PERIPHERALS 222
USER 230
NETWORK INTERFACE 216
INPUT / OUTPUT 220
MEMORY 214
BUS 232
DATA STORE 218
PROCESSOR 212
OTHER HARDWARE 226
COMPUTING DEVICE 210

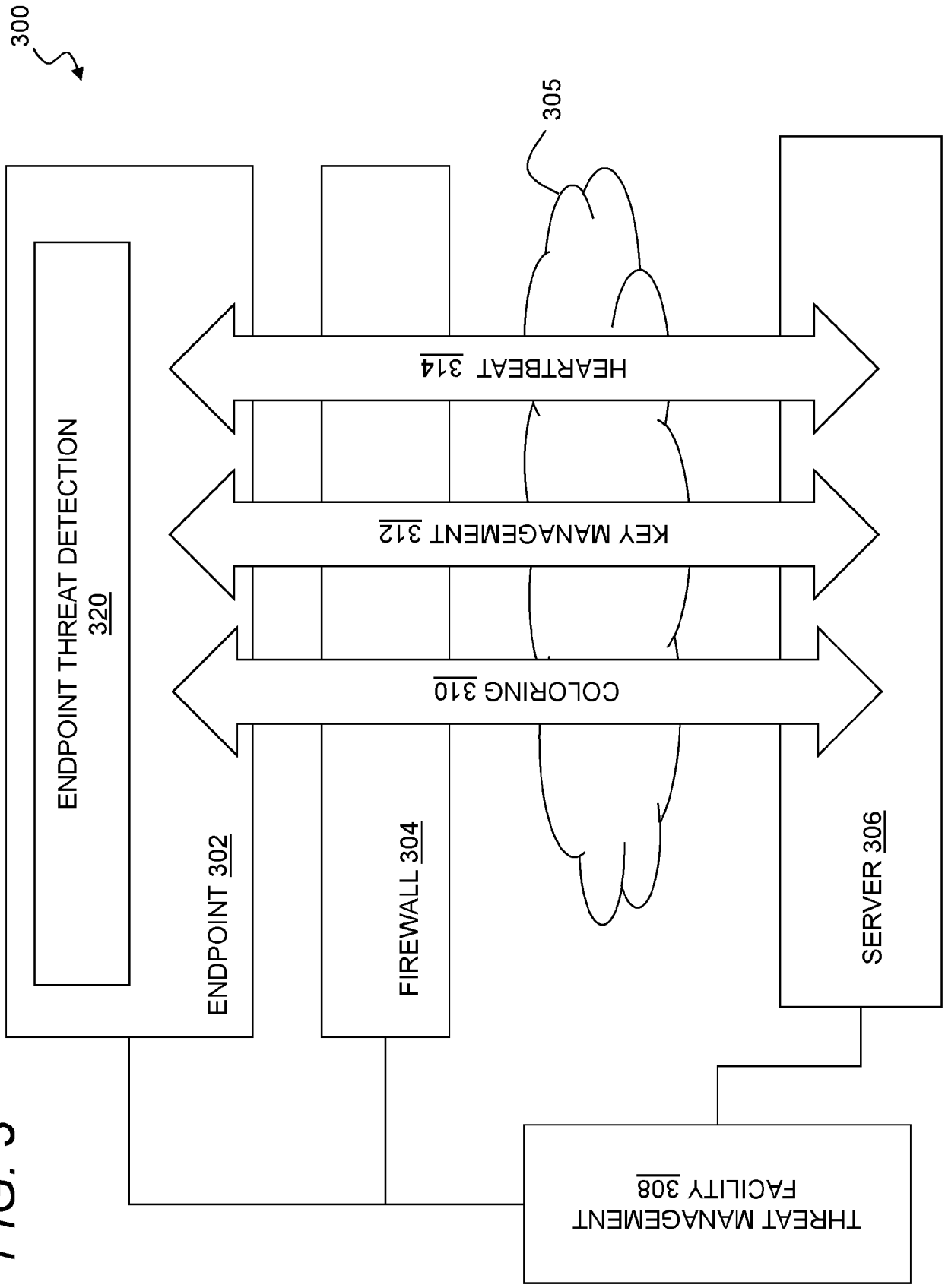

300
305
HEARTBEAT 314
KEY MANAGEMENT 312
COLORING 310
ENDPOINT THREAT DETECTION 320
ENDPOINT 302
FIREWALL 304
SERVER 306
THREAT MANAGEMENT FACILITY 308

FIG. 3

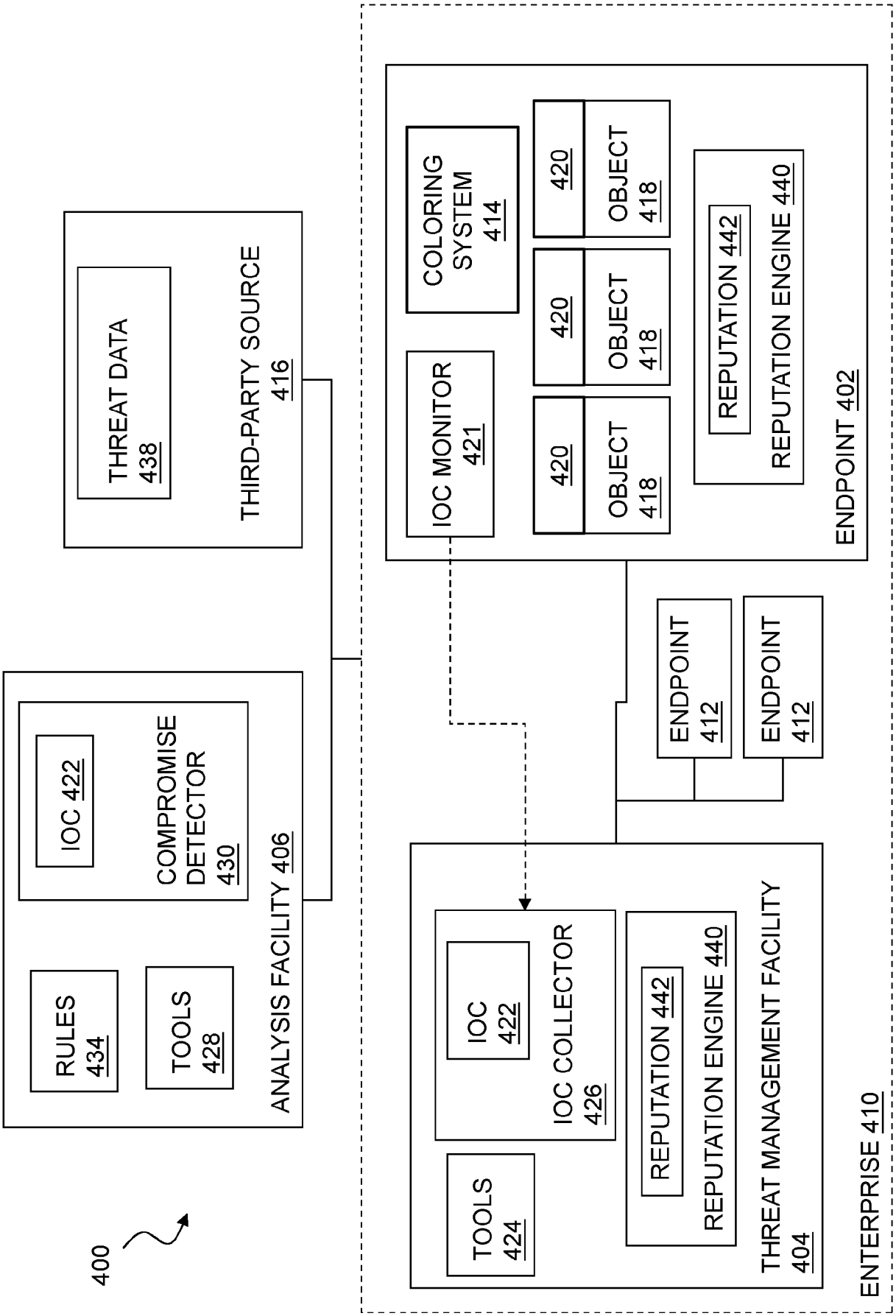

400
ENTERPRISE 410
THREAT MANAGEMENT FACILITY 404
TOOLS 424
IOC COLLECTOR 426
IOC 422
REPUTATION ENGINE 440
REPUTATION 442
ENDPOINT 412
ENDPOINT 412
ENDPOINT 402
REPUTATION ENGINE 440
REPUTATION 442
IOC MONITOR 421
COLORING SYSTEM 414
OBJECT 418
420
OBJECT 418
420
OBJECT 418
420
ANALYSIS FACILITY 406
RULES 434
TOOLS 428
COMPROMISE DETECTOR 430
IOC 422
THIRD-PARTY SOURCE 416
THREAT DATA 438

FIG. 4

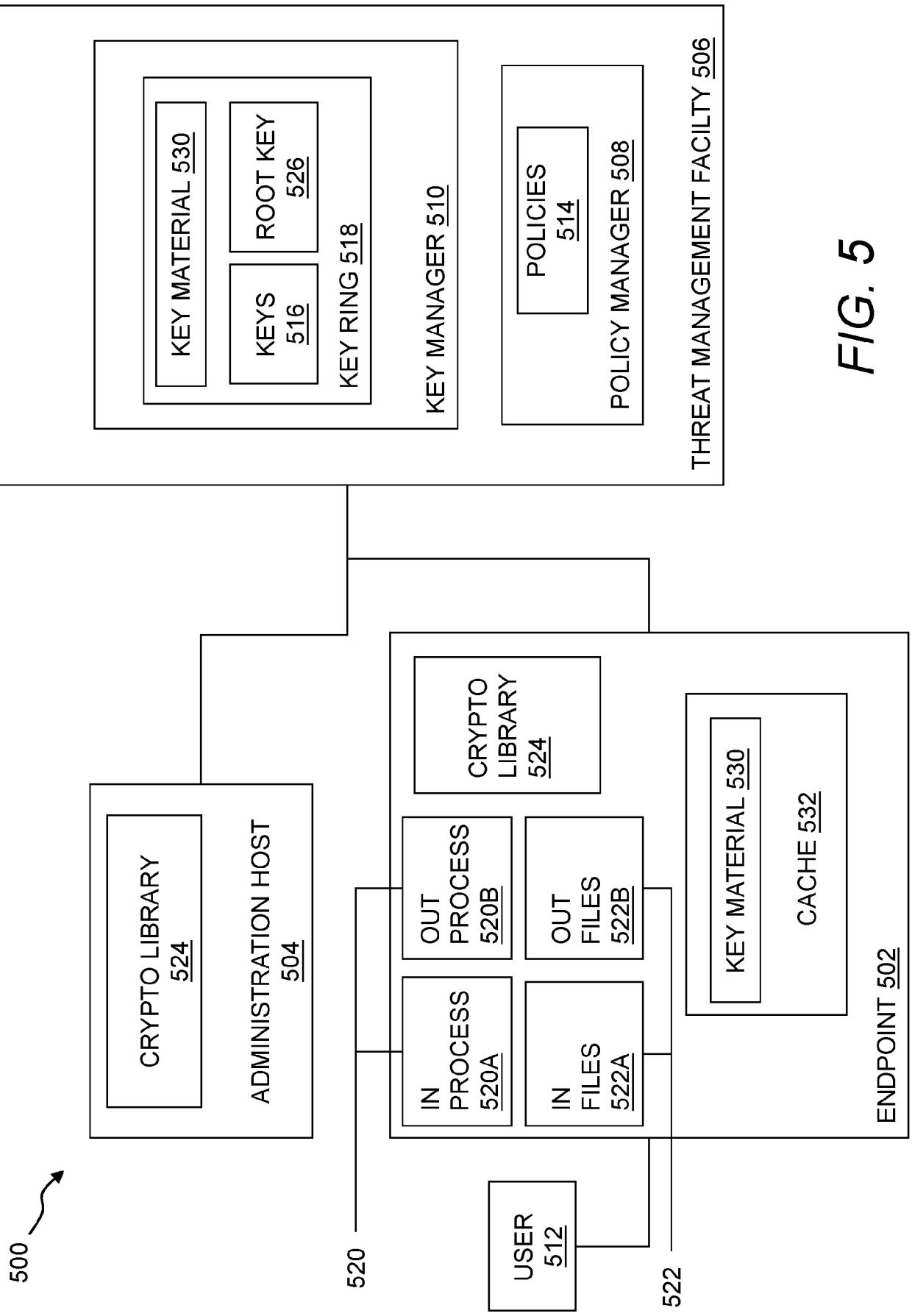

THREAT MANAGEMENT FACILTY 506
KEY MANAGER 510
KEY RING 518
KEY MATERIAL 530
KEYS 516
ROOT KEY 526
POLICY MANAGER 508
POLICIES 514
ADMINISTRATION HOST 504
CRYPTO LIBRARY 524
ENDPOINT 502
CRYPTO LIBRARY 524
OUT PROCESS 520B
OUT FILES 522B
IN PROCESS 520A
IN FILES 522A
KEY MATERIAL 530
CACHE 532
USER 512
500
520
522

FIG. 5

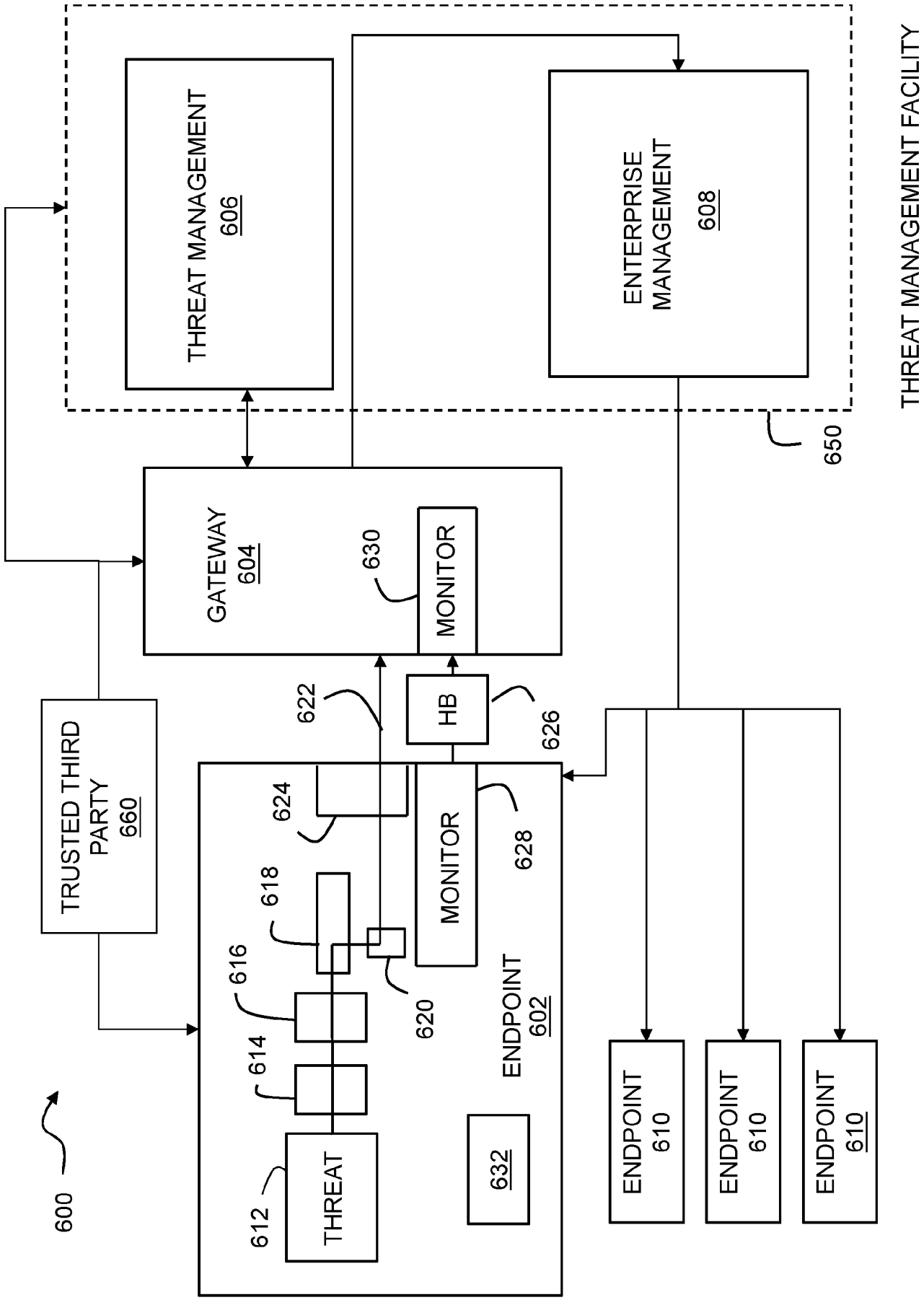

THREAT MANAGEMENT FACILITY
650
THREAT MANAGEMENT 606
ENTERPRISE MANAGEMENT 608
GATEWAY 604
630
MONITOR
622
HB
626
TRUSTED THIRD PARTY 660
624
MONITOR
628
618
616
620
614
ENDPOINT 602
632
612
THREAT
600
ENDPOINT 610
ENDPOINT 610
ENDPOINT 610

FIG. 6

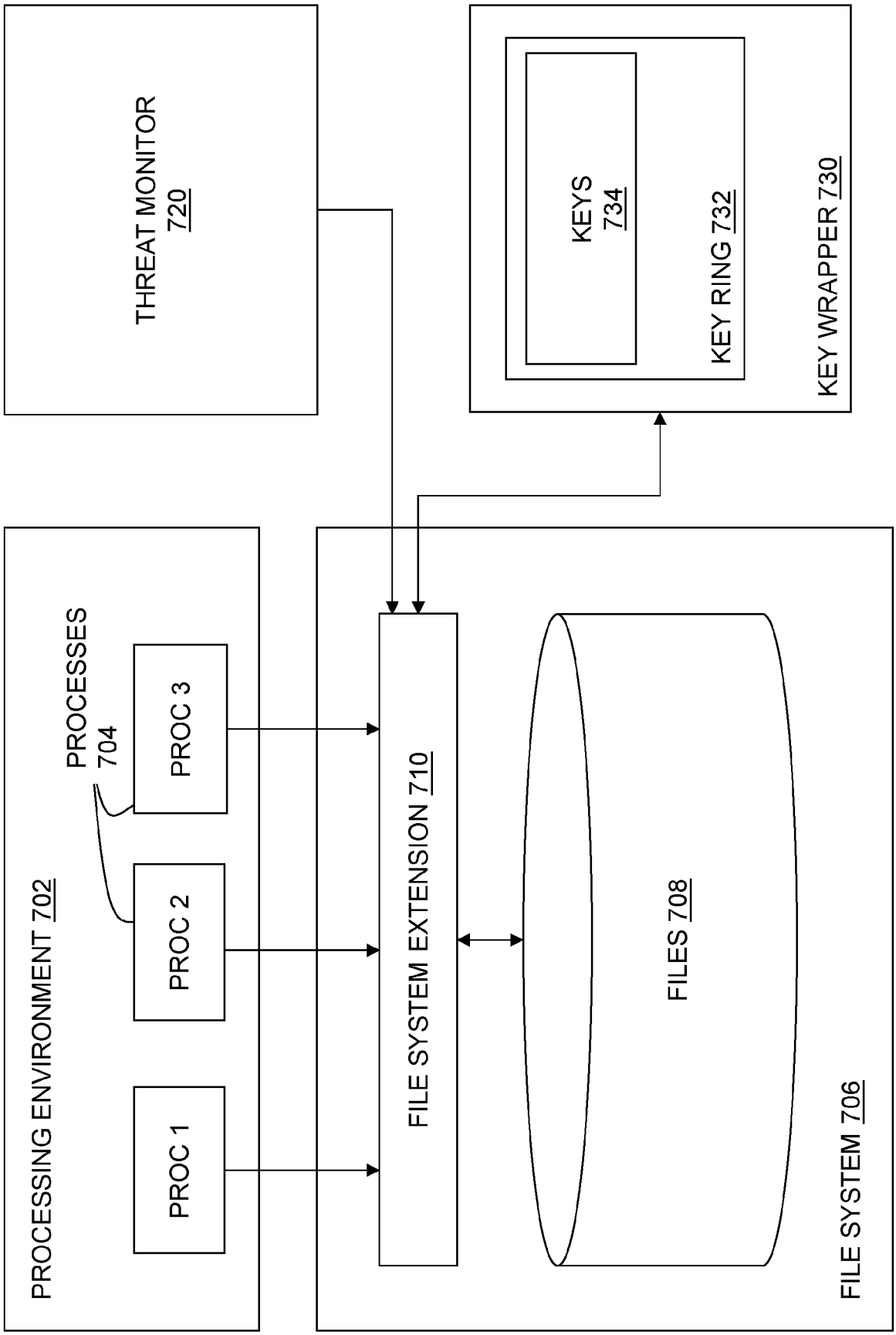

THREAT MONITOR
720
KEY WRAPPER 730
KEY RING 732
KEYS
734
PROCESSING ENVIRONMENT 702
PROCESSES
704
PROC 3
PROC 2
PROC 1
FILE SYSTEM EXTENSION 710
FILES 708
FILE SYSTEM 706

FIG. 7

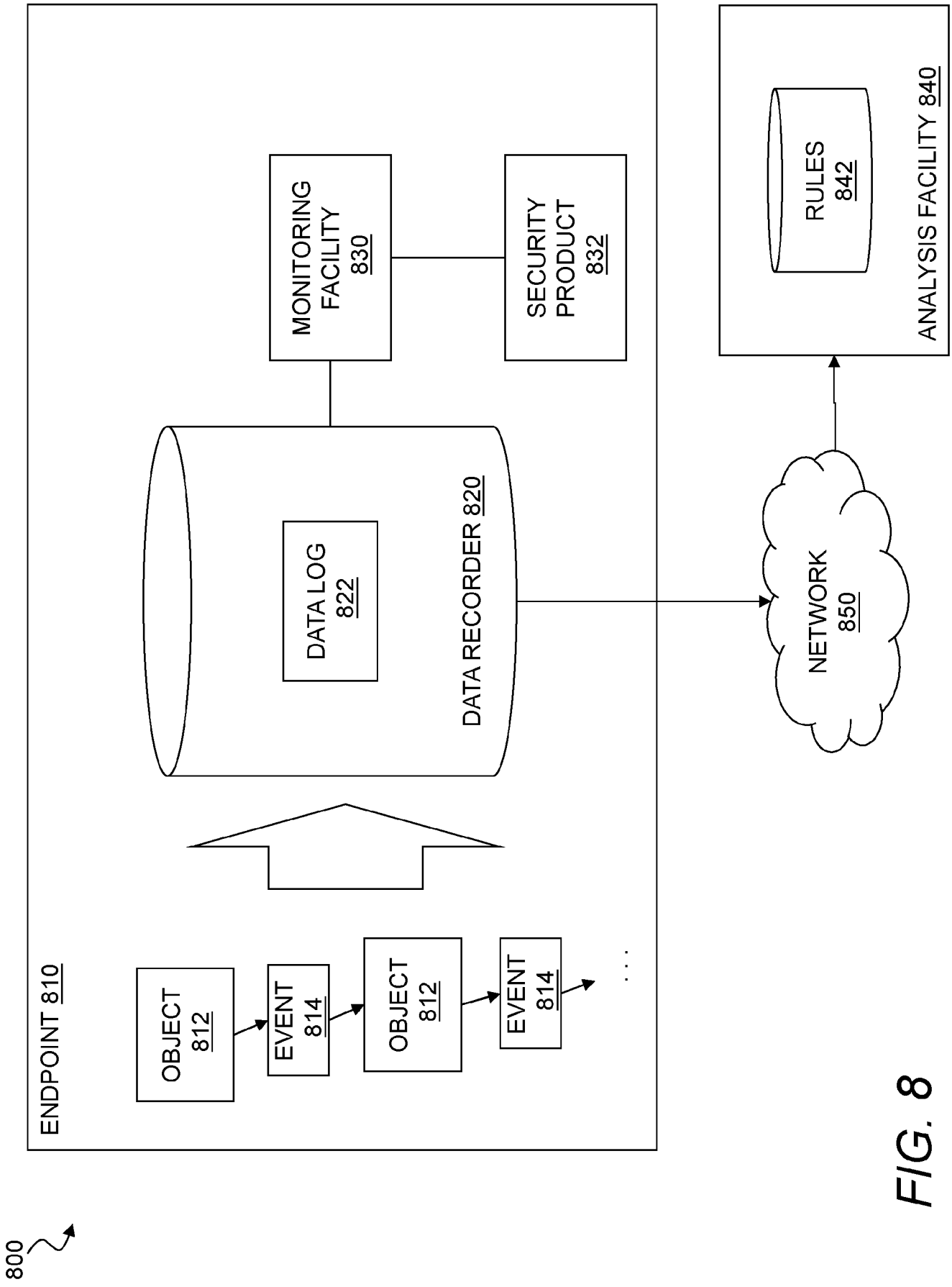

800
ENDPOINT 810
OBJECT 812
EVENT 814
OBJECT 812
EVENT 814
. . .
DATA RECORDER 820
DATA LOG 822
MONITORING FACILITY 830
SECURITY PRODUCT 832
NETWORK 850
ANALYSIS FACILITY 840
RULES 842

FIG. 8

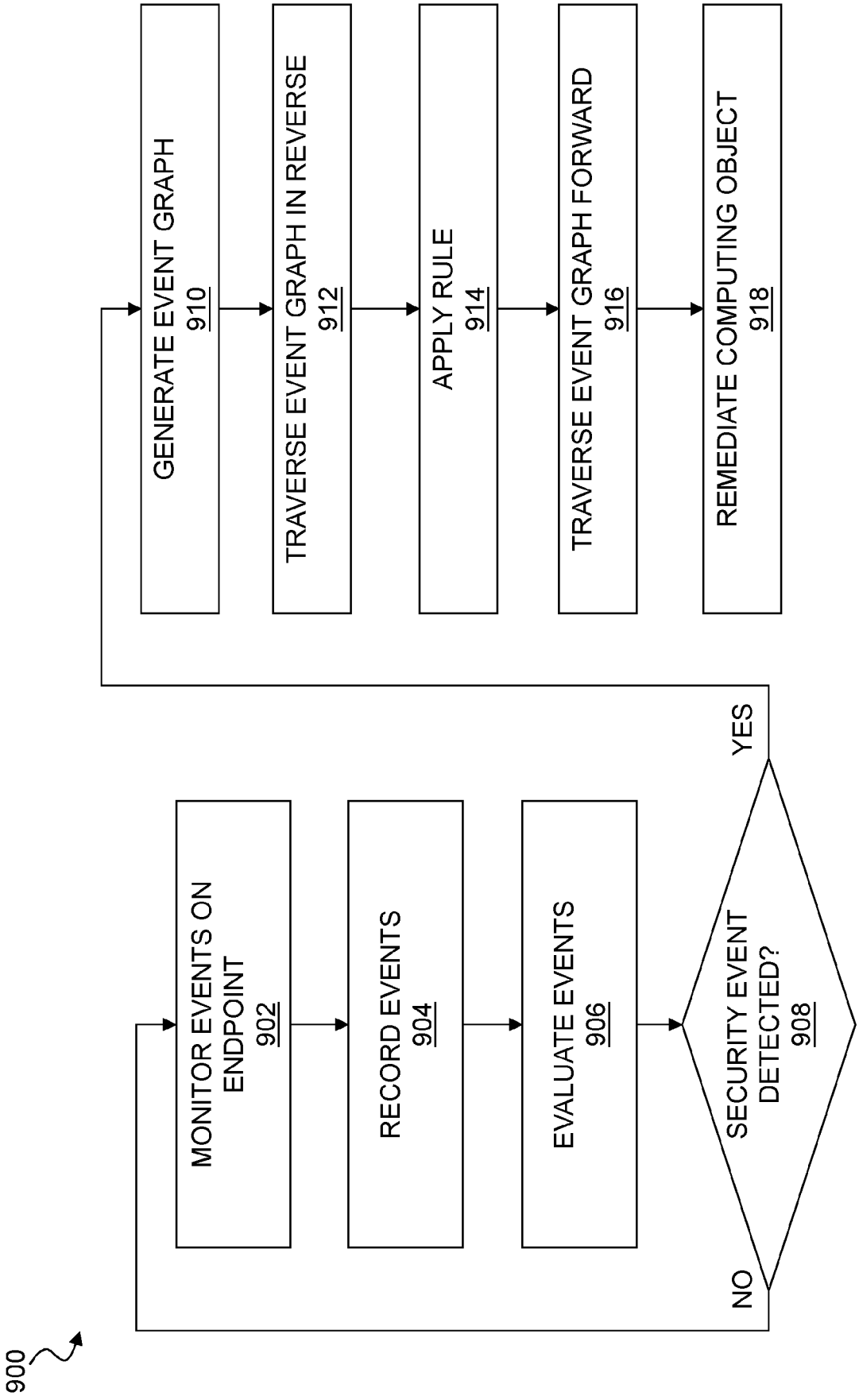

900
MONITOR EVENTS ON ENDPOINT 902
RECORD EVENTS 904
EVALUATE EVENTS 906
SECURITY EVENT DETECTED? 908
NO
YES
GENERATE EVENT GRAPH 910
TRAVERSE EVENT GRAPH IN REVERSE 912
APPLY RULE 914
TRAVERSE EVENT GRAPH FORWARD 916
REMEDIATE COMPUTING OBJECT 918

FIG. 9

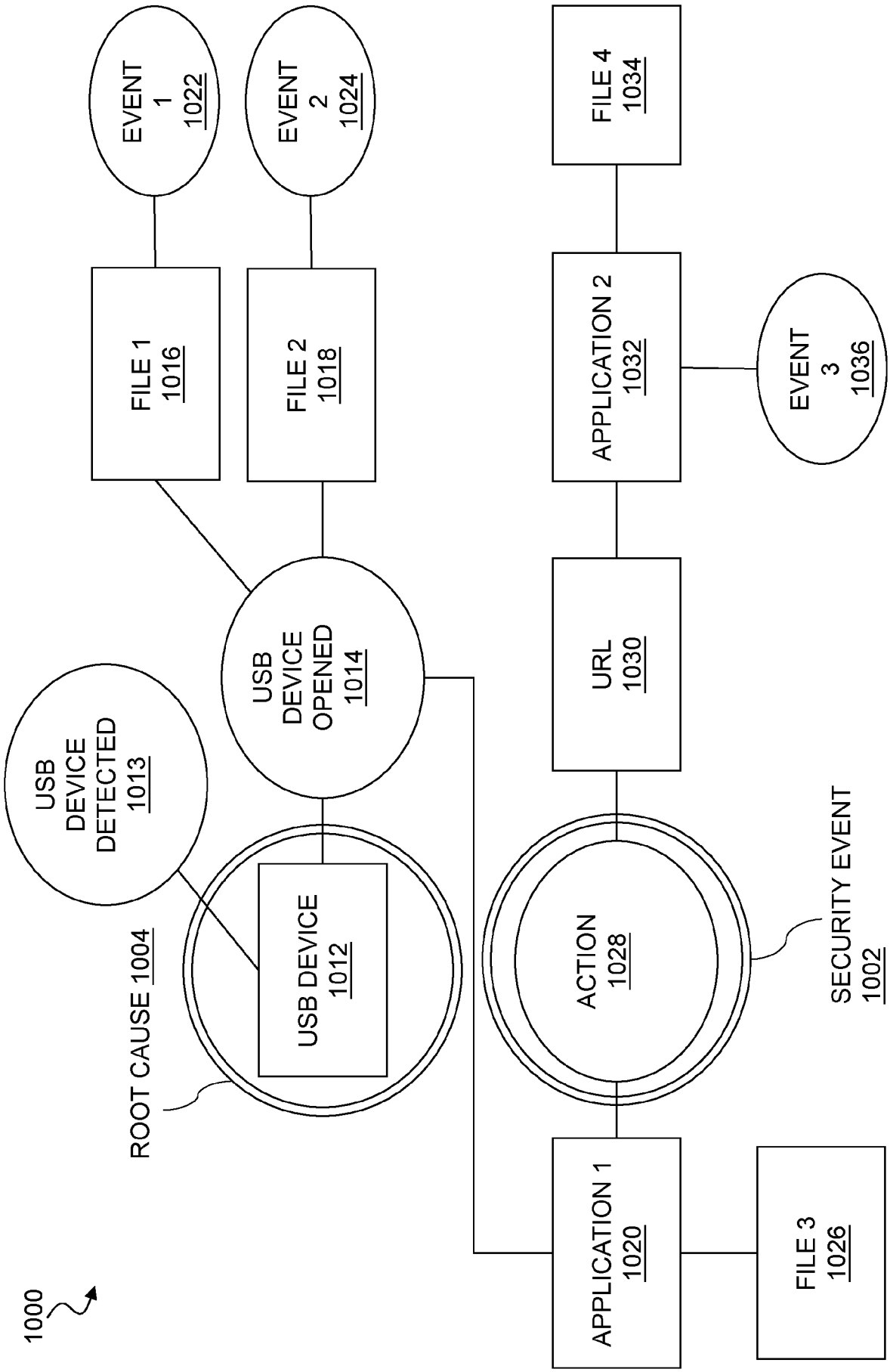

1000
ROOT CAUSE 1004
USB DEVICE DETECTED 1013
USB DEVICE 1012
USB DEVICE OPENED 1014
FILE 1 1016
FILE 2 1018
EVENT 1 1022
EVENT 2 1024
APPLICATION 1 1020
FILE 3 1026
ACTION 1028
URL 1030
APPLICATION 2 1032
FILE 4 1034
EVENT 3 1036
SECURITY EVENT 1002

FIG. 10

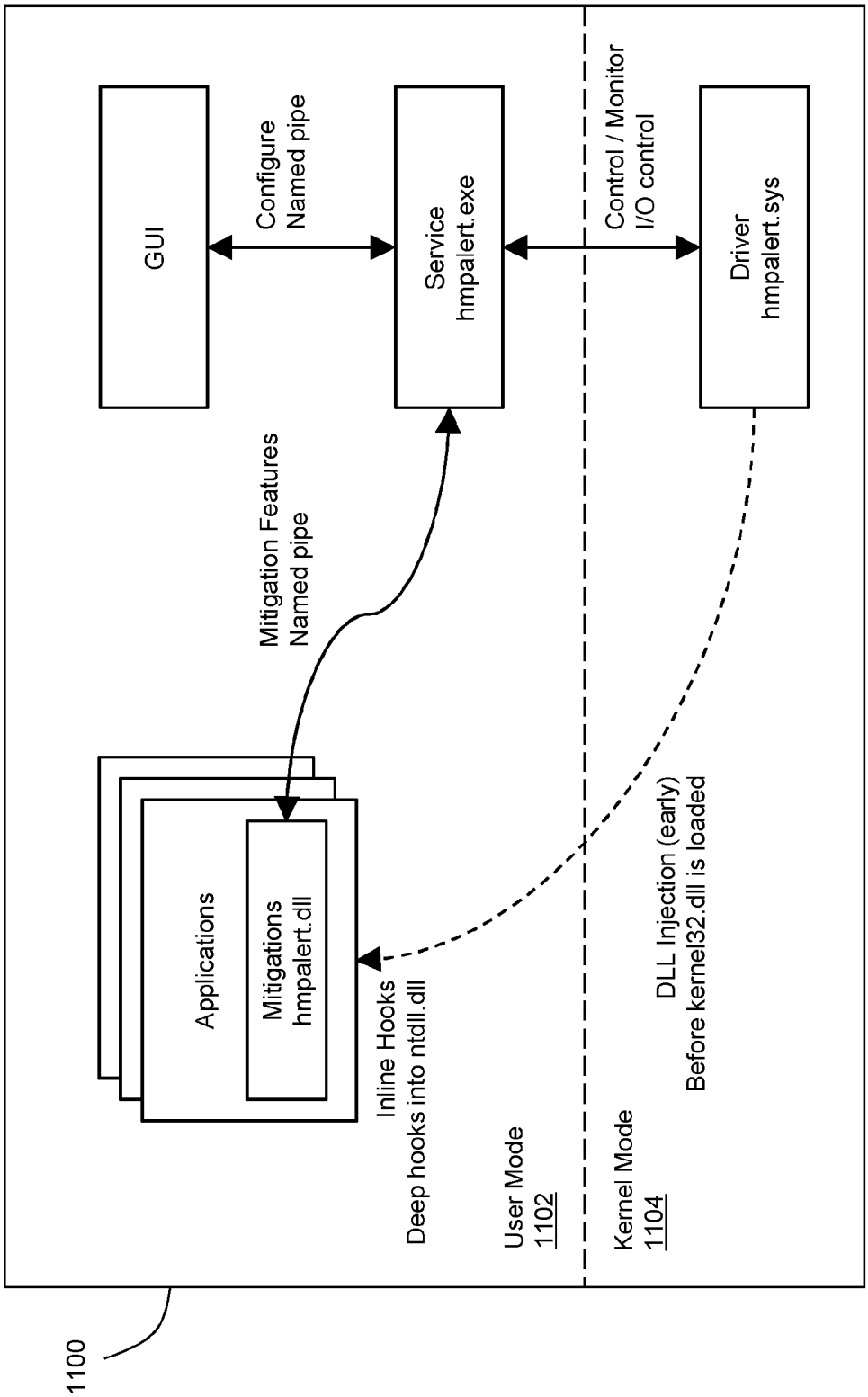

1100
User Mode
1102
Kernel Mode
1104
GUI
Configure
Named pipe
Service
hmpalert.exe
Control / Monitor
I/O control
Driver
hmpalert.sys
Mitigation Features
Named pipe
Applications
Mitigations
hmpalert.dll
Inline Hooks
Deep hooks into ntdll.dll
DLL Injection (early)
Before kernel32.dll is loaded

FIG. 11

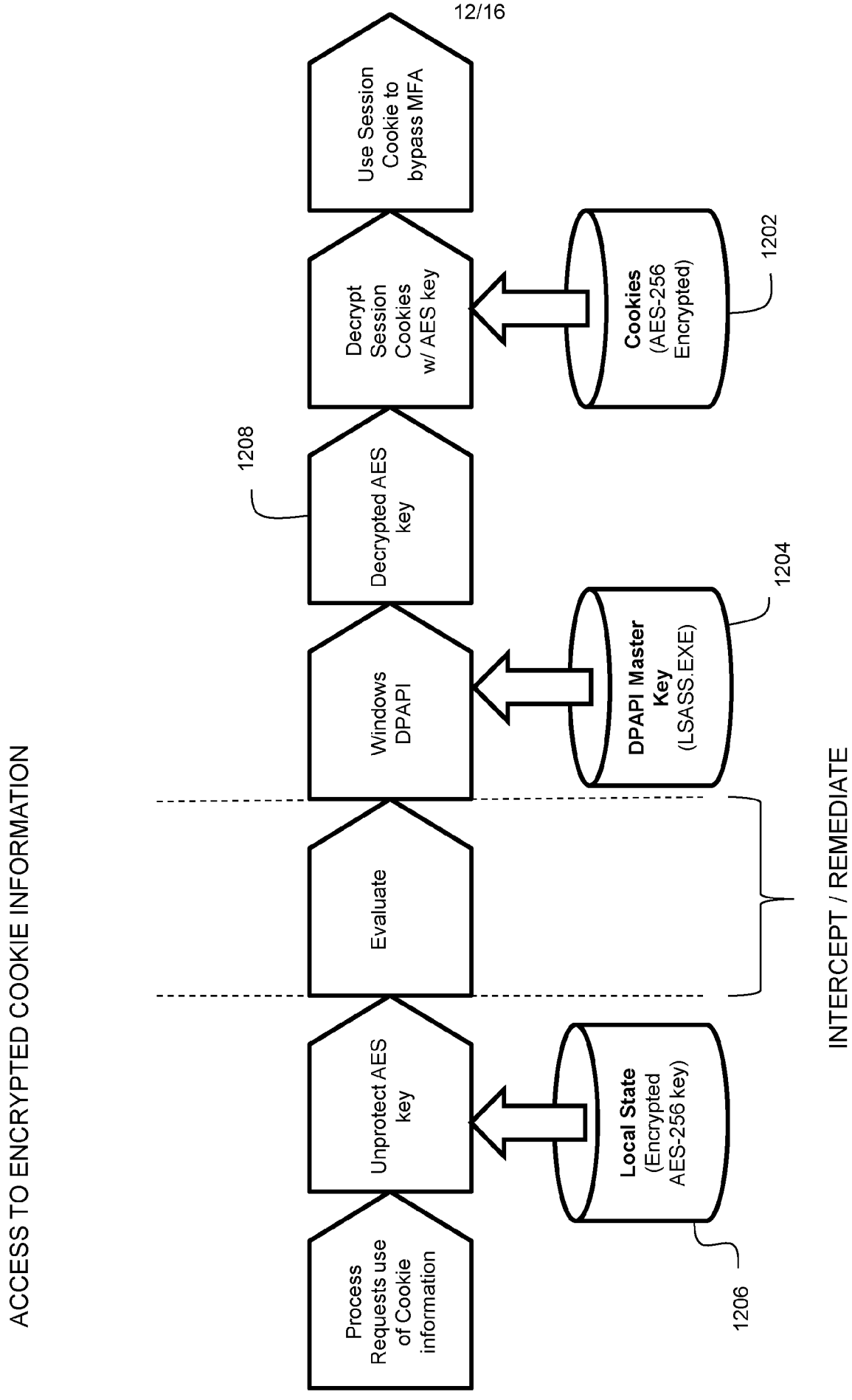

ACCESS TO ENCRYPTED COOKIE INFORMATION
Process Requests use of Cookie information
Unprotect AES key
Evaluate
Windows DPAPI
Decrypted AES key
Decrypt Session Cookies w/ AES key
Use Session Cookie to bypass MFA
Local State (Encrypted AES-256 key)
DPAPI Master Key (LSASS.EXE)
Cookies (AES-256 Encrypted)
1206
1208
1204
1202
INTERCEPT / REMEDIATE

FIG. 12

PREVENT MALWARE FROM DECRYPTING AES KEY

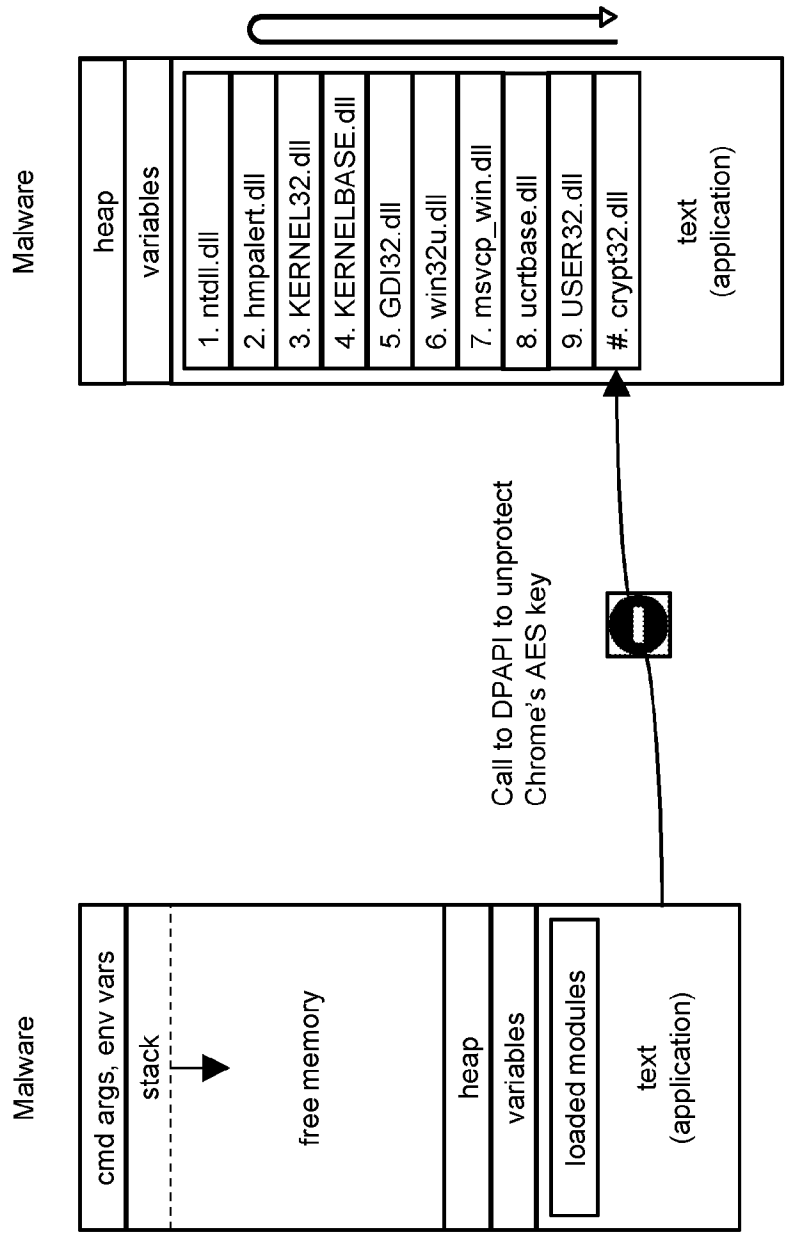


FIG. 13

PREVENT SHELLCODE FROM DECRYPTING AES KEY

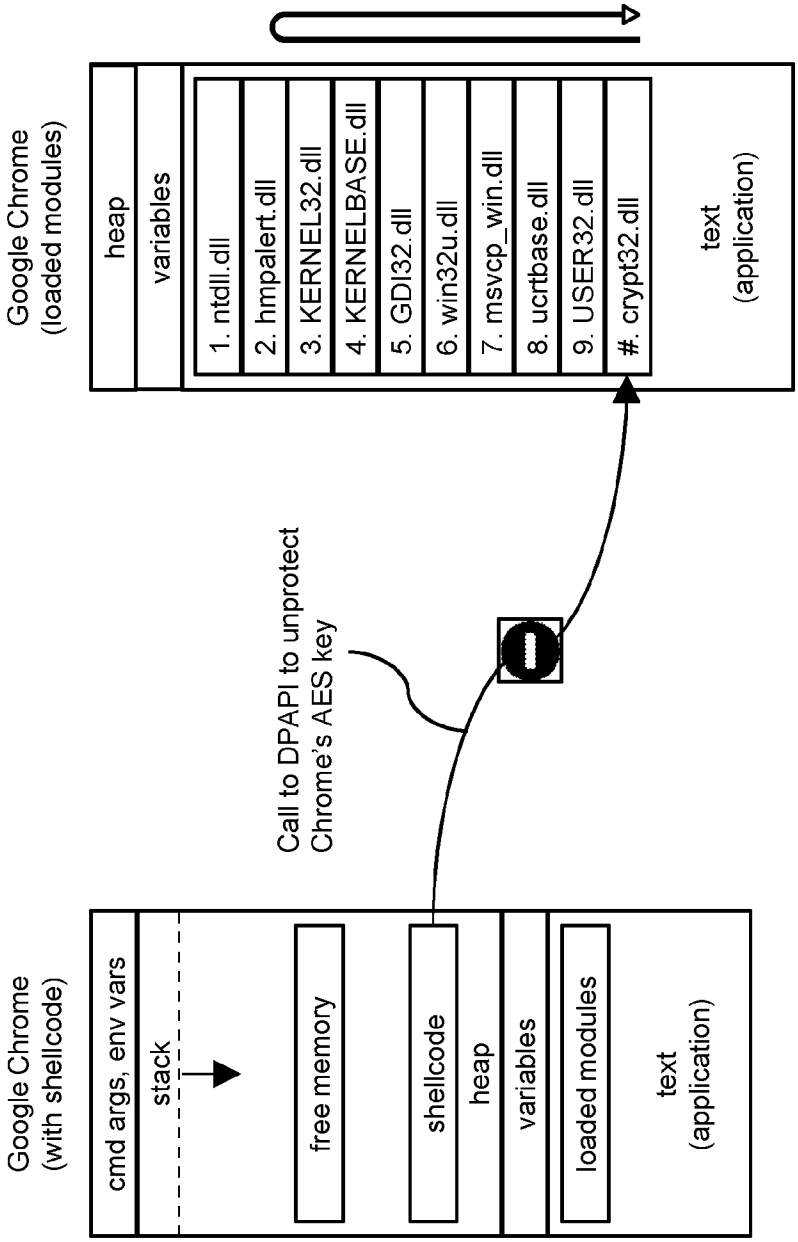


FIG. 14

PREVENT LOADED MODULES FROM DECRYPTING AES KEY

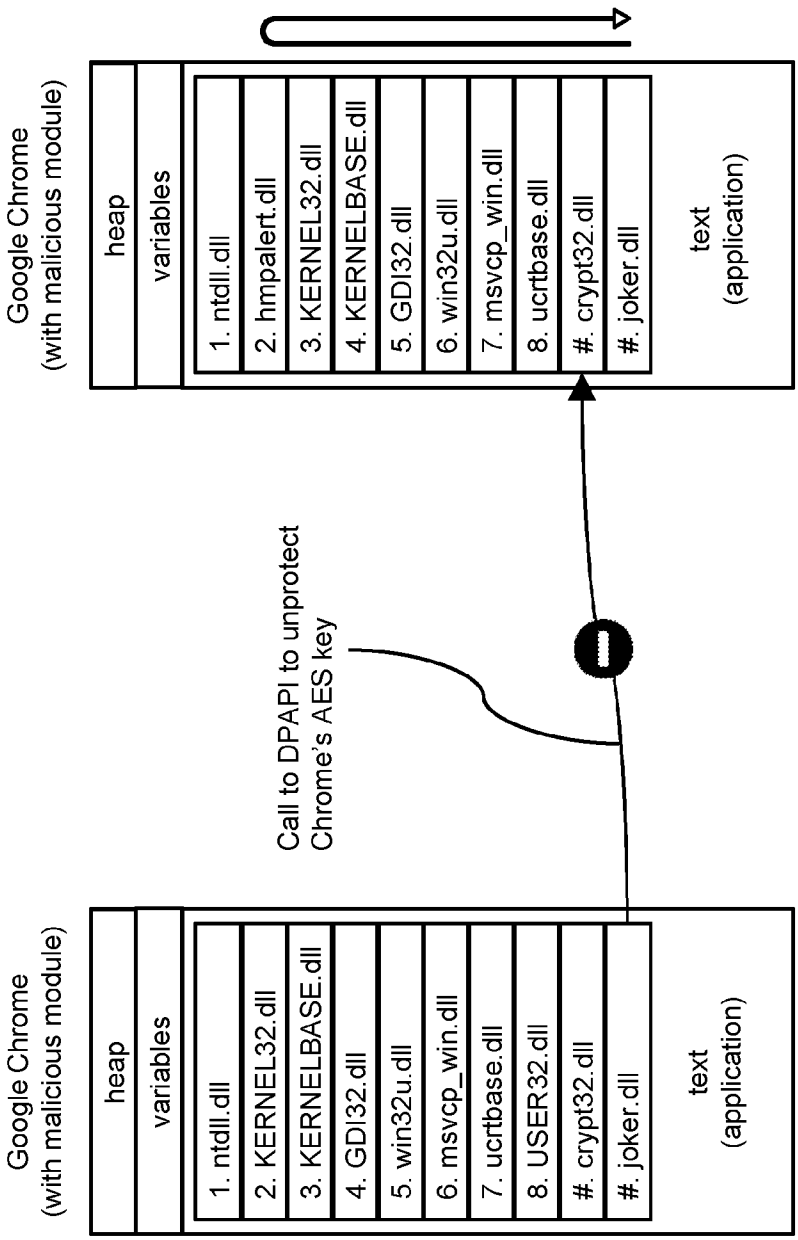


FIG. 15

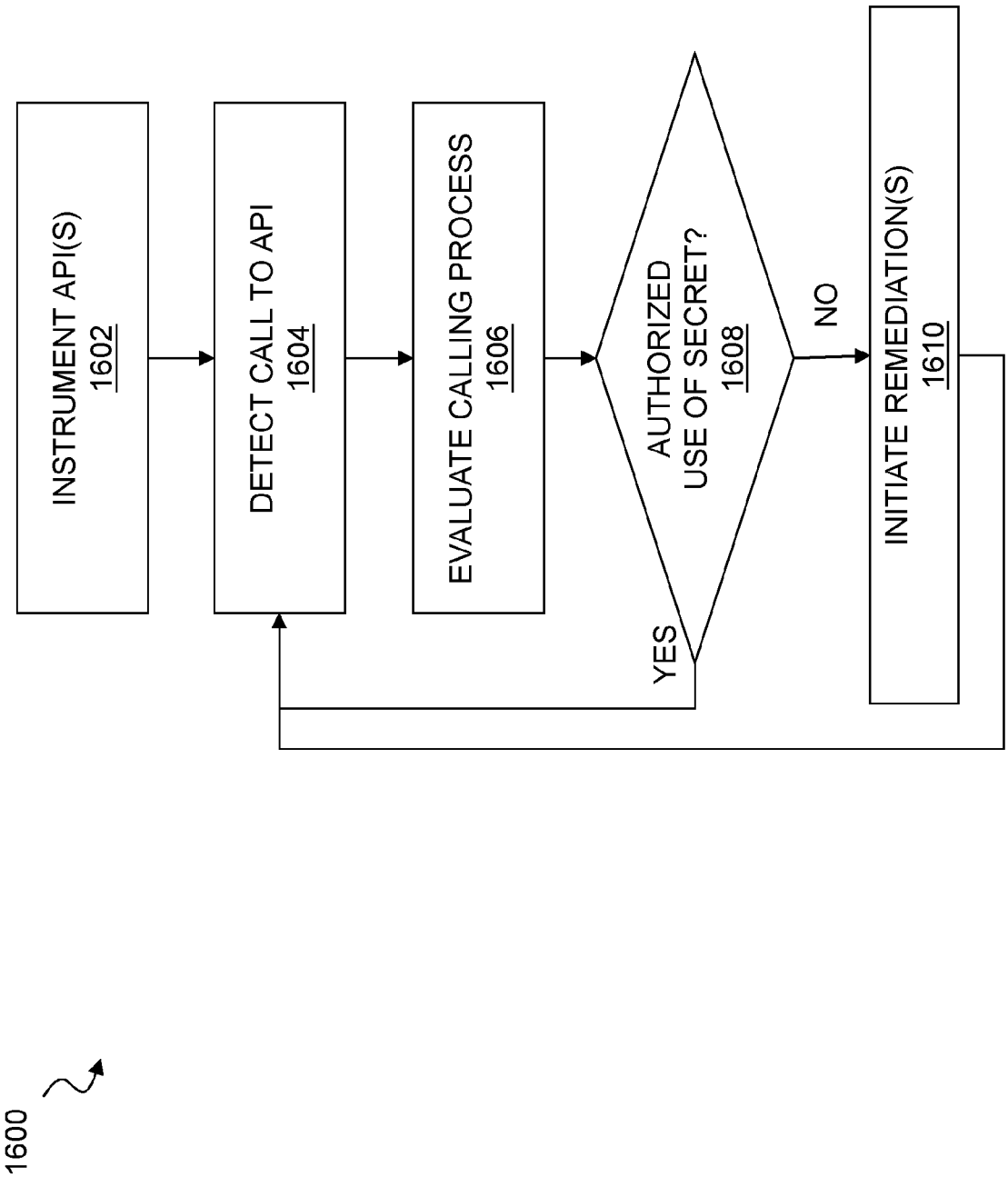

1600
INSTRUMENT API(S)
1602
DETECT CALL TO API
1604
EVALUATE CALLING PROCESS
1606
AUTHORIZED USE OF SECRET?
1608
YES
NO
INITIATE REMEDIATION(S)
1610

FIG. 16

INTERNATIONAL SEARCH REPORT

International application No
PCT/GB2022/050393

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L9/08 G06F21/60 G06F21/62
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 G06F

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2018/102902 A1 (YANG EN-HUI [CA] ET AL) 12 April 2018 (2018-04-12) paragraph [0002] paragraph [0007] paragraph [0009] paragraph [0011] paragraph [0013] paragraph [0016] paragraph [0023] paragraph [0025] paragraph [0057] - paragraph [0058] paragraph [0064] paragraph [0071] - paragraph [0072] paragraph [0076] paragraph [0078] paragraph [0085] - paragraph [0087] paragraph [0089] - paragraph [0090] paragraph [0092] - paragraph [0093] paragraph [0100] - paragraph [0101] -/--	1-20

[x] Further documents are listed in the continuation of Box C.

[x] 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 application or patent but published on or after the international filing date

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 April 2022	21/04/2022

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	Bharucha, Zubin

1

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	paragraph [0121] paragraph [0125] paragraph [0127] - paragraph [0128] paragraph [0135] figures 1-3 -----	
A	US 2018/046635 A1 (NICHOLS JACK ALLEN [US]) 15 February 2018 (2018-02-15) paragraph [0005] paragraph [0014] - paragraph [0016] paragraph [0026] - paragraph [0029] paragraph [0034] - paragraph [0036] paragraph [0040] - paragraph [0043] paragraph [0045] paragraph [0046] - paragraph [0047] paragraph [0066] - paragraph [0067] figures 3, 4, 7 -----	1-20
A	US 2007/124482 A1 (LEE SE H [KR] ET AL) 31 May 2007 (2007-05-31) paragraph [0011] paragraph [0038] - paragraph [0039] paragraph [0046] figures 5, 11, 13, 14, 16 -----	1-20

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/GB2022/050393

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2018102902 A1	12-04-2018	CN 109923548 A	21-06-2019
		US 2018102902 A1	12-04-2018
		WO 2018068133 A1	19-04-2018
US 2018046635 A1	15-02-2018	CN 109564566 A	02-04-2019
		EP 3497586 A1	19-06-2019
		US 2018046635 A1	15-02-2018
		WO 2018031351 A1	15-02-2018
US 2007124482 A1	31-05-2007	CN 1926801 A	07-03-2007
		JP 2007511831 A	10-05-2007
		KR 20050046481 A	18-05-2005
		US 2007124482 A1	31-05-2007
		WO 2005048526 A1	26-05-2005