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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 December 2016 (29.12.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/209449 A1

(51) International Patent Classification:
G06F 21/56 (2013.01) *G06F 21/51* (2013.01)
G06F 21/57 (2013.01) *G06F 21/62* (2013.01)

(21) International Application Number:
PCT/US2016/033691

(22) International Filing Date:
23 May 2016 (23.05.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/751,762 26 June 2015 (26.06.2015) US

(71) Applicant: **MCAFEE, INC.** [US/US]; 2821 Mission College Boulevard, Santa Clara, California 95054-1838 (US).

(72) Inventors: **PIKHUR, Volodymyr**; 17056 SW Pleasanton Lane, Beaverton, Oregon 97003 (US). **MATHUR, Rachit**; 532 NE Caden Avenue, Hillsboro, Oregon 97124 (US).

(74) Agent: **PEMBERTON, John D.**; Patent Capital Group, c/o CPA Global, 900 Second Avenue South, Suite 600, Minneapolis, Minnesota 55402 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, 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, 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))*

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

[Continued on next page]

(54) Title: PROFILING EVENT BASED EXPLOIT DETECTION

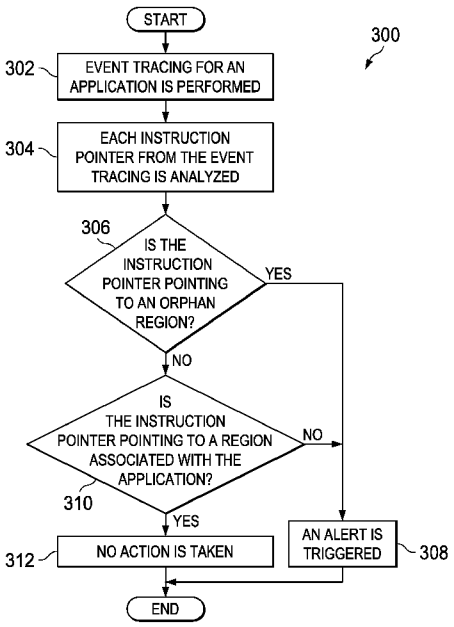


FIG. 3

(57) Abstract: Particular embodiments described herein provide for an electronic device that can be configured to execute an application in a system with an operating system, perform event tracing for the application, analyze each instruction pointer from the event tracing, and determine if an instruction pointer points to an orphan page of memory. The orphan page can be a region of code that is not associated with the application, a region of code that is unidentified, or unusual code that is not associated with the application. In addition, the event tracing can be an embedded application that is part of the operating system.

WO 2016/209449 A1

Published:

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

PROFILING EVENT BASED EXPLOIT DETECTION

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of and priority to U.S. Nonprovisional (Utility) Patent Application No. 14/751,762 filed 26 June 2015 entitled "PROFILING EVENT BASED EXPLOIT DETECTION", which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] This disclosure relates in general to the field of information security, and more particularly, to profiling event based exploit detection.

BACKGROUND

[0003] The field of network security has become increasingly important in today's society. The Internet has enabled interconnection of different computer networks all over the world. In particular, the Internet provides a medium for exchanging data between different users connected to different computer networks via various types of client devices. While the use of the Internet has transformed business and personal communications, it has also been used as a vehicle for malicious operators to gain unauthorized access to computers and computer networks and for intentional or inadvertent disclosure of sensitive information.

[0004] Malicious software ("malware") that infects a host computer may be able to perform any number of malicious actions, such as stealing sensitive information from a business or individual associated with the host computer, propagating to other host computers, and/or assisting with distributed denial of service attacks, sending out spam or malicious emails from the host computer, etc. Hence, significant administrative challenges remain for protecting computers and computer networks from malicious and inadvertent exploitation by malicious software.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] To provide a more complete understanding of the present disclosure and features and advantages thereof, reference is made to the following description, taken in

conjunction with the accompanying figures, wherein like reference numerals represent like parts, in which:

[0006] FIGURE 1 is a simplified block diagram of a communication system for profiling event based exploit detection in a network environment in accordance with an embodiment of the present disclosure;

[0007] FIGURE 2 is a simplified block diagram of a portion of a communication system for profiling event based exploit detection in a network environment in accordance with an embodiment of the present disclosure;

[0008] FIGURE 3 is a simplified flowchart illustrating potential operations that may be associated with the communication system in accordance with an embodiment;

[0009] FIGURE 4 is a simplified flowchart illustrating potential operations that may be associated with the communication system in accordance with an embodiment;

[0010] FIGURE 5 is a block diagram illustrating an example computing system that is arranged in a point-to-point configuration in accordance with an embodiment;

[0011] FIGURE 6 is a simplified block diagram associated with an example ARM ecosystem system on chip (SOC) of the present disclosure; and

[0012] FIGURE 7 is a block diagram illustrating an example processor core in accordance with an embodiment.

[0013] The FIGURES of the drawings are not necessarily drawn to scale, as their dimensions can be varied considerably without departing from the scope of the present disclosure.

DETAILED DESCRIPTION OF EXAMPLE EMBODIMENTS

EXAMPLE EMBODIMENTS

[0014] FIGURE 1 is a simplified block diagram of a communication system 100 for profiling event based exploit detection in a network environment in accordance with an embodiment of the present disclosure. Communication system 100 can include an electronic device 102, cloud services, 104, and a server 106. Electronic device 102 can include a processor 110, an operating system (OS) 112, one or more applications 114a and 114b, a security module 116, and memory 118a. OS 112 can include an event tracing module 120.

Event tracing module 120 can include an instruction pointer queue 122. Security module 116 can include instruction pointer analysis module 124 and an application module table 126. Memory 118a can include one or more application regions 128a and 128b, an orphan application region 130, whitelist 132, and blacklist 134. Cloud services 104 can include memory 118b and a network security module 138a. Memory 118b can include whitelist 132 and blacklist 134. Network security module 138a can include application module table 126 and network instruction pointer analysis module 140. Server 106 can include memory 118c and a network security module 138b. Memory 118c can include whitelist 132 and blacklist 134. Network security module 138b can include application module table 126 and network instruction pointer analysis module 140. Electronic device 102, cloud services 104, and server 106 can be in communication using network 108. Malware 136 may attempt to introduce itself to electronic device 102 using network 108 or through a direct connection (e.g., through a Universal Serial Bus (USB) type connection).

[0015] In example embodiments, communication system 100 can be configured to execute an application (e.g., application 114a) in a system with an OS (e.g., OS 112), perform event tracing for the application using event tracing module 120, analyze each instruction pointer from the event tracing using instruction pointer analysis module 124, and determine if an instruction pointer points to an orphan page of memory (e.g., orphan application region 130). The orphan page can be a region of code that is not associated with the application, a region of code that is unidentified, or unusual code that is not associated with the application. In addition, the event tracing can be an embedded application or feature that is part of the OS. Application module table 126 can include a list of modules or memory regions associated with a process. For example, application 114a may be associated with application region 128a and application 114b may be associated with application region 128b.

[0016] Elements of FIGURE 1 may be coupled to one another through one or more interfaces employing any suitable connections (wired or wireless), which provide viable pathways for network (e.g., network 108) communications. Additionally, any one or more of these elements of FIGURE 1 may be combined or removed from the architecture based on particular configuration needs. Communication system 100 may include a configuration capable of transmission control protocol/Internet protocol (TCP/IP) communications for the transmission or reception of packets in a network. Communication system 100 may also

operate in conjunction with a user datagram protocol/IP (UDP/IP) or any other suitable protocol where appropriate and based on particular needs.

[0017] For purposes of illustrating certain example techniques of communication system 100, it is important to understand the communications that may be traversing the network environment. The following foundational information may be viewed as a basis from which the present disclosure may be properly explained.

[0018] Increased access to the Internet has had the unintended effect of increasing the reach of malware. The term malware as used herein includes any type of software program, application, module, code, etc. designed to infiltrate, modify, change, corrupt, or damage a computer system without the owner's informed consent, regardless of the motivation for the software program, and regardless of the results caused by the software program on the owner' s devices, systems, networks, or data.

[0019] Some malware can take advantage of exploits in document viewers (e.g., Microsoft® Word®, Adobe® viewers, browsers, etc.). For example, a business may get a Word® or PDF® formatted document which may actually be a targeted zero-day attack. When an electronic device in the business opens or reads the document, the malware can compromise the electronic device. In another example, an electronic device may be used to visit a website and malware may unknowingly get installed on the electronic device using exploits from the website.

[0020] Current solutions to the problem are often invasive and have performance or compatibility concerns. For example, sandboxing or emulation based solutions to detect malicious applications try to run third-party software under controlled environments. This can have serious compatibility issues with the electronic device, substantially degrade performance of the electronic device by consuming a relatively large amount of resources, and can create performance bottlenecks making the solutions infeasible to deploy on electronic devices. Other solutions can have performance and compatibility concerns or are based on specific knowledge of an already known vulnerability. In addition, various detection programs may be used to attempt to detect the presence of malware. In some instances, the detection programs rely on detecting a signature in a software program being examined to determine if the program is or contains malware. In some instances, the detection program uses a tracing method to determine whether a software program is malware. However,

malware authors frequently change or alter parts of the malware programs in order to avoid detection by tracing methods. What is needed a malware detection system and method that does not require the target program or suspected malware to run under a controlled environment or use OS control flow interception. It would be beneficial if the system and method could use well supported OS features (e.g., Microsoft Windows® features) to generically identify arbitrary code execution in processes such as document viewers, browsers, etc.

[0021] A communication system for profiling event based exploit detection, as outlined in FIGURE 1, can resolve these issues (and others). In communication system 100 of FIGURE 1, the system may be configured to use event tracing features of an OS to identify the memory regions that are being executed under a given target process. Based on identified memory regions, communication system 100 can be configured to determine if an exploit has taken place in the target application by checking if the target application is executing code from regions that are unidentified or if the code executing inside pages is unusual. In a specific illustrative example, event tracing for Windows (ETW) is a Microsoft Windows® feature where the ETW provides NT Kernel related events including ETW profiling events. The ETW profiling events typically include instruction-pointers (IP) for any given process. This information can be used to identify the memory regions that are being executed under a given target process (e.g., application module table 126 can be used to identify memory regions associated with a target process) and if the target application is executing code from regions that are unidentified or if the code executing inside pages is unusual.

[0022] By using the event tracing features of the OS, the analysis of a target application can be completely transparent to the targeted application and compatibility with the electronic device is not a concern. The profiling event based exploit detection can operate asynchronously and does not block or halt the target application so no perceived performance overhead is created. Also, because the OS provides all the necessary capabilities for the event tracing, driver installation, DLL components, or plugins for targeted applications, is not required and the process does not modify, inject or hook into OS processes.

[0023] In a specific example, a SampledProfile event provided by Microsoft Windows® includes information about Instruction Pointers (IP) in a given thread. This is information that Microsoft® has enabled from within the Windows® OS to aid in troubleshooting and

debugging applications. The information is available to be consumed by any application (even in user-mode) using the ETW framework which is highly efficient. These events are delivered asynchronously so the target application continues execution without being blocked and the process is transparent so that the target application is not even aware of the process and there is little chance of any compatibility issues.

[0024] The IP address information from the event trace can be used to determine if arbitrary code execution has taken place in an application. For example, after receiving an event trace, the event trace can be first associated with a process and the system can determine if the process is a process that is being monitored (e.g., for attacks or if the process is associated with malware). If the process is a process that is being monitored, the IP address reported in the event can be checked against loaded modules or memory regions in application module table 126 associated with the process to determine if any execution occurred outside of the loaded modules or memory regions associated with the process. If an execution did occur outside of the loaded modules or memory regions associated with the process, then the execution is deemed to have occurred in an orphan region (e.g., orphan application region 130).

[0025] Some applications create an orphan page or region by allocating memory on heap and execute code from the heap for virtual-machine or sandbox based executions (e.g., Adobe® Reader 11). To detect this, a generic shell code detection check can be added for the intentionally created orphan page or region. Another approach to detect intentionally created orphan pages or regions is to whitelist the sandbox related code to identify the benign orphan pages and then flag any unexpected code execution within the intentionally created orphan pages.

[0026] Turning to the infrastructure of FIGURE 1, communication system 100 in accordance with an example embodiment is shown. Generally, communication system 100 can be implemented in any type or topology of networks. Network 108 represents a series of points or nodes of interconnected communication paths for receiving and transmitting packets of information that propagate through communication system 100. Network 108 offers a communicative interface between nodes, and may be configured as any local area network (LAN), virtual local area network (VLAN), wide area network (WAN), wireless local area network (WLAN), metropolitan area network (MAN), Intranet, Extranet, virtual private

network (VPN), and any other appropriate architecture or system that facilitates communications in a network environment, or any suitable combination thereof, including wired and/or wireless communication.

[0027] In communication system 100, network traffic, which is inclusive of packets, frames, signals, data, etc., can be sent and received according to any suitable communication messaging protocols. Suitable communication messaging protocols can include a multi-layered scheme such as Open Systems Interconnection (OSI) model, or any derivations or variants thereof (e.g., Transmission Control Protocol/Internet Protocol (TCP/IP), user datagram protocol/IP (UDP/IP)). Additionally, radio signal communications over a cellular network may also be provided in communication system 100. Suitable interfaces and infrastructure may be provided to enable communication with the cellular network.

[0028] The term "packet" as used herein, refers to a unit of data that can be routed between a source node and a destination node on a packet switched network. A packet includes a source network address and a destination network address. These network addresses can be Internet Protocol (IP) addresses in a TCP/IP messaging protocol. The term "data" as used herein, refers to any type of binary, numeric, voice, video, textual, or script data, or any type of source or object code, or any other suitable information in any appropriate format that may be communicated from one point to another in electronic devices and/or networks. Additionally, messages, requests, responses, and queries are forms of network traffic, and therefore, may comprise packets, frames, signals, data, etc.

[0029] In an example implementation, electronic device 102, cloud services 104, and server 106, and are network elements, which are meant to encompass network appliances, servers, routers, switches, gateways, bridges, load balancers, processors, modules, or any other suitable device, component, element, or object operable to exchange information in a network environment. Network elements may include any suitable hardware, software, components, modules, or objects that facilitate the operations thereof, as well as suitable interfaces for receiving, transmitting, and/or otherwise communicating data or information in a network environment. This may be inclusive of appropriate algorithms and communication protocols that allow for the effective exchange of data or information.

[0030] In regards to the internal structure associated with communication system 100, each of electronic device 102, cloud services 104, and server 106 can include memory

elements for storing information to be used in the operations outlined herein. Each of electronic device 102, cloud services 104, and server 106 may keep information in any suitable memory element (e.g., random access memory (RAM), read-only memory (ROM), erasable programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM), application specific integrated circuit (ASIC), etc.), software, hardware, firmware, or in any other suitable component, device, element, or object where appropriate and based on particular needs. Any of the memory items discussed herein should be construed as being encompassed within the broad term 'memory element.' Moreover, the information being used, tracked, sent, or received in communication system 100 could be provided in any database, register, queue, table, cache, control list, or other storage structure, all of which can be referenced at any suitable timeframe. Any such storage options may also be included within the broad term 'memory element' as used herein.

[0031] In certain example implementations, the functions outlined herein may be implemented by logic encoded in one or more tangible media (e.g., embedded logic provided in an ASIC, digital signal processor (DSP) instructions, software (potentially inclusive of object code and source code) to be executed by a processor, or other similar machine, etc.), which may be inclusive of non-transitory computer-readable media. In some of these instances, memory elements can store data used for the operations described herein. This includes the memory elements being able to store software, logic, code, or processor instructions that are executed to carry out the activities described herein.

[0032] In an example implementation, network elements of communication system 100, such as electronic device 102, cloud services 104, and server 106 may include software modules (e.g., security module 116, event tracing module 120, instruction pointer analysis module 124, network security modules 138a and 138b, and network instruction pointer module 140) to achieve, or to foster, operations as outlined herein. These modules may be suitably combined in any appropriate manner, which may be based on particular configuration and/or provisioning needs. In example embodiments, such operations may be carried out by hardware, implemented externally to these elements, or included in some other network device to achieve the intended functionality. Furthermore, the modules can be implemented as software, hardware, firmware, or any suitable combination thereof.

These elements may also include software (or reciprocating software) that can coordinate with other network elements in order to achieve the operations, as outlined herein.

[0033] Additionally, each of electronic device 102, cloud services 104, and server 106 may include a processor that can execute software or an algorithm to perform activities as discussed herein. A processor can execute any type of instructions associated with the data to achieve the operations detailed herein. In one example, the processors could transform an element or an article (e.g., data) from one state or thing to another state or thing. In another example, the activities outlined herein may be implemented with fixed logic or programmable logic (e.g., software/computer instructions executed by a processor) and the elements identified herein could be some type of a programmable processor, programmable digital logic (e.g., a field programmable gate array (FPGA), an EPROM, an EEPROM) or an ASIC that includes digital logic, software, code, electronic instructions, or any suitable combination thereof. Any of the potential processing elements, modules, and machines described herein should be construed as being encompassed within the broad term 'processor.'

[0034] Electronic device 102 can be a network element and includes, for example, desktop computers, laptop computers, mobile devices, personal digital assistants, smartphones, tablets, or other similar devices. Cloud services 104 is configured to provide cloud services to electronic device 102. Cloud services may generally be defined as the use of computing resources that are delivered as a service over a network, such as the Internet. Typically, compute, storage, and network resources are offered in a cloud infrastructure, effectively shifting the workload from a local network to the cloud network. Server 106 can be a network element such as a server or virtual server and can be associated with clients, customers, endpoints, or end users wishing to initiate a communication in communication system 100 via some network (e.g., network 108). The term 'server' is inclusive of devices used to serve the requests of clients and/or perform some computational task on behalf of clients within communication system 100. Although security module 116 is represented in FIGURE 1 as being located in electronic device 102, this is for illustrative purposes only. Security module 116 could be combined or separated in any suitable configuration. Furthermore, security module 116 could be integrated with or distributed in another network accessible by electronic device 102 such as cloud services 104 or server 106.

[0035] Turning to FIGURE 2, FIGURE 2 is a simplified block diagram of a portion of communication system 100 for profiling event based exploit detection in accordance with an embodiment of the present disclosure. Instruction pointer analysis module 124 can be configured to analyze each instruction pointer (e.g., instruction pointers 142a-142d) in instruction pointer queue 122 and determine if the pointer points to an application region (e.g., application region 128a) in memory 118a or an orphan application region 130. If the instruction pointer associated with an application (e.g., application 114a) points to an application region associated with the application, then the application is not flagged as malicious. If the instruction pointer associated with the application points to an application not associated with the application (i.e., orphan application region 130), then the application is flagged or marked as malicious. For example, instruction pointers 142a-142d may each be associated with an application. Instruction pointers 142a, 142b, and 142d each point to module 144a, 144b, and 144c in an application region 128a associated with the application. However, instruction pointer 142c points to an orphan page 146 in orphan application region 130 that is not associated with the application which can be an indication of malicious activity.

[0036] Because instruction pointer 138c points to orphan page 142, the application is flagged or marked as malicious and security module 116 can do further analysis on the application. For example, security module 116 can be configured to compare the application to entries in white list 132 and black list 134. If the application matches an entry in white list 132, the application can be determined to be trusted. If the application matches an entry in black list 134, the application can be determined to be malicious. In an example, data related to the application can be sent to network security module 138a or 138b for a more intensive analysis.

[0037] Turning to FIGURE 3, FIGURE 3 is an example flowchart illustrating possible operations of a flow 300 that may be associated with profiling event based exploit detection, in accordance with an embodiment. In an embodiment, one or more operations of flow 300 may be performed by security module 116, event tracing module 120, and instruction pointer analysis module 124. At 302, event tracing for an application is performed. At 304, each instruction pointer from the event trace is analyzed. At 306, the system determines if the instruction pointer is pointing to an orphan region. If the instruction pointer is pointing to an

orphan region, then an alert is triggered, as in 308. The alert may flag the application as malicious, stop execution of the application, or some other mitigating action.

[0038] If the instruction pointer is not pointing to an orphan region then the system determines if the instruction pointer is pointer to a region associated with the application, as in 310. If the pointer is not pointing to a region associated with the application, then an alert is triggered, as in 308. If the instruction pointer is pointing to a region associated with the application, then no action is taken. For example, the application is not considered malicious and no mitigating action is taken.

[0039] Turning to FIGURE 4, FIGURE 4 is an example flowchart illustrating possible operations of a flow 400 that may be associated with profiling event based exploit detection, in accordance with an embodiment. In an embodiment, one or more operations of flow 400 may be performed by security module 116, event tracing module 120, and instruction pointer analysis module 124. At 402, a profiling event occurs. The profiling event may be for a process or application (e.g., application 114a). At 404, the system determines if the process needs to be monitored. For example, the process may match an entry in whitelist 132 and be a part of a trusted process or application that does not need to be monitored or may match an entry in blacklist 134 and be part of an untrusted process or application that needs to be monitored. If the process does not need to be monitored, then the profiling event is discarded, as in 406. If the process does need to be monitored, then each instruction pointer in the profiling even is analyzed, as in 408.

[0040] At 410, the system determines if the instruction pointer points to an orphan page. If the instruction pointer points to an orphan page, then the memory area is marked as non-executable, as in 418. At 412, the process is flagged as malicious. If the instruction pointer does not point to an orphan page, then a memory area associated with each instruction pointer is analyzed, as in 414.

[0041] At 416, the system determines if the memory (associated with each instruction pointer) includes an anomaly or exploit code. If the memory does not include an anomaly or exploit code, then the profiling event is discarded as in 406. If the memory area does include an anomaly or exploit code, then the memory area is marked as non-executable, as in 418. At 412, the process is flagged as malicious. If the process is flagged as malicious, then the

process may be terminated, sent to network security module 138a or 138b for further analysis, or some other mitigating action.

[0042] FIGURE 5 illustrates a computing system 500 that is arranged in a point-to-point (PtP) configuration according to an embodiment. In particular, FIGURE 5 shows a system where processors, memory, and input/output devices are interconnected by a number of point-to-point interfaces. Generally, one or more of the network elements of communication system 100 may be configured in the same or similar manner as computing system 500.

[0043] As illustrated in FIGURE 5, system 500 may include several processors, of which only two, processors 570 and 580, are shown for clarity. While two processors 570 and 580 are shown, it is to be understood that an embodiment of system 500 may also include only one such processor. Processors 570 and 580 may each include a set of cores (i.e., processor cores 574A and 574B and processor cores 584A and 584B) to execute multiple threads of a program. The cores may be configured to execute instruction code in a manner similar to that discussed above with reference to FIGURES 1-4. Each processor 570, 580 may include at least one shared cache 571, 581. Shared caches 571, 581 may store data (e.g., instructions) that are utilized by one or more components of processors 570, 580, such as processor cores 574 and 584.

[0044] Processors 570 and 580 may also each include integrated memory controller logic (MC) 572 and 582 to communicate with memory elements 532 and 534. Memory elements 532 and/or 534 may store various data used by processors 570 and 580. In alternative embodiments, memory controller logic 572 and 582 may be discrete logic separate from processors 570 and 580.

[0045] Processors 570 and 580 may be any type of processor and may exchange data via a point-to-point (PtP) interface 550 using point-to-point interface circuits 578 and 588, respectively. Processors 570 and 580 may each exchange data with a chipset 590 via individual point-to-point interfaces 552 and 554 using point-to-point interface circuits 576, 586, 594, and 598. Chipset 590 may also exchange data with a high-performance graphics circuit 538 via a high-performance graphics interface 539, using an interface circuit 592, which could be a PtP interface circuit. In alternative embodiments, any or all of the PtP links illustrated in FIGURE 5 could be implemented as a multi-drop bus rather than a PtP link.

[0046] Chipset 590 may be in communication with a bus 520 via an interface circuit 596. Bus 520 may have one or more devices that communicate over it, such as a bus bridge 518 and I/O devices 516. Via a bus 510, bus bridge 518 may be in communication with other devices such as a keyboard/mouse 512 (or other input devices such as a touch screen, trackball, etc.), communication devices 526 (such as modems, network interface devices, or other types of communication devices that may communicate through a computer network 560), audio I/O devices 514, and/or a data storage device 528. Data storage device 528 may store code 530, which may be executed by processors 570 and/or 580. In alternative embodiments, any portions of the bus architectures could be implemented with one or more PtP links.

[0047] The computer system depicted in FIGURE 5 is a schematic illustration of an embodiment of a computing system that may be utilized to implement various embodiments discussed herein. It will be appreciated that various components of the system depicted in FIGURE 5 may be combined in a system-on-a-chip (SoC) architecture or in any other suitable configuration. For example, embodiments disclosed herein can be incorporated into systems including mobile devices such as smart cellular telephones, tablet computers, personal digital assistants, portable gaming devices, etc. It will be appreciated that these mobile devices may be provided with SoC architectures in at least some embodiments.

[0048] Turning to FIGURE 6, FIGURE 6 is a simplified block diagram associated with an example ARM ecosystem SOC 600 of the present disclosure. At least one example implementation of the present disclosure can include the profiling event based exploit detection features discussed herein and an ARM component. For example, the example of FIGURE 6 can be associated with any ARM core (e.g., A-9, A-15, etc.). Further, the architecture can be part of any type of tablet, smartphone (inclusive of Android™ phones, iPhones™), iPad™, Google Nexus™, Microsoft Surface™, personal computer, server, video processing components, laptop computer (inclusive of any type of notebook), Ultrabook™ system, any type of touch-enabled input device, etc.

[0049] In this example of FIGURE 6, ARM ecosystem SOC 600 may include multiple cores 606-607, an L2 cache control 608, a bus interface unit 609, an L2 cache 610, a graphics processing unit (GPU) 615, an interconnect 602, a video codec 620, and a liquid crystal display

(LCD) I/F 625, which may be associated with mobile industry processor interface (MIPI)/ high-definition multimedia interface (HDMI) links that couple to an LCD.

[0050] ARM ecosystem SOC 600 may also include a subscriber identity module (SIM) I/F 630, a boot read-only memory (ROM) 635, a synchronous dynamic random access memory (SDRAM) controller 640, a flash controller 645, a serial peripheral interface (SPI) master 650, a suitable power control 655, a dynamic RAM (DRAM) 660, and flash 665. In addition, one or more example embodiments include one or more communication capabilities, interfaces, and features such as instances of Bluetooth™ 670, a 3G modem 675, a global positioning system (GPS) 680, and an 802.11 Wi-Fi 685.

[0051] In operation, the example of FIGURE 6 can offer processing capabilities, along with relatively low power consumption to enable computing of various types (e.g., mobile computing, high-end digital home, servers, wireless infrastructure, etc.). In addition, such an architecture can enable any number of software applications (e.g., Android™, Adobe® Flash® Player, Java Platform Standard Edition (Java SE), JavaFX, Linux, Microsoft Windows Embedded, Symbian and Ubuntu, etc.). In at least one example embodiment, the core processor may implement an out-of-order superscalar pipeline with a coupled low-latency level-2 cache.

[0052] FIGURE 7 illustrates a processor core 700 according to an embodiment. Processor core 700 may be the core for any type of processor, such as a micro-processor, an embedded processor, a digital signal processor (DSP), a network processor, or other device to execute code. Although only one processor core 700 is illustrated in Figure 7, a processor may alternatively include more than one of the processor core 700 illustrated in Figure 7. For example, processor core 700 represents one example embodiment of processors cores 574a, 574b, 584a, and 584b shown and described with reference to processors 570 and 580 of FIGURE 5. Processor core 700 may be a single-threaded core or, for at least one embodiment, processor core 700 may be multithreaded in that it may include more than one hardware thread context (or "logical processor") per core.

[0053] FIGURE 7 also illustrates a memory 702 coupled to processor core 700 in accordance with an embodiment. Memory 702 may be any of a wide variety of memories (including various layers of memory hierarchy) as are known or otherwise available to those of skill in the art. Memory 702 may include code 704, which may be one or more instructions,

to be executed by processor core 700. Processor core 700 can follow a program sequence of instructions indicated by code 704. Each instruction enters a front-end logic 706 and is processed by one or more decoders 708. The decoder may generate, as its output, a micro operation such as a fixed width micro operation in a predefined format, or may generate other instructions, microinstructions, or control signals that reflect the original code instruction. Front-end logic 706 also includes register renaming logic 710 and scheduling logic 712, which generally allocate resources and queue the operation corresponding to the instruction for execution.

[0054] Processor core 700 can also include execution logic 714 having a set of execution units 716-1 through 716-N. Some embodiments may include a number of execution units dedicated to specific functions or sets of functions. Other embodiments may include only one execution unit or one execution unit that can perform a particular function. Execution logic 714 performs the operations specified by code instructions.

[0055] After completion of execution of the operations specified by the code instructions, back-end logic 718 can retire the instructions of code 704. In one embodiment, processor core 700 allows out of order execution but requires in order retirement of instructions. Retirement logic 720 may take a variety of known forms (e.g., re-order buffers or the like). In this manner, processor core 700 is transformed during execution of code 704, at least in terms of the output generated by the decoder, hardware registers and tables utilized by register renaming logic 710, and any registers (not shown) modified by execution logic 714.

[0056] Although not illustrated in FIGURE 7, a processor may include other elements on a chip with processor core 700, at least some of which were shown and described herein with reference to FIGURE 5. For example, as shown in FIGURE 5, a processor may include memory control logic along with processor core 700. The processor may include I/O control logic and/or may include I/O control logic integrated with memory control logic.

[0057] Note that with the examples provided herein, interaction may be described in terms of two, three, or more network elements. However, this has been done for purposes of clarity and example only. In certain cases, it may be easier to describe one or more of the functionalities of a given set of flows by only referencing a limited number of network elements. It should be appreciated that communication system 100 and its teachings are

readily scalable and can accommodate a large number of components, as well as more complicated/sophisticated arrangements and configurations. Accordingly, the examples provided should not limit the scope or inhibit the broad teachings of communication system 100 as potentially applied to a myriad of other architectures.

[0058] It is also important to note that the operations in the preceding flow diagrams (i.e., FIGURES 3 and 4) illustrate only some of the possible correlating scenarios and patterns that may be executed by, or within, communication system 100. Some of these operations may be deleted or removed where appropriate, or these operations may be modified or changed considerably without departing from the scope of the present disclosure. In addition, a number of these operations have been described as being executed concurrently with, or in parallel to, one or more additional operations. However, the timing of these operations may be altered considerably. The preceding operational flows have been offered for purposes of example and discussion. Substantial flexibility is provided by communication system 100 in that any suitable arrangements, chronologies, configurations, and timing mechanisms may be provided without departing from the teachings of the present disclosure.

[0059] Although the present disclosure has been described in detail with reference to particular arrangements and configurations, these example configurations and arrangements may be changed significantly without departing from the scope of the present disclosure. Moreover, certain components may be combined, separated, eliminated, or added based on particular needs and implementations. Additionally, although communication system 100 has been illustrated with reference to particular elements and operations that facilitate the communication process, these elements and operations may be replaced by any suitable architecture, protocols, and/or processes that achieve the intended functionality of communication system 100.

[0060] Numerous other changes, substitutions, variations, alterations, and modifications may be ascertained to one skilled in the art and it is intended that the present disclosure encompass all such changes, substitutions, variations, alterations, and modifications as falling within the scope of the appended claims. In order to assist the United States Patent and Trademark Office (USPTO) and, additionally, any readers of any patent issued on this application in interpreting the claims appended hereto, Applicant wishes to

note that the Applicant: (a) does not intend any of the appended claims to invoke paragraph six (6) of 35 U.S.C. section 112 as it exists on the date of the filing hereof unless the words "means for" or "step for" are specifically used in the particular claims; and (b) does not intend, by any statement in the specification, to limit this disclosure in any way that is not otherwise reflected in the appended claims.

OTHER NOTES AND EXAMPLES

[0061] Example C1 is at least one machine readable storage medium having one or more instructions that when executed by a processor cause the processor to execute an application in a system, perform event tracing for the application, analyze each instruction pointer from the event tracing, and determine if an instruction pointer points to an orphan page of memory.

[0062] In Example C2, the subject matter of Example C1 can optionally include where the orphan page is a region of code that is not associated with the application.

[0063] In Example C3, the subject matter of any one of Examples C1-C2 can optionally include where the orphan page is a region of code that is unidentified.

[0064] In Example C4, the subject matter of any one of Examples C1-C3 can optionally include where the system includes an operating system and an embedded application that is part of the operating system performs the event tracing.

[0065] In Example C5, the subject matter of any one of Examples C1-C4 can optionally include where the instructions, when executed by the processor, further cause the processor to determine if the instruction pointer points to unusual code that is not associated with the application.

[0066] In Example C6, the subject matter of any one of Example C1-C5 can optionally include where the event tracing is transparent to the application.

[0067] In Example C7, the subject matter of any one of Examples C1-C6 can optionally include where the event tracing is asynchronous with the application such that the target application continues execution without being blocked.

[0068] In Example C8, the subject matter of any one of Examples C1-C7 can optionally include where the instructions, when executed by the processor, further cause the processor to flag the application as malicious if the instruction pointer points to an orphan page of

memory or the instruction pointer points to unusual code that is not associated with the application.

[0069] In Example A1, an apparatus can include a detection module, wherein the detection module is configured to execute an application in a system, perform event tracing for the application, analyze each instruction pointer from the event tracing, and determine if an instruction pointer points to an orphan page of memory.

[0070] In Example, A2, the subject matter of Example A1 can optionally include where the orphan page is a region of code that is not associated with the application.

[0071] In Example A3, the subject matter of any one of Examples A1-A2 can optionally include where the orphan page is a region of code that is unidentified.

[0072] In Example A4, the subject matter of any one of Examples A1-A3 can optionally include where the system includes an operating system and an embedded application that is part of the operating system performs the event tracing.

[0073] In Example A5, the subject matter of any one of Examples A1-A4 can optionally include where the detection module is further configured to determine if the instruction pointer points to unusual code that is not associated with the application.

[0074] In Example A6, the subject matter of any one of Examples A1-A5 can optionally include where the event tracing is transparent to the application.

[0075] In Example A7, the subject matter of any one of Examples A1-A6 can optionally include where the event tracing is asynchronous with the application such that the target application continues execution without being blocked.

[0076] Example M1 is a method including executing an application in a system, performing event tracing for the application, analyzing each instruction pointer from the event tracing, and determining if an instruction pointer points to an orphan page of memory.

[0077] In Example M2, the subject matter of Example M1 can optionally include where the orphan page is a region of code that is not associated with the application.

[0078] In Example M3, the subject matter of any one of the Examples M1-M2 can optionally include where the orphan page is a region of code that is unidentified.

[0079] In Example M4, the subject matter of any one of the Examples M1-M3 can optionally include where the system includes an operating system and an embedded application that is part of the operating system performs the event tracing.

[0080] In Example M5, the subject matter of any one of the Examples M1-M4 can optionally include determining if the instruction pointer points to unusual code that is not associated with the application.

[0081] In Example M6, the subject matter of any one of the Examples M1-M5 can optionally include where the event tracing is transparent to the application.

[0082] In Example M7, the subject matter of any one of the Examples M1-M6 can optionally include where the event tracing is asynchronous with the application such that the target application continues execution without being blocked.

[0083] Example S1 is a system for profiling event based exploit detection, the system including a detection module configured to execute an application in a system, perform event tracing for the application, analyze each instruction pointer from the event tracing, and determine if an instruction pointer points to an orphan page of memory.

[0084] In Example S2, the subject matter of Example S1 can optionally include where the orphan page is a region of code that is not associated with the application.

[0085] In Example S3, the subject matter of any of the Examples S1-S2 can optionally include where the system includes an operating system and an embedded application that is part of the operating system performs the event tracing.

[0086] Example X1 is a machine-readable storage medium including machine-readable instructions to implement a method or realize an apparatus as in any one of the Examples A1-A7, or M1-M7. Example Y1 is an apparatus comprising means for performing of any of the Example methods M1-M7. In Example Y2, the subject matter of Example Y1 can optionally include the means for performing the method comprising a processor and a memory. In Example Y3, the subject matter of Example Y2 can optionally include the memory comprising machine-readable instructions.

CLAIMS:

1. At least one computer-readable medium comprising one or more instructions that when executed by a processor, cause the processor to:

execute an application in a system;

perform event tracing for the application;

analyze each instruction pointer from the event tracing; and

determine if an instruction pointer points to an orphan page of memory.

2. The at least one computer-readable medium of Claim 1, wherein the orphan page is a region of code that is not associated with the application.

3. The at least one computer-readable medium of any of Claims 1 and 2, wherein the orphan page is a region of code that is unidentified.

4. The at least one computer-readable medium of any of Claims 1-3, wherein the system includes an operating system and an embedded application that is part of the operating system performs the event tracing.

5. The at least one computer-readable medium of any of Claims 1-4, further comprising one or more instructions that when executed by the processor:

determine if the instruction pointer points to unusual code that is not associated with the application.

6. The at least one computer-readable medium of any of Claims 1-5, wherein the event tracing is transparent to the application.

7. The at least one computer-readable medium of any of Claims 1-6, wherein the event tracing is asynchronous with the application such that the target application continues execution without being blocked.

8. The at least one computer-readable medium of any of Claims 1-7, further comprising one or more instructions that when executed by the processor:

flag the application as malicious if the instruction pointer points to an orphan page of memory or the instruction pointer points to unusual code that is not associated with the application.

9. An apparatus comprising:

a detection module, wherein the detection module is configured to:

execute an application in a system;

perform event tracing for the application;

analyze each instruction pointer from the event tracing; and

determine if an instruction pointer points to an orphan page of memory.

10. The apparatus of Claim 9, wherein the orphan page is a region of code that is not associated with the application.

11. The apparatus of any of Claims 9 and 10, wherein the orphan page is a region of code that is unidentified.

12. The apparatus of any of Claims 9-11, wherein the system includes an operating system and an embedded application that is part of the operating system performs the event tracing.

13. The apparatus of any of Claims 9-12, wherein the detection module is further configured to:

determine if the instruction pointer points to unusual code that is not associated with the application.

14. The apparatus of any of Claims 9-13, wherein the event tracing is transparent to the application.

15. The apparatus of any of Claims 9-14, wherein the event tracing is asynchronous with the application such that the target application continues execution without being blocked.

16. A method comprising:

executing an application in a system;

performing event tracing for the application;

analyzing each instruction pointer from the event tracing; and

determining if an instruction pointer points to an orphan page of memory.

17. The method of Claim 16, wherein the orphan page is a region of code that is not associated with the application.

18. The method of any of Claims 16 and 17, wherein the orphan page is a region of code that is unidentified.

19. The method of any of Claims 16-18, wherein the system includes an operating system and an embedded application that is part of the operating system performs the event tracing.

20. The method of any of Claims 16-19, further comprising:

determining if the instruction pointer points to unusual code that is not associated with the application.

21. The method of any of Claims 16-20, wherein the event tracing is transparent to the application.

22. The method of any of Claims 16-21, wherein the event tracing is asynchronous with the application such that the target application continues execution without being blocked.

23. A system for profiling event based exploit detection, the system comprising:

a detection module, wherein the detection module is configured to:

execute an application in a system;

perform event tracing for the application;

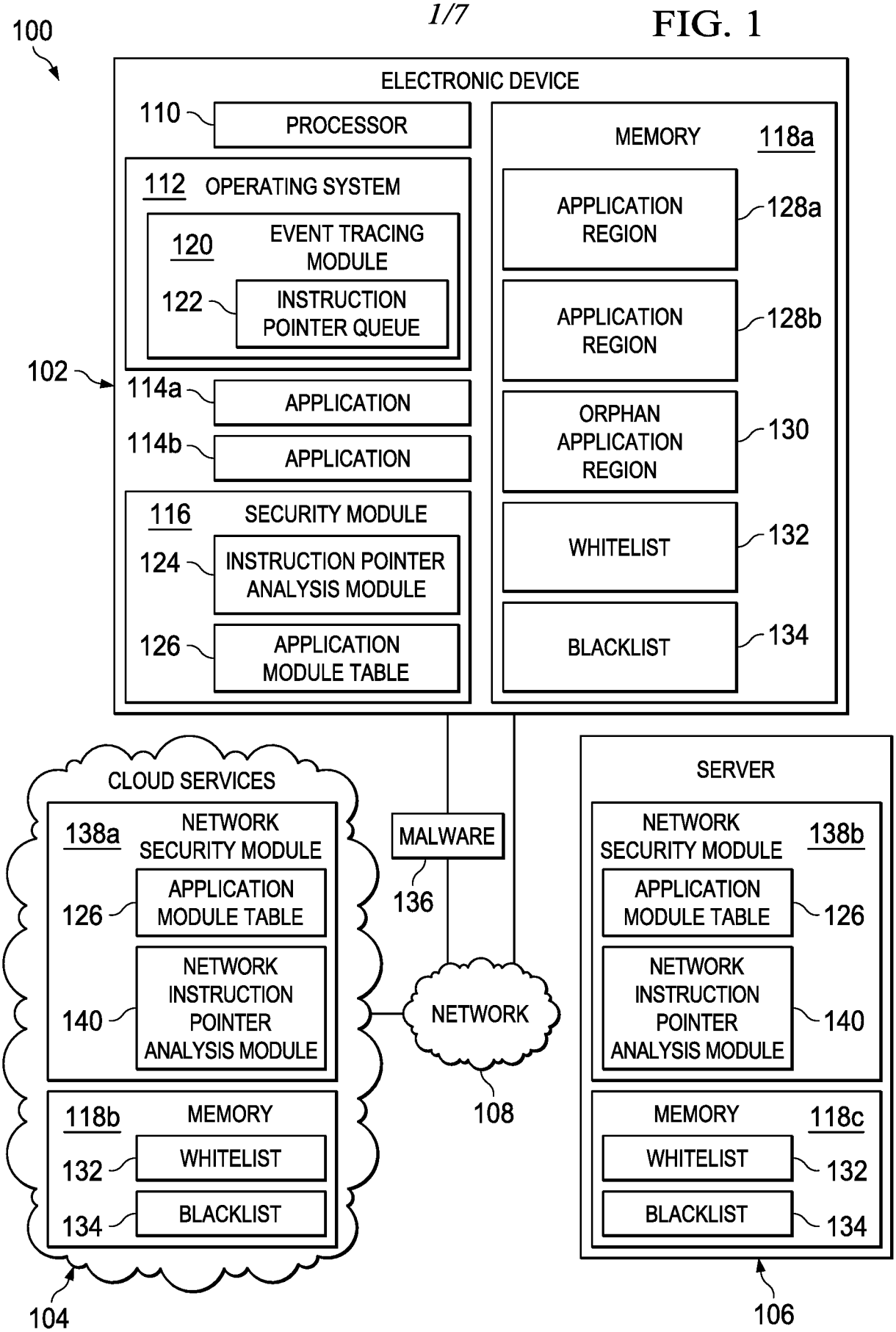
analyze each instruction pointer from the event tracing; and

determine if an instruction pointer points to an orphan page of memory.

24. The system of Claim 23, wherein the orphan page is a region of code that is not associated with the application.

25. The system of any of Claims 23 and 24, wherein the system includes an operating system and an embedded application that is part of the operating system performs the event tracing.

FIG. 1


100
ELECTRONIC DEVICE
110
PROCESSOR
112
OPERATING SYSTEM
120
EVENT TRACING MODULE
122
INSTRUCTION POINTER QUEUE
102
114a
APPLICATION
114b
APPLICATION
116
SECURITY MODULE
124
INSTRUCTION POINTER ANALYSIS MODULE
126
APPLICATION MODULE TABLE
MEMORY
118a
APPLICATION REGION
128a
APPLICATION REGION
128b
ORPHAN APPLICATION REGION
130
WHITELIST
132
BLACKLIST
134
CLOUD SERVICES
138a
NETWORK SECURITY MODULE
126
APPLICATION MODULE TABLE
140
NETWORK INSTRUCTION POINTER ANALYSIS MODULE
118b
MEMORY
132
WHITELIST
134
BLACKLIST
104
MALWARE
136
NETWORK
108
SERVER
NETWORK SECURITY MODULE
138b
APPLICATION MODULE TABLE
126
NETWORK INSTRUCTION POINTER ANALYSIS MODULE
140
MEMORY
118c
WHITELIST
132
BLACKLIST
134
106

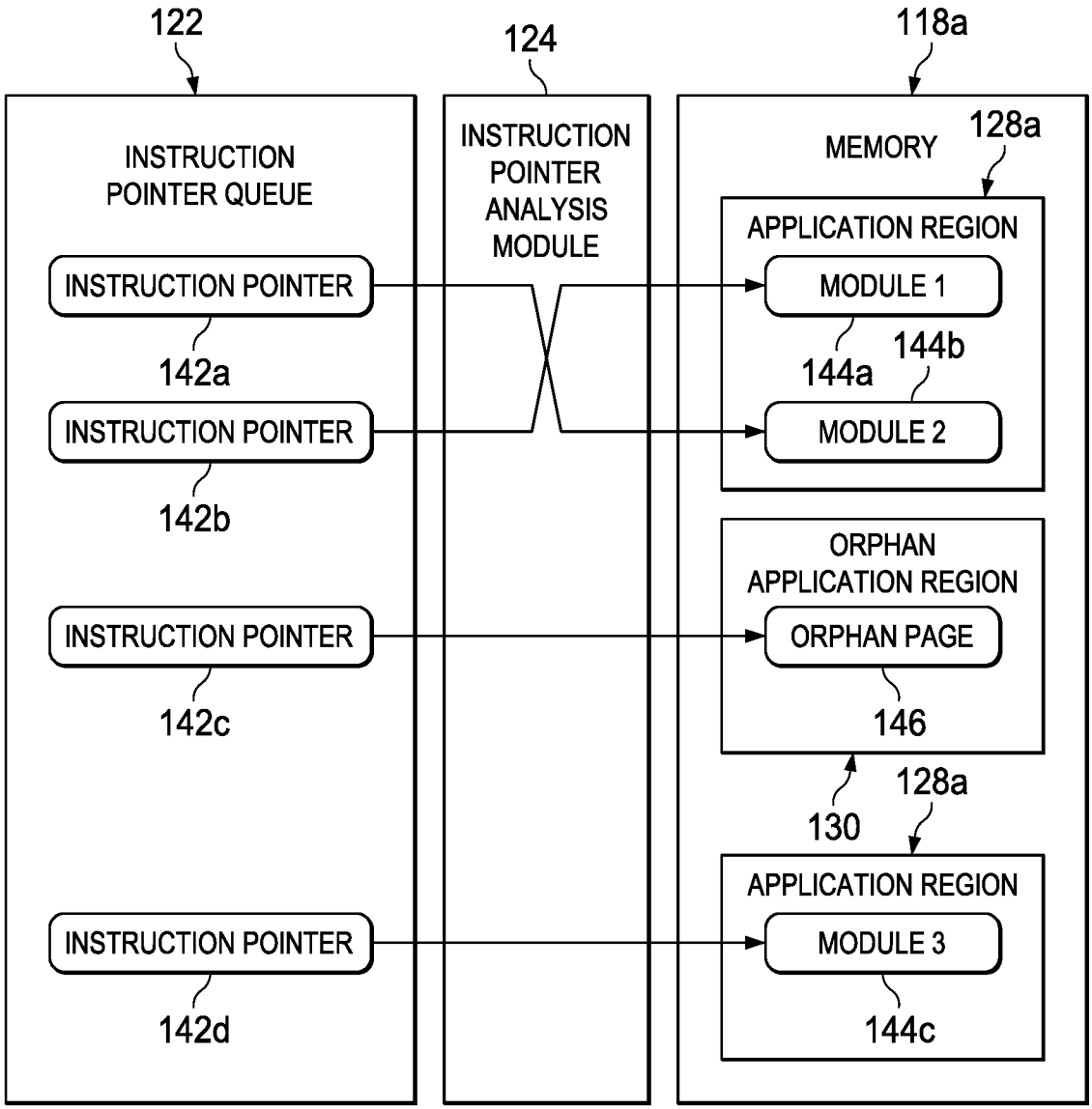

122
124
118a
INSTRUCTION POINTER QUEUE
INSTRUCTION POINTER ANALYSIS MODULE
MEMORY
128a
APPLICATION REGION
INSTRUCTION POINTER
MODULE 1
142a
144a
144b
INSTRUCTION POINTER
MODULE 2
142b
ORPHAN APPLICATION REGION
INSTRUCTION POINTER
ORPHAN PAGE
142c
146
130
128a
APPLICATION REGION
INSTRUCTION POINTER
MODULE 3
142d
144c

FIG. 2

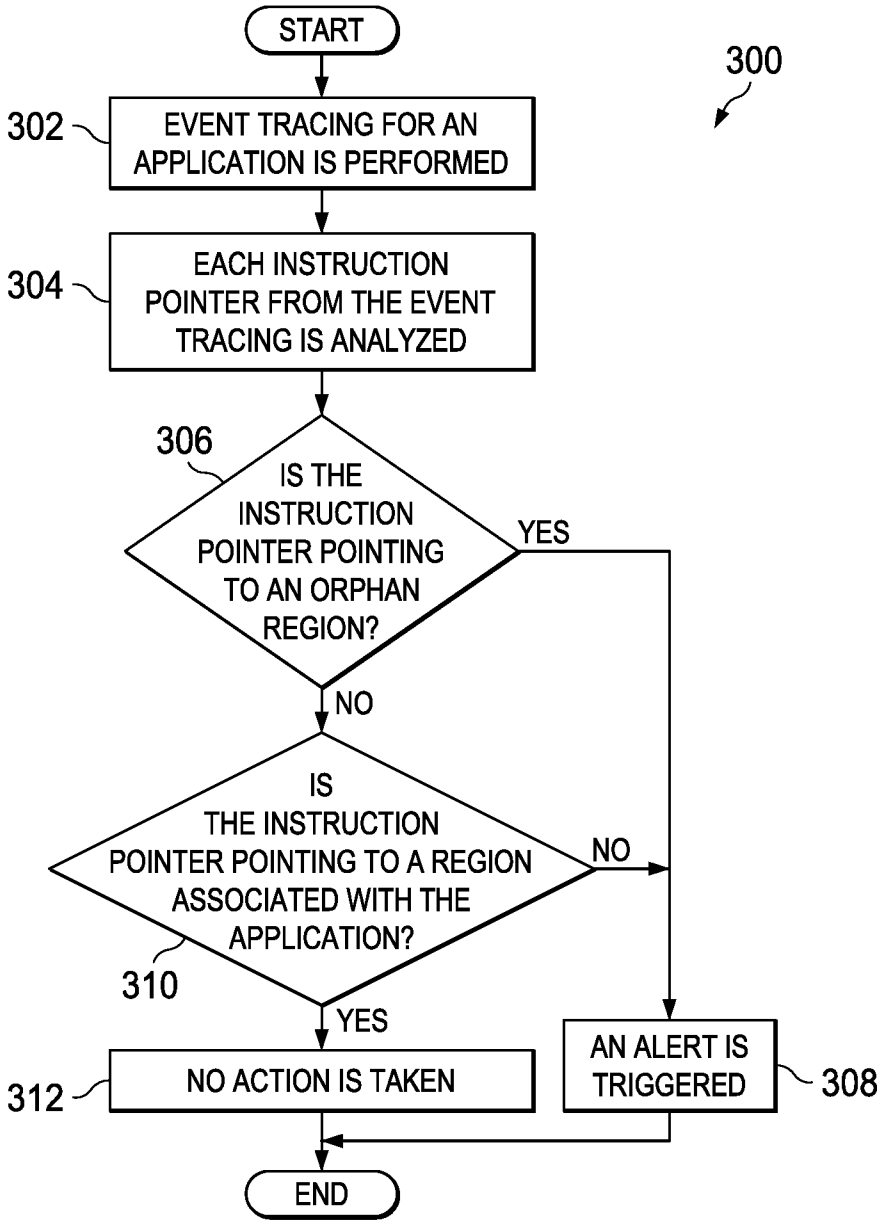

START
300
302
EVENT TRACING FOR AN APPLICATION IS PERFORMED
304
EACH INSTRUCTION POINTER FROM THE EVENT TRACING IS ANALYZED
306
IS THE INSTRUCTION POINTER POINTING TO AN ORPHAN REGION?
YES
NO
IS THE INSTRUCTION POINTER POINTING TO A REGION ASSOCIATED WITH THE APPLICATION?
NO
310
YES
312
NO ACTION IS TAKEN
AN ALERT IS TRIGGERED
308
END

FIG. 3

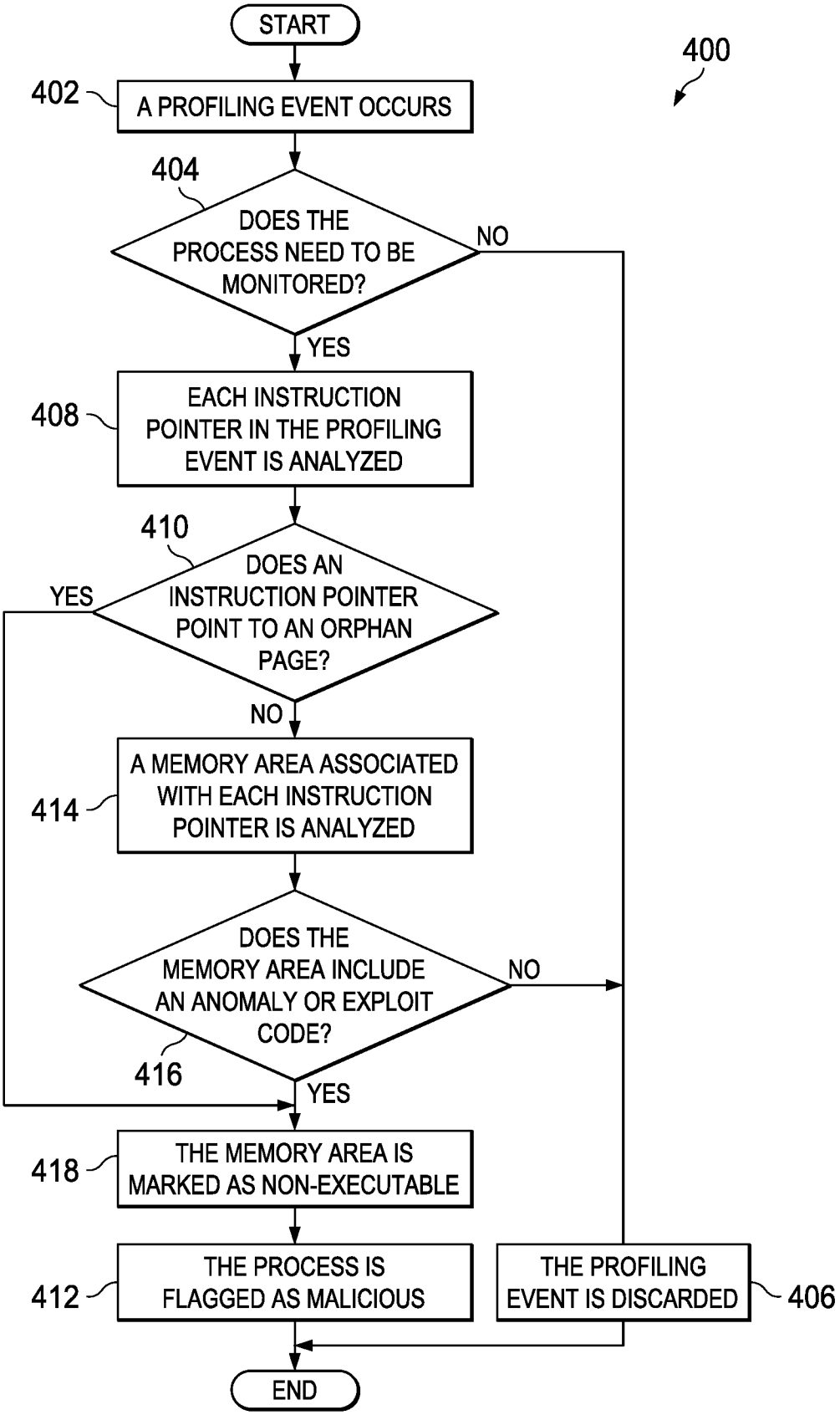

START
400
402
A PROFILING EVENT OCCURS
404
DOES THE PROCESS NEED TO BE MONITORED?
NO
YES
408
EACH INSTRUCTION POINTER IN THE PROFILING EVENT IS ANALYZED
410
DOES AN INSTRUCTION POINTER POINT TO AN ORPHAN PAGE?
YES
NO
414
A MEMORY AREA ASSOCIATED WITH EACH INSTRUCTION POINTER IS ANALYZED
DOES THE MEMORY AREA INCLUDE AN ANOMALY OR EXPLOIT CODE?
NO
416
YES
418
THE MEMORY AREA IS MARKED AS NON-EXECUTABLE
412
THE PROCESS IS FLAGGED AS MALICIOUS
THE PROFILING EVENT IS DISCARDED
406
END

FIG. 4

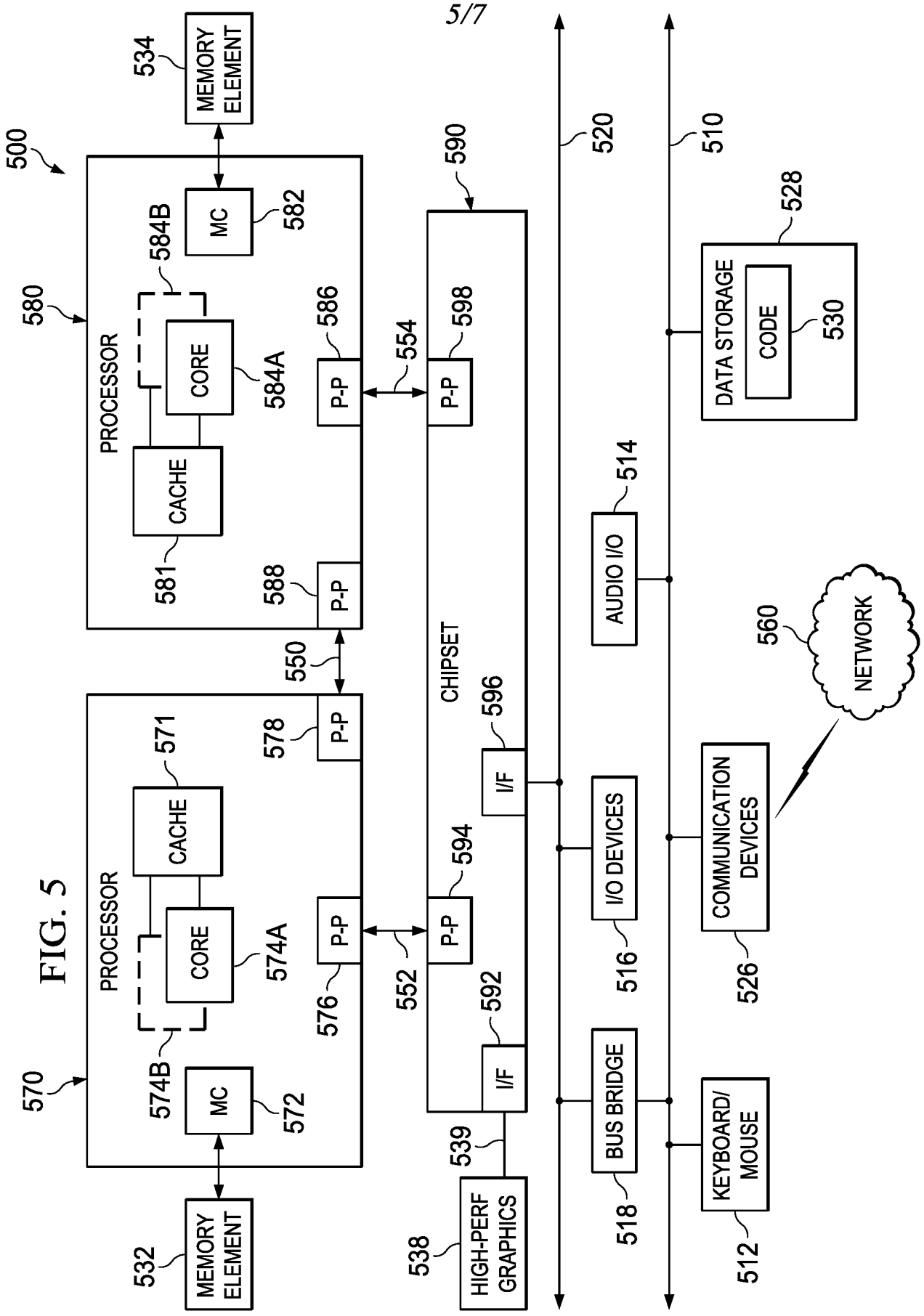

500
534
MEMORY ELEMENT
MC
582
584B
580
PROCESSOR
CORE
584A
CACHE
581
586
P-P
588
P-P
554
550
590
598
P-P
CHIPSET
596
I/F
578
P-P
571
CACHE
CORE
574A
574B
MC
572
570
576
P-P
552
P-P
594
592
I/F
539
538
HIGH-PERF GRAPHICS
532
MEMORY ELEMENT
520
510
528
DATA STORAGE
CODE
530
514
AUDIO I/O
560
NETWORK
I/O DEVICES
516
COMMUNICATION DEVICES
526
BUS BRIDGE
518
KEYBOARD/ MOUSE
512

FIG. 5

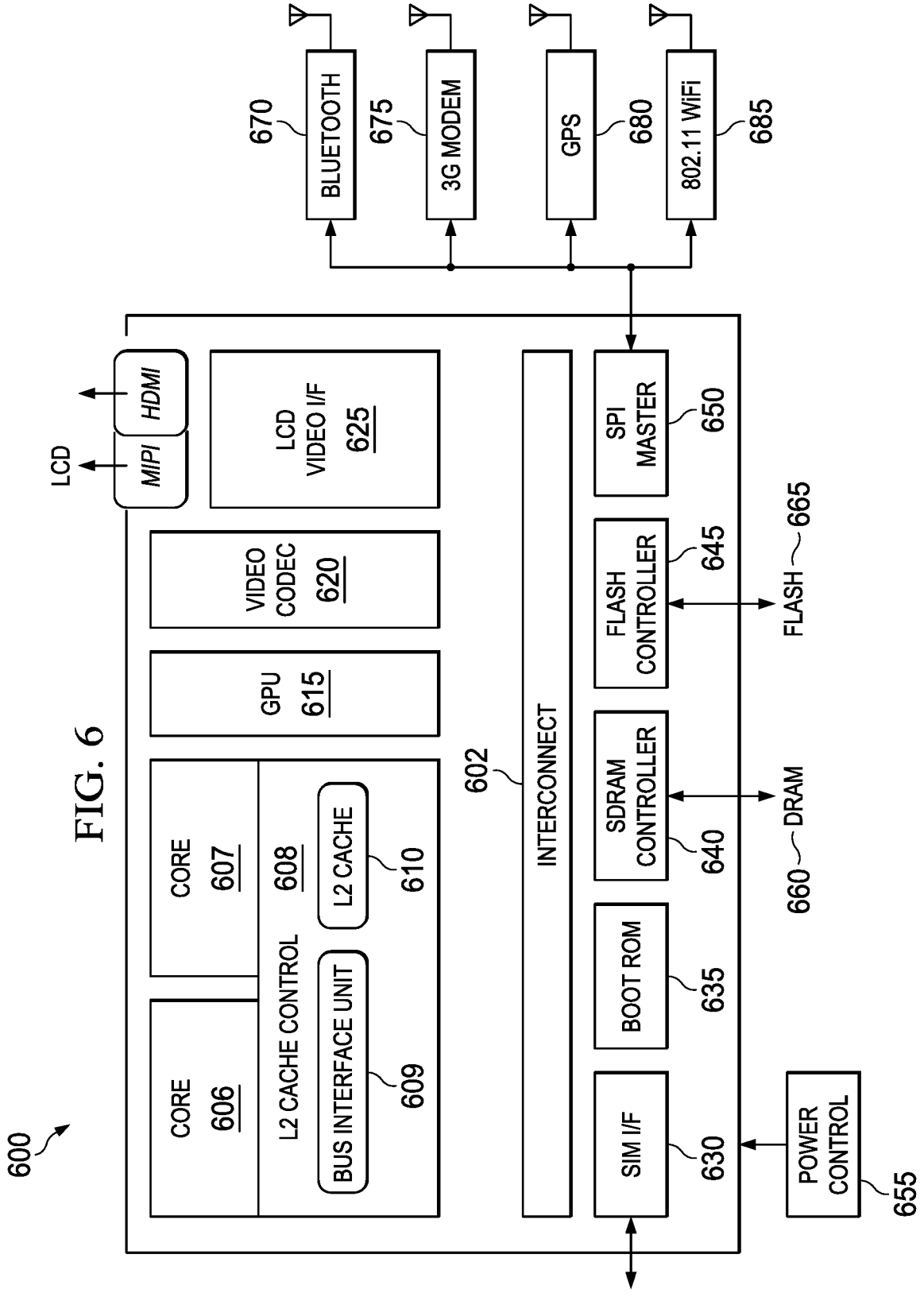

600
CORE 606
CORE 607
L2 CACHE CONTROL 608
BUS INTERFACE UNIT 609
L2 CACHE 610
GPU 615
VIDEO CODEC 620
LCD VIDEO I/F 625
HDMI
MIPI
LCD
602
INTERCONNECT
SIM I/F 630
BOOT ROM 635
SDRAM CONTROLLER 640
FLASH CONTROLLER 645
SPI MASTER 650
POWER CONTROL 655
660 DRAM
FLASH 665
670 BLUETOOTH
675 3G MODEM
680 GPS
685 802.11 WiFi

FIG. 6

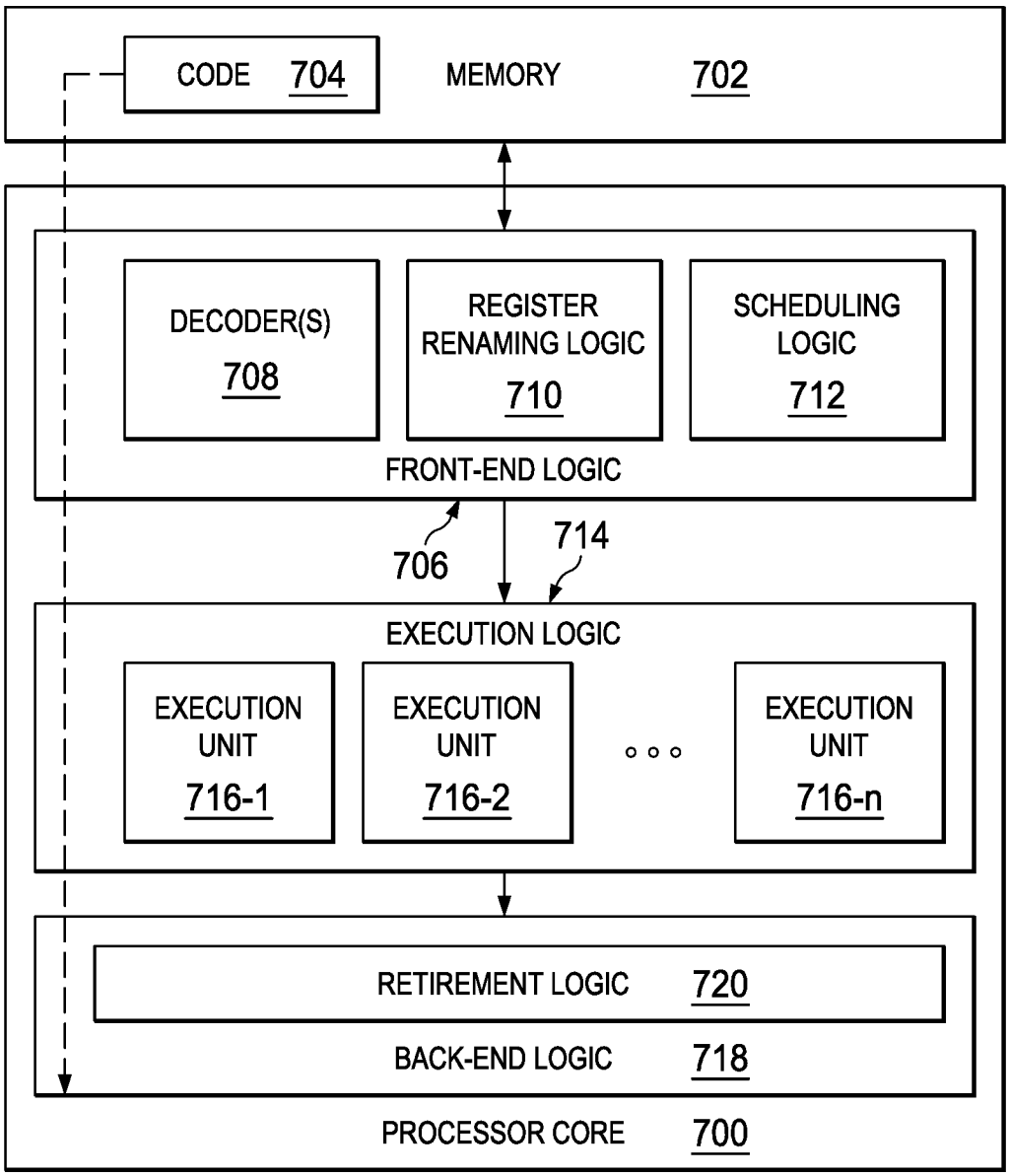

CODE 704
MEMORY 702
DECODER(S) 708
REGISTER RENAMING LOGIC 710
SCHEDULING LOGIC 712
FRONT-END LOGIC
706
714
EXECUTION LOGIC
EXECUTION UNIT 716-1
EXECUTION UNIT 716-2
EXECUTION UNIT 716-n
RETIREMENT LOGIC 720
BACK-END LOGIC 718
PROCESSOR CORE 700

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/033691

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/56(2013.01)I, G06F 21/57(2013.01)I, G06F 21/51(2013.01)I, G06F 21/62(2013.01)I

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 21/56; G06F 21/54; G06F 12/00; G06F 12/14; G06F 21/06; G06F 21/00; G06F 21/57; G06F 21/51; G06F 21/62

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: application, event tracing, instruction pointer, orphan page, detection, code, region

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2011-0265182 A1 (MARCUS PEINADO et al.) 27 October 2011 See paragraphs [0023]-[0024], [0048], [0066]; claim 9; and figures 2-3.	1-25
Y	US 2005-0246522 A1 (ANDERS SAMUELSSON et al.) 03 November 2005 See paragraphs [0018], [0022]; and figure 1.	1-25
A	US 2015-0067763 A1 (GREGORY WILLIAM DALCHER et al.) 05 March 2015 See paragraph [0036]; and figure 2.	1-25
A	US 2013-0227680 A1 (KASPERSKY LAB ZAO) 29 August 2013 See paragraph [0010]; and figure 4.	1-25
A	US 2012-0255013 A1 (AHMED SAID SALLAM) 04 October 2012 See paragraph [0016]; and figure 3.	1-25

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier 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
08 August 2016 (08.08.2016)	**08 August 2016 (08.08.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
LEE, EUN KYU
Telephone No. +82-42-481-3580

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2016/033691

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0265182 A1	27/10/2011	US 8566944 B2	22/10/2013
US 2005-0246522 A1	03/11/2005	US 7530093 B2	05/05/2009
US 2015-0067763 A1	05/03/2015	CN 105408911 A	16/03/2016
		WO 2015-030748 A1	05/03/2015
US 2013-0227680 A1	29/08/2013	EP 2720170 A1	16/04/2014
		RU 2012106465 A	27/08/2013
		RU 2510074 C2	20/03/2014
		US 8990934 B2	24/03/2015
US 2012-0255013 A1	04/10/2012	CN 103620613 A	05/03/2014
		EP 2691908 A2	05/02/2014
		JP 2014-514651 A	19/06/2014
		KR 10-1626424 B1	01/06/2016
		US 2012-0254982 A1	04/10/2012
		US 2012-0254993 A1	04/10/2012
		US 2012-0254994 A1	04/10/2012
		US 2012-0254995 A1	04/10/2012
		US 2012-0254999 A1	04/10/2012
		US 2012-0255000 A1	04/10/2012
		US 2012-0255001 A1	04/10/2012
		US 2012-0255002 A1	04/10/2012
		US 2012-0255003 A1	04/10/2012
		US 2012-0255004 A1	04/10/2012
		US 2012-0255010 A1	04/10/2012
		US 2012-0255011 A1	04/10/2012
		US 2012-0255012 A1	04/10/2012
		US 2012-0255014 A1	04/10/2012
		US 2012-0255016 A1	04/10/2012
		US 2012-0255017 A1	04/10/2012
		US 2012-0255018 A1	04/10/2012
		US 2012-0255021 A1	04/10/2012
		US 2012-0255031 A1	04/10/2012
		US 8549644 B2	01/10/2013
		US 8549648 B2	01/10/2013
		US 8621620 B2	31/12/2013
		US 8650641 B2	11/02/2014
		US 8650642 B2	11/02/2014
		US 8813227 B2	19/08/2014
		US 8863283 B2	14/10/2014
		US 8925089 B2	30/12/2014
		US 8959638 B2	17/02/2015
		US 8966624 B2	24/02/2015
		US 8966629 B2	24/02/2015
		US 9032525 B2	12/05/2015
		US 9038176 B2	19/05/2015
		US 9087199 B2	21/07/2015

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/033691

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 9262246 B2	16/02/2016
		US 9317690 B2	19/04/2016
		WO 2012-135192 A2	04/10/2012
		WO 2012-135192 A3	28/02/2013