

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

(19) World Intellectual Property Organization

International Bureau



(10) International Publication Number

WO 2017/003587 A1

(43) International Publication Date
5 January 2017 (05.01.2017)

WIPO | PCT

(51) International Patent Classification:
G06F 21/56 (2013.01) *H04L 29/06* (2006.01)
G06F 21/57 (2013.01)

(21) International Application Number:
PCT/US2016/033977

(22) International Filing Date:
25 May 2016 (25.05.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/752,901 27 June 2015 (27.06.2015) US

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

(72) Inventors: EDWARDS, Jonathan L.; 2535 NE 46th Avenue, Portland, Oregon 97213 (US). SPURLOCK, Joel R.; 3153 NE Hoyt Street, Portland, Oregon 97232 (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: DETECTION OF MALWARE

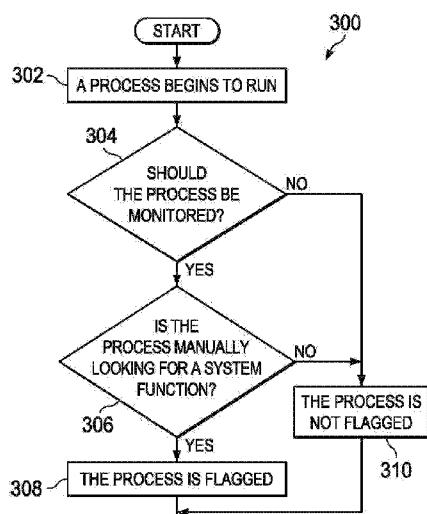


FIG. 3

(57) Abstract: Particular embodiments described herein provide for an electronic device that can be configured to monitor a process, determine if the process is parsing to look for one or more system functions, and flag the process if the process is parsing to look for one or more system functions. In an example, the process can be determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables. In another example, the process can be determined to be parsing to look for one or more system functions if the process calls GetProcAddress.



Published:

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

DETECTION OF MALWARE

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of and priority to U.S. Nonprovisional (Utility) Patent Application No. 14/752,901 filed 27 June 2015 entitled, "DETECTION OF MALWARE", 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 the detection of malware.

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 and devices.

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 the detection of malware in accordance with an embodiment of the present disclosure;

[0007] FIGURE 2 is a simplified block diagram of a portion a communication system for the detection of malware 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 the detection of malware in accordance with an embodiment of the present disclosure. As illustrated in FIGURE 1, an embodiment of communication system 100 can include electronic device 102, cloud services 104, and a server 106. Electronic device 102 can include an operating system (OS) 110, memory 112, a processor 114, a hypervisor 116, a security module 118, and at least one application 120. OS 110 can include OS functions 122 and OS variables 124. Memory 112 can include a shared library 126. Security module 118 can include a system process monitoring module 128, a whitelist 130, and a blacklist 132. Cloud services 104 and

server 106 can each include a network security module 134. Network security module 124 can include whitelist 130 and blacklist 132. Electronic device 102, cloud services 104, and server 106 can be in communication using network 108. In an example, malicious device 136 may attempt to use network 108 or some other means (e.g., a physical connection) to infect electronic device 102 with malicious code 138.

[0015] In example embodiments, communication system 100 can be configured to monitor threads of a process and determine if a thread is trying to lookup a function that the process should already know. Generally, code that is a part of legitimate software or a legitimate application does not need to look for common functions to interact with the operating system because the common functions are available in published libraries and linked against export libraries and a dynamic link library (DLL) loader can resolve addresses automatically. However, malicious code often does not know the location of various function calls and the malicious code must first find the functions before it can execute. By marking certain files and regions related to system functions unreadable, the system can analyze what is reading the files and regions to locate the system function and make a determination if the code is trusted or malicious.

[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] Malicious code 138 may be malware or malicious software that infects a host computer (e.g., electronic device 102) 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. One common malware feature is to use shellcode to exploit a vulnerability in software running on a machine. Shellcode is a piece of code used as the payload in the exploitation of the software vulnerability. It is called "shellcode" because it typically starts a command shell from which the attacker can control the compromised machine. Before the shellcode can effectively infect a machine, it needs to find OS functions or routines (e.g., LoadLibrary, CreateFile, etc.) to execute its payload. In order to find OS routines, the shell code can call GetProcAddress or parse portable executable (PE) headers to find and interpret DLLs' import and export tables. What is needed is a security solution that provides a system and method to detect the shellcode and identify malicious activity.

[0019] A communication system for the detection of malware, as outlined in FIGURE 1, can resolve these issues (and others). Communication system 100 may be configured to use hypervisor (e.g., hypervisor 116) memory based monitoring to monitor code as it is executing and accessing data. For example, memory read monitoring can be used on the data structures that malware needs to read in order to find the OS functions that the malware may need before it can execute. When a DLL exports some function to a process, the information about the location of the start of the function can be found as well as the name of the function that is stored in tables (e.g., export tables) which are pointed to by well known structures at the beginning of the DLL. Communication system 100 can be configured to use the hypervisor to make those structures and tables unreadable so that when a process does read them, the system can analyze the process and look at the pattern of the accesses and the code that is accessing the structures or tables. From the pattern and the bytes being accessed the system can determine what function is being looked for and can determine if the code is a malicious attempt to find the functions.

[0020] For example, system process monitoring module 128 can be configured to analyze code (e.g., from application 120) that is looking up OS functions (e.g., OS functions 122) and OS variables (e.g., OS variables 124). In a shared library of the system (e.g., shared library 126), only the structures that indicate where to find the OS functions or OS variables are made unreadable as there is no benefit in making the code unreadable. Areas of memory

that may be protected and marked unreadable can include the import and export tables, DLL, PE files, etc.

[0021] 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.

[0022] 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.

[0023] 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.

[0024] In an example implementation, electronic device 102, cloud services 104, and server 106 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.

[0025] 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.

[0026] 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.

[0027] 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 118, system process monitoring module 128, and network security module 134) 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.

[0028] 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.'

[0029] 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 118 is represented in FIGURE 1 as being located in electronic device 102, this is for illustrative purposes only. Security module 118 could be combined or separated in any suitable configuration. Furthermore, security module 118 could be integrated with or distributed in another network accessible by electronic device 102 such as cloud services 104 or server 106.

[0030] Turning to FIGURE 2, FIGURE 2 is a simplified block diagram of a portion of a communication system 100 for the detection of malware. As illustrated in FIGURE 2, electronic device 102 can include OS 110, memory 112, security module 118, and application 120. OS 110 can include OS functions 122 and OS variables 124. Memory 112 can include a DLL 140, import and export tables 142, one or more PE file 144, and GetProcAddress 148. Security module 118 can include system process monitoring module 128, whitelist 130, and blacklist 132. Application 120 can include shell code 146. Each PE file 144 can include a header 150. GetProcAddress 148 can retrieve the address of an exported function or variable from DLL 140.

[0031] If application is malicious or includes malicious code 138, before shellcode 146 can effectively infect a machine, it needs to find operating system functions or routines (e.g., example LoadLibrary, CreateFile, etc.) to execute its payload. In order to find OS routines, the shell code can call GetProcAddress 148 or parse to look for PE headers from PE files 144 to find and interpret DLLs' or import and export tables 142. For example, when DLL 140 exports some functions to a process, the information about the start of the function can be found as well as the name of the function. The name of the function can be stored in import and export tables 142 which are pointed to by well known structures at the beginning of DLL 140. Whitelist 122 can include entries of known clean or trusted applications, code, strings, etc. and can be used to reduce false positives. Blacklist 124 can include entries of known malicious or untrusted applications, code, strings, etc.

[0032] Turning to FIGURE 3, FIGURE 3 is an example flowchart illustrating possible operations of a flow 300 that may be associated with the detection of malware, in accordance with an embodiment. At 302, a process begins to run. At 304, the system determines if the

process should be monitored. If the process should not be monitored, then the process is not flagged as in 310. For example, the process may be found in whitelist 130 and may be classified as trusted. In addition, the process may be a process that is not typically monitored for malware. If the process should be monitored (e.g., the application is unknown or is found in blacklist 132), then the system determines if the process is manually looking for (e.g., parsing to look for) a system function, as in 306. If the process is not manually looking for (e.g., parsing to look for) a system function, then the process is not flagged as in 310. If the process is manually looking for (e.g., parsing to look for) a system function, then the process is flagged, as in 308. By flagging the process, the process may be analyzed for malware by security module 118 or sent to a network element for further analysis (e.g. by network security module 134).

[0033] Turning to FIGURE 4, FIGURE 4 is an example flowchart illustrating possible operations of a flow 400 that may be associated with the detection of malware, in accordance with an embodiment. At 402, an application begins to execute. At 404, the application begins to parse PE files to manually (e.g., parsing to) find and interpret DLL tables. At 406, the application is flagged for further analysis to determine if the application is malicious. For example, the process may be analyzed for malware by security module 118 or sent to a network element for further analysis (e.g. by network security module 134).

[0034] 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.

[0035] 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 1-5. 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.

[0036] 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 discreet logic separate from processors 570 and 580.

[0037] 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.

[0038] 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.

[0039] 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.

[0040] 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 detection of malware features discussed herein and an ARM component. For example, the example of FIGURE 6 can be associated with any ARM core (e.g., A-7, 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.

[0041] 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.

[0042] 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.

[0043] 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.

[0044] 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.

[0045] 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.

[0046] 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.

[0047] 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.

[0048] 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.

[0049] 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.

[0050] It is also important to note that the operations in the preceding flow diagrams (i.e., FIGURES 3-5B) 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.

[0051] 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

[0052] 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

[0053] Example C1 is at least one machine readable medium having one or more instructions that when executed by at least one processor, cause the at least one processor to monitor a process, determine if the process is parsing to look for one or more system functions, and flag the process if the process is parsing to look for one or more system functions.

[0054] In Example C2, the subject matter of Example C1 can optionally include where the process is determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables.

[0055] In Example C3, the subject matter of any one of Examples C1-C2 can optionally include where the process is determined to be parsing to look for one or more system functions if the process calls GetProcAddress.

[0056] In Example C4, the subject matter of any one of Examples C1-C3 can optionally include where the process includes shellcode.

[0057] In Example C5, the subject matter of any one of Examples C1-C4 can optionally include where the one or more instructions that when executed by the at least one processor, further cause the at least one processor to analyze the process for malware.

[0058] In Example C6, the subject matter of any one of Example C1-C5 can optionally include where the one or more instructions that when executed by the at least one processor, further cause the at least one processor to remove the flag if the process is found in a whitelist.

[0059] In Example A1, an apparatus can include a system process monitoring module. The system process monitoring module can be configured to monitor a process, determine if the process is parsing to look for one or more system functions, and flag the process if the process is parsing to look for one or more system functions.

[0060] In Example, A2, the subject matter of Example A1 can optionally include where the process is determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables.

[0061] In Example A3, the subject matter of any one of Examples A1-A2 can optionally include where the process is determined to be parsing to look for one or more system functions if the process calls GetProcAddress.

[0062] In Example A4, the subject matter of any one of Examples A1-A3 can optionally include where the process includes shellcode.

[0063] In Example A5, the subject matter of any one of Examples A1-A4 can optionally include where the system process monitoring module is further configured to analyze the process for malware.

[0064] In Example A6, the subject matter of any one of Examples A1-A5 can optionally include where the system process monitoring module is further configured to remove the flag if the process is found in a whitelist.

[0065] Example M1 is a method including monitoring a process, determining if the process is parsing to look for one or more system functions, and flagging the process if the process is parsing to look for one or more system functions.

[0066] In Example M2, the subject matter of Example M1 can optionally include where the process is determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables.

[0067] In Example M3, the subject matter of any one of the Examples M1-M2 can optionally include where the process is determined to be parsing to look for one or more system functions if the process calls GetProcAddress.

[0068] In Example M4, the subject matter of any one of the Examples M1-M3 can optionally include where the process includes shellcode.

[0069] In Example M5, the subject matter of any one of the Examples M1-M4 can optionally include analyzing the process for malware.

[0070] Example S1 is a system for detecting malware, the system can include a system process monitoring module. The system process monitoring module can be configured for monitoring a process, determining if the process is parsing to look for one or more system functions, and flagging the process if the process is parsing to look for one or more system functions.

[0071] In Example S2, the subject matter of Example S1 can optionally include where the process is determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables.

[0072] In Example S2, the subject matter of any one of Examples S1 and S2 can include where the process is determined to be parsing to look for one or more system functions if the process calls GetProcAddress.

[0073] 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-A6, or M1-M5. Example Y1 is an apparatus comprising means for performing of any of the Example methods M1-M5. 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 machine readable medium comprising one or more instructions that when executed by at least one processor, cause the at least one processor to:
 - monitor a process;
 - determine if the process is parsing to look for one or more system functions; and
 - flag the process if the process is parsing to look for one or more system functions.
2. The at least one machine -readable medium of Claim 1, wherein the process is determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables.
3. The at least one machine -readable medium of any of Claims 1 and 2, wherein the process is determined to be parsing to look for one or more system functions if the process calls GetProcAddress.
4. The at least one machine-readable medium of any of Claims 1-3, wherein the process includes shellcode.
5. The at least one machine-readable medium of any of Claims 1-4, further comprising one or more instructions that when executed by the at least one processor, further cause the at least one machine readable medium to:
 - analyze the process for malware.
6. The at least one machine-readable medium of any of Claims 1-5, further comprising one or more instructions that when executed by the at least one processor, further cause the at least one machine readable medium to:
 - remove the flag if the process is found in a whitelist.

7. An apparatus comprising:

a system process monitoring module, wherein the system process monitoring module is configured to:

monitor a process;

determine if the process is parsing to look for one or more system functions;

and

flag the process if the process is parsing to look for one or more system functions.

8. The apparatus of Claim 7, wherein the process is determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables.

9. The apparatus of any of Claims 7 and 8, wherein the process is determined to be parsing to look for one or more system functions if the process calls GetProcAddress.

10. The apparatus of any of Claims 7-9, wherein the process includes shellcode.

11. The apparatus of any of Claims 7-10, wherein the system process monitoring module is further configured to:

analyze the process for malware.

12. The apparatus of any of Claims 7-11, wherein the system process monitoring module is further configured to:

remove the flag if the process is found in a whitelist.

13. A method comprising:

monitoring a process;

determining if the process is parsing to look for one or more system functions; and

flagging the process if the process is parsing to look for one or more system functions.

14. The method of Claim 13, wherein the process is determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables.

15. The method of any of Claims 13 and 14, wherein the process is determined to be parsing to look for one or more system functions if the process calls GetProcAddress.

16. The method of any of Claims 13-15, wherein the process includes shellcode.

17. The method of any of Claims 13-16, further comprising:
analyzing the process for malware.

18. A system for detecting malware, the system comprising:
a system process monitoring module, wherein the system process monitoring module is configured for:

monitoring a process;
determining if the process is parsing to look for one or more system functions;
and
flagging the process if the process is parsing to look for one or more system functions.

19. The system of Claim 18, wherein the process is determined to be parsing to look for one or more system functions if the process parses portable executable headers to find and interpret dynamic link library tables.

20. The system of any of Claims 18 and 19, wherein the process is determined to be parsing to look for one or more system functions if the process calls GetProcAddress.

1/6

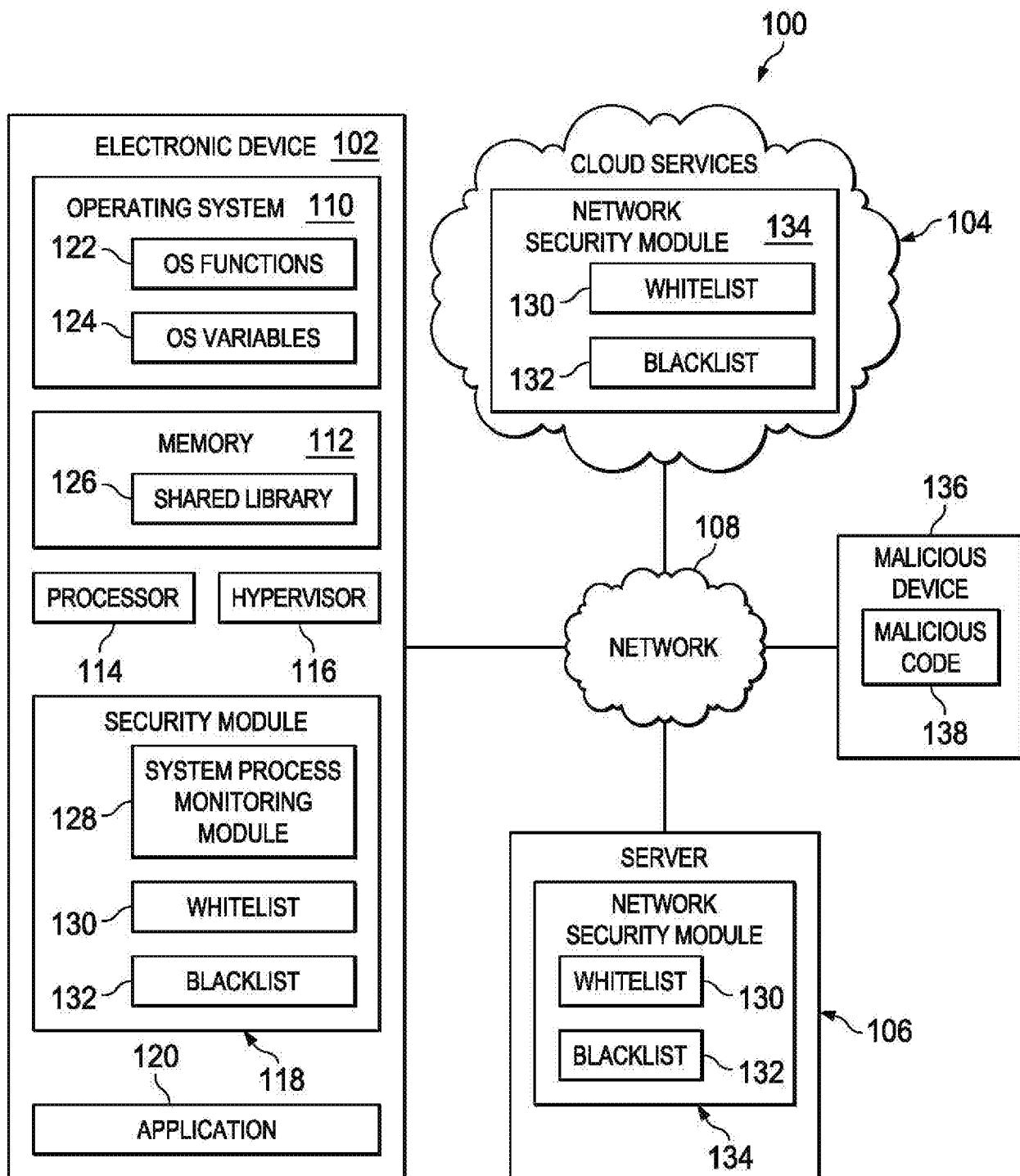


FIG. 1

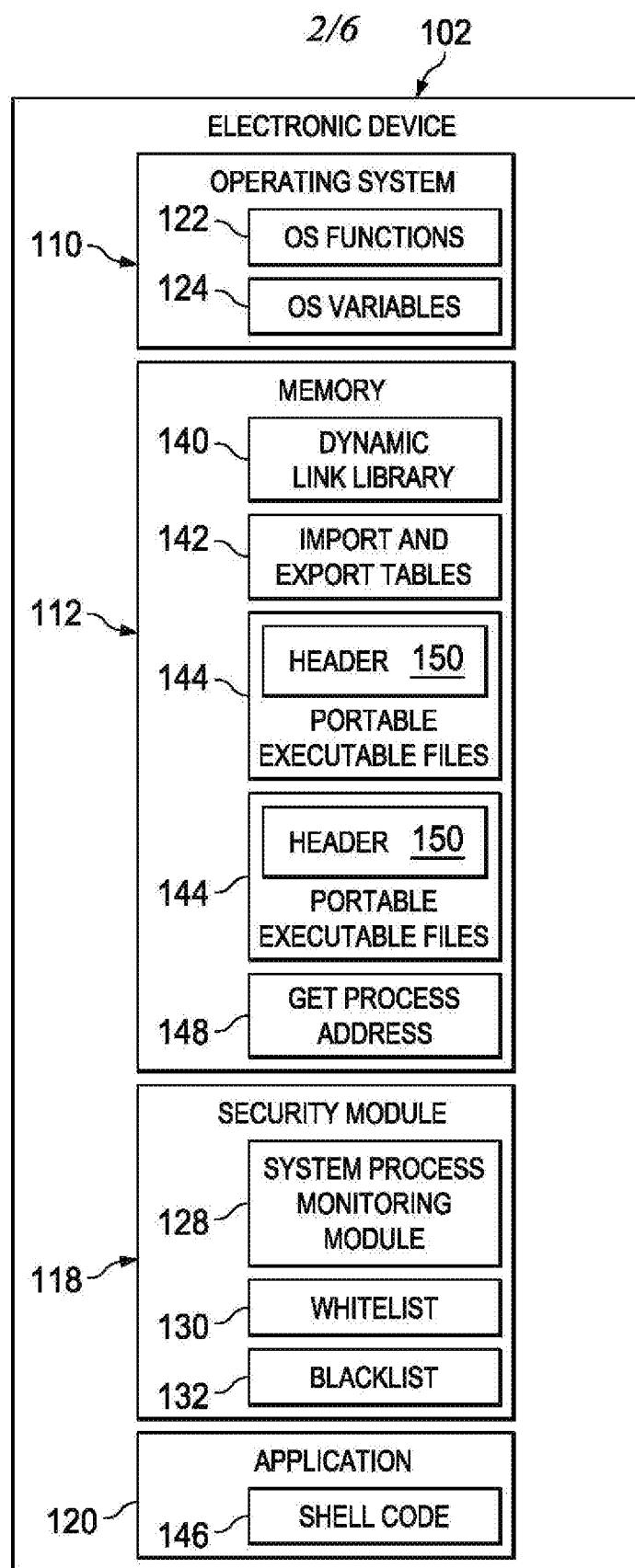


FIG. 2

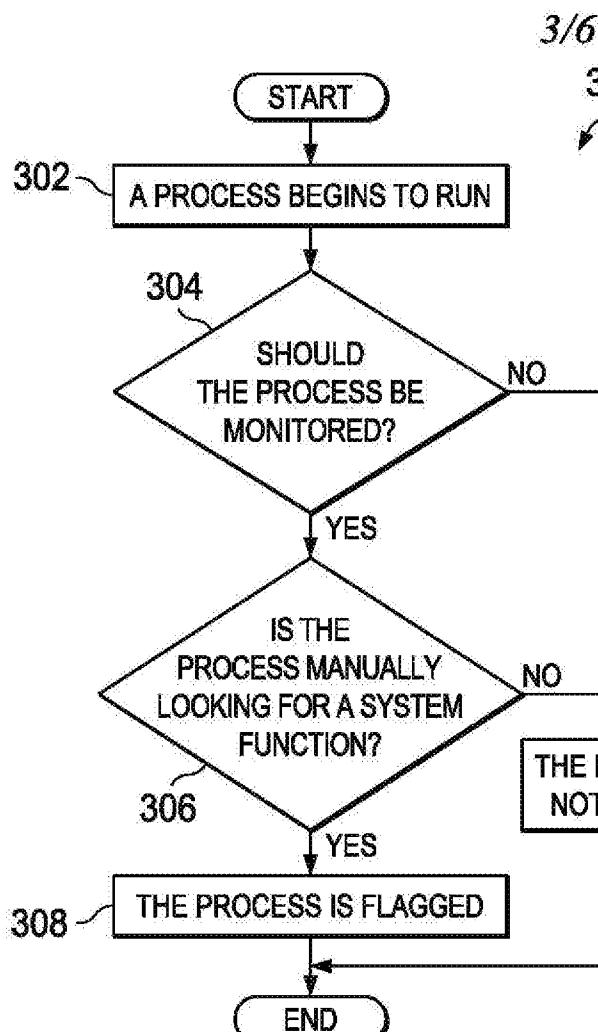


FIG. 3

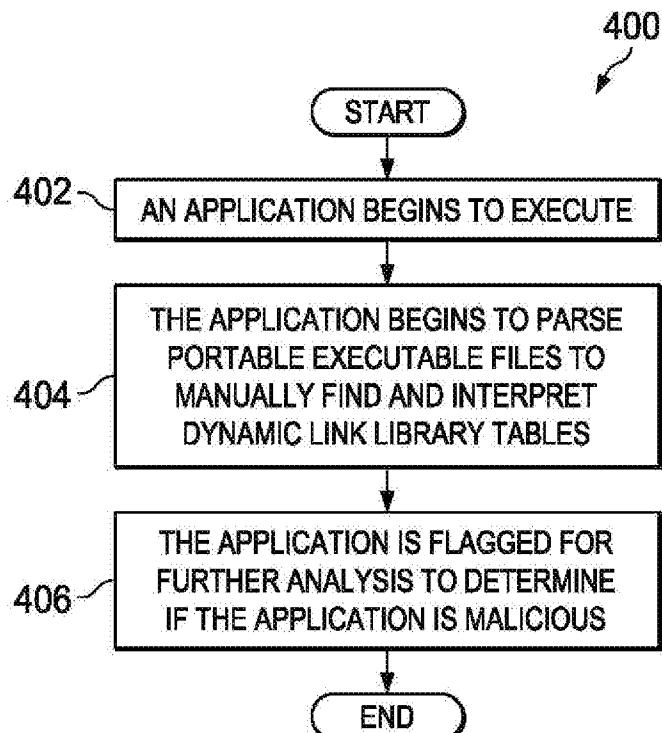
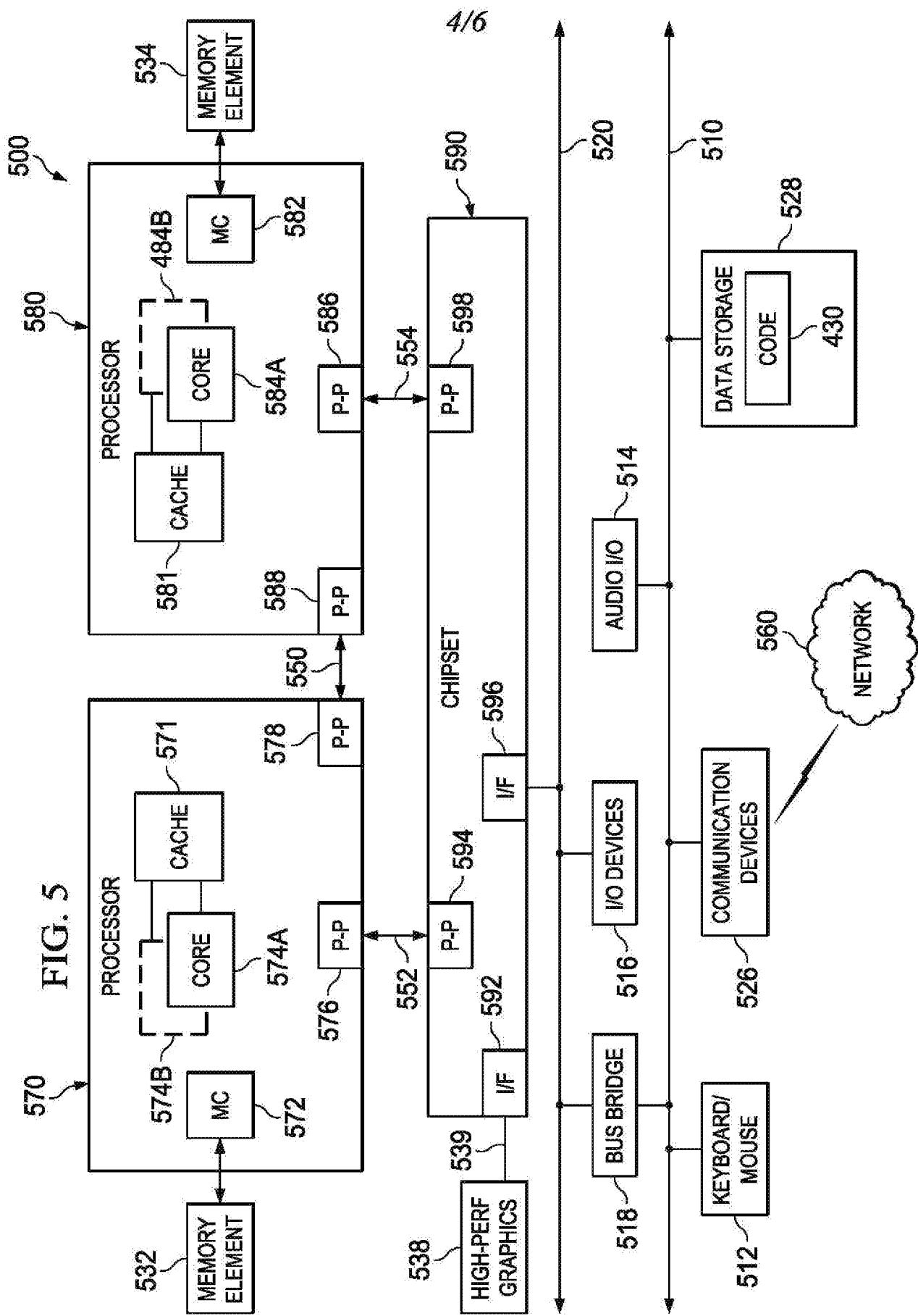
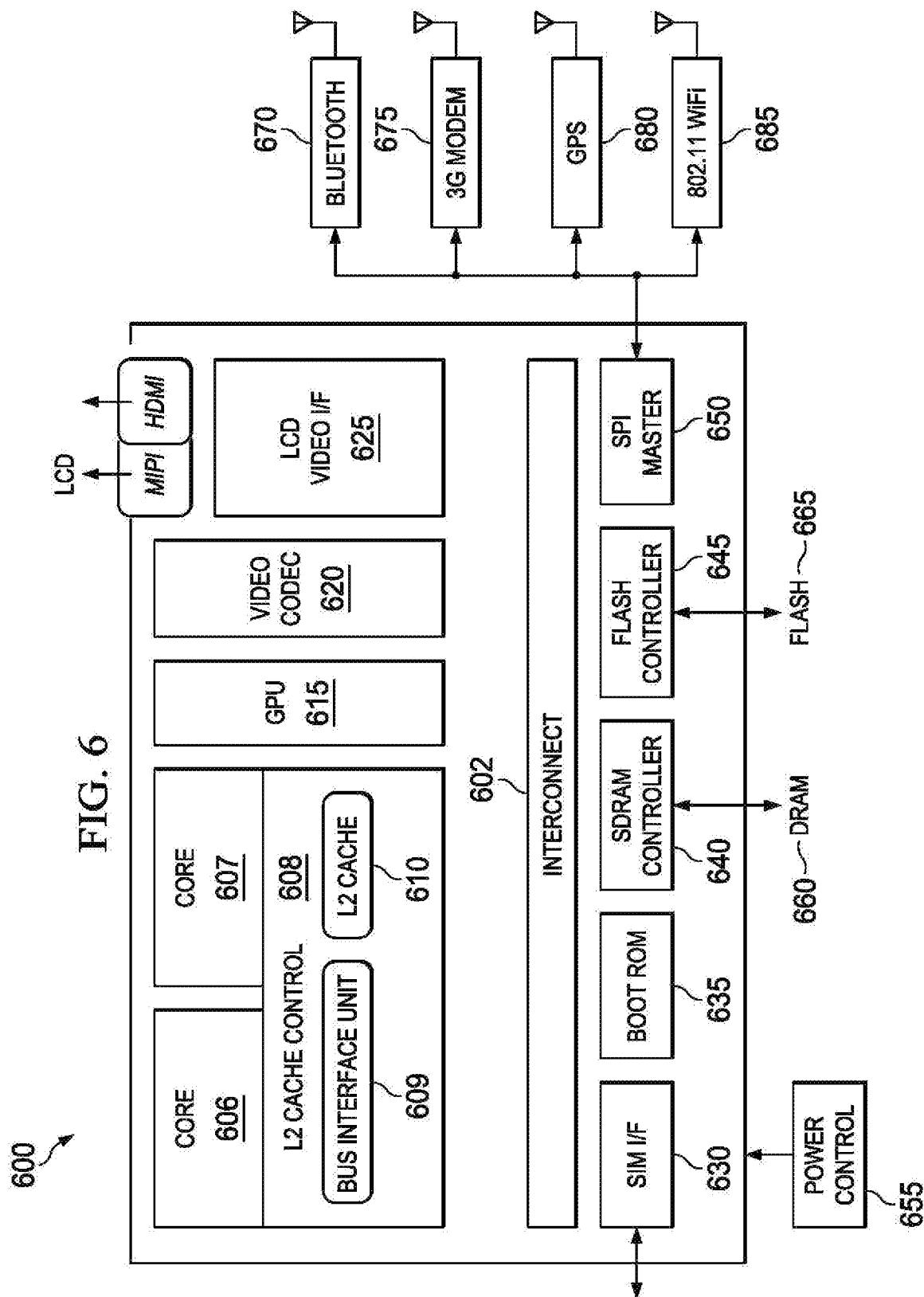


FIG. 4

FIG. 5



5/6



6/6

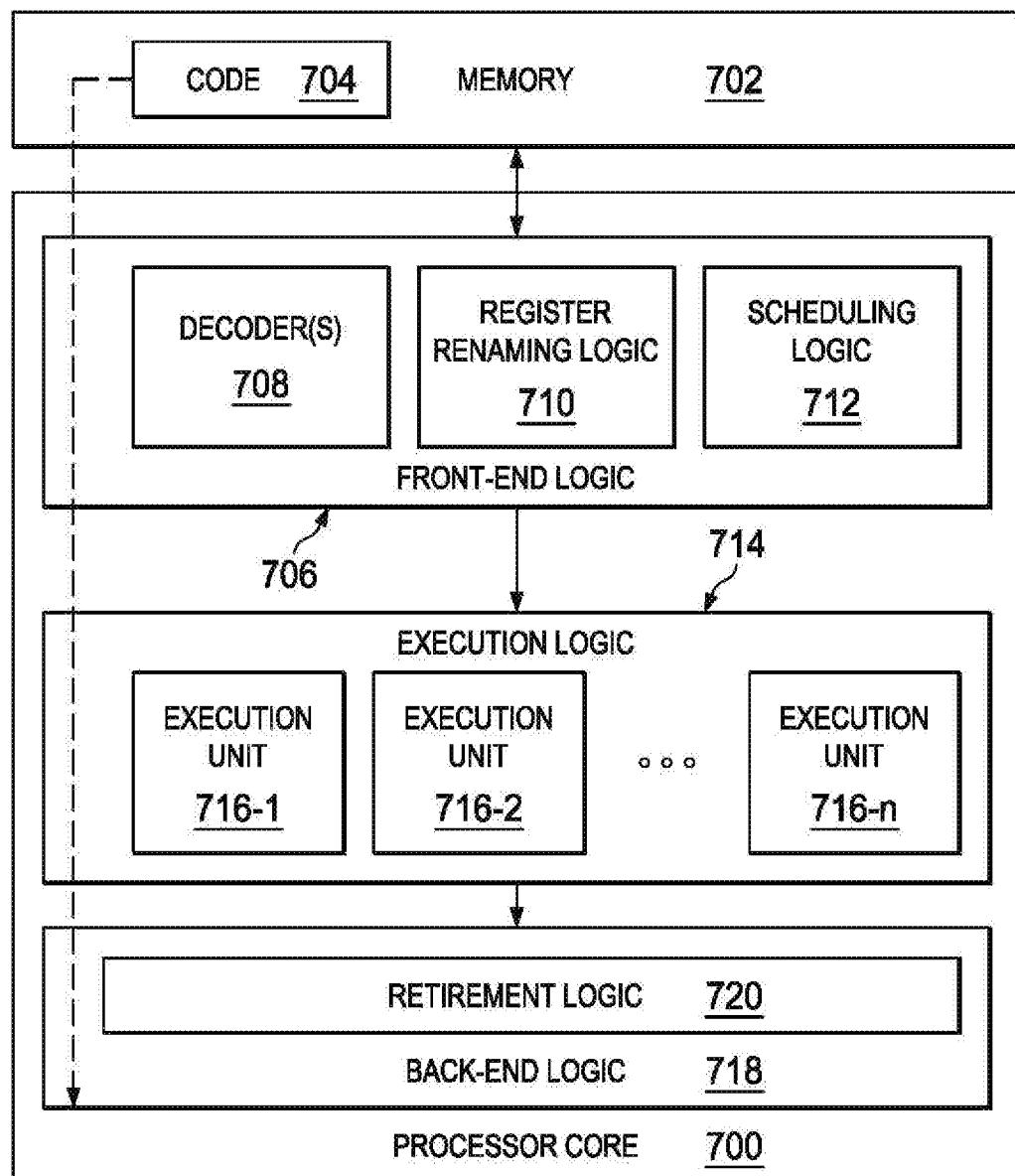


FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/033977**A. CLASSIFICATION OF SUBJECT MATTER****G06F 21/56(2013.01)I, G06F 21/57(2013.01)I, H04L 29/06(2006.01)I**

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

B. FIELDS SEARCHEDMinimum documentation searched (classification system followed by classification symbols)
G06F 21/56; G06F 21/20; H04L 29/06; G06F 21/00; G06F 21/57Documentation 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 modelsElectronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & keywords: malware detection, system function, process monitoring, GetProcAddress, shellcode, dynamic link library (DLL)**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 8307432 B1 (HSIANG-AN FENG) 06 November 2012 See column 7, lines 43-45, column 9, lines 65-67; claims 1-2; and figures 1, 5-6.	1-20
A	US 8931074 B2 (DELL PRODUCTS L.P.) 06 January 2015 See column 4, line 12 - column 6, line 8; claims 1, 9; and figures 2-3.	1-20
A	US 2014-0150101 A1 (XECURE LAB CO., LTD.) 29 May 2014 See paragraphs [0015]-[0027]; and claims 1-3.	1-20
A	US 8407787 B1 (VIC LOU et al.) 26 March 2013 See column 2, line 62 - column 5, line 27; and claims 1-6.	1-20
A	US 2012-0260304 A1 (MELVYN MORRIS et al.) 11 October 2012 See abstract; and claims 1-7.	1-20

 Further documents are listed in the continuation of Box C. See patent family annex.

* Special categories of cited documents:	"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
"A" document defining the general state of the art which is not considered to be of particular relevance	"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
"E" earlier application or patent but published on or after the international filing date	"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
"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)	"&" document member of the same patent family
"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	

Date of the actual completion of the international search 01 September 2016 (01.09.2016)	Date of mailing of the international search report 02 September 2016 (02.09.2016)
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 CHIN, Sang Bum Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/033977

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 8307432 B1	06/11/2012	None	
US 8931074 B2	06/01/2015	US 2014-0101748 A1	10/04/2014
US 2014-0150101 A1	29/05/2014	None	
US 8407787 B1	26/03/2013	None	
US 2012-0260304 A1	11/10/2012	AU 2012-217181 A1 CA 2827385 A1 CN 103493061 A EP 2676219 A1 EP 2750070 A2 EP 2750070 A3 EP 2750071 A2 EP 2750071 A3 HK 1199521 A1 HK 1199522 A1 JP 2014-509007 A MX 2013009373 A US 2012-0260340 A1 US 2012-0266208 A1 US 2014-0304800 A1 WO 2012-110501 A1	05/09/2013 23/08/2012 01/01/2014 25/12/2013 02/07/2014 27/08/2014 02/07/2014 27/08/2014 03/07/2015 03/07/2015 10/04/2014 14/04/2014 11/10/2012 18/10/2012 09/10/2014 23/08/2012