



(51) International Patent Classification:
G06F 21/53 (2013.01)

(21) International Application Number:
PCT/US2017/019636

(22) International Filing Date:
27 February 2017 (27.02.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/083,983 29 March 2016 (29.03.2016) US

(71) Applicant: INTEL CORPORATION [US/US]; 2200
Mission College Boulevard, Santa Clara, California 95054
(US).

(72) Inventor: SHANAHAN, Mark W.; 2841 Anfield Road,
Raleigh, North Carolina 27606 (US).

(74) Agent: KELLETT, Glen M.; Barnes & Thornburg LLP,
11 South Meridian Street, Indianapolis, IN 46204 (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, DJ, 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, KH, KN,
KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA,
MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG,
NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS,
RU, RW, SA, SC, SD, SE, SG, SK, SL, 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))

Published:

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

(54) Title: TECHNOLOGIES FOR MUTUAL APPLICATION ISOLATION WITH PROCESSOR-ENFORCED SECURE ENCLAVES

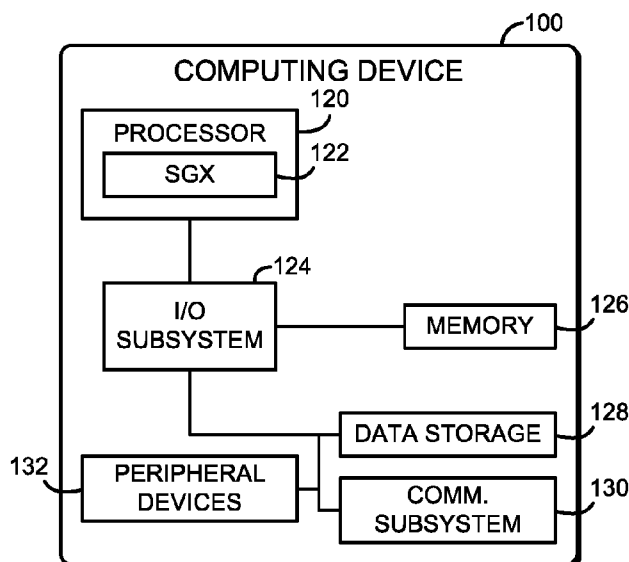


FIG. 1

(57) Abstract: Technologies for mutual application isolation include a computing device having a processor with secure enclave support. The computing device loads an application image to a memory range within a predefined virtual address range and creates a secure enclave with the predefined virtual address range assigned to the secure enclave. The computing device validates control flow integrity of the secure enclave. To validate control flow integrity the computing device may validate that the memory pages of the secure enclave synchronously exit only to an allowed address. Additionally, to validate control flow integrity the computing device may validate an asynchronous exit point associated with an enclave entry instruction. After validating the control flow integrity, the computing device executes the secure enclave, which includes enforcing mutual isolation of the application image and the secure enclave using the secure enclave support of the processor. Other embodiments are described and claimed.

TECHNOLOGIES FOR MUTUAL APPLICATION ISOLATION WITH PROCESSOR-ENFORCED SECURE ENCLAVES

CROSS-REFERENCE TO RELATED U.S. PATENT APPLICATION

[0001] The present application claims priority to U.S. Utility Patent Application Serial No. 15/083,983, entitled “TECHNOLOGIES FOR MUTUAL APPLICATION ISOLATION WITH PROCESSOR-ENFORCED SECURE ENCLAVES,” which was filed on March 29, 2016.

BACKGROUND

[0002] Current processors may provide support for a trusted execution environment such as a secure enclave. Secure enclaves include segments of memory (including code and/or data) protected by the processor from unauthorized access including unauthorized reads and writes. In particular, certain processors may include Intel® Software Guard Extensions (SGX) to provide secure enclave support. SGX provides confidentiality, integrity, and replay-protection to the secure enclave data while the data is resident in the platform memory and thus provides protection against both software and hardware attacks. The on-chip boundary forms a natural security boundary, where data and code may be stored in plaintext and assumed to be secure. The contents of an SGX secure enclave may be authenticated and therefore trusted by the independent software vendor (ISV) that provides the secure enclave. However, an SGX enclave may be capable of reading and/or writing to regular process memory and thus may not be trusted by the hosting application process or operating system.

[0003] Certain operating systems may support sandboxing for native object code. For example, Google® Native Client (NaCl) provides a mechanism for executing native object code in an isolated sandbox provided by a web browser. To ensure isolation, NaCl imposes strict limitations on the operations allowed to be performed by the native object code. In addition, NaCl does not permit code to execute within an SGX secure enclave.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The concepts described herein are illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. Where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements.

[0005] FIG. 1 is a simplified block diagram of at least one embodiment of a computing device for mutual application isolation;

[0006] FIG. 2 is a simplified block diagram of at least one embodiment of an environment that may be established by the computing device of FIG. 1;

[0007] FIG. 3 is a schematic diagram illustrating a virtual address space that may be established by the computing device of FIGS. 1-2; and

[0008] FIGS. 4A and 4B are a simplified flow diagram of at least one embodiment of a method for mutual application isolation that may be executed by the computing device of FIGS. 1-2.

DETAILED DESCRIPTION OF THE DRAWINGS

[0009] While the concepts of the present disclosure are susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and will be described herein in detail. It should be understood, however, that there is no intent to limit the concepts of the present disclosure to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives consistent with the present disclosure and the appended claims.

[0010] References in the specification to “one embodiment,” “an embodiment,” “an illustrative embodiment,” etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may or may not necessarily include that particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described. Additionally, it should be appreciated that items included in a list in the form of “at least one of A, B, and C” can mean (A); (B); (C); (A and B); (A and C); (B and C); or (A, B, and C). Similarly, items listed in the form of “at least one of A, B, or C” can mean (A); (B); (C); (A and B); (A and C); (B and C); or (A, B, and C).

[0011] The disclosed embodiments may be implemented, in some cases, in hardware, firmware, software, or any combination thereof. The disclosed embodiments may also be implemented as instructions carried by or stored on one or more transitory or non-transitory machine-readable (e.g., computer-readable) storage media, which may be read and executed by one or more processors. A machine-readable storage medium may be embodied as any storage

device, mechanism, or other physical structure for storing or transmitting information in a form readable by a machine (e.g., a volatile or non-volatile memory, a media disc, or other media device).

[0012] In the drawings, some structural or method features may be shown in specific arrangements and/or orderings. However, it should be appreciated that such specific arrangements and/or orderings may not be required. Rather, in some embodiments, such features may be arranged in a different manner and/or order than shown in the illustrative figures. Additionally, the inclusion of a structural or method feature in a particular figure is not meant to imply that such feature is required in all embodiments and, in some embodiments, may not be included or may be combined with other features.

[0013] Referring now to FIG. 1, in an illustrative embodiment, a computing device 100 for mutual application isolation includes a processor 120 that supports secure enclaves (e.g., a processor 120 that supports Intel® SGX technology). In use, as described further below, the computing device 100 loads one or more application images into a predefined range of virtual memory known as a secure enclave memory range. The application images may be trusted or untrusted by the operating system of the computing device 100. The computing device 100 creates a secure enclave that is assigned the predefined secure enclave memory range. The computing device 100 validates memory pages of the secure enclave and the instructions used to enter the secure enclave to protect control flow integrity of the secure enclave. The secure enclave support of the processor 120 enforces mutual isolation between the application images and the secure enclave. Thus, the computing device 100 may securely execute enclave code provided by an independent software vendor (ISV), and the secure enclave code is isolated from application or operating system code executed by the computing device 100. Additionally, the computing device 100 protects the application and/or operating system code that hosts the secure enclave from attack by the secure enclave code. As described further below, the computing device 100 may provide mutual isolation with relatively simple validation of the secure enclave code and without imposing overly restrictive limitations on the contents of the secure enclave code. Accordingly, the computing device 100 may allow an ISV to take advantage of the security features of secure enclaves while still executing the enclave code within a sandboxed environment that is isolated from the host application and/or operating system.

[0014] The computing device 100 may be embodied as any type of computation or computer device capable of performing the functions described herein, including, without limitation, a computer, a desktop computer, a workstation, a server, a laptop computer, a

notebook computer, a tablet computer, a mobile computing device, a wearable computing device, a network appliance, a web appliance, a distributed computing system, a processor-based system, and/or a consumer electronic device. As shown in FIG. 1, the computing device 100 illustratively includes the processor 120, an input/output subsystem 124, a memory 126, a data storage device 128, and communication circuitry 130. Of course, the computing device 100 may include other or additional components, such as those commonly found in a desktop computer (e.g., various input/output devices), in other embodiments. Additionally, in some embodiments, one or more of the illustrative components may be incorporated in, or otherwise form a portion of, another component. For example, the memory 126, or portions thereof, may be incorporated in the processor 120 in some embodiments.

[0015] The processor 120 may be embodied as any type of processor capable of performing the functions described herein. The processor 120 may be embodied as a single or multi-core processor(s), digital signal processor, microcontroller, or other processor or processing/controlling circuit. As shown, the processor 120 includes secure enclave support 122, which allows the processor 120 to establish a trusted execution environment known as a secure enclave, in which executing code may be measured, verified, and/or otherwise determined to be authentic. Additionally, code and data included in the secure enclave may be encrypted or otherwise protected from being accessed by code executing outside of the secure enclave. For example, code and data included in the secure enclave may be protected by hardware protection mechanisms of the processor 120 while being executed or while being stored in certain protected cache memory of the processor 120. The code and data included in the secure enclave may be encrypted when stored in a shared cache or the main memory 126. The secure enclave support 122 may be embodied as a set of processor instruction extensions that allows the processor 120 to establish one or more secure enclaves in the memory 126. For example, the secure enclave support 122 may be embodied as Intel® Software Guard Extensions (SGX) technology.

[0016] The memory 126 may be embodied as any type of volatile or non-volatile memory or data storage capable of performing the functions described herein. In operation, the memory 126 may store various data and software used during operation of the computing device 100 such as operating systems, applications, programs, libraries, and drivers. Additionally, part of the memory 126 may be used as an enclave page cache (EPC) to store encrypted code and/or data for the secure enclaves established by the processor 120. The processor 120 may encrypt memory pages of each secure enclave before transferring the encrypted pages out of the processor 120 to the EPC within the memory 126.

[0017] The memory 126 is communicatively coupled to the processor 120 via the I/O subsystem 124, which may be embodied as circuitry and/or components to facilitate input/output operations with the processor 120, the memory 126, and other components of the computing device 100. For example, the I/O subsystem 124 may be embodied as, or otherwise include, memory controller hubs, input/output control hubs, platform controller hubs, integrated control circuitry, firmware devices, communication links (i.e., point-to-point links, bus links, wires, cables, light guides, printed circuit board traces, etc.) and/or other components and subsystems to facilitate the input/output operations. In some embodiments, the I/O subsystem 124 may form a portion of a system-on-a-chip (SoC) and be incorporated, along with the processor 120, the memory 126, and other components of the computing device 100, on a single integrated circuit chip.

[0018] The data storage device 128 may be embodied as any type of device or devices configured for short-term or long-term storage of data such as, for example, memory devices and circuits, memory cards, hard disk drives, solid-state drives, or other data storage devices. In some embodiments, the data storage device 128 may be used to store one or more executable binary images. Each executable binary image may include code and/or data that is to be initially loaded into a secure enclave and then measured or otherwise verified before the secure enclave is executed.

[0019] The communication circuitry 130 of the computing device 100 may be embodied as any communication circuit, device, or collection thereof, capable of enabling communications between the computing device 100 and other remote devices over a network. The communication circuitry 130 may be configured to use any one or more communication technology (e.g., wired or wireless communications) and associated protocols (e.g., Ethernet, Bluetooth®, Wi-Fi®, WiMAX, etc.) to effect such communication.

[0020] As shown, the computing device 100 may also include one or more peripheral devices 132. The peripheral devices 132 may include any number of additional input/output devices, interface devices, and/or other peripheral devices. For example, in some embodiments, the peripheral devices 132 may include a display, touch screen, graphics circuitry, keyboard, mouse, speaker system, microphone, network interface, and/or other input/output devices, interface devices, and/or peripheral devices.

[0021] Referring now to FIG. 2, in an illustrative embodiment, the computing device 100 establishes an environment 200 during operation. The illustrative environment 200 includes an operating system loader module 202, an enclave loader module 204, and a mutual isolation module 206. The various modules of the environment 200 may be embodied as

hardware, firmware, software, or a combination thereof. As such, in some embodiments, one or more of the modules of the environment 200 may be embodied as circuitry or collection of electrical devices (e.g., operating system loader circuitry 202, enclave loader circuitry 204, and/or mutual isolation circuitry 206). It should be appreciated that, in such embodiments, one or more of the operating system loader circuitry 202, the enclave loader circuitry 204, and/or the mutual isolation circuitry 206 may form a portion of one or more of the processor 120, the I/O subsystem 124, and/or other components of the computing device 100. Additionally, in some embodiments, one or more of the illustrative modules may form a portion of another module and/or one or more of the illustrative modules may be independent of one another.

[0022] The operating system loader module 202 is configured to load an application image to a memory range within a predefined virtual address range. The operating system loader module 202 may be further configured to verify that the application image is an operating-system trusted application image and execute the application image in response to verification that the application image is an operating-system-trusted application image. The operating system loader module 202 may be configured to load an untrusted application image that is not trusted by an operating system of the computing device 100. The operating system loader module 202 may be further configured to execute the untrusted application image within a sandbox established by the operating system.

[0023] The enclave loader module 204 is configured to create a secure enclave with the predefined virtual address range assigned to the secure enclave. The enclave loader module 204 is further configured to validate control flow integrity of the secure enclave in response to creating the secure enclave. Validating the control flow integrity of the secure enclave may include validating secure enclave memory pages to ensure that the secure enclave exits to an allowed address and/or verifying that each executable secure enclave memory page is not writable. The enclave loader module 204 may be further configured to add the secure enclave memory pages to an enclave page cache in response to validating the secure enclave memory pages and initialize the secure enclave in response to adding the secure enclave memory pages. The enclave page cache is included within the predefined virtual address range. Validating the control flow integrity of the secure enclave may also include validating an asynchronous exit point associated with an enclave entry instruction. The enclave loader module 204 may be further configured to enter the secure enclave in response to validating the control flow integrity of the secure enclave.

[0024] The mutual isolation module 206 is configured to execute the secure enclave in response to validating the control flow integrity of the secure enclave. Executing the secure

enclave includes enforcing mutual isolation of the secure enclave and the application image using the secure enclave support 122 of the processor 120. Enforcing mutual isolation may include generating abort transaction semantics by the processor 120 in response to a memory access from the application image into the enclave secure enclave or generating a fault by the processor 120 in response to a memory access from the secure enclave into the application image. The mutual isolation module 206 may be further configured to share data stored in an untrusted memory buffer between the application image and the secure enclave. The untrusted memory buffer is located outside of the predefined virtual address range.

[0025] Referring now to FIG. 3, schematic diagram 300 illustrates a virtual memory address space 302 that may be established by the computing device 100. The virtual memory address space 302 includes a contiguous range of linear memory addresses (also called virtual addresses) that may be addressed by one or more programs executed the computing device 100. The computing device 100 translates between virtual addresses in the virtual memory address space 302 to physical memory pages in the memory 126, using one or more page tables managed by the processor 120 and operating system software. As shown, the virtual memory address space 302 is a 32-bit range capable of addressing 4 gigabytes of memory 126. However, it should be understood that in some embodiments the virtual memory address space 302 may have a different size (e.g., a 64-bit range).

[0026] The virtual memory address space 302 includes a secure enclave address range 304, which is illustratively located in the lower half of the virtual memory address space 302. The secure enclave address range 304 may include code and/or data used by programs executed by the computing device 100, including an enclave image 306 and an application-trusted image 308. The enclave image 306 corresponds to code and/or data memory pages included in a secure enclave. The contents of the enclave image 306 are added to an enclave page cache, which is encrypted and otherwise protected from unauthorized access by the secure enclave support 122 of the processor 120. The application-trusted image 308 corresponds to code and/or data that has been verified or is otherwise trusted by application software (or operating-system level software) executed by the computing device 100. As described further below, the enclave image 306 and the application-trusted image 308 are mutually isolated by the processor 120, meaning that code executing from the enclave image 306 cannot access the application-trusted image 308 and vice versa. In some embodiments, the enclave image 306 and the application-trusted image 308 may communicate through an untrusted data marshaling buffer 310 that is positioned in the virtual memory address space 302 outside of the secure enclave address range 304. Additionally or alternatively, in some embodiments the secure enclave

address range 304 may also include an application-untrusted image 312. The application-untrusted image 312 may be code and/or data that is not trusted by the application software or the operating-system level software, such as a downloaded applet, plug-in, library, or other untrusted code. The enclave image 306 and the application-untrusted image 312 are also mutually isolated by the processor 120.

[0027] Referring now to FIGS. 4A and 4B, in use, the computing device 100 may execute a method 400 for mutual application isolation. The method 400 begins in block 402, in which the computing device 100 verifies an application-trusted image 308 by an operating system of the computing device 100. The application-trusted image 308 may be embodied as any executable application, library, or other native object code module that may be executed by the computing device 100. For example, the application-trusted image 308 may be embodied as an interactive application such as a web browser or an applet, plug-in, library, or other code module that may be embedded by an interactive application. The computing device 100 may use any technique to authenticate or otherwise verify the contents of the application-trusted image 308. For example, an operating system loader or virtual machine monitor (VMM) of the computing device 100 may measure and verify the application-trusted image 308. After being verified, the application-trusted image 308 is deemed to be trusted by the operating system of the computing device 100.

[0028] In block 404, the computing device 100 loads the application-trusted image 308 within the predefined secure enclave address range 304. For example, a dynamic loader of the computing device 100 may load the application-trusted image 308 into a range of the virtual memory address space 302 that is included within the secure enclave address range 304. In the illustrative embodiment, as shown in FIG. 3, the computing device 100 may load the application-trusted image 308 into the lower half of the virtual memory address space 302. Although located within the secure enclave address range 304, the application-trusted image 308 is not loaded into an enclave page cache (EPC) and thus is not encrypted, integrity-protected, or confidentiality-protected by the secure enclave support 122 of the processor 120.

[0029] In block 406, the computing device 100 executes the application-trusted image 308. For example, the computing device 100 may execute interactive application code included in the application-trusted image 308. During execution, the application-trusted image 308 (or an associated host application) may attempt to load a secure enclave. For example, the computing device 100 may download or otherwise install a new secure enclave and attempt to load and execute that secure enclave. In block 408, the computing device 100 determines whether to load a secure enclave. If not, the method 400 branches ahead to block 430, shown in

FIG. 4B. If the computing device 100 determines to load a secure enclave, the method 400 advances to block 410.

[0030] In block 410, the computing device 100 creates a secure enclave and assigns the predefined secure enclave address range 304 to the newly created secure enclave. The computing device 100 creates the secure enclave using the secure enclave support 122 of the processor 120. For example, certain processors 120 supporting Intel SGX technology may support one or more processor instructions with multiple associated leaf functions. In particular, the processor 120 may support an ENCLS system-level instruction and an ENCLU user-level instruction. In those embodiments, to create the secure enclave the computing device 100 executes the ENCLS[ECREATE] leaf function. To improve clarity, in this disclosure SGX leaf functions may be referred to as instructions using the mnemonic of the corresponding leaf function (e.g., ENCLS[ECREATE] may be referred to as the ECREATE instruction). The computing device 100 may set the ELRANGE of the newly created secure enclave to correspond to the predefined secure enclave address range 304 (for example, by specifying a base address and size of the secure enclave when creating the secure enclave). The secure enclave is created in an uninitialized state, and when first created the enclave page cache (EPC) does not include any pages for the secure enclave.

[0031] In block 412, the computing device 100 validates that secure enclave memory pages to be added to the secure enclave do not synchronously exit to arbitrary memory addresses. Validating that the memory pages do not synchronously exit to arbitrary memory addresses is part of protecting the control flow integrity of the secure enclave. The secure enclave support 122 of the processor 120 may support well-defined control flow mechanisms. For example, a secure enclave may only be exited by a synchronous enclave exit instruction EEXIT (ENCLU[EEXIT]) or by an asynchronous exit (described further below in connection with block 420). To validate that the memory pages do not include a synchronous exit to arbitrary memory addresses, the computing device 100 may statically analyze the contents of the enclave memory pages, for example by disassembling object code included in the memory pages. In some embodiments, in block 414 the computing device 100 may validate that the exit address of any enclave exit instruction in the enclave memory page is allowed. For example, the computing device 100 may validate that the exit address of each EEXIT instruction points to a predefined address within the virtual memory address space 302 or to a predefined offset relative to the base of the secure enclave image 306. An exit trampoline or other trusted code may be located at those allowed addresses. In some embodiments, in block 416, the computing device 100 may ensure that executable secure enclave memory pages are not writable.

Preventing executable and writable pages may prevent self-modifying enclave code from generating EEXIT instructions that exit to arbitrary addresses.

[0032] In block 418, the computing device 100 adds the validated pages to the enclave page cache (EPC) and then initializes the secure enclave. For example, the computing device 100 may execute the EADD instruction (ENCLS[EADD]) or the EAUG instruction (ENCLS[EAUG]) to add each page to the EPC. The computing device 100 may add the validated pages at a predefined offset within the secure enclave address range 304. Adding the page to the EPC may also update a measurement of the secure enclave. The enclave may be initialized by executing the EINIT instruction (ENCLS[EINIT]), which finalizes the measurement of the secure enclave. The measurement of the secure enclave may be verified, for example by checking against a signature provided by the ISV that produced the secure enclave. Thus, after being loaded and initialized, the secure enclave is deemed to be trusted by the ISV (or other party trusted by the ISV).

[0033] In block 420, the computing device 100 enters the secure enclave and validates an asynchronous exit point (AEP) for the secure enclave. The computing device 100 may execute the EENTER instruction (ENCLU[EENTER]) to enter the secure enclave. The computing device 100 supplies an address for the AEP with the EENTER instruction. During execution of the secure enclave, the processor 120 may generate an asynchronous exit in response to an interrupt or exception. In response to the asynchronous exit, the processor 120 stores thread state within the secure enclave (including the current instruction pointer) and then jumps to the address of the AEP. Because the thread state is stored within the secure enclave, code executing outside of the secure enclave (e.g., the application-trusted image 308) may not access that thread state. The computing device 100 may validate the AEP by, for example, determining that the AEP points to an address of application-trusted code such as an exception handler, interrupt handler, or other trusted asynchronous exit handler. In some embodiments, the computing device 100 may verify that the code pointed to by the AEP ends with an enclave resume instruction such as ERESUME (ENCLU[ERESUME]). In response to executing the ERESUME instruction, the processor 120 re-enters the secure enclave and uses the stored thread state to resume execution.

[0034] In block 422, the computing device 100 executes the secure enclave with mutual isolation from application code enforced by the secure enclave support 122 of the processor 120. The particular techniques for enforcing mutual isolation may depend on the particular memory accesses encountered at runtime. In some embodiments, in block 424 the computing device 100 may abort the transaction for a non-enclave memory access to a memory page

within the enclave page cache. For example, a memory access from the application-trusted image 308 and/or the application-untrusted image 312 to a page within the enclave image 306 may be aborted. To abort the transaction, the processor 120 may generate page abort semantics. The particular page abort semantics are undefined and may depend on the particular model of the processor 120. For example, in some embodiments the processor 120 may return an abort page with all bits set to the logical value “1.” Aborting the page prevents the application-trusted image 308 and/or the application-untrusted image 312 from accessing the contents of the enclave image 306.

[0035] In some embodiments, in block 426 the computing device 100 may generate a fault for an enclave access to a page that is within the secure enclave address range 304 but outside of the enclave page cache. For example, a memory access from the enclave image 306 to a page within the application-trusted image 308 and/or the application-untrusted image 312 may generate a page fault. The page fault may prevent the secure enclave image 306 from accessing the contents of the application-trusted image 308 or the application-untrusted image 312.

[0036] In block 428, the computing device 100 may share data between the secure enclave and application code executing outside of the secure enclave using an untrusted buffer outside of the secure enclave address range 304. For example, the computing device 100 may access shared data in the untrusted data marshaling buffer 310. The computing device 100 may use any technique for marshaling the data in the buffer 310. For example, in some embodiments, the secure enclave image 306 may establish a transitional stack in the buffer 310 to pass data to the application-trusted image 308. After loading and executing the secure enclave, the method 400 advances to block 430, shown in FIG. 4B.

[0037] In block 430, the computing device 100 determines whether to load an application-untrusted image 312. During execution, the secure enclave image 306, the application-trusted image 308, and/or other code executed by the computing device 100 may attempt to load an application-untrusted image 312. For example, the computing device 100 may attempt to load an application executable, library, plug-in, applet, or other code module that is not trusted by the operating system of the computing device 100. If the computing device 100 determines not to load the application-untrusted image 312, the method 400 loops back to block 406, shown in FIG. 4A, to continue executing the application-trusted image 308 and/or the secure enclave image 306 with mutual isolation. If the computing device 100 determines to load the application-untrusted image 312, the method 400 advances to block 432.

[0038] In block 432, the computing device 100 validates that the application-untrusted image 312 does not include transitions to a secure enclave. Validating the application-untrusted image 312 may protect the control flow integrity of the secure enclave image 306 by preventing unauthorized entry into the secure enclave image 306. As described above, the secure enclave support 122 of the processor 120 may support well-defined control flow mechanisms. For example, a secure enclave may only be entered by an enclave enter instruction EENTER (ENCLU[EENTER]). Jumps directly to locations within the secure enclave image 306 may generate abort page semantics, as described above in connection with block 422. To validate that the application-untrusted image 312 does not include EENTER instructions, the computing device 100 may statically analyze the contents of the application-untrusted image 312, for example by disassembling the application-untrusted image 312.

[0039] In block 434, the computing device 100 loads the application-untrusted image 312 within the predefined secure enclave address range 304. For example, a dynamic loader of the computing device 100 may load the application-untrusted image 312 into a range of the virtual memory address space 302 that is included within the secure enclave address range 304. In the illustrative embodiment, as shown in FIG. 3, the computing device 100 may load the application-untrusted image 312 into the lower half of the virtual memory address space 302. Although located within the secure enclave address range 304, the application-untrusted image 312 is not loaded into an enclave page cache (EPC) and is thus not encrypted, integrity-protected, or confidentiality-protected by the secure enclave support 122 of the processor 120.

[0040] In block 436, the computing device 100 executes the application-untrusted image 312 using an operating-system-trusted sandbox. The sandbox isolates the application-untrusted image 312 from other application code such as the application-trusted image 308. The sandbox is not required to isolate the application-untrusted image 312 from the enclave image 306; as described above in connection with block 422 shown in FIG. 4A, the secure enclave support 122 of the processor 120 enforces isolation between the secure enclave image 306 and application code such as the application-untrusted image 312. After starting execution of the application-untrusted image 312, the method 400 loops back to block 406, shown in FIG. 4A, to continue executing the application-trusted image 308, the application-untrusted image 312, and/or the secure enclave image 306 with mutual isolation.

[0041] It should be appreciated that, in some embodiments, the method 400 may be embodied as various instructions stored on a computer-readable media, which may be executed by the processor 120, the I/O subsystem 124, and/or other components of the computing device 100 to cause the computing device 100 to perform the method 400. The computer-readable

media may be embodied as any type of media capable of being read by the computing device 100 including, but not limited to, the memory 126, the data storage device 128, firmware devices, and/or other media.

EXAMPLES

[0042] Illustrative examples of the technologies disclosed herein are provided below. An embodiment of the technologies may include any one or more, and any combination of, the examples described below.

[0043] Example 1 includes a computing device for mutual application isolation, the computing device comprising a processor with secure enclave support; an operating system loader module to load an application image to a memory range within a predefined virtual address range; an enclave loader module to (i) create a secure enclave, wherein the predefined virtual address range is assigned to the secure enclave and (ii) validate control flow integrity of the secure enclave in response to creation of the secure enclave; and a mutual isolation module to execute the secure enclave in response to validation of the control flow integrity of the secure enclave, wherein to execute the secure enclave comprises to enforce mutual isolation of the secure enclave and the application image with the secure enclave support of the processor.

[0044] Example 2 includes the subject matter of Example 1, and wherein to enforce mutual isolation of the secure enclave and the application image comprises to generate abort transaction semantics by the processor in response to a memory access from the application image into the enclave secure enclave.

[0045] Example 3 includes the subject matter of any of Examples 1 and 2, and wherein to enforce mutual isolation of the secure enclave and the application image comprises to generate a fault by the processor in response to a memory access from the secure enclave into the application image.

[0046] Example 4 includes the subject matter of any of Examples 1-3, and wherein to validate the control flow integrity of the secure enclave comprises to validate one or more secure enclave memory pages associated with the secure enclave to ensure that the secure enclave synchronously exits to an allowed address.

[0047] Example 5 includes the subject matter of any of Examples 1-4, and wherein to validate the one or more secure enclave memory pages comprises to statically analyze the one or more secure enclave memory pages.

[0048] Example 6 includes the subject matter of any of Examples 1-5, and wherein to validate the one or more secure enclave memory pages further comprises to validate that an exit address associated with an enclave exit instruction is an allowed address.

[0049] Example 7 includes the subject matter of any of Examples 1-6, and wherein to validate that the exit address associated with the enclave exit instruction is an allowed address comprises to determine whether the exit address is at a predefined address within the application image or to determine whether the exit address is at a predefined offset from a base address of the secure enclave.

[0050] Example 8 includes the subject matter of any of Examples 1-7, and wherein to validate the one or more secure enclave memory pages comprises to verify that each executable secure enclave memory page is not writable.

[0051] Example 9 includes the subject matter of any of Examples 1-8, and wherein the enclave loader module is further to add the one or more secure enclave memory pages to an enclave page cache in response to validation of the one or more secure enclave memory pages, wherein the enclave page cache is included within the predefined virtual address range; and initialize the secure enclave in response to addition of the one or more secure enclave memory pages to the enclave page cache, wherein to initialize the secure enclave comprises to measure and verify the secure enclave.

[0052] Example 10 includes the subject matter of any of Examples 1-9, and wherein to add the one or more secure enclave memory pages to the enclave page cache comprises to add the memory pages at a predefined offset within the predefined virtual address range.

[0053] Example 11 includes the subject matter of any of Examples 1-10, and wherein to validate the control flow integrity of the secure enclave comprises to validate an asynchronous exit point associated with an enclave entry instruction.

[0054] Example 12 includes the subject matter of any of Examples 1-11, and wherein to validate the asynchronous exit point associated with the enclave entry instruction comprises to determine whether the asynchronous exit point is associated with a code segment of the application image that ends with an enclave resume instruction.

[0055] Example 13 includes the subject matter of any of Examples 1-12, and wherein the operating system loader module is further to (i) verify that the application image is an operating-system trusted application image and (ii) execute the application image in response to verification that the application image is an operating-system-trusted application image; the enclave loader module is further to enter the secure enclave in response to the validation of the control flow integrity of the secure enclave; and to validate the asynchronous exit point

associated with the enclave entry instruction comprises to validate the asynchronous exit point associated with an enclave entry instruction of the application image in response to execution of the application image.

[0056] Example 14 includes the subject matter of any of Examples 1-13, and wherein the mutual isolation module is further to share data stored in an untrusted memory buffer between the application image and the secure enclave, wherein the untrusted memory buffer is located outside of the predefined virtual address range.

[0057] Example 15 includes the subject matter of any of Examples 1-14, and wherein to share the data stored in the untrusted memory buffer comprises to write the data by the secure enclave and to read the data by the application image.

[0058] Example 16 includes the subject matter of any of Examples 1-15, and wherein to share the data stored in the untrusted memory buffer comprises to write the data by the application image and to read the data by the secure enclave.

[0059] Example 17 includes the subject matter of any of Examples 1-16, and wherein to load the application image comprises to load an untrusted application image that is not trusted by an operating system of the computing device; and to validate the control flow integrity of the secure enclave comprises to validate that the untrusted application image does not include an enclave entry instruction.

[0060] Example 18 includes the subject matter of any of Examples 1-17, and wherein the operating system loader module is further to execute the untrusted application image within a sandbox established by the operating system in response to loading of the untrusted application image.

[0061] Example 19 includes a method for mutual application isolation, the method comprising loading, by a computing device, an application image to a memory range within a predefined virtual address range; creating, by the computing device, a secure enclave, wherein the predefined virtual address range is assigned to the secure enclave; validating, by the computing device, control flow integrity of the secure enclave in response to creating the secure enclave; and executing, by the computing device, the secure enclave in response to validating the control flow integrity of the secure enclave, wherein executing the secure enclave comprises enforcing mutual isolation of the secure enclave and the application image with secure enclave support of a processor of the computing device.

[0062] Example 20 includes the subject matter of Example 19, and wherein enforcing mutual isolation of the secure enclave and the application image comprises generating abort

transaction semantics by the processor in response to a memory access from the application image into the enclave secure enclave.

[0063] Example 21 includes the subject matter of any of Examples 19 and 20, and wherein enforcing mutual isolation of the secure enclave and the application image comprises generating a fault by the processor in response to a memory access from the secure enclave into the application image.

[0064] Example 22 includes the subject matter of any of Examples 19-21, and wherein validating the control flow integrity of the secure enclave comprises validating one or more secure enclave memory pages associated with the secure enclave to ensure that the secure enclave synchronously exits to an allowed address.

[0065] Example 23 includes the subject matter of any of Examples 19-22, and wherein validating the one or more secure enclave memory pages comprises statically analyzing the one or more secure enclave memory pages.

[0066] Example 24 includes the subject matter of any of Examples 19-23, and wherein validating the one or more secure enclave memory pages further comprises validating that an exit address associated with an enclave exit instruction is an allowed address.

[0067] Example 25 includes the subject matter of any of Examples 19-24, and wherein validating that the exit address associated with the enclave exit instruction is an allowed address comprises determining whether the exit address is at a predefined address within the application image or determining whether the exit address is at a predefined offset from a base address of the secure enclave.

[0068] Example 26 includes the subject matter of any of Examples 19-25, and wherein validating the one or more secure enclave memory pages comprises verifying that each executable secure enclave memory page is not writable.

[0069] Example 27 includes the subject matter of any of Examples 19-26, and further comprising adding, by the computing device, the one or more secure enclave memory pages to an enclave page cache in response to validating the one or more secure enclave memory pages, wherein the enclave page cache is included within the predefined virtual address range; and initializing, by the computing device, the secure enclave in response to adding the one or more secure enclave memory pages to the enclave page cache, wherein initializing the secure enclave comprises measuring and verifying the secure enclave.

[0070] Example 28 includes the subject matter of any of Examples 19-27, and wherein adding the one or more secure enclave memory pages to the enclave page cache comprises adding the memory pages at a predefined offset within the predefined virtual address range.

[0071] Example 29 includes the subject matter of any of Examples 19-28, and wherein validating the control flow integrity of the secure enclave comprises validating an asynchronous exit point associated with an enclave entry instruction.

[0072] Example 30 includes the subject matter of any of Examples 19-29, and wherein validating the asynchronous exit point associated with the enclave entry instruction comprises determining whether the asynchronous exit point is associated with a code segment of the application image that ends with an enclave resume instruction.

[0073] Example 31 includes the subject matter of any of Examples 19-30, and further comprising verifying, by the computing device, that the application image is an operating-system trusted application image; executing, by the computing device, the application image in response to verifying that the application image is an operating-system-trusted application image; and entering, by the computing device, the secure enclave in response to validating the control flow integrity of the secure enclave; wherein validating the asynchronous exit point associated with the enclave entry instruction comprises validating the asynchronous exit point associated with an enclave entry instruction of the application image in response to executing the application image.

[0074] Example 32 includes the subject matter of any of Examples 19-31, and further comprising sharing, by the computing device, data stored in an untrusted memory buffer between the application image and the secure enclave, wherein the untrusted memory buffer is located outside of the predefined virtual address range.

[0075] Example 33 includes the subject matter of any of Examples 19-32, and wherein sharing the data stored in the untrusted memory buffer comprises writing the data by the secure enclave and reading the data by the application image.

[0076] Example 34 includes the subject matter of any of Examples 19-33, and wherein sharing the data stored in the untrusted memory buffer comprises writing the data by the application image and reading the data by the secure enclave.

[0077] Example 35 includes the subject matter of any of Examples 19-34, and wherein loading the application image comprises loading an untrusted application image that is not trusted by an operating system of the computing device; and validating the control flow integrity of the secure enclave comprises validating that the untrusted application image does not include an enclave entry instruction.

[0078] Example 36 includes the subject matter of any of Examples 19-35, and further comprising executing, by the computing device, the untrusted application image within a

sandbox established by the operating system in response to loading the untrusted application image.

[0079] Example 37 includes a computing device comprising a processor; and a memory having stored therein a plurality of instructions that when executed by the processor cause the computing device to perform the method of any of Examples 19-36.

[0080] Example 38 includes one or more machine readable storage media comprising a plurality of instructions stored thereon that in response to being executed result in a computing device performing the method of any of Examples 19-36.

[0081] Example 39 includes a computing device comprising means for performing the method of any of Examples 19-36.

[0082] Example 40 includes a computing device for mutual application isolation, the computing device comprising means for loading an application image to a memory range within a predefined virtual address range; means for creating a secure enclave, wherein the predefined virtual address range is assigned to the secure enclave; means for validating control flow integrity of the secure enclave in response to creating the secure enclave; and means for executing the secure enclave in response to validating the control flow integrity of the secure enclave, wherein the means for executing the secure enclave comprises means for enforcing mutual isolation of the secure enclave and the application image with secure enclave support of a processor of the computing device.

[0083] Example 41 includes the subject matter of Example 40, and wherein the means for enforcing mutual isolation of the secure enclave and the application image comprises means for generating abort transaction semantics by the processor in response to a memory access from the application image into the enclave secure enclave.

[0084] Example 42 includes the subject matter of any of Examples 40 and 41, and wherein the means for enforcing mutual isolation of the secure enclave and the application image comprises means for generating a fault by the processor in response to a memory access from the secure enclave into the application image.

[0085] Example 43 includes the subject matter of any of Examples 40-42, and wherein the means for validating the control flow integrity of the secure enclave comprises means for validating one or more secure enclave memory pages associated with the secure enclave to ensure that the secure enclave synchronously exits to an allowed address.

[0086] Example 44 includes the subject matter of any of Examples 40-43, and wherein the means for validating the one or more secure enclave memory pages comprises means for statically analyzing the one or more secure enclave memory pages.

[0087] Example 45 includes the subject matter of any of Examples 40-44, and wherein the means for validating the one or more secure enclave memory pages further comprises means for validating that an exit address associated with an enclave exit instruction is an allowed address.

[0088] Example 46 includes the subject matter of any of Examples 40-45, and wherein the means for validating that the exit address associated with the enclave exit instruction is an allowed address comprises means for determining whether the exit address is at a predefined address within the application image or determining whether the exit address is at a predefined offset from a base address of the secure enclave.

[0089] Example 47 includes the subject matter of any of Examples 40-46, and wherein the means for validating the one or more secure enclave memory pages comprises means for verifying that each executable secure enclave memory page is not writable.

[0090] Example 48 includes the subject matter of any of Examples 40-47, and further comprising means for adding the one or more secure enclave memory pages to an enclave page cache in response to validating the one or more secure enclave memory pages, wherein the enclave page cache is included within the predefined virtual address range; and means for initializing the secure enclave in response to adding the one or more secure enclave memory pages to the enclave page cache, wherein initializing the secure enclave comprises measuring and verifying the secure enclave.

[0091] Example 49 includes the subject matter of any of Examples 40-48, and wherein the means for adding the one or more secure enclave memory pages to the enclave page cache comprises means for adding the memory pages at a predefined offset within the predefined virtual address range.

[0092] Example 50 includes the subject matter of any of Examples 40-49, and wherein the means for validating the control flow integrity of the secure enclave comprises means for validating an asynchronous exit point associated with an enclave entry instruction.

[0093] Example 51 includes the subject matter of any of Examples 40-50, and wherein the means for validating the asynchronous exit point associated with the enclave entry instruction comprises means for determining whether the asynchronous exit point is associated with a code segment of the application image that ends with an enclave resume instruction.

[0094] Example 52 includes the subject matter of any of Examples 40-51, and further comprising means for verifying that the application image is an operating-system trusted application image; means for executing the application image in response to verifying that the application image is an operating-system-trusted application image; and means for entering the

secure enclave in response to validating the control flow integrity of the secure enclave; wherein the means for validating the asynchronous exit point associated with the enclave entry instruction comprises means for validating the asynchronous exit point associated with an enclave entry instruction of the application image in response to executing the application image.

[0095] Example 53 includes the subject matter of any of Examples 40-52, and further comprising means for sharing data stored in an untrusted memory buffer between the application image and the secure enclave, wherein the untrusted memory buffer is located outside of the predefined virtual address range.

[0096] Example 54 includes the subject matter of any of Examples 40-53, and wherein the means for sharing the data stored in the untrusted memory buffer comprises means for writing the data by the secure enclave and reading the data by the application image.

[0097] Example 55 includes the subject matter of any of Examples 40-54, and wherein the means for sharing the data stored in the untrusted memory buffer comprises means for writing the data by the application image and reading the data by the secure enclave.

[0098] Example 56 includes the subject matter of any of Examples 40-55, and wherein the means for loading the application image comprises means for loading an untrusted application image that is not trusted by an operating system of the computing device; and the means for validating the control flow integrity of the secure enclave comprises means for validating that the untrusted application image does not include an enclave entry instruction.

[0099] Example 57 includes the subject matter of any of Examples 40-56, and further comprising means for executing the untrusted application image within a sandbox established by the operating system in response to loading the untrusted application image.

WHAT IS CLAIMED IS:

1. A computing device for mutual application isolation, the computing device comprising:

- a processor with secure enclave support;
- an operating system loader module to load an application image to a memory range within a predefined virtual address range;
- an enclave loader module to (i) create a secure enclave, wherein the predefined virtual address range is assigned to the secure enclave and (ii) validate control flow integrity of the secure enclave in response to creation of the secure enclave; and
- a mutual isolation module to execute the secure enclave in response to validation of the control flow integrity of the secure enclave, wherein to execute the secure enclave comprises to enforce mutual isolation of the secure enclave and the application image with the secure enclave support of the processor.

2. The computing device of claim 1, wherein to enforce mutual isolation of the secure enclave and the application image comprises to:

- generate abort transaction semantics by the processor in response to a memory access from the application image into the enclave secure enclave; and
- generate a fault by the processor in response to a memory access from the secure enclave into the application image.

3. The computing device of claim 1, wherein to validate the control flow integrity of the secure enclave comprises to validate one or more secure enclave memory pages associated with the secure enclave to ensure that the secure enclave synchronously exits to an allowed address.

4. The computing device of claim 3, wherein to validate the one or more secure enclave memory pages comprises to:

- statically analyze the one or more secure enclave memory pages; and
- validate that an exit address associated with an enclave exit instruction is an allowed address.

5. The computing device of claim 4, wherein to validate that the exit address associated with the enclave exit instruction is an allowed address comprises to determine

whether the exit address is at a predefined address within the application image or to determine whether the exit address is at a predefined offset from a base address of the secure enclave.

6. The computing device of claim 3, wherein to validate the one or more secure enclave memory pages comprises to verify that each executable secure enclave memory page is not writable.

7. The computing device of claim 3, wherein the enclave loader module is further to:

add the one or more secure enclave memory pages to an enclave page cache at a predefined offset within the predefined virtual address range in response to validation of the one or more secure enclave memory pages, wherein the enclave page cache is included within the predefined virtual address range; and

initialize the secure enclave in response to addition of the one or more secure enclave memory pages to the enclave page cache, wherein to initialize the secure enclave comprises to measure and verify the secure enclave.

8. The computing device of any of claims 1-7, wherein to validate the control flow integrity of the secure enclave comprises to validate an asynchronous exit point associated with an enclave entry instruction.

9. The computing device of claim 8, wherein to validate the asynchronous exit point associated with the enclave entry instruction comprises to determine whether the asynchronous exit point is associated with a code segment of the application image that ends with an enclave resume instruction.

10. The computing device of claim 8, wherein:
the operating system loader module is further to (i) verify that the application image is an operating-system trusted application image and (ii) execute the application image in response to verification that the application image is an operating-system-trusted application image;

the enclave loader module is further to enter the secure enclave in response to the validation of the control flow integrity of the secure enclave; and

to validate the asynchronous exit point associated with the enclave entry instruction comprises to validate the asynchronous exit point associated with an enclave entry instruction of the application image in response to execution of the application image.

11. The computing device of any of claims 1-7, wherein the mutual isolation module is further to share data stored in an untrusted memory buffer between the application image and the secure enclave, wherein the untrusted memory buffer is located outside of the predefined virtual address range.

12. The computing device of any of claims 1-7, wherein:
to load the application image comprises to load an untrusted application image that is not trusted by an operating system of the computing device; and
to validate the control flow integrity of the secure enclave comprises to validate that the untrusted application image does not include an enclave entry instruction.

13. The computing device of claim 12, wherein the operating system loader module is further to execute the untrusted application image within a sandbox established by the operating system in response to loading of the untrusted application image.

14. A method for mutual application isolation, the method comprising:
loading, by a computing device, an application image to a memory range within a predefined virtual address range;
creating, by the computing device, a secure enclave, wherein the predefined virtual address range is assigned to the secure enclave;
validating, by the computing device, control flow integrity of the secure enclave in response to creating the secure enclave; and
executing, by the computing device, the secure enclave in response to validating the control flow integrity of the secure enclave, wherein executing the secure enclave comprises enforcing mutual isolation of the secure enclave and the application image with secure enclave support of a processor of the computing device.

15. The method of claim 14, wherein enforcing mutual isolation of the secure enclave and the application image comprises:
generating abort transaction semantics by the processor in response to a memory access from the application image into the enclave secure enclave; and

generating a fault by the processor in response to a memory access from the secure enclave into the application image.

16. The method of claim 14, wherein validating the control flow integrity of the secure enclave comprises validating one or more secure enclave memory pages associated with the secure enclave to ensure that the secure enclave synchronously exits to an allowed address.

17. The method of claim 16, wherein validating the one or more secure enclave memory pages comprises statically analyzing the one or more secure enclave memory pages.

18. The method of claim 16, wherein validating the one or more secure enclave memory pages comprises verifying that each executable secure enclave memory page is not writable.

19. The method of claim 16, further comprising:
adding, by the computing device, the one or more secure enclave memory pages to an enclave page cache in response to validating the one or more secure enclave memory pages, wherein the enclave page cache is included within the predefined virtual address range;
and

initializing, by the computing device, the secure enclave in response to adding the one or more secure enclave memory pages to the enclave page cache, wherein initializing the secure enclave comprises measuring and verifying the secure enclave.

20. The method of claim 14, wherein validating the control flow integrity of the secure enclave comprises validating an asynchronous exit point associated with an enclave entry instruction.

21. The method of claim 20, further comprising:
verifying, by the computing device, that the application image is an operating-system trusted application image;
executing, by the computing device, the application image in response to verifying that the application image is an operating-system-trusted application image; and
entering, by the computing device, the secure enclave in response to validating the control flow integrity of the secure enclave;

wherein validating the asynchronous exit point associated with the enclave entry instruction comprises validating the asynchronous exit point associated with an enclave entry instruction of the application image in response to executing the application image.

22. The method of claim 14, wherein:

loading the application image comprises loading an untrusted application image that is not trusted by an operating system of the computing device; and

validating the control flow integrity of the secure enclave comprises validating that the untrusted application image does not include an enclave entry instruction.

23. A computing device comprising:

a processor; and

a memory having stored therein a plurality of instructions that when executed by the processor cause the computing device to perform the method of any of claims 14-22.

24. One or more machine readable storage media comprising a plurality of instructions stored thereon that in response to being executed result in a computing device performing the method of any of claims 14-22.

25. A computing device comprising means for performing the method of any of claims 14-22.

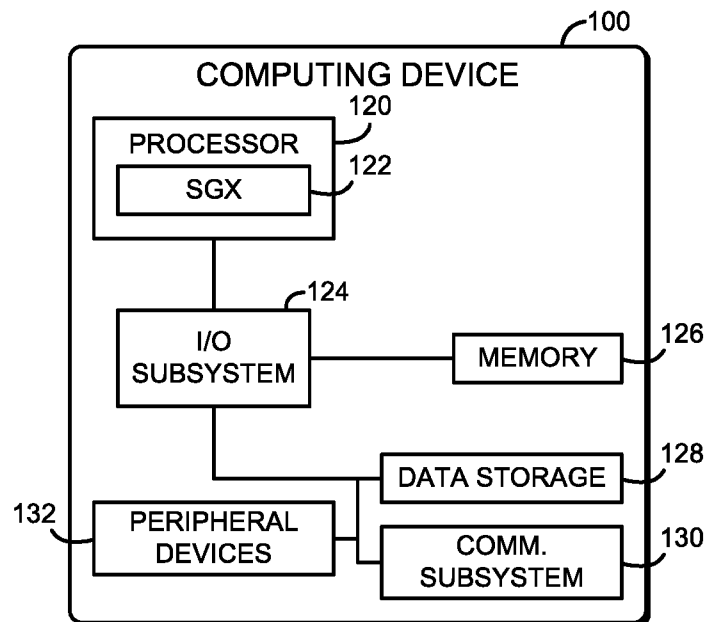


FIG. 1

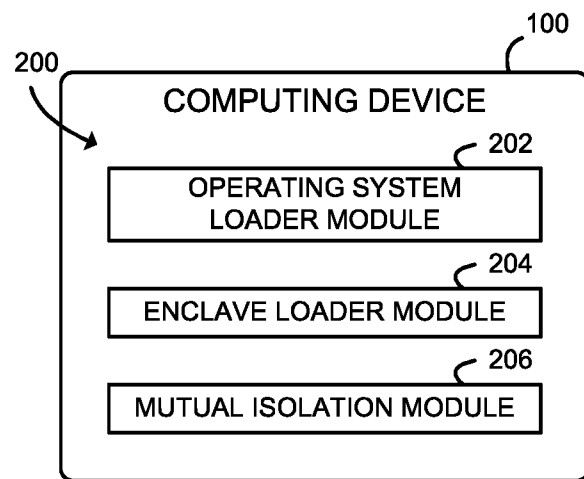


FIG. 2

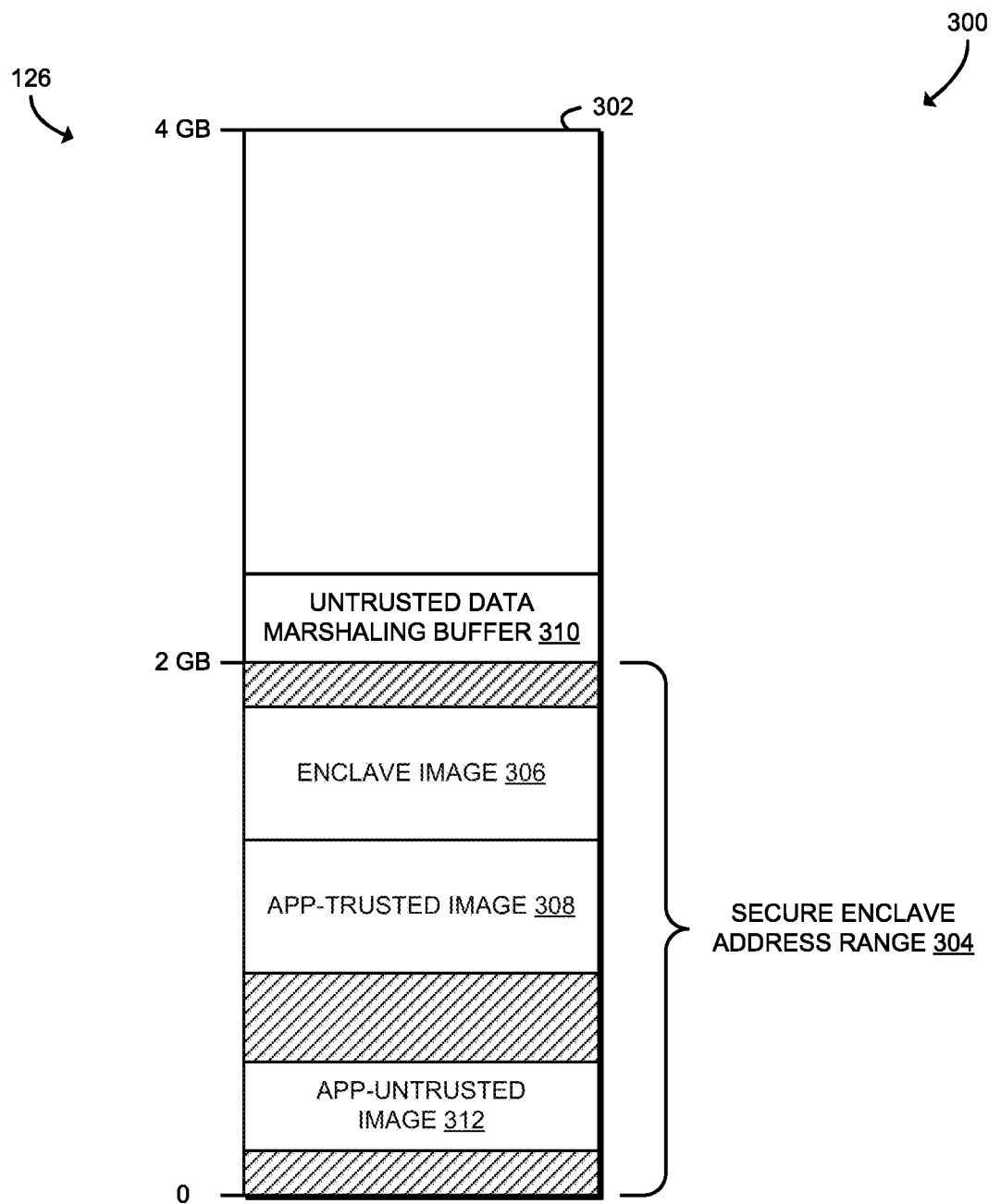
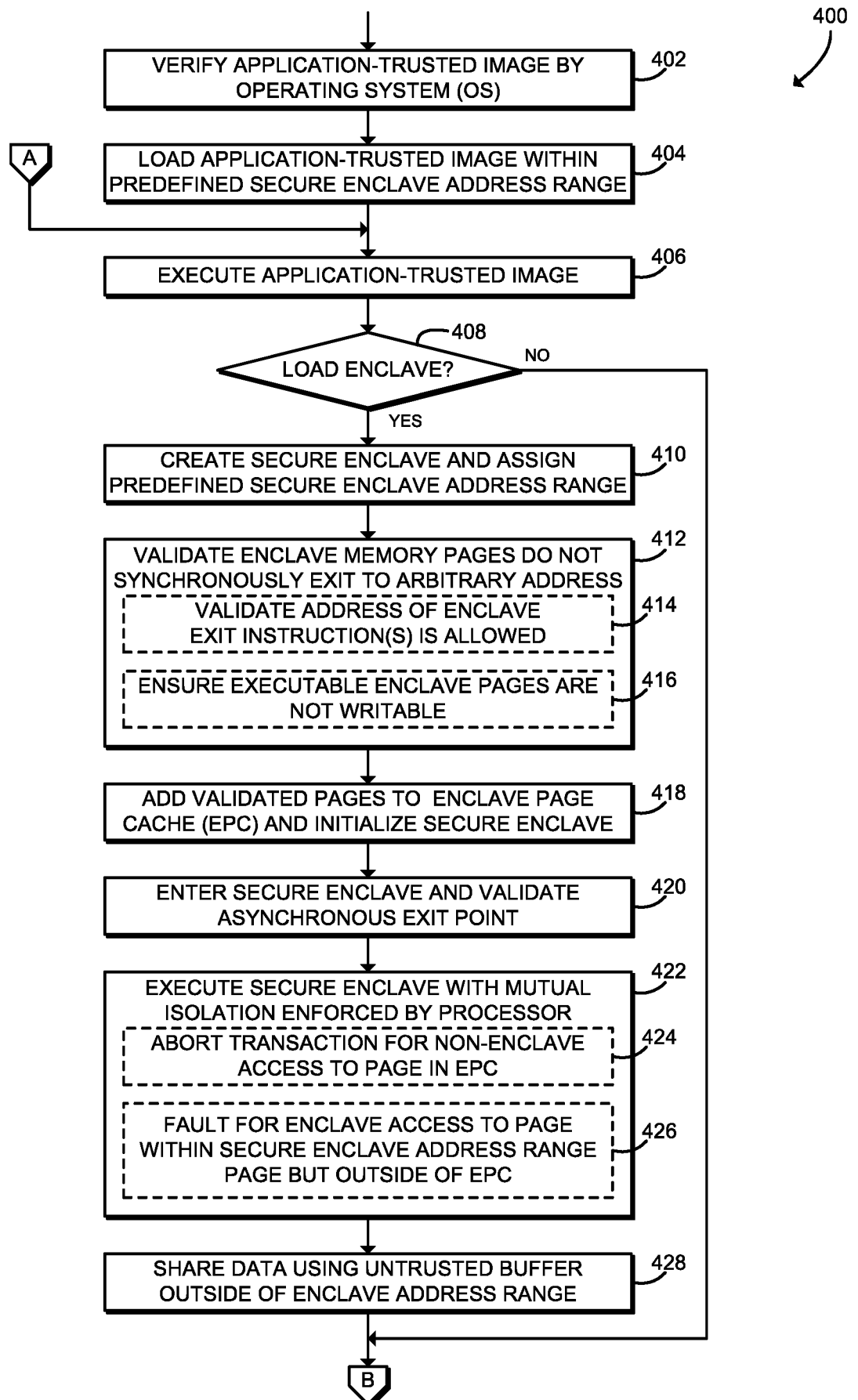
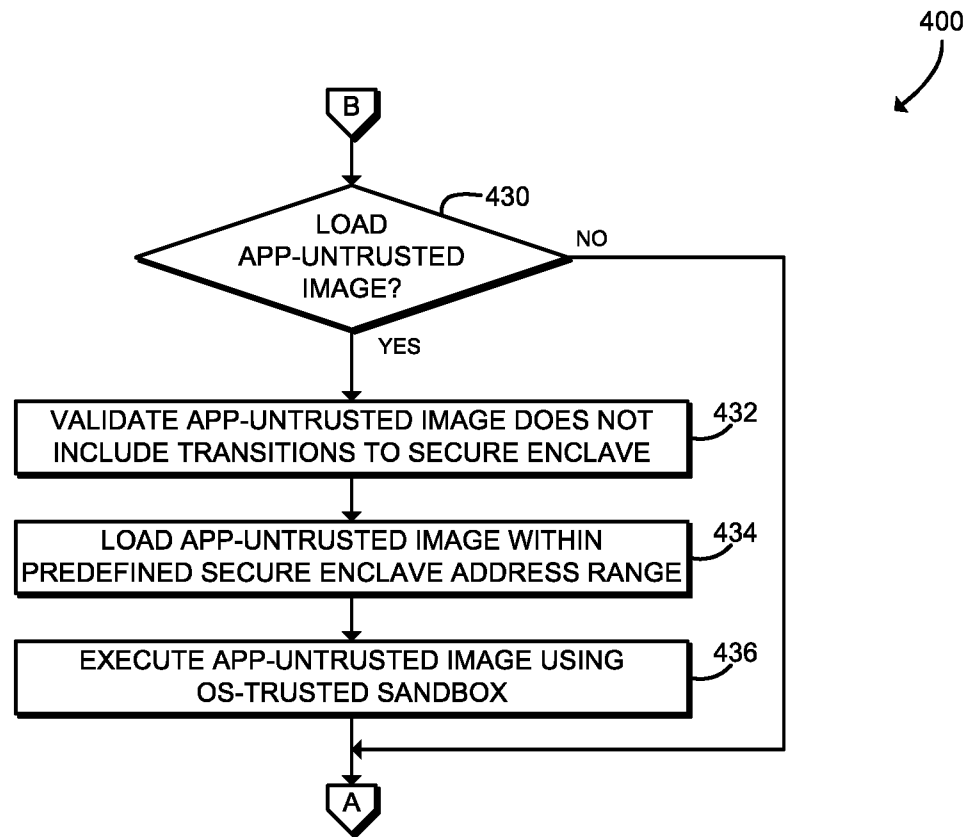


FIG. 3





A. CLASSIFICATION OF SUBJECT MATTER**G06F 21/53(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 21/53; G06F 12/10; G06F 12/14; G06F 21/60; G06F 12/08

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: secure enclave, mutual application isolation, validate

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2013-0198853 A1 (FRANCIS X. MCKEEN et al.) 01 August 2013 See paragraphs [0064]-[0094], [0128]-[0169], [0394]-[0403], [0658]-[0660]; claim 1; and figures 7, 14, 25.	1-2, 11-15, 22-25
Y		3-10, 16-21
Y	US 2014-0337983 A1 (XIAOZHU KANG et al.) 13 November 2014 See paragraphs [0018]-[0023], [0049]-[0055]; and figures 1, 4.	3-10, 16-21
A	US 2015-0186272 A1 (MICHAEL GOLDSMITH et al.) 02 July 2015 See paragraphs [0021]-[0031]; and figure 2.	1-25
A	US 2015-0278528 A1 (INTEL CORPORATION) 01 October 2015 See paragraphs [0077]-[0083]; and figure 3.	1-25
A	US 2014-0006711 A1 (BIN XING) 02 January 2014 See paragraphs [0018]-[0028]; and figure 1.	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

08 June 2017 (08.06.2017)

Date of mailing of the international search report

09 June 2017 (09.06.2017)

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/US2017/019636

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2013-0198853 A1	01/08/2013	US 2013-0159726 A1 US 9087200 B2	20/06/2013 21/07/2015
US 2014-0337983 A1	13/11/2014	US 09652609 B2 US 2015-0278514 A1 US 9087202 B2	16/05/2017 01/10/2015 21/07/2015
US 2015-0186272 A1	02/07/2015	EP 2889778 A1	01/07/2015
US 2015-0278528 A1	01/10/2015	CN 106415506 A EP 3123340 A1 KR 10-2016-0113254 A TW 201544955 A TW 1563385 B WO 2015-148881 A1	15/02/2017 01/02/2017 28/09/2016 01/12/2015 21/12/2016 01/10/2015
US 2014-0006711 A1	02/01/2014	US 2016-0054945 A1 US 9053042 B2	25/02/2016 09/06/2015