US 20160283721A1

(19) United States
(12) Patent Application Publication
Nemiroff et al.
(10) Pub. No.: US 2016/0283721 A1
(43) Pub. Date: Sep. 29, 2016

(54) OUT-OF-BAND HOST OS BOOT SEQUENCE VERIFICATION

(71) Applicant: **Intel Corporation**, Santa Clara, CA (US)

(72) Inventors: **Daniel Nemiroff**, Folsom, CA (US); **Paul J. Thadikaran**, Rancho Cordova, CA (US); **Andrew H. Gafken**, Folsom, CA (US); **Purushottam Goel**, Beaverton, OR (US); **Nicholas D. Triantafillou**, Portland, OR (US); **Paritosh Saxena**, Portland, OR (US); **Debra Cablao**, Hillsboro, OR (US)

(21) Appl. No.: **15/179,665**

(22) Filed: **Jun. 10, 2016**

Related U.S. Application Data

(63) Continuation of application No. 13/536,859, filed on Jun. 28, 2012, now Pat. No. 9,367,328.

Publication Classification

(51) **Int. Cl.**

G06F 21/57	(2006.01)
G06F 21/64	(2006.01)
G06F 21/62	(2006.01)
G06F 21/56	(2006.01)

(52) **U.S. Cl.**
CPC ***G06F 21/577*** (2013.01); ***G06F 21/575*** (2013.01); ***G06F 21/561*** (2013.01); ***G06F 21/64*** (2013.01); ***G06F 21/6218*** (2013.01); *G06F 2221/033* (2013.01)

(57) ABSTRACT

Embodiments of techniques and systems for out-of-band verification of host OS components are described. In embodiments, a out-of-band host OS boot sequence verification system ("BSVS") may access system memory without detection by a host OS process, or "out of band." The BSVS may access host OS components in the system memory and may generate signatures from memory footprints of the host OS components. These signatures may then be compared to trusted signatures to verify integrity of the host OS components. In embodiments, this verification may be performed during a boot of a host OS or on demand. In embodiments, the trusted signatures may be pre-stored by the BSVS before a boot; in some embodiments, the trusted signatures may be previously-computed and then stored by the BSVS. Other embodiments may be described and claimed.

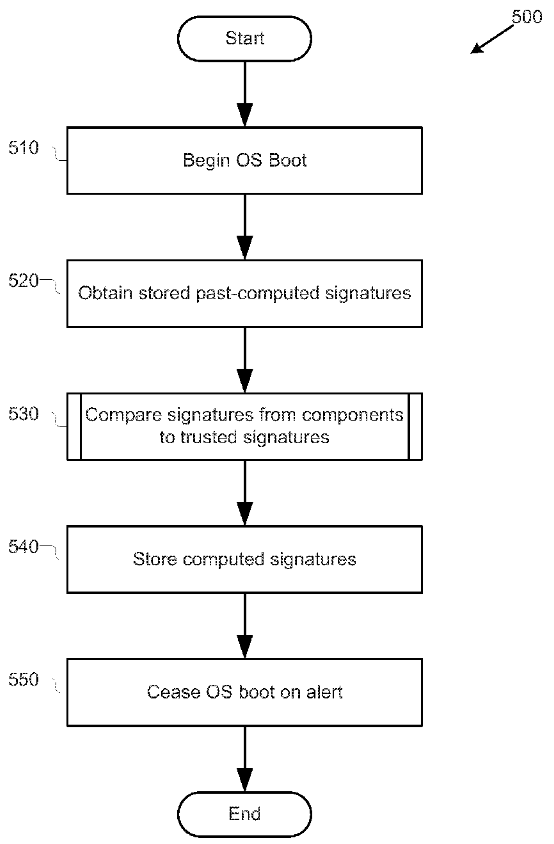

500
Start
510
Begin OS Boot
520
Obtain stored past-computed signatures
530
Compare signatures from components to trusted signatures
540
Store computed signatures
550
Cease OS boot on alert
End

Fig. 1

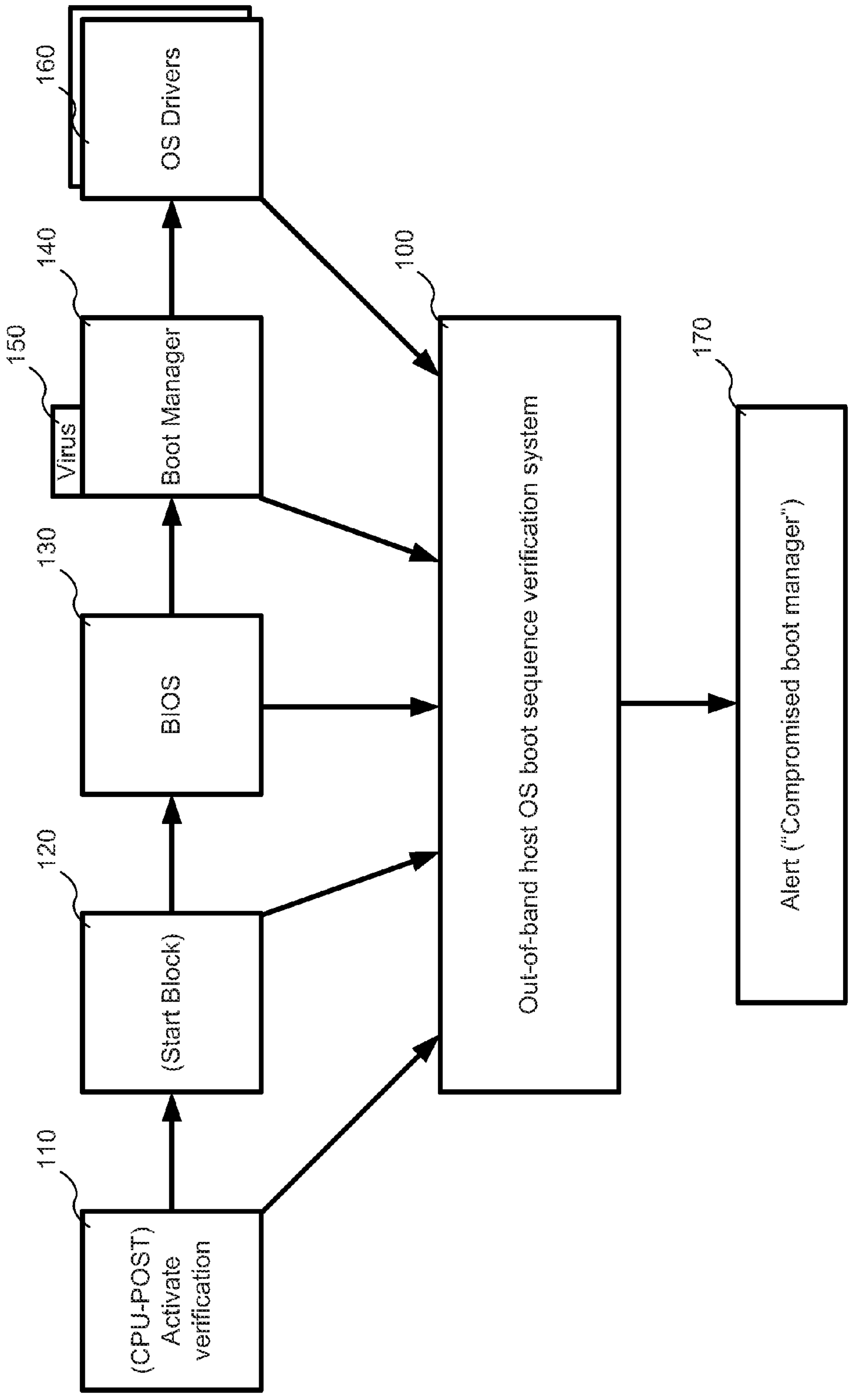

(CPU-POST) Activate verification
110
(Start Block)
120
BIOS
130
Boot Manager
140
Virus
150
OS Drivers
160
Out-of-band host OS boot sequence verification system
100
Alert ("Compromised boot manager")
170

Fig. 2

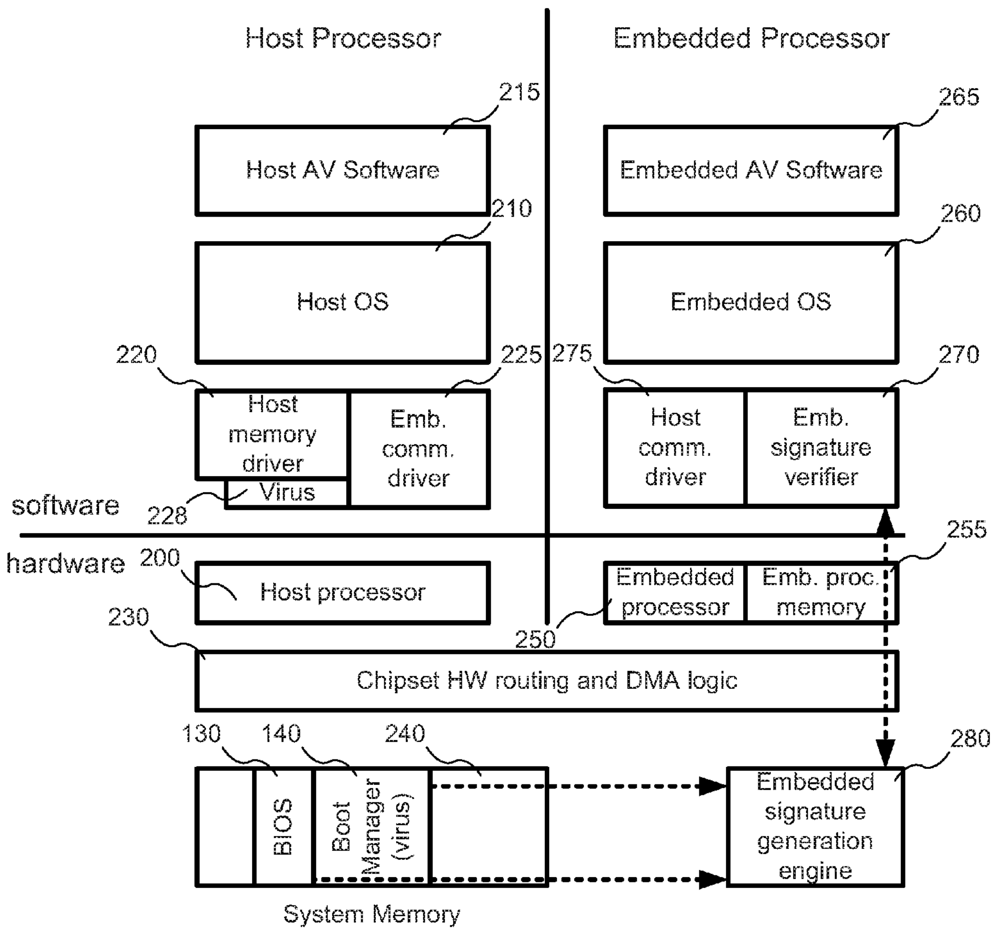

Host Processor
Embedded Processor
215
Host AV Software
265
Embedded AV Software
210
Host OS
260
Embedded OS
220
225
275
270
Host memory driver
Emb. comm. driver
Host comm. driver
Emb. signature verifier
Virus
software
228
hardware
200
Host processor
255
Embedded processor
Emb. proc. memory
250
230
Chipset HW routing and DMA logic
130
140
240
280
BIOS
Boot Manager (virus)
Embedded signature generation engine
System Memory

Fig. 3

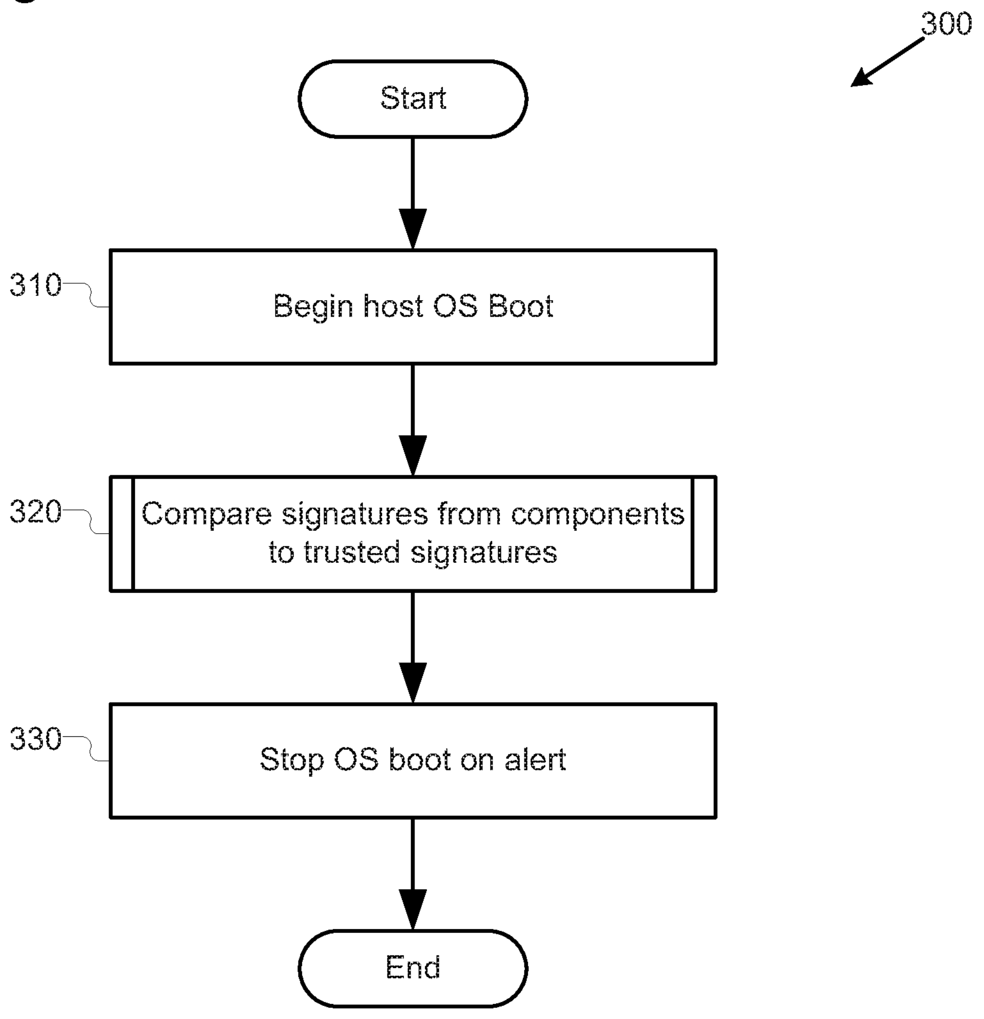

300
Start
310
Begin host OS Boot
320
Compare signatures from components to trusted signatures
330
Stop OS boot on alert
End

Fig. 4

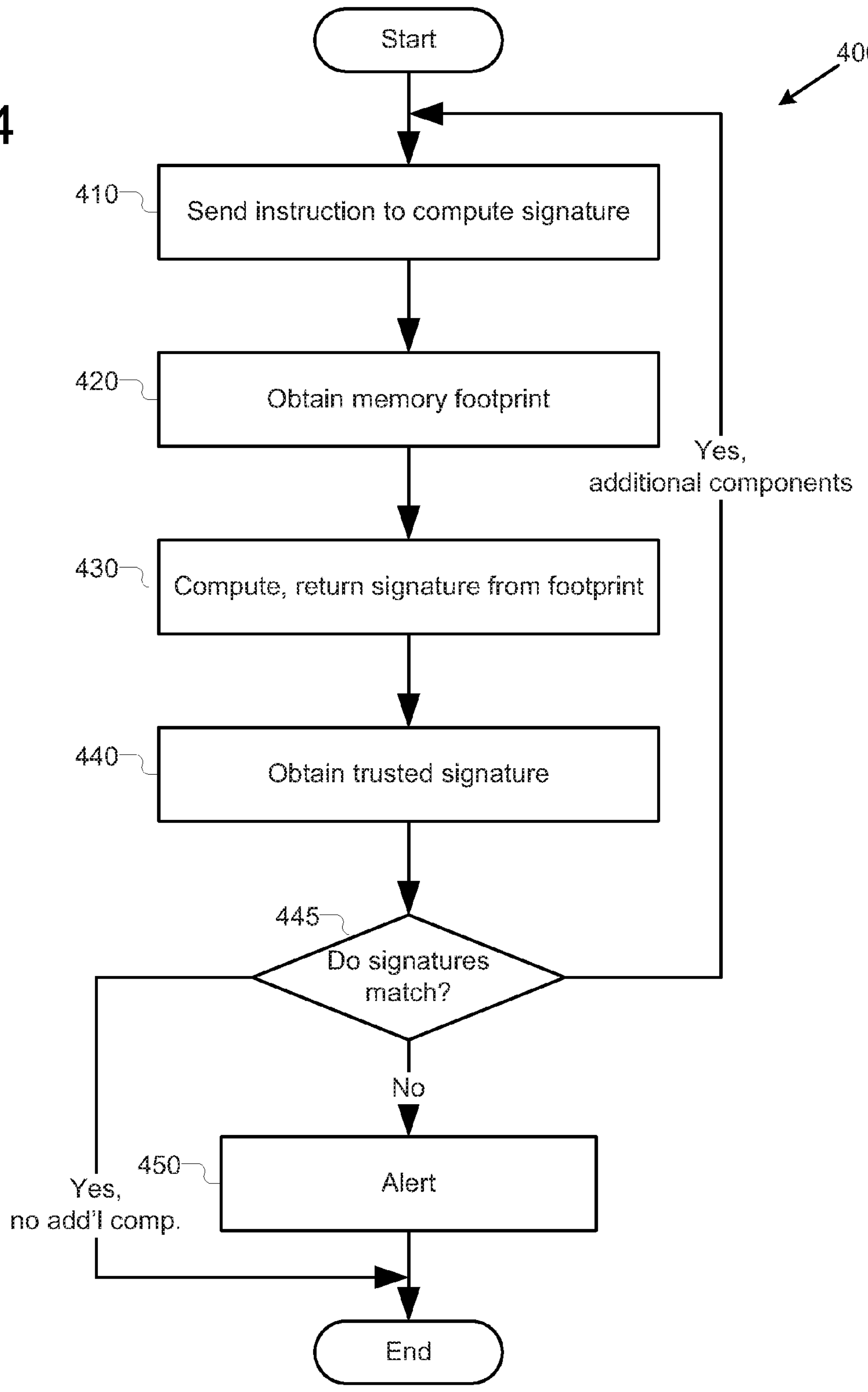

Start
400
410
Send instruction to compute signature
420
Obtain memory footprint
Yes,
additional components
430
Compute, return signature from footprint
440
Obtain trusted signature
445
Do signatures match?
No
450
Alert
Yes,
no add'l comp.
End

Fig. 5

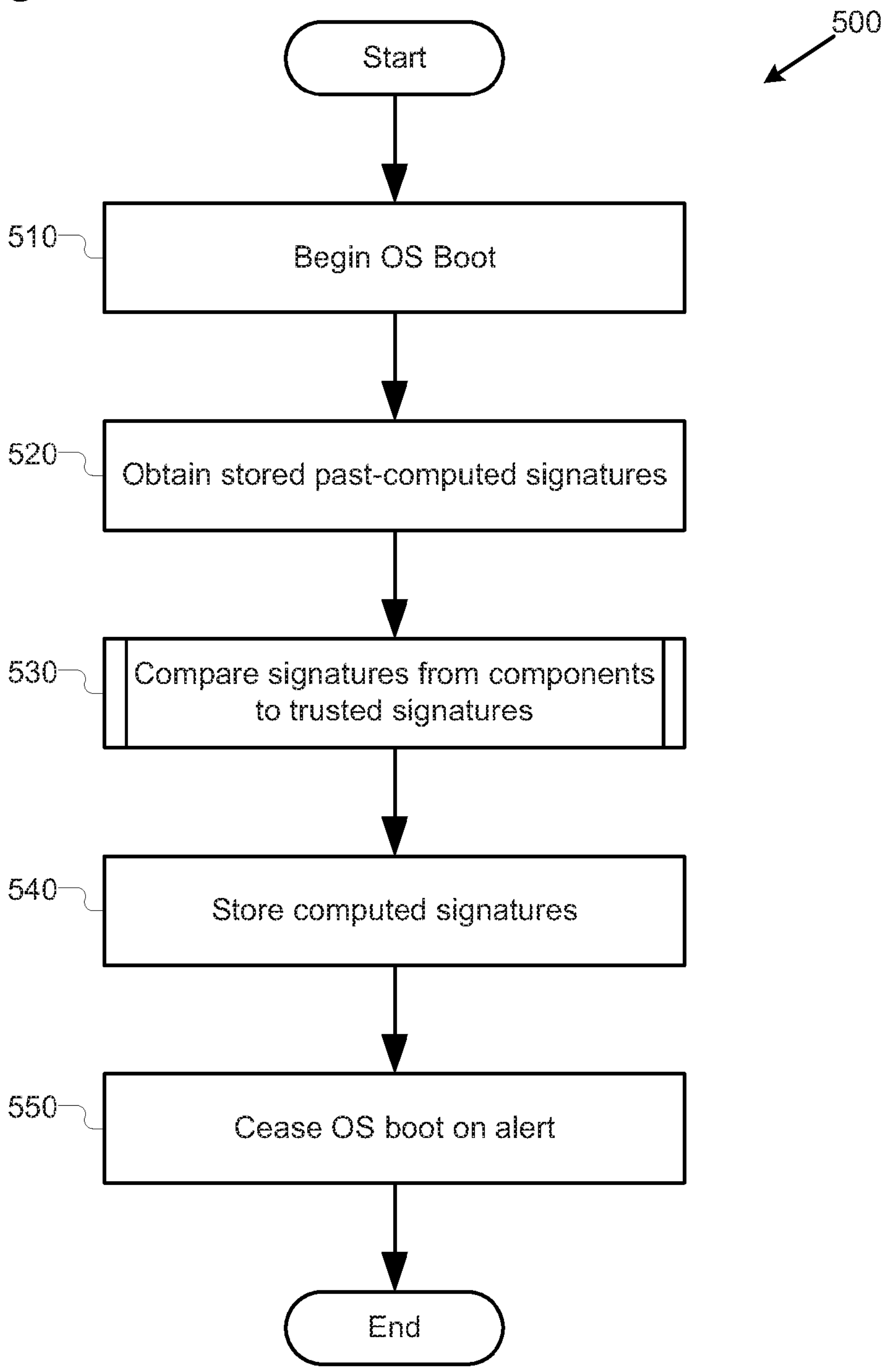

500
Start
510
Begin OS Boot
520
Obtain stored past-computed signatures
530
Compare signatures from components to trusted signatures
540
Store computed signatures
550
Cease OS boot on alert
End

Fig. 6

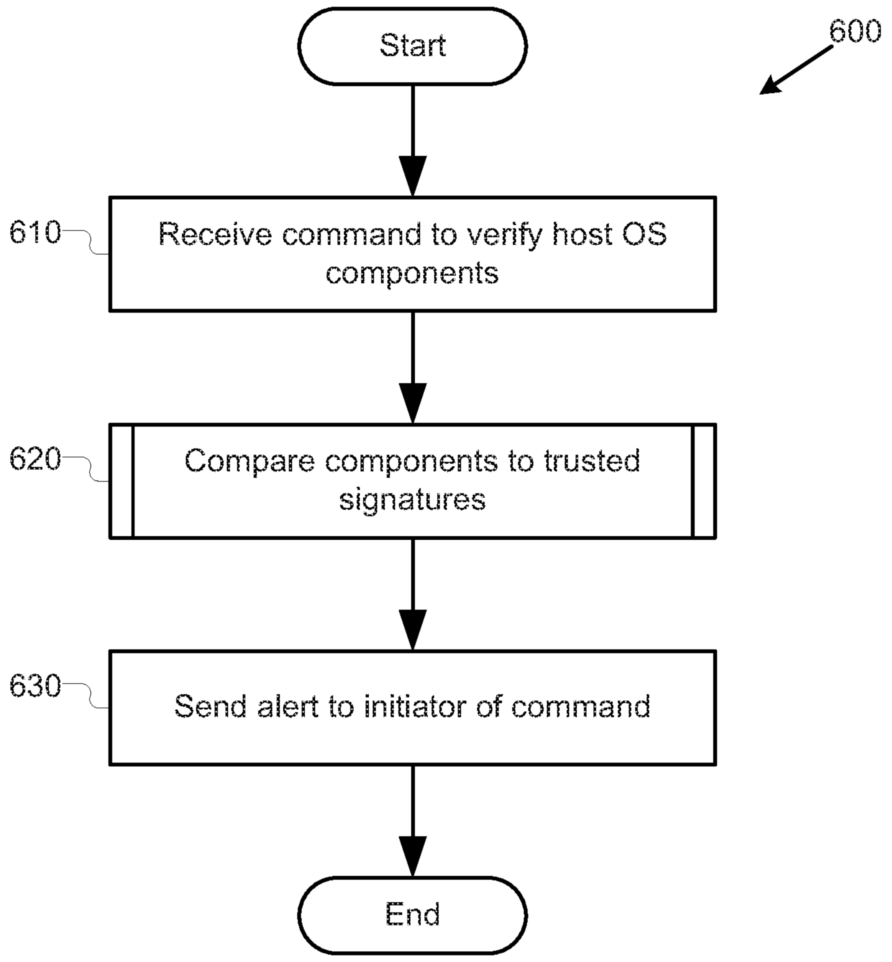
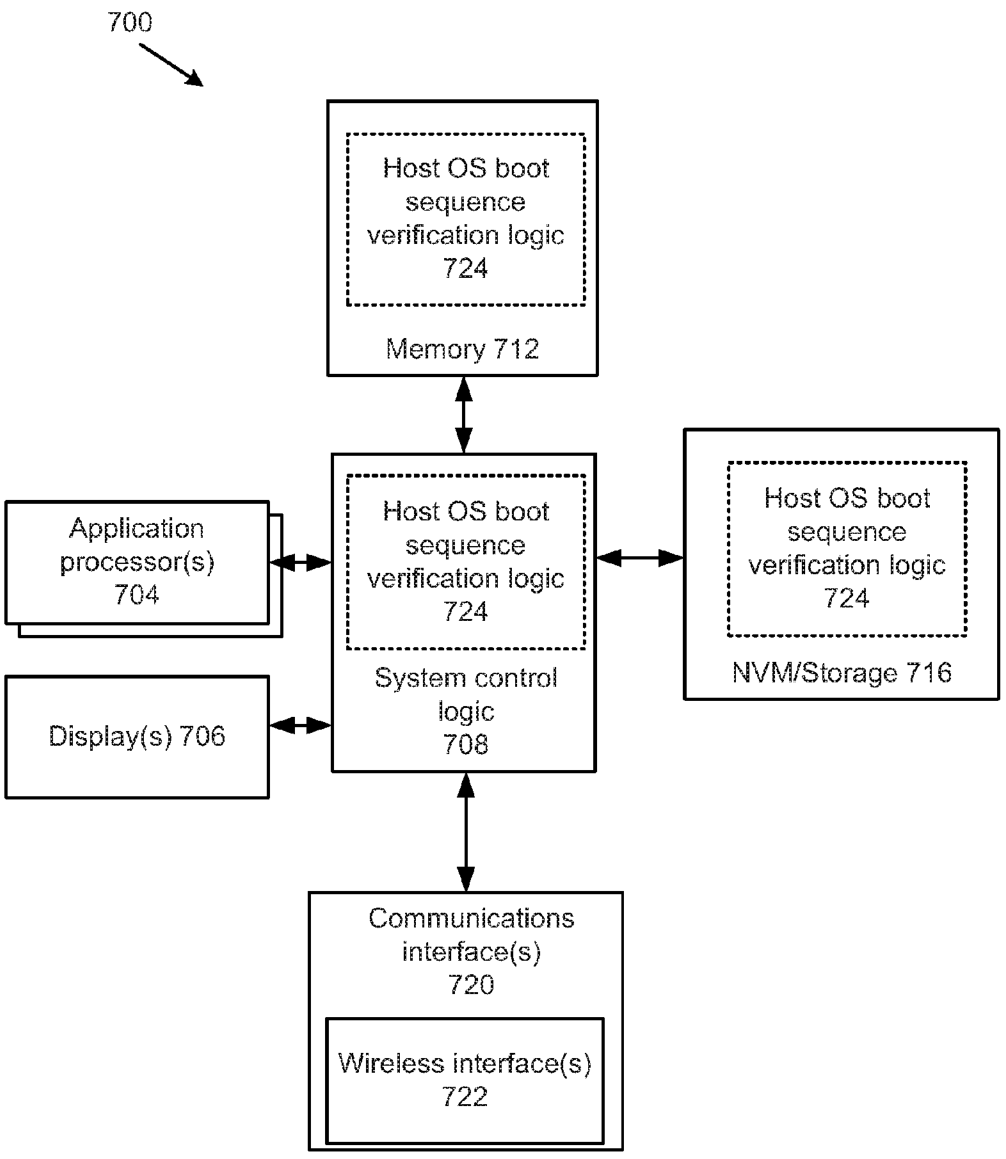

Start
600
610
Receive command to verify host OS components
620
Compare components to trusted signatures
630
Send alert to initiator of command
End

Fig. 7


700
Host OS boot sequence verification logic 724
Memory 712
Application processor(s) 704
Host OS boot sequence verification logic 724
System control logic 708
Host OS boot sequence verification logic 724
NVM/Storage 716
Display(s) 706
Communications interface(s) 720
Wireless interface(s) 722

OUT-OF-BAND HOST OS BOOT SEQUENCE VERIFICATION

BACKGROUND

[0001] Viruses, worms, and other malware continue to create problems for users of computing devices. Various systems and devices may utilize anti-virus software to protect against infection by identifying and removing viruses. However, many viruses may thwart this protection by modifying files in a host operating system ("host OS") to prevent detection by the anti-virus software. For example, a virus may infect a boot manager for the host OS. The virus, either by itself or through the infected boot manager, may modify other host OS files that are utilized during detection of viruses. If the virus modifies the host OS files to intercept requests that are used by anti-virus software, the virus may then modify these requests and thus prevent the detection.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Embodiments will be readily understood by the following detailed description in conjunction with the accompanying drawings. To facilitate this description, like reference numerals designate like structural elements. Embodiments are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings.

[0003] FIG. **1** is a block diagram illustrating an example out-of-band host OS boot sequence verification system, in accordance with various embodiments.

[0004] FIG. **2** illustrates example components of the out-of-band host OS boot sequence verification system, in accordance with various embodiments.

[0005] FIG. **3** illustrates an example out-of-band host OS boot sequence verification process of the host OS boot sequence verification system, in accordance with various embodiments.

[0006] FIG. **4** illustrates an example out-of-band host OS component verification process of the out-of-band host OS boot sequence verification system, in accordance with various embodiments.

[0007] FIG. **5** illustrates an example time-based out-of-band host OS boot sequence verification process of the host OS boot sequence verification system, in accordance with various embodiments.

[0008] FIG. **6** illustrates an example on-demand out-of-band OS boot sequence verification process of the out-of-band host OS boot sequence verification system, in accordance with various embodiments.

[0009] FIG. **7** illustrates an example computing environment suitable for practicing the disclosed embodiments, in accordance with various embodiments.

DETAILED DESCRIPTION

[0010] In the following detailed description, reference is made to the accompanying drawings which form a part hereof wherein like numerals designate like parts throughout, and in which is shown by way of illustration embodiments that may be practiced. It is to be understood that other embodiments may be utilized and structural or logical changes may be made without departing from the scope of the present disclosure. Therefore, the following detailed description is not to be taken in a limiting sense, and the scope of embodiments is defined by the appended claims and their equivalents.

[0011] Various operations may be described as multiple discrete actions or operations in turn, in a manner that is most helpful in understanding the claimed subject matter. However, the order of description should not be construed as to imply that these operations are necessarily order dependent. In particular, these operations may not be performed in the order of presentation. Operations described may be performed in a different order than the described embodiment. Various additional operations may be performed and/or described operations may be omitted in additional embodiments.

[0012] For the purposes of the present disclosure, the phrase "A and/or B" means (A), (B), or (A and B). For the purposes of the present disclosure, the phrase "A, B, and/or C" means (A), (B), (C), (A and B), (A and C), (B and C), or (A, B and C).

[0013] The description may use the phrases "in an embodiment," or "in embodiments," which may each refer to one or more of the same or different embodiments. Furthermore, the terms "comprising," "including," "having," and the like, as used with respect to embodiments of the present disclosure, are synonymous.

[0014] As may be used herein, the term "module" may refer to, be part of, or include an Application Specific Integrated Circuit ("ASIC"), an electronic circuit, a processor (shared, dedicated, or group) and/or memory (shared, dedicated, or group) that execute one or more software or firmware programs, a combinational logic circuit, and/or other suitable components that provide the described functionality.

[0015] Referring now to FIG. **1**, an out-of-band host OS boot sequence verification system **100** ("BSVS **100**") is illustrated. In various embodiments, the BSVS **100** may be configured to access and verify the integrity of one or more components of a host OS executing on a computing device. For example, and as illustrated in the example of FIG. **1**, the BSVS **100** may be configured to access and verify integrity of a BIOS **130**, a boot manager **140**, and/or one or more OS drivers **160**. In various embodiments, the BSVS **100** may perform such verification during, for example, a boot of the host OS; in other embodiments, the verification may be performed on demand. In various embodiments, the BSVS **100** may be configured to access the components directly in a system memory without intervention by, or even knowledge of, the host OS or any host processors the host OS is executing on. This action outside of the attention of the host OS and host processors may be referred to as acting "out of band."

[0016] In various embodiments, the BSVS **100** may be configured to generate an alert **170** if a component of the host OS is found to be compromised. Thus, in the example illustrated in FIG. **1**, where the boot manager **140** has been compromised by a virus **150**, the BSVS **100** may generate an alert **170** that the boot manager is compromised. In various embodiments, the BSVS **100** may be configured to display the alert **170** to a user of the computing device. In other embodiments, the BSVS **100** may be configured to send an alert **170** remotely, such as, for example, to a user on a remote computing device. In various embodiments, in addition to sending the alert **170**, the BSVS **100** may be configured to perform other actions in response to determining that a host OS component is corrupt. For example, in various embodiments, the BSVS **100** may be configured to interrupt a boot process of the host OS and/or begin a cleaning process, such as using an anti-virus application.

[0017] In various embodiments, the BSVS **100** may interact with various components and actions during a boot process of the host OS. Thus the BSVS **100** may receive a command to be activated during a boot process. For example, and as illustrated in FIG. **1**, during a power on self text process of a central processing unit ("CPU-POST"), a CPU may send a command to the BSVS **100** to activate verification of host OS components. Next, a start block **120** of the boot process may be loaded by an OS loader. In various embodiments, the start block **120** may determine settings prior to BIOS start and may send a notification to the BSVS **100** that the host OS boot process has started. The BSVS **100** may then be configured to access various components of the host OS during the boot process. Thus, for example, the BSVS **100** may access the BIOS **130**, the boot manager **140** and/or the OS drivers **160**. In various embodiments, the BSVS **100** may be configured to perform this access by accessing the components in memory once they are loaded during the boot of the host OS. Thus, the BSVS **100** may obtain a memory footprint for one or more of the BIOS **130**, the boot manager **140** and/or the OS drivers **160**. The BSVS **100** may then generate signatures, or other identifiers, from the memory footprint, and may compare these to trusted signatures. This comparison may, in various embodiments, allow the BSVS **100** to determine if the components are compromised or not, and thus whether to generate an alert.

[0018] FIG. **2** illustrates example components of the BSVS **100**, in accordance with various embodiments. FIG. **2** illustrates various components included in and/or executing on a host processor **200** and an embedded processor **250**. In various embodiments, the host processor **200** may include a CPU. As FIG. **2** illustrates, in various embodiments, one or more components may be executed in software on the host processor **200** or embedded processor **250**. Additionally, one or more hardware components may be found either in or coupled to the host processor **200** or embedded processor **250**, while some components may be accessible to both. In various embodiments, illustrated components may include components of the BSVS **100**, such as the embedded signature verifier **270** and/or embedded signature generation engine **280**, as described herein. In various embodiments, the host processor **200** and embedded processor **250** may be in separate packages or may be contained in a single package. In some embodiments, the embedded processor may be located on a card on a PCI express bus; in other embodiments, the embedded processor may be located on another hardware device coupled to a motherboard that can allow for direct memory access, such as, for example, a network device. In various embodiments, the embedded processor **250** may be resident in a platform control hub, a system on a chip, a multi-chip package, and/or a chipset that manages I/O interfaces for the host processor.

[0019] In various embodiments, the embedded processor **250** may be associated with embedded memory **225** which may not be accessible to the host processor **200**. Additionally, in various embodiments, a system memory **240** may be accessible by both the host processor **200** and the embedded processor **250**. In various embodiments, the embedded processor **250** may be configured to access system memory **240** without detection of the host processor **200** or host OS **210**, such as through the chipset hardware routing and direct memory access logic **230**.

[0020] In various embodiments, each of the host processor **200** and the embedded processor **250** may execute operating systems in software (e.g. the host OS **210** and the embedded OS **260**, respectively). In various embodiments, the host OS **210** may include one or more of components that may be loaded during a boot of the host OS **210**, such as the BIOS **130**, the boot manager **140** and/or the OS drivers **160**.

[0021] In various embodiments, anti-virus software, such as host anti-virus software **215** and/or embedded anti-virus software **265**, may execute on one of the host OS **210** or embedded OS **260**. In various embodiments, the BSVS **100** may, after detection of a compromised host OS component, interact with one of the host anti-virus software **215** and/or embedded anti-virus software **265** to clean or quarantine the component. In various embodiments, the embedded anti-virus software **265** may be utilized for this cleaning purpose if it is believed that the host OS **210** (and therefore the host anti-virus software **215**) is too compromised to trust that the cleaning/quarantining can be effective.

[0022] In various embodiments, both the host processor and the embedded processor may have one or more drivers to allow for communication outside of the processor. For example, the host processor may execute to host memory driver **220** to provide for communication with system memory **240**. In various embodiments, however, the host memory driver **220** may be compromised by a virus **228**. As described above, the virus **228** may, through compromising the host memory driver **220**, intercept and control accesses to the system memory **240**. This may prevent software, such as the host anti-virus software **215**, from properly recognizing the presence of a virus in the system memory. Thus, if the boot manager **140** is compromised with the virus, the host anti-virus software **215** may be unable to detect the virus and may not identify the compromised nature of the boot manager **140**. In various embodiments, the host processor **200** and embedded processor **250** may also execute communication drivers for communication with each other. For example, the host processor **200** may execute an embedded communication driver **225** and the embedded processor **250** may execute a host communication driver **275**. In various embodiments, communications between the host processor **200** and embedded processor **250** may be facilitated through chipset hardware routing and direct memory access logic **230**.

[0023] In some embodiments, the embedded processor **250** may access one or more components in the system memory **240** using the embedded signature verifier **270**. In various embodiments, the embedded signature verifier **270** may communicate with an hardware-based embedded signature generation engine **280** accessible by the embedded processor **250**. In various embodiments, the embedded signature verifier **270** may be configured to execute to cause the embedded signature generation engine **280** to access one or more host OS components in the system memory **240**. In some embodiments, the embedded signature generation engine **280** may be configured to access a memory footprint of one or more components. In various embodiments, the embedded signature generation engine **280** may be configured to access the memory out-of-band, as described herein.

[0024] The embedded signature verifier **270** may be further configured to cause the embedded signature generation engine **280** to generate a signature from the accessed components and to return that generated signature to the embedded signature verifier **270**. In various embodiments, the embedded signature generation engine **280** may be configured to perform one or more cryptographic hashes to generate the signature such as, but not limited to, for example, the SHA-

256 hash function. The embedded signature verifier **270** may then compare the generated signatures to trusted signatures for the components in order to determine if the components may have been compromised.

[0025] FIG. **3** illustrates an example out-of-band host OS boot verification process **300** of the host OS boot sequence verification system, in accordance with various embodiments. It may be recognized that, while the operations of process **300** are arranged in a particular order and illustrated once each, in various embodiments, one or more of the operations may be repeated, omitted, or performed out of order. The process may begin at operation **310**, where the host OS **210** may begin a boot process. Next, at operation **320**, the embedded signature verifier **270** of the embedded processor **250** may compare one or more signatures from host OS components to trusted signatures. Particular embodiments of operation **320** are described below with reference to process **400** of FIG. **4**. Next, at operation **330**, if an alert is received based on the comparing, the OS boot process may be stopped. The process may then end.

[0026] FIG. **4** illustrates an example out-of-band host OS component verification process of the out-of-band host OS boot sequence verification system, in accordance with various embodiments. In various embodiments, process **400** may include one or more embodiments of operation **320** of process **300**. It may be recognized that, while the operations of process **400** are arranged in a particular order and illustrated once each, in various embodiments, one or more of the operations may be repeated, omitted, or performed out of order. The process may begin at operation **410**, where the embedded signature verifier **270** may send an instruction to generate a signature to the embedded signature generation engine **280**. In various embodiments, the instruction may include an indication of a component for which the embedded signature generation engine **280** should generate a signature.

[0027] Next, at operation **420**, the embedded signature generation engine **280** may obtain a memory footprint for the host OS component for which it will generate the signature. Then at operation **430**, the embedded signature generation engine **280** may generate a signature from the memory footprint of the component to the embedded signature verifier **270**. At operation **430** the embedded signature generation engine **280** may also provide the generated signature to the embedded signature verifier **270**. In various embodiments, the embedded signature generation engine **280** may generate the signature by performing a cryptographic hash function on all or part of the memory footprint.

[0028] Next, at operation **440**, the embedded signature verifier **270** may obtain a trusted signature. As discussed herein, in various embodiments, the embedded signature verifier **270** may obtain the trusted signature as a pre-stored trusted signature. For example, in some embodiments, the embedded signature verifier **270** may obtain the trusted signature from a trusted signature stored on the computing device before any boot of the host OS, such as in the embedded memory **255**. Thus, in some such embodiments, the computing device may be pre-loaded with one or more trusted signatures for comparison, each of which is assumed to represent the memory footprint of a host OS component in an uncompromised state. By checking against these known trusted signatures, the embedded signature verifier **270** may be able to verify the integrity of the host OS components during boot (or at another time).

[0029] In other embodiments, the embedded signature verifier **270** may obtain the trusted signature from a signature previously stored on the device by the embedded signature verifier **270** during operation of the computing device. Thus, in some embodiments, the embedded signature verifier **270** may utilize previously-generated signatures to compare against, allowing the embedded signature verifier **270** to confirm that host OS components have not been modified since a previous point in time when the previously-generated signatures were generated. Particular examples of these embodiments are described below. In other implementations, the obtained trusted signatures may be obtained from other sources, such as from other computing devices. In some such embodiments, the trusted signatures may be sent to the computing device before or along with an instruction to perform a host OS boot sequence verification as described herein.

[0030] Next, at decision operation **445**, the embedded signature verifier **270** may determine if the signature generated at operation **430** matches the signature obtained at operation **440**. If not, then at operation **450** the embedded signature verifier **270** may generate an alert, such as alert **170**. In some embodiments, release of the alert may lead to cessation of a host OS boot process, as described above. If, however, the signatures match, then if additional components may be verified, the process may repeat at operation **410**. If, however, no additional host OS components may be verified, then the process may end.

[0031] FIG. **5** illustrates an example time-based out-of-band host OS boot sequence verification process **500** of the host OS boot sequence verification system, in accordance with various embodiments. It may be recognized that, while the operations of process **500** are arranged in a particular order and illustrated once each, in various embodiments, one or more of the operations may be repeated, omitted, or performed out of order. It may be recognized that aspects of process **500** may be similar to aspects of process **300** of FIG. **3**. However, in various embodiments, process **500** may be performed to verify one or more host OS components against prior-generated and -stored host OS component signatures. By comparing signatures against previously-generated signatures, the embedded signature verifier **270** may be able to determine if any of the host OS components have been modified and/or compromised since the last signature generation.

[0032] The process may begin at operation **510**, where the host OS **210** may begin a boot process. Next, at operation **520**, the embedded signature verifier **270** may obtain stored previously-generated signatures. In various embodiments, these signatures may be obtained from signatures stored by the embedded processor **250**, such as in the embedded processor memory **255**. In other embodiments, the signatures may be stored elsewhere, such as on a storage device.

[0033] Next, at operation **530**, the embedded signature verifier **270** of the embedded processor **250** may compare one or more signatures from host OS components to trusted signatures. Particular embodiments of operation **530** are described above with reference to process **400** of FIG. **4**. Then at operation **540**, the embedded signature verifier **270** may store the generated signatures for use in later verification of host OS components. Next, at operation **550**, if an alert is received based on the comparing, the OS boot process may be stopped. The process may then end.

[0034] FIG. **6** illustrates an example out-of-band on-demand OS boot sequence verification process **600** of the out-of-band host OS boot sequence verification system, in accordance with various embodiments. It may be recognized that, while the operations of process **600** are arranged in a particular order and illustrated once each, in various embodiments, one or more of the operations may be repeated, omitted, or performed out of order. It may be recognized that aspects of process **600** may be similar to aspects of process **300** of FIG. **3**. However, in various embodiments, process **600** may be performed to perform verification of host OS components on demand, rather than only at boot time. The process may begin at operation **610**, where the embedded signature verifier **270** may receive a command to verify host OS components. In various embodiments, the command may be received from a trusted agent operating in the host OS such as, for example, the host OS anti-virus software **215**. In other embodiments, a user, either located locally or remotely from the computing device, may initiate the command. Thus, in some embodiments, the command may be received over a network. In various embodiments the embedded processor **250** may be configured to receive one or more network commands to perform the host OS boot sequence verification techniques described herein. In various embodiments, the embedded processor **250** may be configured to receive the one or more network commands out-of-band of the host processor **200**.

[0035] Next, at operation **620**, the embedded signature verifier **270** of the embedded processor **250** may compare one or more signatures from host OS components to trusted signatures. Particular embodiments of operation **620** are described above with reference to process **400** of FIG. **4**. Next, at operation **630**, if an alert is received based on the comparing, the alert may be sent to the initiator of the command received at operation **610**. Thus, in some embodiments, the alert may be sent to a trusted agent operating in the host OS. In other embodiments, an alert message may be sent over a network to a user initiating the command. The process may then end.

[0036] FIG. **7** illustrates, for one embodiment, an example computer system **700** suitable for practicing embodiments of the present disclosure. As illustrated, example computer system **700** may include control logic **708** coupled to at least one of the processor(s) **704**, system memory **712** coupled to system control logic **708**, non-volatile memory (NVM)/storage **716** coupled to system control logic **708**, and one or more communications interface(s) **720** coupled to system control logic **708**. In various embodiments, the one or more processors **704** may be a processor core.

[0037] System control logic **708** for one embodiment may include any suitable interface controllers to provide for any suitable interface to at least one of the processor(s) **704** and/or to any suitable device or component in communication with system control logic **708**. System control logic **708** may also interoperate with a display **706** for display of information, such as to as user. In various embodiments, the display may include one of various display formats and forms, such as, for example, liquid-crystal displays, cathode-ray tube displays, and e-ink displays. In various embodiments, the display may include a touch screen.

[0038] System control logic **708** for one embodiment may include one or more memory controller(s) to provide an interface to system memory **712**. System memory **712** may be used to load and store data and/or instructions, for example, for system **700**. In one embodiment, system memory **712** may include any suitable volatile memory, such as suitable dynamic random access memory ("DRAM"), for example.

[0039] System control logic **708**, in one embodiment, may include one or more input/output ("I/O") controller(s) to provide an interface to NVM/storage **716** and communications interface(s) **720**.

[0040] NVM/storage **716** may be used to store data and/or instructions, for example. NVM/storage **716** may include any suitable non-volatile memory, such as flash memory, for example, and/or may include any suitable non-volatile storage device(s), such as one or more hard disk drive(s) ("HDD (s)"), one or more solid-state drive(s), one or more compact disc ("CD") drive(s), and/or one or more digital versatile disc ("DVD") drive(s), for example.

[0041] The NVM/storage **716** may include a storage resource physically part of a device on which the system **700** is installed or it may be accessible by, but not necessarily a part of, the device. For example, the NVM/storage **716** may be accessed over a network via the communications interface (s) **720**.

[0042] System memory **712**, NVM/storage **716**, and system control logic **708** may include, in particular, temporal and persistent copies of host OS boot sequence verification logic **724**. The host OS boot sequence verification logic **724** may include instructions that when executed by at least one of the processor(s) **704** result in the system **700** practicing one or more of the protected memory management operations described above.

[0043] Communications interface(s) **720** may provide an interface for system **700** to communicate over one or more network(s) and/or with any other suitable device. Communications interface(s) **720** may include any suitable hardware and/or firmware, such as a network adapter, one or more antennas, a wireless interface **722**, and so forth. In various embodiments, communication interface(s) **720** may include an interface for system **700** to use NFC, optical communications (e.g., barcodes), BlueTooth or other similar technologies to communicate directly (e.g., without an intermediary) with another device. In various embodiments, the wireless interface **722** may interoperate with radio communications technologies such as, for example, WCDMA, GSM, LTE, and the like.

[0044] For one embodiment, at least one of the processor(s) **704** may be packaged together with system control logic **708** and/or host OS boot sequence verification logic **724**. For one embodiment, at least one of the processor(s) **704** may be packaged together with system control logic **708** and/or host OS boot sequence verification logic **724** to form a System in Package ("SiP"). For one embodiment, at least one of the processor(s) **704** may be integrated on the same die with system control logic **708** and/or host OS boot sequence verification logic **724**. For one embodiment, at least one of the processor(s) **704** may be integrated on the same die with system control logic **708** and/or host OS boot sequence verification logic **724** to form a System on Chip ("SoC").

[0045] Computer-readable media (including non-transitory computer-readable media), methods, systems and devices for performing the above-described techniques are illustrative examples of embodiments disclosed herein. Additionally, other devices in the above-described interactions may be configured to perform various disclosed techniques.

[0046] Although certain embodiments have been illustrated and described herein for purposes of description, a wide variety of alternate and/or equivalent embodiments or implementations calculated to achieve the same purposes may be substituted for the embodiments shown and described without departing from the scope of the present disclosure. This application is intended to cover any adaptations or variations of the embodiments discussed herein. Therefore, it is manifestly intended that embodiments described herein be limited only by the claims.

[0047] Where the disclosure recites "a" or "a first" element or the equivalent thereof, such disclosure includes one or more such elements, neither requiring nor excluding two or more such elements. Further, ordinal indicators (e.g., first, second or third) for identified elements are used to distinguish between the elements, and do not indicate or imply a required or limited number of such elements, nor do they indicate a particular position or order of such elements unless otherwise specifically stated.

1.-**30**. (canceled)

31. A system for verifying a host operating system, the system comprising:
- computer system memory;
- a first computer processor, separate but coupled with the computer system memory to operate out-of-band of the host operating system, which is to be executed on a second computer processor, wherein the first computer processor is an embedded processor co-located with the second computer processor on a computing platform, and the first computer processor is to:
  - access one or more components of the host operating system in the computer system memory; and
  - verify the one or more components.

32. The system of claim **31**, wherein the first computer processor is to access the computer system memory without detection by the second computer processor.

33. The system of claim **31**, wherein the first computer processor is to access the one or more components during a boot of the host operating system.

34. The system of claim **31**, wherein the first computer processor comprises a signature generation engine to access the one or more components and generate one or more signatures for the accessed one or more components.

35. The system of claim **34**, wherein the signature generation engine executes in hardware.

36. The system of claim **34**, wherein the first computer processor is to execute a signature verifier in software to compare the one or more generated signatures to one or more trusted signatures.

37. The system of claim **36**, wherein the one or more trusted signatures comprise first one or more trusted signatures, and the first computer processor is to store the one or more generated signatures as second one or more trusted signatures.

38. The system of claim **37**, wherein the first computer processor is further, during a subsequent boot of the host operating system, to repeat the access, generate, and compare with respect to the second one or more trusted signatures.

39. The system of claim **31**, wherein the first computer processor is further to receive a command to perform the verify.

40. The system of claim **39**, wherein the first computer processor is to receive the command from a trusted agent executing in the host operating system.

41. The system of claim **39**, wherein the first computer processor is to receive the command via a network connection.

42. The system of claim **31**, wherein the first computer processor is further to generate an alert if one or more of the components is not successfully verified.

43. The system of claim **31**, further comprising the second computer processor.

44. A computer-implemented method for verifying a host operating system operating on a first computer processor, the method comprising:
- accessing, by a second computer processor, one or more components of the host operating system in a computer system memory, separate from at least the second computer processor, without detection by the host operating system; and
- verifying, by the second computer processor, the one or more components, wherein the second computer processor is an embedded processor co-located with the first computer processor on a computing platform.

45. The method of claim **44**, wherein verifying the one or more components includes generating one or more signatures by a signature generation engine of the second computer processor.

46. The method of claim **44**, wherein accessing the one or more components includes accessing the one or more components during a boot of the host operating system.

47. The method of claim **44**, wherein verifying the one or more components is based at least in part on one or more trusted signatures known to the second computer processor prior to a first boot of the host operating system on the first computer processor.

48. The method of claim **47**, wherein:
- the one or more trusted signatures include first one or more trusted signatures, and
- the method further comprises generating one or more signatures and storing the one or more generated signatures as second one or more trusted signatures.

49. The method of claim **48**, wherein the method further comprise repeating the verifying, during a subsequent boot of the host operating system, with respect to the second one or more trusted signatures.

50. The method of claim **44**, further comprising receiving a command to perform the verifying.

51. The method of claim **44**, further comprising generating an alert if one or more of the one or more components is not successfully verified.

52. An embedded processor circuitry to verify a host operating system operating on a host central processing unit executing a host operating system, the embedded processor circuitry to operate to:
- access one or more components of the host operating system in a computer system memory, separate from the embedded processor circuitry, without detection by any processes executing on the host central processing unit; and
- verify the one or more components,
- wherein the embedded processor circuitry is to be co-located with the host central processing unit on a computing platform.

53. The embedded processor circuitry of claim **52**, wherein the embedded processor circuitry is to access the one or more components during a boot of the host operating system.

54. The embedded processor circuitry of claim **52**, wherein the embedded processor circuitry is to verify the one or more components based at least in part on one or more trusted signatures known to the embedded processor circuitry prior to a first boot of the host operating system.

55. The embedded processor circuitry of claim **54**, wherein:

the one or more trusted signatures comprise first one or more trusted signatures, and

the embedded processor circuitry is further to generate one or more signatures and store the one or more generated signatures as second one or more trusted signatures.

56. The embedded processor circuitry of claim **52**, wherein the embedded processor circuitry is further to receive a command to perform the verify.

57. The embedded processor circuitry of claim **52**, wherein the embedded processor circuitry is further to generate an alert if one or more of the one or more components is not successfully verified.

* * * * *