

SPECIFICATION

TITLE OF INVENTION: APPARATUS AND METHOD FOR HARDWARE-BASED DETECTION OF KERNEL CODE INJECTION ATTACK

TECHNICAL FIELD

[0001] The inventive concept relates to an apparatus and method for detecting an attempt to inject malicious code into a kernel code, and more particularly, to an apparatus embodied as hardware independent from a target system to detect an attempt to write to a kernel code region of the target system or detect an execution attempt occurring outside the kernel code while the target system is operated in a kernel mode, and a method thereof.

BACKGROUND ART

[0002] Many security specialists have tried to make security monitors be independent from a host system so as to protect the integrity of an operating system (OS) kernel. Efforts to monitor the integrity of a kernel may be largely classified into two access methods. One of these methods is a hardware-based access method and the other is a hypervisor-based access method.

[0003] The hypervisor-based method has recently become more popular but software vulnerability thereof may increasingly heighten as hypervisor becomes more complicated. Although an additional software layer may be inserted to protect the integrity of the hypervisor, the additional software layer may also cause new vulnerabilities and thus does not provide a fundamental solution thereto.

[0004] Most existing solutions for kernel-integrity monitoring have been developed to detect a malicious write attempt to a memory employed by the kernel. However, defending against attempts to inject malicious code into the kernel is also just as important. Many attacks on the kernel are made to cause code designed by an attacker to be operated as a part of the kernel, because code injection is an easier way of making such an attack. As methods of detecting code injection, the hypervisor-based access method has been employed and research has been conducted on a technique of detecting code injection using a specific function of a specific processor. However, such access methods are disadvantageous in terms of performance or have the security problem described above.

DETAILED DESCRIPTION OF THE INVENTION

TECHNICAL PROBLEM

[0005] The inventive concept provides an apparatus and method for hardware-based detection of an attempt to inject malicious code into kernel code, which are capable of detecting an attempt to write to a kernel code region and an execution attempt occurring outside the kernel code region while a system is operated in a kernel mode.

TECHNICAL SOLUTION

[0006] According to an aspect of the present invention, an apparatus for hardware-based detection of a kernel code injection attack includes a trace manager configured to receive a packet containing runtime information from a program trace interface (PTI) of a host system and to parse the packet to detect indirect branch target addresses; a notification manager configured to receive a notification about an event through execution of a trampoline when the event occurs in a state in which a code block related to the event among code blocks of a kernel is replaced with a jump command on the trampoline while the host system is booted, wherein the event includes a change in an address translation base register (ATBR), mode conversion, or a change in a PTI control register; and a traffic manager configured to detect an attempt to write to a physical code region of a memory by snooping traffic on a bus.

[0007] According to another aspect of the present invention, a method of hardware-based detection of a kernel code injection attack includes receiving a packet containing runtime information from a program trace interface (PTI) of a host system and parsing the packet to detect indirect branch target addresses; receiving a notification about an event through execution of a trampoline when the event occurs in a state in which a code block related to the event among code blocks of a kernel is replaced with a jump command on the trampoline while the host system is booted, wherein the event includes a change in an address translation base register (ATBR), mode conversion, or a change in a PTI control register; and detecting an attempt to write to a physical code region of a memory by snooping traffic on a bus.

ADVANTAGEOUS EFFECTS

[0008] According to the present invention, an attack on a kernel may be detected without receiving assistance of a software layer assigned higher authority than that of an operating system (OS) kernel of a target system and without newly designing the architecture of existing hardware.

DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 is a diagram illustrating a structure of an apparatus for hardware-based detection of a kernel code injection attack, according to an embodiment of the present invention,

[0010] FIG. 2 is a diagram illustrating an example of a mode-conversion tracing method according to the present invention,

[0011] FIG. 3 is a diagram illustrating a method of notifying about an event such as a change in an address translation base register (ATBR), according to an embodiment of the present invention,

[0012] FIG. 4 is a diagram illustrating one example of trampoline code according to the present invention,

[0013] FIG. 5 is a diagram illustrating a method of detecting against a replay attack by using an apparatus for hardware-based detection of a kernel code injection attack, according to an embodiment of the present invention,

[0014] FIG. 6 is a diagram illustrating an example of interdependencies between elements of an apparatus for hardware-based detection of a kernel code injection attack, according to the present invention,

[0015] FIG. 7 is a diagram illustrating a method of notifying about an event, according to another embodiment of the present invention,

[0016] FIG. 8 is a diagram illustrating a structure of an apparatus for hardware-based detection of a kernel code injection attack, according to another embodiment of the present invention, and

[0017] FIG. 9 is a flowchart of a method of hardware-based detection of a kernel code injection attack, according to an embodiment of the present invention.

MODE OF THE INVENTION

[0018] Hereinafter, an apparatus and method for hardware-based detection of a kernel code injection attack, according to the present invention, will be described in detail with reference to the accompanying drawings.

[0019] FIG. 1 is a diagram illustrating a structure of an apparatus for hardware-based detection of a kernel code injection attack, according to an embodiment of the present invention.

[0020] In the present embodiment, it is assumed that correct values of an address translation base register (ATBR) of a host system 135 have been known beforehand to an apparatus 100 for hardware-based detection of a kernel code injection attack (hereinafter referred to as 'attack detecting apparatus'). The ATBR includes a primary address of a first-level page table (or a page global directory). The host system 135 is a type of computing device which includes a processor 140, a memory 150, a peripheral device 160, etc. The host system 135 is a target into which an attacker tries to inject kernel code to attack the host system 135.

[0021] It is assumed that in kernel page tables of the host system 135, a virtual code region and a physical code region are exactly mapped to each other during initial booting. Here, the virtual code region means a virtual memory region storing kernel code and the physical code region means a physical memory region storing the kernel code.

[0022] When the host system 135 is booted, the attack detecting apparatus 100 detects the locations of a code block of a kernel which causes a mode switch between a privileged mode and a user mode, a code block to be performed by a processor when the user mode is switched to the privileged mode (i.e., a gateway code block), a code block related to a change in the ATBR, a code block which changes an address of the gateway code block, a code block which reconstructs a program trace interface (PTI), etc. The attack detecting apparatus 100 replaces each of the detected code blocks with a jump command for appropriate code snippets (hereinafter referred to as 'trampolines'). Here, a trampoline corresponding to each of the code blocks has a function of performing an original code block corresponding thereto and notifying the attack detecting apparatus 100 of an event, such as mode conversion, an ATBR address change, or a change in a PTI control register, caused by the performing of the original code block. An example of trampoline code for notifying about an event is illustrated in FIG. 4.

[0023] The location of each of the code blocks may be detected according to various methods according to the related art. For example, when the host system 135 is switched from the user mode to the privileged mode, the processor 140 detects a designated address indicating the gateway code block. Most processors

each include a special register to calculate the designated address. Thus, when the host system 135 is booted, the attack detecting apparatus 100 may detect the location of the gateway code block by using values of the special register.

[0024] Referring to FIG. 1, the attack detecting apparatus 100 is connected to hardware elements of the host system 135, and includes a trace manager 110, a notification manager 120, and a traffic manager 130.

[0025] The trace manager 110 is connected to a PTI 142 of the processor 140 and receives a data packet containing various runtime information from the PTI 142. Here, the PTI 142 is an external interface included in the processor 140 for the purpose of debugging, etc. and is also referred to as a debugging interface. The PTI 142 may be called various names according to the type of the processor 140. For example, the PTI 142 is called a 'processor tracer' when the processor 140 is an 'Intel x86 processor', and is called a 'program trace macrocell (PTM)' when the processor 140 is an 'ARM processor'. Thus, in the present embodiment, the PTI 142 is defined to represent various types of interfaces capable of outputting various information stored in the processor 140 to the outside.

[0026] The trace manager 110 examines indirect branch target addresses by parsing a received packet to determine whether injected malicious code is executed in the privileged mode. As will be described below, when an event such as a change in the ATBR or mode conversion occurs, the notification manager 120 receives a notification about the event, such as a change in the ATBR or mode conversion, through a trampoline. Thus, the trace manager 110 may detect an attacker's attempt to compulsorily convert the processor 140 into the privileged mode outside a virtual code region on the basis of the indirect branch target addresses detected through the PTI 142 and the event, such as a change in the ATBR change or mode conversion, detected through the notification manager 120.

[0027] For example, during initial booting of the host system 135, if a list of indirect branch target addresses and a mode related to each address included in this list have been known to the trace manager 110, the trace manager 110 detects whether an attack occurs by determining whether the mode related to the indirect branch target addresses detected through the PTI 142 is correct on the basis of the mode notified through the notification manager 120, and generates an alarm when it is determined that an attack has occurred.

[0028] The notification manager 120 is connected to a peripheral-device bus 180

of the host system 135, and includes a keying register 122, a notification register 124, an acknowledgement (ACK) register 126, etc. The notification manager 120 may have two interconnections 192 and 194 with the peripheral-device bus 180. Based on the first interconnection 194, the processor 140 notifies the notification manager 120 of an event, such as a change in the ATBR, mode conversion, or a change in the PTI control register, during execution of the trampoline. Based on the second interconnection 192, the notification manager 120 accesses the main memory 150 of the host system 135 and analyzes contents of the memory 150 to verify the notification received from the processor 140.

[0029] More specifically, when a jump command regarding a trampoline is input to the processor 140 while a kernel command is executed, the processor 140 branches to the trampoline and notifies the notification manager 120 of an event, such as mode conversion, an ATBR address change, or a change in the PTI control register, through execution of the trampoline. Methods of receiving a notification about an event through each of the registers 122, 124, and 126, performed by the notification manager 120, according to various embodiments are illustrated in FIGS. 3 and 7. For example, the notification manager 120 may be notified about an ATBR address change or the like by using a protocol of FIG. 3, and the notification manager 120 may be notified about mode conversion or the like by notified using a protocol of FIG. 7.

[0030] The traffic manager 130 is connected to a memory bus 170 and the peripheral-device bus 180 of the host system 135 (see interconnections 196, 198, and 199), and monitors a write operation performed on the main memory 150, a write operation performed on a control register of the PTI 142, etc. by analyzing traffic of the buses 170 and 180. More specifically, the traffic manager 130 detects a write operation performed on the main memory 150 by the processor 140 or the peripheral device 160 by snooping traffic on the buses 170 and 180 connected to the main memory 150 to detect a first-level page table (a page global directory) indicated by the ATBR or the like or detect all accesses for controlling the PTI 142 through a memory-mapped control register by snooping traffic on the peripheral-device bus 180.

[0031] For example, it is assumed that an attacker tries to inject code into a physical code region. The traffic manager 130 snoops all write traffic related to the main memory 150 on the basis of the interconnections 198 and 199 with the

peripheral-device bus 180 and generates an alarm when a write attempt occurs in the physical code region. The traffic manager 130 continuously snoops write traffic from the processor 140 or the peripheral device 160 and may thus detect a write attempt or the like through direct memory access.

[0032] As another example, the traffic manager 130 may have an address of an invariable kernel region beforehand, snoop traffic on the buses 170 and 180, and generate an alarm when a write attempt to the invariable kernel region is detected. Here, the invariable kernel region indicates a kernel code region, a system call table, an interrupt descriptor table (IDT), or the like. The address of the invariable kernel region may be detected according to various methods when the host system 135 is booted.

[0033] As another example, when a change in the ATBR is detected by the notification manager 120, the traffic manager 130 starts to detect a first-level page table (or a page global directory) which is being currently used on the basis of a changed ATBR and monitors it by snooping a packet corresponding to this table rather than snooping packets regarding page tables of all ATBRs.

[0034] FIG. 2 is a diagram illustrating an example of a mode-conversion tracing method according to the present invention.

[0035] Referring to FIGS. 1 and 2 together, the PTI 142 of the processor 140 may sequentially provide direct or indirect branch target addresses to the trace manager 110, and the trace manager 110 may detect information regarding these addresses by receiving them from the PTI 142. During mode conversion, the processor 140 branches to a trampoline and then executes the trampoline. Thus the notification manager 120 may receive a notification about the mode conversion from the processor 140.

[0036] For example, a mode associated with a branch target address 200 at a point of time (a) is the user mode and a mode associated with a subsequent branch target address is the privileged mode. That is, mode conversion occurs. Thus, the processor 140 notifies the notification manager 120 of the mode conversion from the user mode to the privileged mode through execution of the trampoline.

[0037] As another example, a mode associated with a branch target address 210 at a point of time (b) is the privileged mode and a mode associated with a subsequent branch target address is the user mode. That is, mode conversion occurs. Thus, the processor 140 notifies the notification manager 120 of the mode

conversion from the privileged mode to the user mode through execution of the trampoline in the privileged mode.

[0038] FIG. 3 is a diagram illustrating a method of notifying about an event such as a change in an ATBR according to an embodiment of the present invention.

[0039] Referring to FIGS. 1 and 3 together, when the host system 135 branches to a trampoline as a change in an ATBR, mode conversion, or the like occurs, the host system 135 notifies the attack detecting apparatus 100 of a start of a trampoline through the PTI 142 (operation S300). When recognizing the start of the trampoline notified by the host system 135 through the PTI 142, the attack detecting apparatus 100 generates a key (a value other than '0') and stores it in the keying register 122 and the host system 135 obtains the key by reading the keying register 122 through execution of the trampoline (operation S310). In order to prevent the key from being read two or more times, the keying register 122 resets the keying register 122 when the host system 135 reads the keying register 122 once.

[0040] When the key is obtained from the keying register 122, the host system 135 records the content of an event (e.g., mode conversion, a change in the ATBR, a change in the PTI control register, etc.) on the notification register 124 of the attack detecting apparatus 100 (operation S330). The attack detecting apparatus 100 may read the content of the event from the notification register 124 to determine the event occurring in the host system 135.

[0041] To block an attacker's bypass attack or replay attack, the host system 135 may transmit the key, which is previously obtained, together with the content of the event (operation S320). That is, the host system 135 sequentially transmits the key and the content of the event to the attack detecting apparatus 100 (operations S320 and S330).

[0042] The attack detecting apparatus 100 reads the content of a notification newly stored in the notification register 124, and accesses the main memory 150 or the like and verifies the content of the notification when the content of the notification needs to be verified. For example, when the notification is related to a change in the ATBR, the attack detecting apparatus 100 determines whether the change in the ATBR has a malicious purpose or not on the basis of content stored in the main memory 150 indicated by a new ATBR. When the notification is related to a change in the PTI control register, the attack detecting apparatus 100 reads the PTI control register stored in the main memory 150 and compares it with a PTI control register

obtained when the host system 135 is booted to verify whether the stored PTI control register is to be changed. When the notification is related to mode conversion, verification thereof may not be needed. After the content of the notification is verified, the attack detecting apparatus 100 records a value indicating an acknowledgement (ACK) of the verification on the ACK register 126 and a session notifying about the event is ended when the host system 135 reads this ACK from the ACK register 126 (operation S340).

[0043] FIG. 4 is a diagram illustrating an example of a trampoline for notifying about an address change event according to the present invention.

[0044] FIG. 4 illustrates an example of a trampoline for notifying an attack detecting apparatus about an event which changes a specific register such as a vector base address register (VBAR) of an ARM system. A value of the VBAR is used to determine the location of a gateway code block in the ARM system.

[0045] In the trampoline, first, an exception or an interrupt is changed to be in an inactivate state (see line 2). The trampoline does not include an indirect branch command and includes two direct branches to form an internal loop to wait for a key and an ACK from the attack detecting apparatus 100. Thus, any other commands may be prevented from being executed between the execution of a command on line 10 of the trampoline and the execution of a command on line 11 of the trampoline.

[0046] The trace manager 110 of the attack detecting apparatus 100 may detect an attack by comparing a branch target address (jump target address) (see line 11) received from the PTI 142 with a list of correct branch target addresses (i.e., starting addresses of trampolines, see line 1) generated when the attack detecting apparatus 100 is started.

[0047] FIG. 5 is a diagram illustrating a method of detecting against a replay attack by using an apparatus for hardware-based detection of a kernel code injection attack according to an embodiment of the present invention.

[0048] Referring to FIGS. 1 and 5, when the processor 140 jumps to a trampoline to start notifying the attack detecting apparatus 100 of an event such as mode conversion, an attacker may try to notify the attack detecting apparatus 100 of a false event (operation S500) between starting a trampoline (operation S510) and notifying about a true event (operation S540).

[0049] In this case, the attack detecting apparatus 100 may differentiate between the true event notified in operation S540 and the false event notified in operation

S500 on the basis of whether a key is transmitted when a notification about an event is notified after the starting of the trampoline (operation S510). When a session is ended by receiving a message notifying the true event and performing acknowledgement thereof (operation S550), the attacker cannot record a message notifying the false event on the notification register 124.

[0050] FIG. 6 is a diagram illustrating an example of interdependencies between elements of an apparatus for hardware-based detection of a kernel code injection attack according to the present invention.

[0051] Referring to FIG. 6, each of most elements of an attack detecting apparatus depends on the states of the other elements or a state of a host system. However, two elements of the attack detecting apparatus, i.e., a write monitoring unit 600 and a secure boot unit 610, have no dependencies on the other elements.

[0052] As indicated by dotted lines, the secure boot unit 610 guarantees an injection-free state and an address translation redirection attack (ATRA)-free state of the host system at an initial stage. Here, solid lines represent matters which an attacker should overcome to invalidate a specific element or a specific state of the attack detecting apparatus. For example, unless the attacker successfully performs an ATRA by changing a page table of an ATBR or a kernel, a target system is in the ATRA-free state. In other words, the attacker should invalidate ATBR tracking to change the ATBR or avoid monitoring a write operation to change the contents of the page table, and thus the attack detecting apparatus may easily detect the attacker's attack.

[0053] FIG. 7 is a diagram illustrating a method of notifying about an event such as mode conversion according to another embodiment of the present invention.

[0054] Referring to FIG. 7, as in FIG. 3, when a host system 135 branches to a trampoline due to mode conversion or the like, the host system 135 starts notification of an event while notifying the attack detecting apparatus 100 of start of the trampoline (operation S710). However, the host system 135 does not obtain a key during execution of the trampoline and notifies about the event by using a key obtained during execution of a previous trampoline in operation S700 (operations S720 and S730). In other words, the host system 135 uses the key, which was obtained during the execution of the previous trampoline, as a key for notifying about the event at a current stage and transmits the key to the attack detecting apparatus (operation S720).

[0055] When the host system 135 records the event such as mode conversion on a notification register, together with the key (operations S720 and S730), the host system 135 receives beforehand a key to be used to notify a subsequent event (operation S740) and ends a session notifying the event without performing an additional ACK process (operation S750).

[0056] FIG. 8 is a diagram illustrating a structure of an apparatus for hardware-based detection of a kernel code injection attack according to another embodiment of the present invention.

[0057] Referring to FIG. 8, the attack detecting apparatus includes a traffic manager 800, a notification manager 810, a trace manager 830, and a system manager 850. The traffic manager 800 includes a snoopers 802, a first verification unit 804, and a first alarm generator 806. The notification manager 810 includes a notification transmitter 812, a keying register 814, an ACK register 816, a notification register 818, a notification management finite-state machine (FSM) 820, a key generator 822, a second alarm generator 824, a memory verification unit 826, and a memory access unit 828. The trace manager 830 includes a trace receiver 832, a trace analyzer 834, a trace verification unit 836, a trace management FSM 838, a temporary alarm storage unit 840, and a third alarm generator 842.

[0058] First, the system manager 850 includes a processor and a memory thereof and initializes and controls the traffic manager 800, the notification manager 810, the trace manager 830, etc. The system manager 850 checks the alarm generators 806, 824, and 842 of the traffic manager 800, the notification manager 810, and the trace manager 830 to determine whether a kernel code injection attack occurs. When one of the alarm generators 806, 824, and 842 generates an alarm, the system manager 850 informs a system manager of the generation of the alarm.

[0059] Next, the traffic manager 800 will be described below.

[0060] The snoopers 802 snoops traffic on a memory bus and a peripheral-device bus and transmits a result of snooping the traffic to the verification unit 804 as described above with reference to FIG. 1. The verification unit 804 determines whether the result of snooping the traffic received from the snoopers 802 represents a sign of a code injection attack such as kernel code alteration and transmits a result of the determination to the alarm generator 806. The verification unit 804 may determine, as a sign of an attack, a write attempt to the physical code regions and a page table and a memory-mapped register for mapping a virtual code region. The

alarm generator 806 generates and outputs an alarm when the verification unit 804 determines the result of snooping the traffic as a sign of a code injection attack.

[0061] Next, the trace manager 830 will be described below.

[0062] The trace receiver 832 receives a data packet from a PTI of a host system. The trace analyzer 834 processes the data packet to be meaningful information and transmits it to the trace verification unit 836. For example, the trace analyzer 834 parses a packet received via the trace receiver 832 to detect indirect branch target addresses.

[0063] The trace verification unit 836 determines whether there is an attempt to make a kernel code injection attack on the basis of information received from the trace analyzer 834, and transmits a result of the determination to the temporary alarm storage unit 840 or the alarm generator 842 according to a state of the host system.

[0064] For example, the trace verification unit 836 removes indirect branch addresses generated in the user mode from among the indirect branch target addresses detected by the trace analyzer 834. The trace management FSM 838 may receive mode information from the notification manager 810 and provide it to the trace verification unit 836. When there is an address related to the privileged mode or an address which is not related to any mode, the trace verification unit 836 compares this address with address information of a virtual code region of the host system detected when the attack detecting apparatus is initially driven. The trace verification unit 836 transmits this address to the temporary alarm storage unit 840 when this address is not within the virtual code region but not related to any mode, and transmits this address to the alarm generator 842 when this address is no within the virtual code region and related to the privileged mode.

[0065] When receiving information such as mode conversion from the notification manager 810, the trace management FSM 838 analyzes information stored in the temporary alarm storage unit 840 and transmits a result of analyzing the information to the alarm generator 842. The trace verification unit 836 compares a start address and an end address of trampolines stored in a database with an address analyzed by the trace analyzer 834 to detect an attempt to bypass a notification about a trampoline by jumping from a midpoint on the trampolines. When an address representing an attempt to bypass the notification is received, the trace verification unit 836 generates an alarm through the alarm generator 842.

[0066] Lastly, the notification manager 810 will be described below.

[0067] The notification manager 810 realizes a protocol defined to be followed by the host system to determine a current state of the host system. When start of a trampoline is notified through the trace manager 830, the key generator 822 prepares a key so that the keying register 814 may fetch the key. The memory verification unit 826 accesses and reads a main memory of the host system via the memory access unit 828 to verify the content of an event which the host system notifies through a notification register (e.g., whether an ATBR is changed). The notification management FSM 820 controls other modules and generates an alarm through the alarm generator 824 when a malicious action such as a false notification message is detected.

[0068] The notification manager 810 may include registers as illustrated in FIG. 1. The keying register 814 rejects all write accesses and permits only read accesses. For example, the keying register 814 includes a value field storing a key for each of the trampolines and a verification field representing whether the key is available. When the notification management FSM 820 sets the verification field such that the host system uses a key, the keying register 814 quickly fetches a new key from the key generator 822 and stores the new key in the value field. When an available key is fetched by the host system, the keying register 814 clears the verification field to represent that this key is unavailable. The ACK register 816 operates very similar to the keying register 814 except that a fixed value representing acknowledgement (ACK) is stored in the value field. The notification register 818 does not permit any read access and transmits a write access to the notification management FSM 820 when there is the write access.

[0069] For example, the notification manager 810 waits for start of a trampoline to process a notification about a change in an ATBR. When the trace manager 830 receives a notification about the start of the trampoline from the PTI, the notification management FSM 820 configures the keying register 814 such that a key is available, receives the key from the key generator 822, and waits for a notification to be received via the notification register 818. When a notification about an event is received together with a key, the notification management FSM 820 controls the memory verification unit 826 to verify contents of a page table, and provides an ACK message via the ACK register 816 when contents of a table of a virtual code region are correct.

[0070] FIG. 9 is a flowchart of a method of hardware-based detection of a kernel code injection attack according to an embodiment of the present invention.

[0071] Referring to FIG. 9, an attack detecting apparatus detects the locations of code blocks of a kernel related to mode conversion, an ATBR address change, PTI control, etc. while a host system is booted, and replaces these code blocks with branch commands related to trampolines (operation S900).

[0072] When a packet containing runtime information is received through a PTI of a host system, the attack detecting apparatus parses the packet to detect information such as indirect branch target addresses (operation S910). At the same time, when mode conversion or the like occurs in the host system, the attack detecting apparatus receives a notification about mode conversion or the like through a trampoline (operation S920). An example of a method of receiving a notification about mode conversion or the like is illustrated in detail in FIGS. 3 and 7. Furthermore, the attack detecting apparatus detects a malicious write attempt to a physical code region by snooping traffic on a memory bus or a peripheral-device bus (operation S930).

[0073] The attack detecting apparatus detects an attacker's attack which compulsorily converts a processor or the like into the privileged mode outside a virtual code region to inject code, based on the indirect branch target addresses detected by parsing the packet received through the PTI and information (such as mode conversion or the like) notified through a trampoline (operation S940). Furthermore, when an event such as ATBR address change or control occurs, the attack detecting apparatus may receive a notification about the event through a trampoline and detect an attacker's attack made through a malicious address change or the like (operation S940).

[0074] The present invention can be embodied as computer-readable code in a non-transitory computer-readable medium. The non-transitory computer-readable medium may be any recording apparatus capable of storing data that can be read by a computer system, e.g., a read-only memory (ROM), a random access memory (RAM), a compact disc (CD)-ROM, a magnetic tape, a floppy disk, an optical data storage device, and so on. The non-transitory computer-readable medium can be distributed among computer systems that are interconnected through a network, and the computer-readable code may be stored and implemented in the distributed system.

[0075] While the present invention has been particularly shown and described with reference to exemplary embodiments thereof, it will be apparent to those of ordinary skill in the art that various changes in form and details may be made therein without departing from the spirit and scope of the following claims. Thus, the embodiments set forth herein should be considered in a descriptive sense only and not for purposes of limitation. Accordingly, the spirit and scope of the present invention is defined not by the above description of the invention but by the appended claims, and all differences within the scope will be construed as being included in the present invention.

CLAIMS

1. An apparatus for hardware-based detection of a kernel code injection attack, the apparatus comprising:

a trace manager configured to receive a packet containing runtime information from a program trace interface (PTI) of a host system and to parse the packet to detect indirect branch target addresses;

a notification manager configured to receive a notification about an event through execution of a trampoline when the event occurs in a state in which a code block related to the event among code blocks of a kernel is replaced with a jump command on the trampoline while the host system is booted, wherein the event comprises a change in an address translation base register (ATBR), mode conversion, or a change in a PTI control register; and

a traffic manager configured to detect an attempt to write to a physical code region of a memory by snooping traffic on a bus.

2. The apparatus of claim 1, wherein the trace manager detects whether an attack occurs by determining whether a mode related to the indirect branch target addresses is correct on the basis of the event notified by the notification manager.

3. The apparatus of claim 1, wherein the trace manager receives a notification about a start of the trampoline from the PTI, and

the notification manager generates a key and provides the key to the host system when receiving the notification about the start of the trampoline from the trace manager, and receives the key, and information regarding mode conversion or information regarding a change in the ATBR from the host system.

4. The apparatus of claim 3, wherein the notification manager comprises a keying register, a notification register, and an acknowledgement (ACK) register,

wherein the notification manager generates the key and stores the key in the keying register when receiving the notification about the start of the trampoline from the trace manager, resets the keying register when the host system reads the keying register once, and reads the notification register to detect the information regarding the mode conversion when the host system records the information regarding the

mode conversion or the information regarding the change in the ATBR on the notification register.

5. The apparatus of claim 1, wherein the traffic manager snoops the traffic on the bus to monitor an invariable code region, a first-level page table of the ATBR, or an access to the PTI.

6. A method of hardware-based detection of a kernel code injection attack, the method comprising:

receiving a packet containing runtime information from a program trace interface (PTI) of a host system and parsing the packet to detect indirect branch target addresses;

receiving a notification about an event through execution of a trampoline when the event occurs in a state in which a code block related to the event among code blocks of a kernel is replaced with a jump command on the trampoline while the host system is booted, wherein the event comprises a change in an address translation base register (ATBR), mode conversion, or a change in a PTI control register; and

detecting an attempt to write to a physical code region of a memory by snooping traffic on a bus.

7. The method of claim 6, further comprising detecting whether an attack occurs by determining whether a mode related to the indirect branch target addresses is correct on the basis of the notification about the event.

8. The method of claim 6, further comprising:
receiving a notification about a start of the trampoline from the PTI;
generating a key and providing the key to the host system; and
sequentially receiving the key and information regarding the event from the host system.

9. A non-transitory computer-readable recording medium having recorded thereon a program which performs the method of any one of claims 6 to 8.