

LOGGING METHOD USING HYPERVISOR AND APPARATUS THEREOF

TECHNICAL FIELD

[0001] The present invention relates to a method of storing log data and an apparatus thereof, and more particularly, to a method of storing log data of a kernel, etc., by using a micro hypervisor and an apparatus thereof.

BACKGROUND ART

[0002] A micro hypervisor is a type of software which allows different operating systems to run on one computer system. In other words, a hypervisor is used as a type of middleware between hardware and software, and provides an effect of producing a virtual computer using different operating systems in one computer.

DETAILED DESCRIPTION OF THE INVENTION

TECHNICAL PROBLEM

[0003] The present invention provides a method of storing log data of a kernel or an application program, etc., by using a hypervisor and an apparatus thereof.

TECHNICAL SOLUTION

[0004] According to an embodiment of the present invention, a method of storing log data by using a hypervisor includes a logging method using a hypervisor in a computing device including the hypervisor, a kernel, and an application program, the logging method including, via the hypervisor: replacing, among commands of the application program or the kernel, a command in a position defined by a log definition module with a trap command; when the trap command is executed, or an interruption or an exception occurs, obtaining a system privilege; storing log data regarding the

application program or the kernel; and handing over the system privilege to the application program or the kernel.

[0005] According to an embodiment of the present invention, a computing device includes: a log definition module configured to define a position of a command, from among commands of an application program or a kernel, which is to be replaced with a trap command; a command replacement unit configured to replace the command of the application program or the kernel with the trap command according to the log definition module; a privilege controller configured to obtain a system privilege when the trap command is executed, and hand over the system privilege to the application program or the kernel when storing log data is completed; and a log storage unit configured to grasp and store log data of the application program or the kernel.

ADVANTAGEOUS EFFECTS OF THE INVENTION

[0006] According to the present invention, various log data according to an operation of a kernel or an application program, a system interruption or an exception, etc. may be easily grasped and stored.

DESCRIPTION OF THE DRAWINGS

[0007] FIG. 1 shows a structure of a computing device according to an embodiment of the present invention.

[0008] FIG. 2 shows a system privilege order of a computing device according to the present invention.

[0009] FIG. 3 shows a schematic flow of a logging method using a hypervisor according to the present invention.

[0010] FIG. 4 shows an example of a log definition module according to the present invention.

[0011] FIG. 5 shows an implementation example of a log definition module according to the present invention.

[0012] FIG. 6 shows an example of inserting a trap command according to a log definition module, according to the present invention.

[0013] FIGS. 7A and 7B show an implementation example of inserting a trap command according to a log definition module, according to the present invention.

[0014] FIG. 8 shows an example of a hypervisor according to the present invention.

[0015] FIG. 9 shows a flow of a logging method using a hypervisor according to an embodiment of the present invention.

MODE OF THE INVENTION

[0016] Hereinafter, a logging method using a hypervisor and an apparatus thereof, according to the present invention, will be described in detail with reference to the accompanying drawings.

[0017] FIG. 1 shows a structure of a computing device 100 according to an embodiment of the present invention.

[0018] Referring to FIG. 1, the computing device 100 includes a central processing unit 110, a memory 120, a storage unit 130, an input/output unit 140, and a display 150. The central processing unit 110, which is a processor, controls each of the memory 120, the storage unit 130, the input/output unit 140, and the display 150, and drives a kernel, an application program, or software of a hypervisor. The memory 120 includes random-access memory (RAM), read-only memory (ROM), etc., and the storage unit 130 is implemented in various forms such as a hard disk, a solid state disk (SSD), etc. The input/output unit 140 receives a user command via an input interface such as a keypad or a touchscreen, communicates with an external device via wired and wireless communication networks such as the Internet, or connects to various external devices via a universal serial bus (USB) port, etc. The display 150 may include a liquid crystal display (LCD), a light-emitting diode (LED) display, an organic light-emitting diode (OLED) display, etc.

[0019] FIG. 2 shows a system privilege order of a computing device according to the present invention.

[0020] Referring to FIG. 2, a system privilege is high in order of hardware 200, a hypervisor 210, a kernel 220, and an application program 230. Accordingly, the hypervisor 210 may monitor and control various actions of the kernel 220 and the application program 230.

[0021] FIG. 3 shows a schematic flow of a logging method using a hypervisor according to the present invention.

[0022] Referring to FIG. 3, a manager sets and stores a log definition 300 including a time when logging is required during various operations of the application program or kernel 220, 230 and an interruption or exception 360 of the hardware 200 and a type of log data which is to be stored at each logging time. In this regard, logging refers to a process of recording data related to various operations of the application program or kernel 220, 230 and the interruption or exception 360 of the hardware 200, for example, data of a certain memory area, a register value, input and output data, etc. An example of a log definition module including the log definition 300 is illustrated in FIGS. 4 and 5.

[0023] The hypervisor 210 inserts a trap command 350 into an application program or a kernel in order to obtain a system privilege at a time when logging is required during operation of the application program or kernel 220, 230, or changes a value of a control register in order to obtain a system privilege in an interruption or an exception. An example of a method by which the hypervisor 210 inserts a trap command into the application program or kernel 220, 230 is illustrated in FIGS. 6 to 7B.

[0024] When the application program or kernel 220, 230 meets the trap command 350 inserted by the hypervisor 210 while the application program or kernel 220, 230 is in operation, the hypervisor 210 obtains a system privilege and then performs generation 310 of log data related to various operations of the application program or kernel 220, 230. Also, when the interruption or exception 360 of the hardware 200 occurs, and a value of a control register regarding the corresponding interruption or exception is set to change an execution flow into the hypervisor 210, the hypervisor 210 obtains a system privilege and then generates log data. For example, when a certain event, such as a change in a status value related to a virtual memory or the occurrence of a general interruption, occurs, a value of a control register may be changed to hand over a system privilege to the hypervisor 210.

[0025] The log data may include various pieces of information, such as a log occurrence time, a log occurrence location, and a log occurrence cause, as well as a type of log data defined in the log definition module. For example, when a log definition for storing log data related to I/O processing is set, the hypervisor 210 may obtain a system privilege through a trap command at times of I/O processing and occurrence of an operation and then may collect what a processor being currently executed is, what data was intended to be left in a file, etc. A digital forensic technique may be used to collect log data.

[0026] The hypervisor 210 stores log data in a log buffer 320 in order to minimize an effect on system performance in connection with a logging operation. The log buffer 320 records data according to a time order in which log data occurs, and when a certain amount of data is accumulated in the log buffer 320, stores the log data in a log database 340, 342 through a compression and encryption process 330. By encrypting and storing log data, the content thereof may be prevented from being read or modified without a privilege by a kernel or an application program instead of the manager.

[0027] The log database 340, 342 may be implemented in the hypervisor 210 or may be implemented in the application program or kernel 220, 230. The log database 340 implemented in the hypervisor may increase memory usage and complexity, and thus, it may be better to use the log database 342 implemented in the application program or kernel. The structure of a database does not affect a logging operation, and thus, the log database 340, 342 in various structures may be employed according to embodiments.

[0028] The log database 340, 342 may back up collected items of log data in a log repository 344 at a local location or a network-coupled remote repository and a server.

[0029] FIG. 4 shows an example of a log definition module according to the present invention.

[0030] Referring to FIG. 4, the log definition module includes at least one log definition defined by a manager. The log definition includes a position 410 when a logging operation is required in an application program or a kernel, a system interruption or an exception, etc., and a type 420 of log data which is to be stored. If necessary, the log

definition module may include a comment 400 on the corresponding logging operation, and further, various pieces of information may be defined according to embodiments.

[0031] As shown in FIG. 5, the manager may set a log definition through various programming languages or may set a log definition through a graphical user interface (GUI). For example, a computing device may display, on a graphics screen, a comment input field, a position input field for a command position into which a trap command is to be inserted, and a type input field for a type of log data which is to be stored, and may provide a menu list selectable in each field to a user, thereby allowing the user to easily set a log definition. Further, there may be various methods of receiving pieces of information for log definition from the manager.

[0032] The command position 410 into which a trap command is to be inserted indicates a position of a certain command from among commands which are loaded into a memory when an application program or kernel starts. The manager is assumed to grasp a position of a command beforehand through various conventional methods. For example, in the case of a kernel, commands are loaded into a predetermined position in a memory, and a position of each command within the memory is fixed, and accordingly, the manager may grasp a position of a command within the kernel through various methods such as a conventional debugging process. As another example, in the case of an application program, a position into which commands are loaded at the time of execution may change every time. However, a relative position (an offset) of each command within the application program is static, and accordingly, the manager may grasp a relative position of a command within the application program through various conventional methods such as a debugging process. The method itself for grasping a command position of a kernel or an application program is beyond the scope of the present invention, and thus, an additional description thereof is omitted herein.

[0033] The log data type 420 may be variously set by the manager. For example, when the manager intends to store a memory status before operation of a memory allocation or memory release in an application program or a kernel, the manager may define a command position before a command related to memory allocation or release of the application program or the kernel as an insertion position of a trap command, and may

also define log data which is to be stored as data of a corresponding area of a memory where the memory allocation or release operation is performed.

[0034] When the manager intends to set a log definition regarding a hardware interruption or an exception, the manager defines a type of a control register related to a certain event (for example, a change in a status value related to a virtual memory) of the interruption or the exception and a type of log data which is to be stored. The log definition regarding the interruption or the exception may be defined by a programming language or may be defined via various GUI environments.

[0035] FIG. 5 shows an implementation example of a log definition module 500 according to the present invention.

[0036] Referring to FIG. 5, the log definition module 500 includes the comment 400 indicating what a corresponding log definition is about, the command position 410 into which a trap command is to be inserted, and the type 420 of log data which is to be stored. FIG. 5, which shows an example of performing a logging operation before memory release of a kernel, defines that the insertion position 410 of the trap command is a position '0xc011a060' in a memory and the type 420 of stored log data is register number '0th' data.

[0037] FIG. 6 shows an example of inserting a trap command according to a log definition module, according to the present invention.

[0038] Referring to FIG. 6, when an application program or kernel is executed, a hypervisor replaces, among commands 600 of the application program or kernel loaded into a memory, a command 610 in a position defined by a log definition module with a trap command 630.

[0039] For example, when a command position into which a trap command is to be inserted is defined as a position of an Nth command 610 in the log definition module, the hypervisor replaces the Nth command 610 with the trap command 630 when the application program or kernel starts. Accordingly, when the application program or kernel meets the trap command 630 while executing commands, a system privilege is handed over to the hypervisor. The hypervisor having obtained the system privilege grasps and stores log data defined in the log definition module.

[0040] FIGS. 7A and 7B show an implementation example of inserting a trap command according to a log definition module, according to the present invention.

[0041] An example in which a source code regarding memory release and an actual command for execution of the corresponding source code are loaded into a memory is illustrated in FIGS. 7A and 7B. When a manager intends to perform logging at a memory release time, the manager writes the log definition module of FIG. 5 described above, based on a position regarding a memory release instruction (`_vummap(addr,1)`).

[0042] Referring to the example of FIG. 5 again, a position of a memory release command (`_vummap`) is '0xc011a64', and accordingly, the manager defines '0xc011a060', which is a position right before execution of a release command 710, as a command position into which a trap command is to be inserted and defines register number '0' indicating a memory release position as a type of log data in the log definition module.

[0043] Referring to FIGS. 7A and 7B again, when a kernel or application program starts, a hypervisor replaces a command 700 in the memory position '0xc011a060' defined in the log definition module of FIG. 5 with a trap command 750. When the kernel or application program starts and meets the trap command 750 in '0xc011a060', a system privilege is handed over to the hypervisor.

[0044] FIG. 8 shows an example of a hypervisor 210 according to the present invention.

[0045] Referring to FIG. 8, the hypervisor 210 includes a log definition module 800, a command replacement unit 810, a log storage unit 820, and a privilege controller 830. The hypervisor 210 may be implemented as a type of software, and may be executed through hardware such as a memory, a central processing unit, etc. of a computing system described with reference to FIG. 1.

[0046] The log definition module 800 includes at least one log definition including a command position of an application program or kernel into which a trap command is to be inserted and a type of log data which is to be stored. A manager is assumed to know beforehand a position of each command of the application program or kernel through various conventional methods such as debugging.

[0047] The command replacement unit 810 inserts a trap command into the application program or kernel according to the log definition module. The command replacement

unit 810 may insert a trap command between commands of the application program or kernel, and more preferably, may replace an existing command of the application program or kernel with a trap command and insert the trap command. The command replacement unit 810 performs a process of inserting a trap command at a time when the kernel boots or at a time when the application program starts.

[0048] When the trap command inserted in the application program or kernel by the command replacement unit 810 is executed, and thus, a system privilege is handed over to the hypervisor, the log storage unit 820 performs a logging process of grasping log data which is to be stored according to what is defined in the log definition module and storing the log data.

[0049] The privilege controller 830 obtains a system privilege from the application program or kernel when the trap command inserted in the application program or kernel is executed, and returns the system privilege to the application program or kernel when a logging operation of the log storage unit 820 is completed.

[0050] When a command of the application program or kernel is changed from an existing command to a trap command by the command replacement unit 810, the application program or kernel may not normally operate if the privilege controller 830 hands over only the system privilege after the logging operation. Accordingly, the privilege controller 830 executes the existing command replaced with the trap command before handing over the privilege to the application program or kernel, or hands over the privilege after decoding the existing command and executing the command via emulation. For example, the privilege controller 830 executes the pre-replacement command of FIG. 7A and then hands over the system privilege.

[0051] When an original command before being replaced with a trap command is decoded and executed via emulation, and then, a privilege is handed over, the hypervisor may freely manage execution of the original command and may process all relevant commands, whereas implementation of the hypervisor may become complex. When only an original command is simply executed, and then, a privilege is handed over, the hypervisor may be implemented without complexity and an operation of the original command may be performed rapidly, whereas some operations such as an operation related to a stack pointer may not be supported. Accordingly, the privilege

controller 830 selectively performs execution of only an original command or emulation execution according to circumstances.

[0052] For example, if an original command overwritten with a trap command is 'mov r1 r2', and r2 is used during a logging process, the result becomes different when the original command 'mov r1 r2' is simply executed after a logging operation. Accordingly, a method by which r2 is not used during the logging operation, or 'operand' of an instruction is changed has to be performed, and thus, the privilege controller performs emulation in which various processes for computing an exact result are performed instead of simply performing the original command.

[0053] The privilege controller 830 grasps whether to perform an additional operation before handing over a system privilege even in an interruption or exception. For example, when an interruption or exception does not affect an operation of an existing kernel or application program, and the interruption or exception related to the operation of the kernel or application program is forcedly delivered to the hypervisor merely for logging, the privilege controller 830 sets, as in a case where an interruption and an exception occur, related register and memory values in order to notify the occurrence of an interruption and an exception regarding the kernel, etc. before returning a system privilege and then returns the privilege.

[0054] FIG. 9 shows a flow of a logging method using a hypervisor according to an embodiment of the present invention.

[0055] Referring to FIG. 9, the hypervisor receives a log definition from a manager and stores the log definition in a log definition module (operation S900). When an application program or kernel starts, the hypervisor inserts a trap command into the application program or kernel based on the log definition module (operation S910).

[0056] When the application program or kernel meets the trap command inserted by the hypervisor (operation S920), the hypervisor obtains a system privilege from the application program or kernel (operation S930). The hypervisor backs up system information, etc. of a register, etc. (operation S940), and grasps and stores corresponding log data according to a definition of the log definition module (operation S950). When a logging operation is completed, the hypervisor restores the register to a register before obtainment of the system privilege by using a backed up register

(operation S960). The hypervisor simply executes a pre-replacement original command or executes an original command via emulation, etc. (operation S970), and returns the system privilege to the application program or kernel (operation S980).

[0057] In addition, the present invention may be embodied as computer-readable code on a computer-readable recording medium. The computer-readable recording medium includes any type of recording device for storing data which can be read by a computer system. Examples of the computer-readable recording medium include ROM, RAM, compact disc read-only memory (CD-ROM), a magnetic tape, a floppy disk, and an optical data storage device. The computer-readable recording medium may also be distributed over network-coupled computer systems so that the computer-readable code is stored and executed in a distributed fashion.

[0058] While the present invention has been particularly shown and described with reference to exemplary embodiments thereof, it will be understood by one of ordinary skill in the art that various changes in form and detail may be made therein without departing from essential qualities of the present invention. The disclosed exemplary embodiments should be considered in a descriptive sense only and not for purposes of limitation. Therefore, the scope of the present invention is defined not by the detailed 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. A logging method using a hypervisor in a computing device comprising the hypervisor, a kernel, and an application program, the logging method comprising, via the hypervisor:

replacing, among commands of the application program or the kernel, a command in a position defined by a log definition module with a trap command;

when the trap command is executed, or an interruption or an exception occurs, obtaining a system privilege;

storing log data regarding the application program or the kernel; and

handing over the system privilege to the application program or the kernel.

2. The logging method of claim 1, wherein the log definition module comprises a position of a command of the application program or the kernel which is to be replaced with the trap command, and a type of log data which is to be stored.

3. The logging method of claim 1, further comprising backing up a register in the obtaining of the system privilege, and restoring the register to a backed up register in the handing over of the system privilege.

4. The logging method of claim 1, wherein the handing over of the system privilege comprises executing a pre-replacement original command and then handing over the system privilege.

5. The logging method of claim 1, wherein the storing of the log data comprises:

storing the log data in a buffer of the hypervisor; and

when a size of the log data stored in the buffer corresponds to a threshold, encrypting the log data and storing encrypted log data in a database.

6. A computing device comprising a memory and a central processing unit, the computing device configured to drive a hypervisor comprising:

a log definition module configured to define a position of a command, from among commands of an application program or a kernel, which is to be replaced with a trap command;

a command replacement unit configured to replace the command of the application program or the kernel with the trap command according to the log definition module;

a privilege controller configured to obtain a system privilege when the trap command is executed, and hand over the system privilege to the application program or the kernel when storing log data is completed; and

a log storage unit configured to grasp and store log data of the application program or the kernel.

7. The computing device of claim 6, wherein the command replacement unit is further configured to change a value of a control register so as to obtain the system privilege when an interruption or an exception occurs.

8. The computing device of claim 6, wherein the log storage unit is further configured to back up a register when the obtaining of the system privilege is performed, and restore the register to a backed up register when the handing over of the system privilege is performed.

9. The computing device of claim 6, wherein the privilege controller is further configured to execute a pre-replacement original command and then hand over the system privilege when the handing over of the system privilege.

10. The computing device of claim 6, wherein the log storage unit is further configured to store the log data in a buffer of the hypervisor, and when a size of the log data stored in the buffer corresponds to a threshold, encrypt the log data and store encrypted log data in a database.

11. A computer-readable recording medium having recorded thereon a program for executing the method of one of claims 1 to 5.