US 20070028225A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2007/0028225 A1

Whittaker et al. (43) Pub. Date: Feb. 1, 2007

(54) METHOD AND APPARATUS FOR PREEMPTIVE MONITORING OF SOFTWARE BINARIES BY INSTRUCTION INTERCEPTION AND DYNAMIC RECOMPILATION

(76) Inventors: **James A. Whittaker**, Indialantic, FL (US); **Rahul Chaturvedi**, Melbourne, FL (US); **John R. Wagner**, Melbourne, FL (US)

Correspondence Address:
IP STRATEGIES
12 1/2 WALL STREET
SUITE I
ASHEVILLE, NC 28801 (US)

(21) Appl. No.: **11/528,982**

(22) Filed: **Sep. 27, 2006**

Related U.S. Application Data

(63) Continuation of application No. 10/390,397, filed on Mar. 17, 2003.

(60) Provisional application No. 60/364,907, filed on Mar. 16, 2002.

Publication Classification

(51) **Int. Cl.**
G06F 9/44 (2006.01)

(52) **U.S. Cl.** **717/162**; 717/166

(57) **ABSTRACT**

A method of executing a program in a controlled environment includes initiating execution of an operating system with which the program is adapted to execute, inserting redirection logic at the beginning of the program, and executing the program such that the redirection logic is executed. A current instruction pointer is stored, and execution control is redirected to a program loader. The program loader selects a first block of instructions of the program, based at least in part on the stored current instruction pointer. This selected block of instructions is manipulated to provide a first phantom instruction block, which is executed in the controlled environment. This manipulation includes copying at least a portion of the selected first block to form the first phantom instruction block.

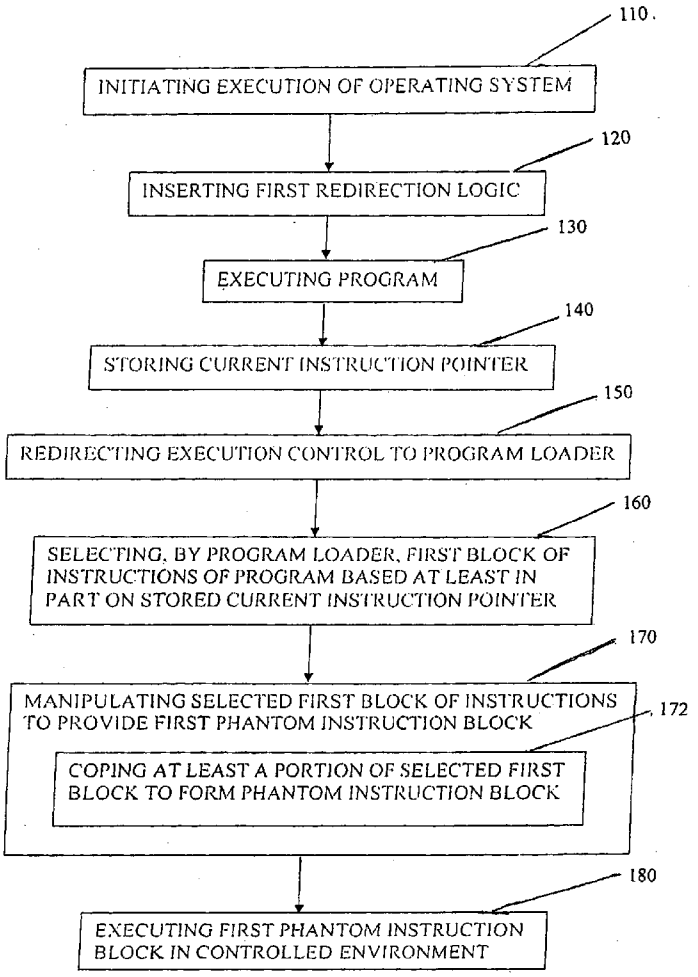


FIGURE 1

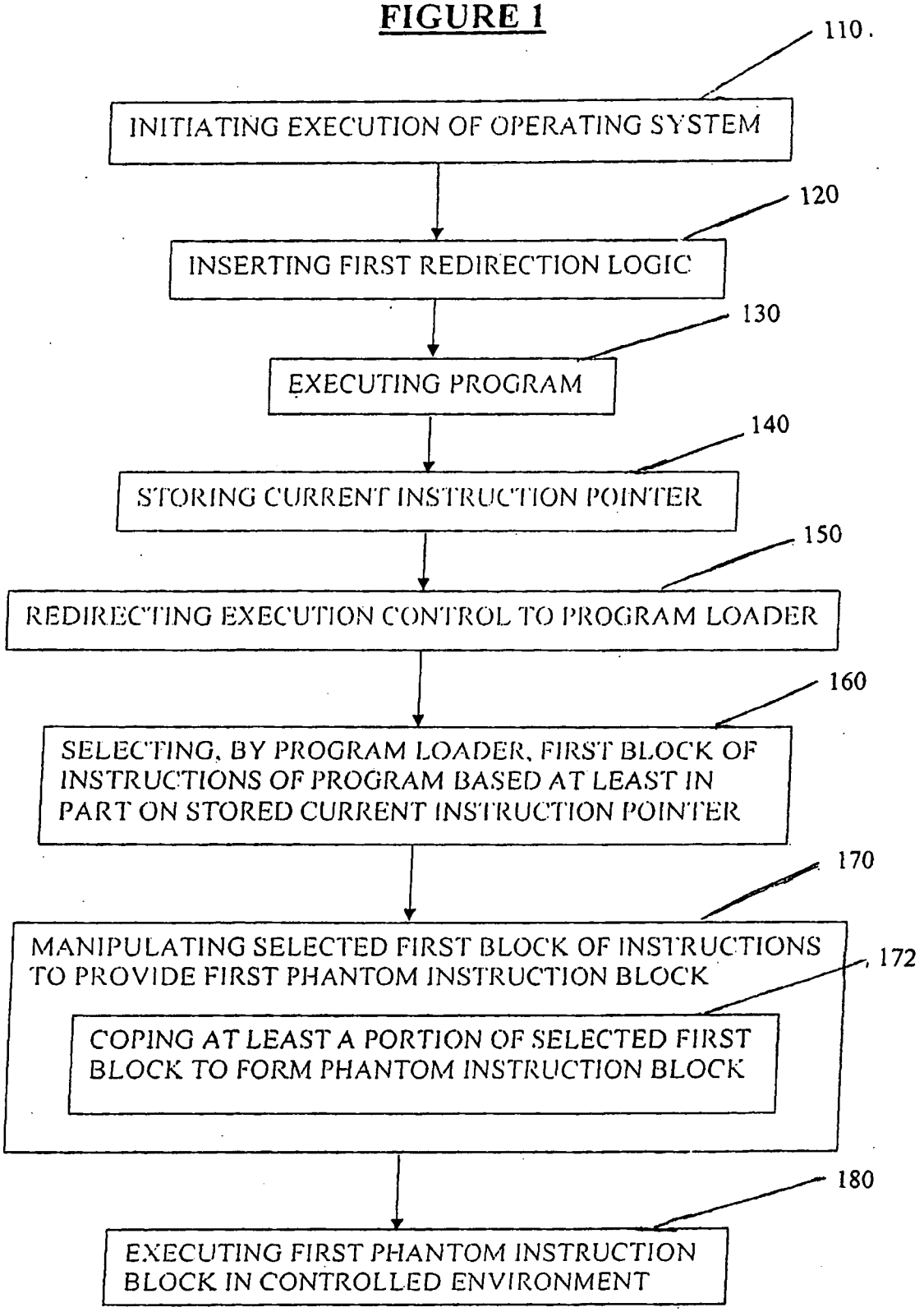

110
INITIATING EXECUTION OF OPERATING SYSTEM
120
INSERTING FIRST REDIRECTION LOGIC
130
EXECUTING PROGRAM
140
STORING CURRENT INSTRUCTION POINTER
150
REDIRECTING EXECUTION CONTROL TO PROGRAM LOADER
160
SELECTING, BY PROGRAM LOADER, FIRST BLOCK OF INSTRUCTIONS OF PROGRAM BASED AT LEAST IN PART ON STORED CURRENT INSTRUCTION POINTER
170
MANIPULATING SELECTED FIRST BLOCK OF INSTRUCTIONS TO PROVIDE FIRST PHANTOM INSTRUCTION BLOCK
172
COPING AT LEAST A PORTION OF SELECTED FIRST BLOCK TO FORM PHANTOM INSTRUCTION BLOCK
180
EXECUTING FIRST PHANTOM INSTRUCTION BLOCK IN CONTROLLED ENVIRONMENT

FIGURE 2

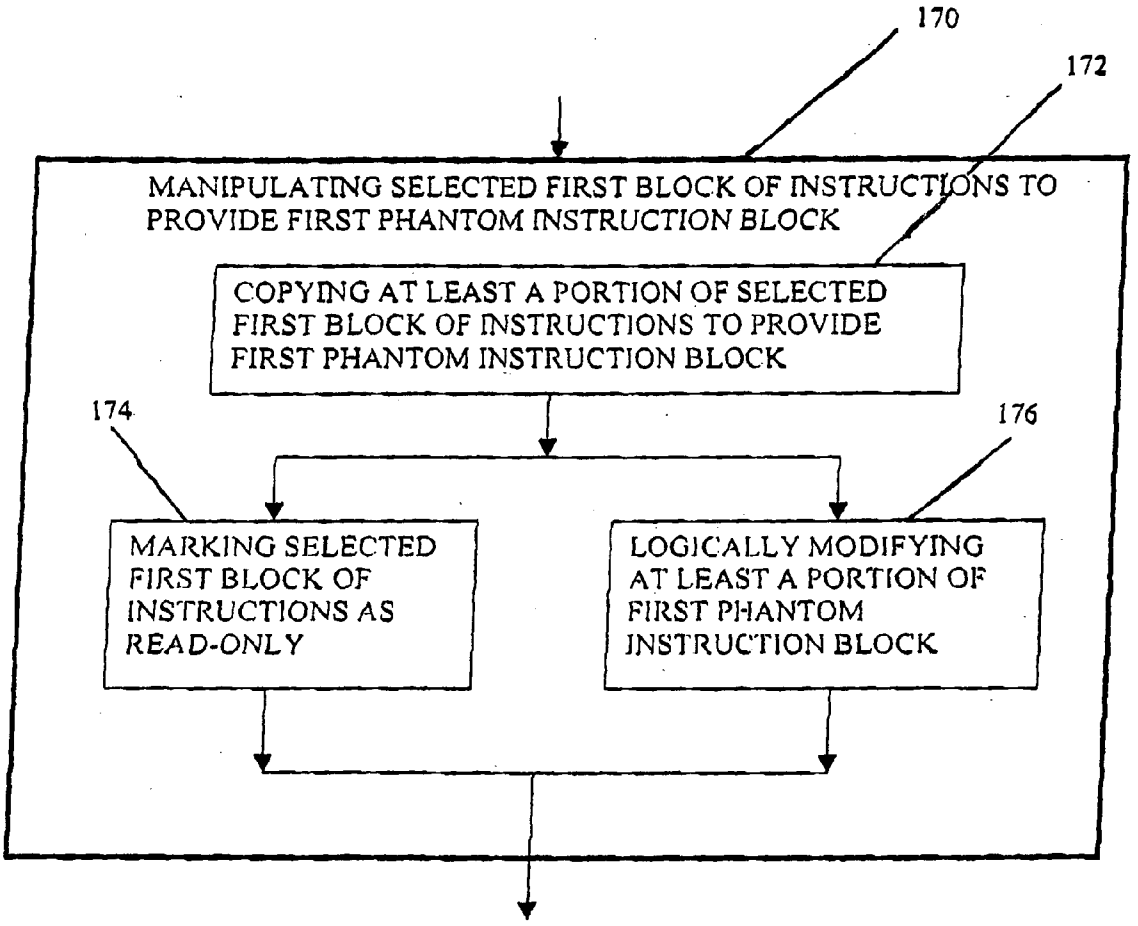

170
172
MANIPULATING SELECTED FIRST BLOCK OF INSTRUCTIONS TO PROVIDE FIRST PHANTOM INSTRUCTION BLOCK
COPYING AT LEAST A PORTION OF SELECTED FIRST BLOCK OF INSTRUCTIONS TO PROVIDE FIRST PHANTOM INSTRUCTION BLOCK
174
176
MARKING SELECTED FIRST BLOCK OF INSTRUCTIONS AS READ-ONLY
LOGICALLY MODIFYING AT LEAST A PORTION OF FIRST PHANTOM INSTRUCTION BLOCK

FIGURE 3

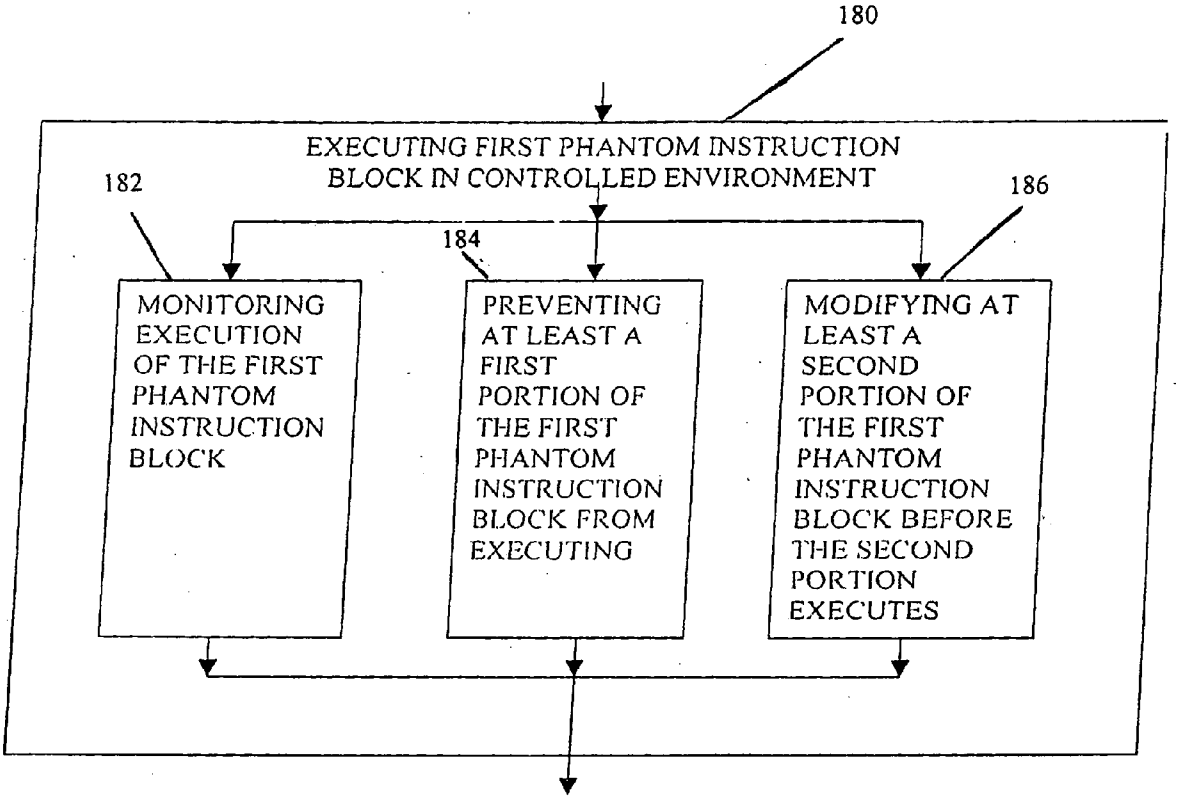

180
EXECUTING FIRST PHANTOM INSTRUCTION BLOCK IN CONTROLLED ENVIRONMENT
182
184
186
MONITORING EXECUTION OF THE FIRST PHANTOM INSTRUCTION BLOCK
PREVENTING AT LEAST A FIRST PORTION OF THE FIRST PHANTOM INSTRUCTION BLOCK FROM EXECUTING
MODIFYING AT LEAST A SECOND PORTION OF THE FIRST PHANTOM INSTRUCTION BLOCK BEFORE THE SECOND PORTION EXECUTES

FIGURE 4

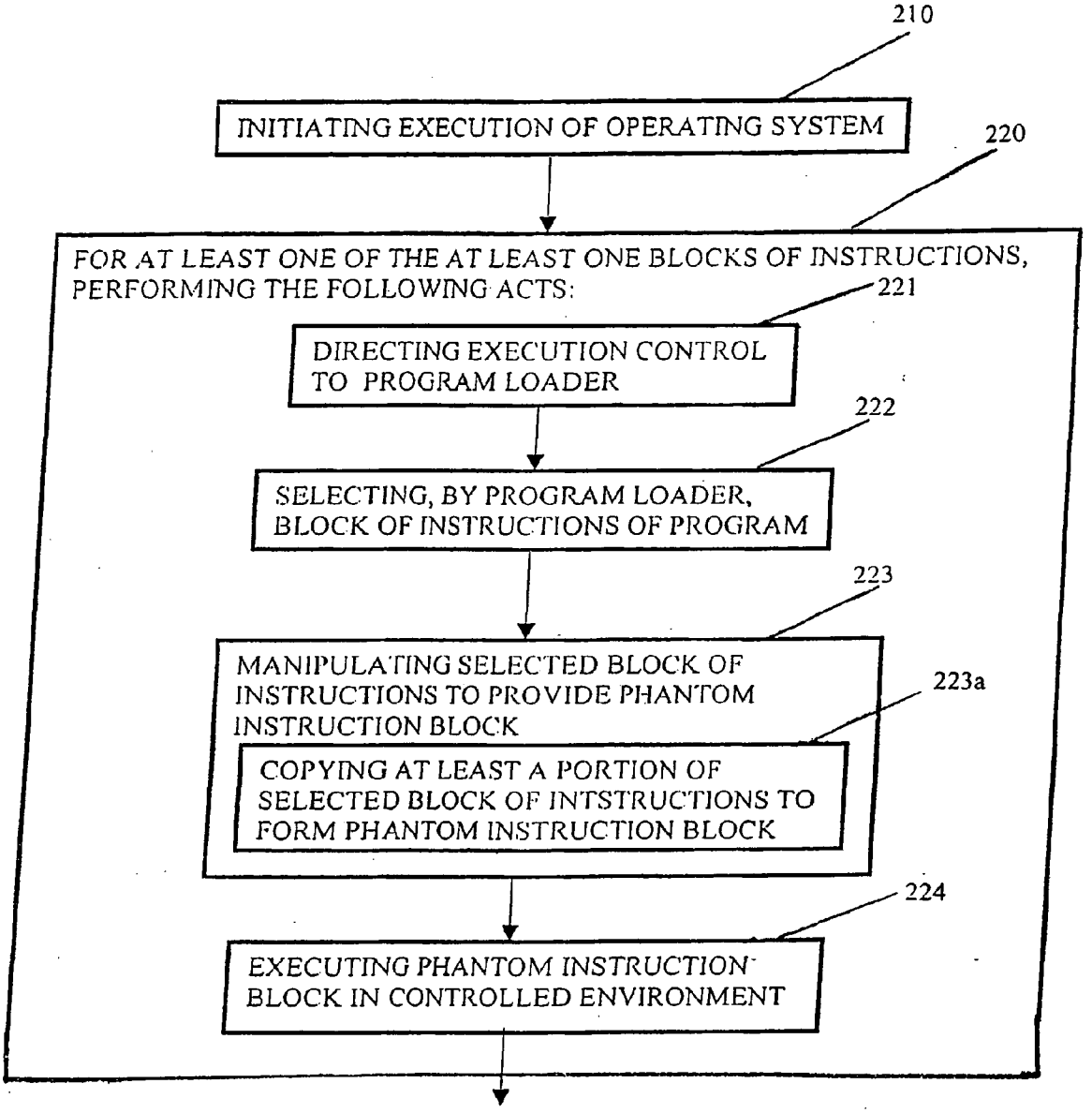
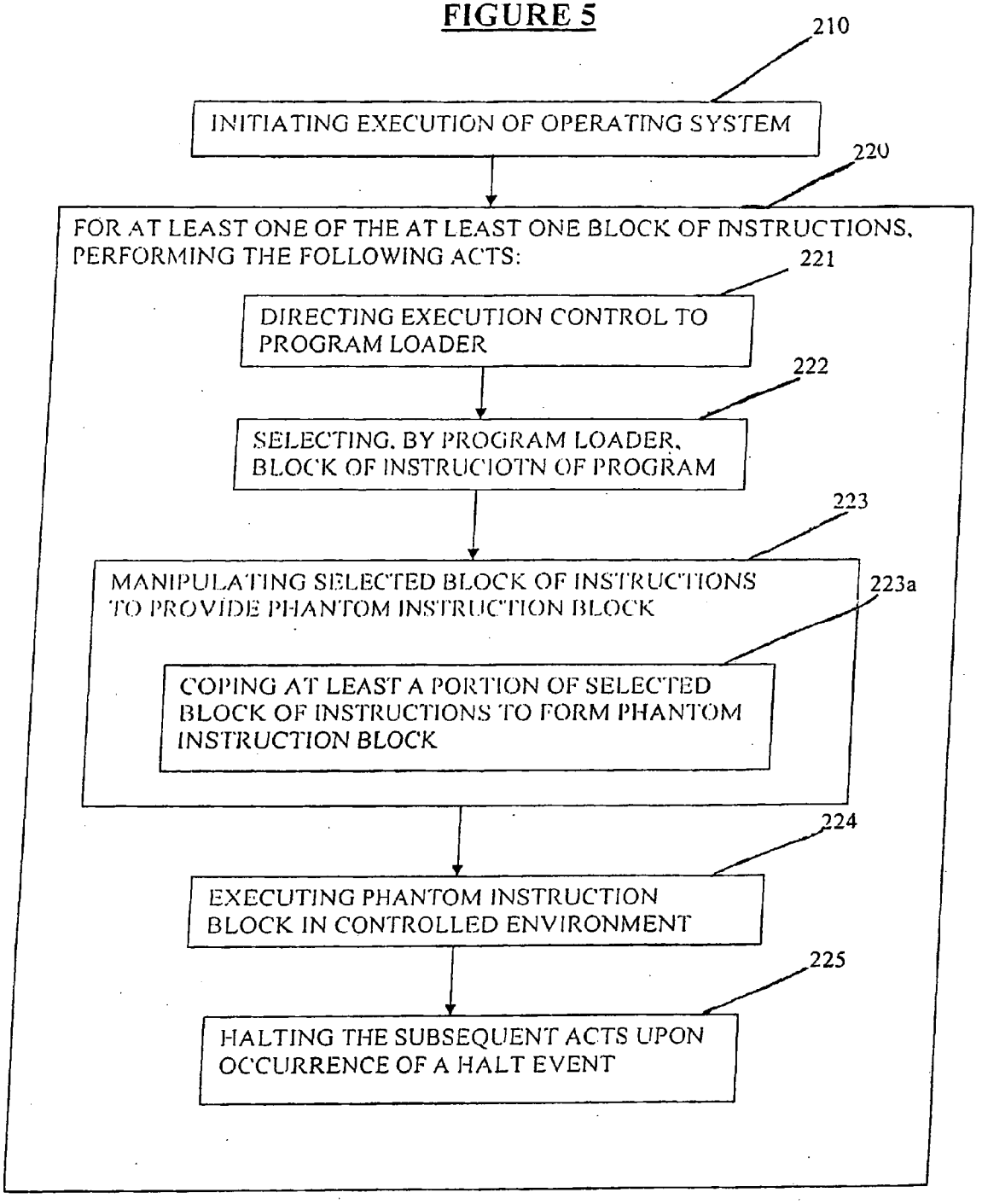

210
INITIATING EXECUTION OF OPERATING SYSTEM
220
FOR AT LEAST ONE OF THE AT LEAST ONE BLOCKS OF INSTRUCTIONS, PERFORMING THE FOLLOWING ACTS:
221
DIRECTING EXECUTION CONTROL TO PROGRAM LOADER
222
SELECTING, BY PROGRAM LOADER, BLOCK OF INSTRUCTIONS OF PROGRAM
223
MANIPULATING SELECTED BLOCK OF INSTRUCTIONS TO PROVIDE PHANTOM INSTRUCTION BLOCK
223a
COPYING AT LEAST A PORTION OF SELECTED BLOCK OF INTSTRUCTIONS TO FORM PHANTOM INSTRUCTION BLOCK
224
EXECUTING PHANTOM INSTRUCTION BLOCK IN CONTROLLED ENVIRONMENT

FIGURE 5


210
INITIATING EXECUTION OF OPERATING SYSTEM
220
FOR AT LEAST ONE OF THE AT LEAST ONE BLOCK OF INSTRUCTIONS, PERFORMING THE FOLLOWING ACTS:
221
DIRECTING EXECUTION CONTROL TO PROGRAM LOADER
222
SELECTING, BY PROGRAM LOADER, BLOCK OF INSTRUCIOTN OF PROGRAM
223
MANIPULATING SELECTED BLOCK OF INSTRUCTIONS TO PROVIDE PHANTOM INSTRUCTION BLOCK
223a
COPING AT LEAST A PORTION OF SELECTED BLOCK OF INSTRUCTIONS TO FORM PHANTOM INSTRUCTION BLOCK
224
EXECUTING PHANTOM INSTRUCTION BLOCK IN CONTROLLED ENVIRONMENT
225
HALTING THE SUBSEQUENT ACTS UPON OCCURRENCE OF A HALT EVENT

METHOD AND APPARATUS FOR PREEMPTIVE MONITORING OF SOFTWARE BINARIES BY INSTRUCTION INTERCEPTION AND DYNAMIC RECOMPILATION

CROSS-REFERENCE TO RELATED DOCUMENTS

[0001] This document claims the priority benefit, and incorporates by reference in its entirety, U.S. Provision Patent Application No. 60/364,907 filed on Mar. 16, 2002.

FIELD OF THE INVENTION

[0002] The present invention relates to software monitoring, testing and analysis.

BACKGROUND OF THE INVENTION

[0003] Processes for monitoring software behavior have many applications in both offensive and defensive cyberwarfare and also in software testing/debugging. Whenever intelligence about an application needs to be gathered, the behavior of the application is monitored and conclusions drawn by analyzing the results.

[0004] When the source code of the application is available, one can monitor the behavior of the application on an instruction-by-instruction basis. Such monitoring allows the behavior to be better understood for the purposes of reengineering or debugging. Access to the source code allows exposure of all relevant information about the software's behavior, both external behavior and internal behavior. This includes calls to external components and the parameters of these calls, calls to internal components and the parameters of these calls, internally stored data, and control structures.

[0005] However, when the source code is not available, one must work with only the compiled binary of the application (for example, the executable, library or component). This means that only the machine code is available for analysis. As such it is much more difficult to get the detailed information that source debuggers can get.

[0006] Conventional systems include technology to intercept external behavior such as calls to the operating system or third party components. This technique is called system call interception. Conventional techniques also include the use of disassemblers to extract certain bits of internal information, but disassembly is a painstaking process fraught with trial and error. In general, information inside the binary executes without scrutiny and with no opportunity to intervene to change behavior in any predictable fashion. Internal functions, stored data, and individual control statements can be unreachable externally.

[0007] System call interception has many uses including proactive antivirus tools and testing/debugging tools. However, there are many behaviors that escape such external scrutiny. For example, instructions that call exception handlers or interrupts execute without making system calls and can bypass monitoring or protective software.

[0008] The only currently-known solution to intercept such dispatch mechanisms is to insert code into the operating system kernel that detects these dispatches and notifies the monitoring application. There is a disadvantage to this solution in that because the actual operating system kernel is modified, every single application is intercepted instead of just the application being monitored, and such "kernel mode" solutions tend to destabilize the operating system.

[0009] Therefore, there is a need for an improved software monitoring solution.

BRIEF SUMMARY OF THE INVENTION

[0010] The present invention includes a method of executing a program in a controlled environment, which can be embodied in software components and user interface programs that can either be incorporated into other software or that can be used in a standalone fashion. The broad process of the invention can be realized in any of many specific embodiments, and there are many advantageous uses for the process. For example, the process can be applied to blocking the behaviors of malicious programs such as viruses, worms, and Trojan horse and zombie programs. Further, the process can be used to monitor behavior so that an application can be coaxed into revealing hidden features and functionality. For example, the process can be used to crack encryption keys (to discover private data and information) and CD keys (to enable software piracy). These applications are useful as evaluation tools.

[0011] Therefore, according to an exemplary aspect of the invention, a method of executing a program in a controlled environment includes initiating execution of an operating system with which the program is adapted to execute, inserting redirection logic at the beginning of the program, and executing the program such that the redirection logic is executed. A current instruction pointer is stored, and execution control is redirected to a program loader. The program loader selects a first block of instructions of the program, based at least in part on the stored current instruction pointer. This selected block of instructions is manipulated to provide a first phantom instruction block, which is executed in the controlled environment. This manipulation includes copying at least a portion of the selected first block to form the first phantom instruction block.

[0012] The selected first block of instructions includes one or more instructions. That is, although referred to as a block, the block can be a single instruction, or it can include a number of instructions.

[0013] The manipulation of the selected block of instructions can also include marking the selected first block as read-only, or logically modifying at least a portion of the first phantom instruction block, or both. In turn, logically modifying at least a portion of the first phantom instruction block can include any one or combination of inserting program logic into the first phantom instruction block, deleting program logic from the first phantom instruction block, and changing program logic in the first phantom instruction block.

[0014] Execution of the first phantom instruction block in the controlled environment can include any one or combination of monitoring execution of the first phantom instruction block, preventing at least a first portion of the first phantom instruction block from executing, and modifying at least a second portion of the first phantom instruction block before the second portion executes. The controlled environment can include a user interface by which a user can execute and monitor execution of the first phantom instruction block.

[0015] The current instruction pointer can be stored by pushing the current instruction pointer onto a stack.

[0016] The acts can further include storing a next instruction pointer, for example, after executing the first phantom instruction block. In this case, the program loader selects a second block of instructions of the program based at least in part on the stored next instruction pointer, and the selected second block of instructions is manipulated to provide a second phantom instruction block. This manipulation of the selected second block includes copying at least a portion of the selected second block to form the second phantom instruction block, and executing the second phantom instruction block in the controlled environment. The selected second block of instructions includes one or more instructions. The manipulation of the selected first and second blocks can also include marking the first and second blocks as read-only, or logically modifying at least a portion of the first and second phantom instruction blocks, or both. In turn, logically modifying at least a portion of the first and second phantom instruction blocks can include any one or combination of inserting program logic into the first and second phantom instruction blocks, deleting program logic from the first and second phantom instruction blocks, and changing program logic in the first and second phantom instruction blocks.

[0017] The operating system can include a thread spawning routine having at least one block of instructions. In this case, the method of the invention can also include inserting second redirection logic at a beginning of the thread spawning routine that directs execution control to the program loader and executing the thread spawning routine.

[0018] The operating system can include an exception handling routine having at least one block of instructions. In this case, the method of the invention can also include inserting, at a beginning of the exception handling routine, second redirection logic that directs execution control to the program loader, and executing the exception handling routine.

[0019] According to another aspect of the present invention a method of executing, in a controlled environment, a program having at least one block of instructions includes initiating execution of an operating system with which the program is adapted to execute, and performing a number of subsequent acts for at least one of the blocks of instructions. These acts include directing execution control to a program loader. Also, a block of instructions of the program is selected by the program loader, and the selected block of instructions is manipulated to provide a phantom instruction block. This phantom instruction block is executed in the controlled environment. The manipulation of the selected block of instructions includes copying at least a portion of the selected block to form the phantom instruction block. Each selected block includes at least one instruction.

[0020] The subsequent acts can also include halting the subsequent acts based on the occurrence of a halt event.

[0021] The manipulation of the selected block of instructions can also include marking the selected block as read-only, or logically modifying at least a portion of the phantom instruction block, or both. In turn, logically modifying at least a portion of the phantom instruction block can include any one or combination of inserting program logic into the phantom instruction block, deleting program logic from the phantom instruction block, and changing program logic in the phantom instruction block.

[0022] Execution of the phantom instruction block in the controlled environment can include any one or combination of monitoring execution of the phantom instruction block, preventing at least a first portion of the phantom instruction block from executing, and modifying at least a second portion of the phantom instruction block before the second portion executes. The controlled environment can include a user interface by which a user can execute and monitor execution of the phantom instruction block.

[0023] The method of the invention can also include inserting first redirection logic within the program that directs execution control to the program loader, and executing the program. The subsequent acts can include storing a respective current instruction pointer. In this case, the program loader selects the block of instructions of the program based at least in part on the stored respective current instruction pointer.

[0024] The operating system can include a thread spawning routine having at least one set of instructions, wherein the set of instructions can include as few as a single instruction. Thus, the method of the invention can also include inserting second redirection logic at the beginning of the thread spawning routine that directs execution control to the program loader, and executing the thread spawning routine. In this case, the method of the invention can also include performing additional actions for at least one set of instructions. These additional actions include redirecting execution control to the program loader. Also, a set of instructions of the thread spawning routine is selected by the program loader, and the selected set of instructions is manipulated to provide a phantom instruction set. The phantom instruction set is executed in the controlled environment. Each selected set of instructions includes at least one instruction, and manipulating the selected set of instructions includes copying at least a portion of the selected set of instructions to form the phantom instruction set.

[0025] Alternatively, or in addition, the operating system can include an exception handling routine having at least one set of instructions. Thus, the method of the invention can also include inserting additional redirection logic at a beginning of the exception handling routine that directs execution control to the program loader, and executing the exception handling routine. In this case, the method of the invention also includes performing subsequent actions for at least one set of instructions. These subsequent actions can include redirecting execution control to the program loader. A set of instructions of the exception handling routine is selected by the program loader, and the selected set of instructions is manipulated to provide a phantom instruction set. The phantom instruction set is executed in the controlled environment. Each selected set of instructions can include as few as a single instruction, and manipulating the selected set of instructions includes copying at least a portion of the selected set of instructions to form the phantom instruction set.

[0026] The manipulation of the selected block can also include determining if the selected block is represented by current data stored in a cache. If the selected block is not represented by the current data stored in the cache, at least

a portion of the selected block is copied to form the phantom instruction block, and additional data representative of the formed phantom instruction block is added to the cache. On the other hand, if the selected block is represented by the current data stored in the cache, the current data representative of the selected block of instruction is referenced to provide the phantom instruction block. The manipulation of the selected block can also include determining if the selected block of instructions invariably directs execution control to a different block of instructions represented by the current data stored in the cache. If the selected block of instructions is determined to invariably direct execution control to the different block of instructions, redirection logic is inserted into the selected block that directs execution control to a cached phantom instruction block representative of the different block.

BRIEF DESCRIPTION OF THE DRAWINGS

[0027] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which:

[0028] FIG. **1** is a flow diagram illustrating an exemplary process of the invention.

[0029] FIG. **2** shows an exemplary detail of an instruction manipulation action according to the invention.

[0030] FIG. **3** shows an exemplary detail of a phantom instruction execution action according to the invention.

[0031] FIG. **4** is a flow diagram illustrating another exemplary process of the invention.

[0032] FIG. **5** is a flow diagram showing yet another exemplary process of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0033] FIG. **1** illustrates an exemplary aspect of the invention, in which a method of executing a program in a controlled environment includes initiating execution of an operating system with which the program is adapted to execute (**110**), inserting redirection logic at the beginning of the program (**120**), and executing the program such that the redirection logic is executed (**130**). Further, a current instruction pointer is stored (**140**), and execution control is redirected to a program loader (**150**). The program loader selects a first block of instructions of the program (**160**), based at least in part on the stored current instruction pointer. This selected block of instructions is manipulated to provide a first phantom instruction block (**170**), which is executed in the controlled environment (**180**). This manipulation includes copying at least a portion of the selected first block to form the first phantom instruction block (**172**). Thus, controlled execution of the program (or selected block or blocks) is achieved by executing a phantom block or blocks. As shown, the original program itself is not executed; instead, a phantom copy of the program (a phantom block or blocks instructions) is executed.

[0034] It should be noted that the selected first block of instructions can include one or more instructions. That is, although referred to as a block, the selected block can be a single instruction, or it can include multiple instructions.

[0035] As shown in FIG. **2**, the manipulation of the selected block of instructions (**170**) can also include marking the selected first block as read-only (**174**), or logically modifying at least a portion of the first phantom instruction block (**176**), or both. By marking the selected first block as read-only (**174**), the risk that self-modifying code will modify a previously selected block of instructions is avoided, so as to ensure, for example, that program control is not lost. In turn, logically modifying at least a portion of the first phantom instruction block (**176**) can include any one or combination of inserting program logic into the first phantom instruction block, deleting program logic from the first phantom instruction block, and changing program logic in the first phantom instruction block.

[0036] As illustrated in FIG. **3**, execution of the first phantom instruction block in the controlled environment (**180**) can include any one or combination of monitoring execution of the first phantom instruction block (**182**), preventing at least a first portion of the first phantom instruction block from executing (**184**), and modifying at least a second portion of the first phantom instruction block before the second portion executes (**186**). The controlled environment can include a user interface by which a user can execute and monitor execution of the first phantom instruction block, such as monitoring the values of system and/or program variables and states, and system memory, as well as other system or program states and conditions. According to another exemplary aspect of the invention, a user interface can provide a "debugging" environment, with which a user can execute, monitor, modify and/or prevent execution of one or more instructions contained in a phantom instruction block. In another exemplary aspect of the invention, a user interface can allow a user to trace and step through, on an instruction-by-instruction basis, the one or more instructions contained in a phantom instruction block. Further, a user interface can allow a user to watch changes in the values of variables as instructions are stepped through or traced.

[0037] The current instruction pointer can be stored in a memory, such as RAM, hard drive, or other memory, for example and not in limitation. Further, the current instruction pointer can be stored by pushing the current instruction pointer onto a stack, for example and not in limitation.

[0038] The method can further include executing a second block of instructions in the controlled environment, following storage of the current instruction pointer (**140**). For example, the method can further include storing a next instruction pointer, after executing the first phantom instruction block. In this case, the program loader selects a second block of instructions of the program based at least in part on the stored next instruction pointer, and the selected second block of instructions is manipulated to provide a second phantom instruction block. This manipulation of the selected second block includes copying at least a portion of the selected second block to form the second phantom instruction block, and executing the second phantom instruction block in the controlled environment. The selected second block of instructions includes one or more instructions. The manipulation of the selected first and second blocks can also include marking the first and second blocks as read-only, or logically modifying at least a portion of the first and second phantom instruction blocks, or both. In turn, logically modifying at least a portion of the first and second phantom instruction blocks can include any one or combination of

inserting program logic into the first and second phantom instruction blocks, deleting program logic from the first and second phantom instruction blocks, and changing program logic in the first and second phantom instruction blocks.

[0039] According to a further exemplary aspect of the invention, the operating system can include a thread spawning routine having at least one block of instructions, and therefore, the method can further include actions for accounting for thread spawning, which can cause a loss of execution control. Accordingly, the method can further include actions that logically mirror other actions in execution of the method, starting from initiation of execution of the operating system. Thus, the method of the invention can also include inserting second redirection logic at a beginning of the thread spawning routine that directs execution control to the program loader, and executing the thread spawning routine.

[0040] Alternatively, or in addition, the operating system can include an exception handling routine having at least one set of instructions. Accordingly, the method of the present invention can further include actions for accounting for exceptions, which can also cause a loss of execution control. Accordingly, the method can further include inserting second redirection logic, at a beginning of the exception handling routine, that directs execution control to the program loader, and executing the exception handling routine.

[0041] As shown in FIG. **4**, according to another exemplary aspect of the present invention, a method of executing, in a controlled environment, a program having at least one block of instructions includes initiating execution of an operating system with which the program is adapted to execute (**210**), and performing a number of subsequent actions for at least one of the blocks of instructions (**220**). These actions include directing execution control to a program loader (**221**). Also, a block of instructions of the program is selected by the program loader (**222**), and the selected block of instructions is manipulated to provide a phantom instruction block (**223**). This phantom instruction block is executed in the controlled environment (**224**). The manipulation of the selected block of instructions includes copying at least a portion of the selected block to form the phantom instruction block (**223**a). Each selected block includes at least one instruction.

[0042] As shown in FIG. **5**, the subsequent actions can also include halting further actions based on the occurrence of a halt event (**225**), which can include, for example and not in limitation, a stop command, and a logical condition within program logic (for example, WHILE NOT END OF FILE).

[0043] The manipulation of the selected block of instructions can also include marking the selected block as read-only, or logically modifying at least a portion of the phantom instruction block, or both. In turn, logically modifying at least a portion of the phantom instruction block can include any one or combination of inserting program logic into the phantom instruction block, deleting program logic from the phantom instruction block, and changing program logic in the phantom instruction block.

[0044] Execution of the phantom instruction block in the controlled environment can include any one or combination of monitoring execution of the phantom instruction block, preventing at least a first portion of the phantom instruction block from executing, and modifying at least a second portion of the phantom instruction block before the second portion executes. The controlled environment can include a user interface by which a user can execute and monitor execution of the phantom instruction block, such as monitoring the values of system and/or program variables and states, and system memory, as well as other system or program states and conditions. According to an exemplary aspect of the invention, a user interface can provide a "debugging" environment, with which a user can execute, monitor, modify and/or prevent execution of one or more instructions contained in a phantom instruction block. According to another exemplary aspect of the invention, a user interface can allow a user to trace and step through, on an instruction-by-instruction basis, the one or more instructions contained in a phantom instruction block. Further, a user interface can allow a user to watch changes in the values of variables as instructions are stepped through or traced.

[0045] This method of the present invention can also include inserting, within the program, first redirection logic that directs execution control to the program loader, and executing the program. In this case, the subsequent actions can further include storing a respective current instruction pointer, such that the program loader selects the block of instructions of the program based at least in part on the stored respective current instruction pointer.

[0046] According to another exemplary aspect of this invention, as per an earlier described exemplary aspect, the operating system can include a thread spawning routine having at least one set of instructions, wherein a set of instructions can include as few as a single instruction. Thus, this method of the invention can also include inserting, at the beginning of the thread spawning routine, second redirection logic that directs execution control to the program loader, and executing the thread spawning routine. In this case, this method of the invention can also include performing additional actions for at least one set of instructions. These additional actions can include redirecting execution control to the program loader. Also, a set of instructions of the thread spawning routine is selected by the program loader, and the selected set of instructions is manipulated to provide a phantom instruction set. The phantom instruction set is executed in the controlled environment. Each selected set of instructions includes at least one instruction, and manipulating the selected set of instructions includes copying at least a portion of the selected set of instructions to form the phantom instruction set.

[0047] Alternatively, or in addition, the operating system, as per a previously described exemplary aspect of the present invention, can include an exception handling routine having at least one set of instructions. Thus, the method of the invention can also include inserting, at a beginning of the exception handling routine, additional redirection logic that directs execution control to the program loader, and executing the exception handling routine. In this case, the method of the invention also includes performing subsequent actions for at least one set of instructions. These subsequent actions can include redirecting execution control to the program loader. A set of instructions of the exception handling routine is selected by the program loader, and the selected set of instructions is manipulated to provide a phantom instruction set. The phantom instruction set is executed in the controlled

environment. Each selected set of instructions can include as few as a single instruction, and manipulating the selected set of instructions includes copying at least a portion of the selected set of instructions to form the phantom instruction set.

[0048] According to yet another exemplary aspect of the present invention, the manipulation of the selected block can also include determining if the selected block is represented by current data stored in a cache. If the selected block is not represented by the current data stored in the cache, at least a portion of the selected block is copied to form the phantom instruction block, and additional data representative of the formed phantom instruction block is added to the cache. On the other hand, if the selected block is represented by the current data stored in the cache, the current data representative of the selected block of instructions is referenced to provide the phantom instruction block. The manipulation of the selected block can also include determining if the selected block of instructions invariably directs execution control to a different block of instructions represented by the current data stored in the cache. If it is determined that the selected block of instructions invariably directs execution control to the different block of instructions, redirection logic is inserted into the selected block that directs execution control to a cached phantom instruction block representative of the different block.

[0049] It should be noted that the present invention can be further embodied in an apparatus, as well as a computer readable medium, each of which being based on the process embodiments described herein.

[0050] In the foregoing written description, the invention has been described with reference to specific embodiments thereof. However, it will be evident that various modifications and/or changes may be made thereto without departing from the broader spirit and scope of the invention. Accordingly, the specification and drawings are to be regarded in an illustrative and enabling, rather than a restrictive, sense.

1. A method of executing a program in a controlled environment, comprising:

initiating execution of an operating system with which the program is adapted to execute;

inserting, at a beginning of the program, first redirection logic;

executing the program such that the first redirection logic is executed;

storing a current instruction pointer;

redirecting execution control to a program loader;

selecting, by the program loader, a first block of instructions of the program based at least in part on the stored current instruction pointer;

manipulating the selected first block of instructions to provide a first phantom instruction block; and

executing the first phantom instruction block in the controlled environment;

wherein the selected first block of instructions includes at least one instruction, and manipulating the selected first block includes copying at least a portion of the selected first block to form the first phantom instruction block.

2. The method of claim 1, wherein manipulating the selected first block of instructions further includes at least one of

marking the selected first block as read-only, and

logically modifying at least a portion of the first phantom instruction block.

3. The method of claim 2, wherein logically modifying the at least a portion of the first phantom instruction block includes at least one of

inserting program logic into the first phantom instruction block,

deleting program logic from the first phantom instruction block, and

changing program logic in the first phantom instruction block.

4. (canceled)

5. (canceled)

6. The method of claim 1, wherein the selected first block of instructions includes a plurality of instructions.

7. The method of claim 1, wherein storing a current instruction pointer includes pushing the current instruction pointer onto a stack.

8. The method of claim 1, wherein the acts further comprise:

storing a next instruction pointer;

selecting, by the program loader, a second block of instructions of the program based at least in part on the stored next instruction pointer;

manipulating the selected second block of instructions to provide a second phantom instruction block, wherein manipulating the selected second block includes copying at least a portion of the selected second block to form the second phantom instruction block; and

executing the second phantom instruction block in the controlled environment;

wherein the selected second block includes at least one instruction.

9. The method of claim 8, wherein manipulating the selected first and second blocks further include at least one of

marking the first and second blocks as read-only, and

logically modifying at least a portion of the first and second phantom instruction blocks.

10. The method of claim 9, wherein logically modifying the at least a portion of the first and second phantom instruction blocks includes at least one of

inserting program logic into the first and second phantom instruction blocks,

deleting program logic from the first and second phantom instruction blocks, and

changing program logic in the first and second phantom instruction blocks.

11. The method of claim 1, wherein the operating system includes a thread spawning routine having at least one block of instructions, and said method further comprises:

inserting, at a beginning of the thread spawning routine, second redirection logic that directs execution control to the program loader; and

executing the thread spawning routine.

12. The method of claim 1, wherein the operating system includes an exception handling routine having at least one block of instructions, and said method further comprises:

inserting, at a beginning of the exception handling routine, second redirection logic that directs execution control to the program loader; and

executing the exception handling routine.

13. A method of executing, in a controlled environment, a program having at least one block of instructions, comprising:

initiating execution of an operating system with which the program is adapted to execute; and

for at least one of the at least one blocks of instructions, performing acts including

directing execution control to a program loader,

selecting, by the program loader, a block of instructions of the program;

manipulating the selected block of instructions to provide a phantom instruction block, and

executing the phantom instruction block in the controlled environment;

wherein each selected block includes at least one instruction, and manipulating the selected block includes copying at least a portion of the selected block to form the phantom instruction block.

14. The method of claim 13, further comprising halting said acts based on the occurrence of a halt event.

15. The method of claim 13, wherein manipulating the selected block of instructions further includes at least one of

marking the selected block as read-only, and

logically modifying at least a portion of the phantom instruction block.

16. The method of claim 15, wherein logically modifying the at least a portion of the phantom instruction block includes at least one of

inserting program logic into the phantom instruction block,

deleting program logic from the phantom instruction block, and

changing program logic in the phantom instruction block.

17. (canceled)

18. (canceled)

19. The method of claim 13, wherein the method further comprises:

inserting, within the program, first redirection logic that directs execution control to the program loader; and

executing the program;

wherein said acts further include storing a respective current instruction pointer, and the program loader selects the block of instructions of the program based at least in part on the stored respective current instruction pointer.

20. The method of claim 13, wherein the operating system includes a thread spawning routine having at least one set of instructions, and said method further comprises:

inserting, at a beginning of the thread spawning routine, second redirection logic that directs execution control to the program loader; and

executing the thread spawning routine;

wherein each set of instructions includes at least one instruction.

21. The method of claim 20, further comprising

for at least one of the at least one set of instructions, performing actions including

redirecting execution control to the program loader,

selecting, by the program loader, a set of instructions of the thread spawning routine;

manipulating the selected set of instructions to provide a phantom instruction set, and

executing the phantom instruction set in the controlled environment;

wherein each selected set of instructions includes at least one instruction, and manipulating the selected set of instructions includes copying at least a portion of the selected set of instructions to form the phantom instruction set.

22. The method of claim 13, wherein the operating system includes an exception handling routine having at least one set of instructions, and said method further comprises:

inserting, at a beginning of the exception handling routine, second redirection logic that directs execution control to the program loader; and

executing the exception handling routine;

wherein each set of instructions includes at least one instruction.

23. The method of claim 22, further comprising

for at least one of the at least one set of instructions, performing actions including

redirecting execution control to the program loader,

selecting, by the program loader, a set of instructions of the exception handling routine;

manipulating the selected set of instructions to provide a phantom instruction set, and

executing the phantom instruction set in the controlled environment;

wherein each selected set of instructions includes at least one instruction, and manipulating the selected set of instructions includes copying at least a portion of the selected set of instructions to form the phantom instruction set.

24. The method of claim 13, wherein manipulating the selected block of instructions further includes

determining if the selected block is represented by current data stored in a cache;

if the selected block is not represented by the current data stored in the cache,

copying at least a portion of the selected block to form the phantom instruction block, and

adding additional data representative of the formed phantom instruction block to the cache; and

if the selected block is represented by the current data stored in the cache, referencing the current data representative of the selected block of instruction to provide the phantom instruction block.

25. (canceled)

* * * * *