

CLAIMS

What is claimed is:

1. A host system comprising a hardware processor and a memory, the hardware processor configured to execute a virtual machine (VM), an event filter, and an introspection engine, the event filter executing within the VM, the introspection engine executing outside of the VM, the hardware processor further configured to:
 - generate a first exception in response to detecting an attempt by a software entity executing within the VM to access a memory location in a manner which violates a memory access permission;
 - in response to the first exception, switch from executing the software entity to executing the event filter; and
 - in response to a second exception, switch from executing the event filter to executing the introspection engine,wherein the event filter is configured to:
 - in response to the first exception, determine according to an event eligibility criterion whether the attempt is eligible for notification to the introspection engine;
 - in response to determining whether the attempt is eligible for notification, when the attempt is eligible, cause the processor to generate the second exception; and
 - in response to determining whether the attempt is eligible for notification, when the attempt is not eligible, prevent the processor from generating the second exception and instead, cause the processor to resume executing the software entity,and wherein the introspection engine is configured to determine whether the attempt is malicious.

- 1 2. The host system of claim 1, wherein determining whether the attempt is eligible for
2 notification comprises:
3 determining whether a section of a memory page stores a part of a protected object,
4 the section indicated by the memory location;
5 in response to determining whether the section stores the part of the protected object,
6 when the section stores the part of the protected object, determine that the
7 attempt is eligible; and
8 in response to determining whether the section stores the part of the protected object,
9 when the section does not store the part of the protected object, determine that
10 the attempt is not eligible.
- 11
- 1 3. The host system of claim 2, wherein the memory page further stores a part of an
2 unprotected object.
- 3
- 1 4. The host system of claim 2, wherein the protected object comprises a page table used
2 by the hardware processor to perform memory address translations for the VM.
- 3
- 1 5. The host system of claim 3, wherein determining whether the attempt is
2 eligible for notification further comprises determining whether the attempt
3 comprises an attempt to overwrite a value of a control bit of a page table entry
4 of the page table.
- 5
- 1 6. The host system of claim 3, wherein determining whether the attempt is
2 eligible for notification further comprises determining whether the attempt
3 comprises an attempt to overwrite a content of an address field of a page table
4 entry of the page table.
- 5
- 1 7. The host system of claim 1, wherein determining whether the attempt is malicious
2 comprises:

3 employing the introspection engine to determine whether the attempt was performed
4 by a component of an operating system executing within the virtual machine;
5 and
6 in response, when the attempt was performed by the component of the operating
7 system, determining that the attempt is not malicious.

- 8
- 1 8. A method of protecting a host system from computer security threats, wherein the host
2 system comprises a hardware processor and a memory, the hardware processor
3 configured to execute a virtual machine (VM), an event filter, and an introspection
4 engine, the event filter executing within the VM, the introspection engine executing
5 outside of the VM, the method comprising:
6 configuring the hardware processor to generate a first exception in response to detecting
7 an attempt by a software entity executing within the VM to access a memory
8 location in a manner which violates a memory access permission;
9 configuring the hardware processor to switch, in response to the first exception, from
10 executing the software entity to executing the event filter;
11 configuring the hardware processor to switch, in response to a second exception, from
12 executing the event filter to executing the introspection engine;
13 in response to the first exception, employing the event filter to determine according to an
14 event eligibility criterion whether the attempt is eligible for notification to the
15 introspection engine;
16 in response to determining whether the attempt is eligible for notification, when the
17 attempt is eligible, causing the processor to generate the second exception;
18 in response to determining whether the attempt is eligible for notification, when the
19 attempt is not eligible, preventing the processor from generating the second
20 exception and instead, causing the processor to resume executing the software
21 entity; and
22 in response to the second exception, employing the introspection engine to determine
23 whether the attempt is malicious.

24

1

9. The method of claim 8, wherein determining whether the attempt is eligible for notification comprises:

2

3

determining whether a section of a memory page stores a part of a protected object, the section indicated by the memory location;

4

5

in response to determining whether the section stores the part of the protected object,

6

when the section stores the part of the protected object, determine that the

7

attempt is eligible; and

8

in response to determining whether the section stores the part of the protected object,

9

when the section does not store the part of the protected object, determine that

10

the attempt is not eligible.

11

1

10. The method of claim 9, wherein the memory page further stores a part of an unprotected object.

2

3

1

11. The method of claim 9, wherein the protected object comprises a page table used by the hardware processor to perform memory address translations for the VM.

2

3

1

12. The method of claim 11, wherein determining whether the attempt is eligible for notification further comprises determining whether the attempt comprises an attempt to overwrite a value of a control bit of a page table entry of the page table.

2

3

4

5

1

13. The method of claim 11, wherein determining whether the attempt is eligible for notification further comprises determining whether the attempt comprises an attempt to overwrite a content of an address field of a page table entry of the page table.

2

3

4

5

1 14. The method of claim 8, wherein determining whether the attempt is malicious
2 comprises:

3 employing the introspection engine to determine whether the attempt was performed
4 by a component of an operating system executing within the virtual machine;
5 and

6 in response, when the attempt was performed by the component of the operating
7 system, determining that the attempt is not malicious.

1 15. A non-transitory computer-readable medium storing instructions which, when executed
2 by a hardware processor of a host system exposing a virtual machine, cause the host
3 system to form an event filter and an introspection engine, the event filter executing
4 within the VM, the introspection engine executing outside of the VM, wherein:
5 the introspection engine is configured to:

6 configure the hardware processor to generate a first exception in response to
7 detecting an attempt by a software entity executing within the VM to
8 access a memory location in a manner which violates a memory access
9 permission;

10 configure the hardware processor to switch, in response to the first exception,
11 from executing the software entity to executing the event filter;

12 configure the hardware processor to switch, in response to a second exception,
13 from executing the event filter to executing the introspection engine; and

14 in response to the second exception, determine whether the attempt is malicious,
15 and wherein

16 the event filter is configured to:

17 in response to the first exception, determine according to an event eligibility
18 criterion whether the attempt is eligible for notification to the introspection
19 engine;

20 in response to determining whether the attempt is eligible for notification, when
21 the attempt is eligible, cause the processor to generate the second
22 exception; and

23 in response to determining whether the attempt is eligible for notification, when
24 the attempt is not eligible, prevent the processor from generating the
25 second exception and instead, cause the processor to resume executing the
26 software entity.

27
1 16. The computer-readable medium of claim 15, wherein determining whether the
2 attempt is eligible for notification comprises:

3 determining whether a section of a memory page stores a part of a protected object,
4 the section indicated by the memory location;

5 in response to determining whether the section stores the part of the protected object,
6 when the section stores the part of the protected object, determine that the
7 attempt is eligible; and

8 in response to determining whether the section stores the part of the protected object,
9 when the section does not store the part of the protected object, determine that
10 the attempt is not eligible.

11
1 17. The computer-readable medium of claim 16, wherein the memory page further stores
2 a part of an unprotected object.

3
1 18. The computer-readable medium of claim 16, wherein the protected object comprises a
2 page table used by the hardware processor to perform memory address
3 translations for the VM.

4
1 19. The computer-readable medium of claim 18, wherein determining whether the
2 attempt is eligible for notification further comprises determining whether the

3 attempt comprises an attempt to overwrite a value of a control bit of a page
4 table entry of the page table.

5
1 20. The computer-readable medium of claim 18, wherein determining whether the
2 attempt is eligible for notification further comprises determining whether the
3 attempt comprises an attempt to overwrite a content of an address field of a
4 page table entry of the page table.

5
1 21. The computer-readable medium of claim 15, wherein determining whether the
2 attempt is malicious comprises:
3 employing the introspection engine to determine whether the attempt was performed
4 by a component of an operating system executing within the virtual machine;
5 and
6 in response, when the attempt was performed by the component of the operating
7 system, determining that the attempt is not malicious.
8

CLAIMS

1. A host system comprising a hardware processor and a memory, the hardware processor configured to execute a virtual machine (VM), an event filter, and an introspection engine, the event filter executing within the VM, the introspection engine executing outside of the VM, the hardware processor further configured to:

generate an exception in response to detecting an attempt by a software entity executing within the VM to access a memory location in a manner which violates a memory access permission; and

in response to the exception, switch from executing the software entity to executing the event filter;

wherein the event filter is configured to:

in response to the exception, determine whether the memory location currently stores a part of a page table entry used by the hardware processor to perform memory address translations for the VM, the part comprising a control bit or a memory address,

in response, when the memory location currently stores the part of the page table entry, trigger a VM exit event to cause the hardware processor to suspend execution of the VM and to switch to executing the introspection engine, and

when the memory location does not store the part of the page table entry, cause the hardware processor to resume executing the software entity without triggering the VM exit event;

and wherein the introspection engine is configured to determine whether the attempt is malicious in response to the VM exit event.

2. The host system of claim 1, wherein determining whether the attempt is malicious comprises:

employing the introspection engine to determine whether the attempt was performed by a component of an operating system executing within the virtual machine; and

in response, when the attempt was performed by the component of the operating system, determining that the attempt is not malicious.

3. The host system of claim 1, wherein the introspection engine is further configured, in preparation for execution of the event filter, to inject the event filter into another software entity executing within the VM, wherein injecting the event filter comprises writing at least a

part of the event filter to a section of the memory currently allocated for the other software entity.

4. The host system of claim 3, wherein the other software entity is a driver of an operating system of the VM.

5. The host system of claim 1, wherein the event filter is further configured to:

assemble a roster of protected memory sections, the roster including, for each protected memory section, an address of the each protected memory section and an indicator of a type of asset stored within the each protected memory section; and

determine whether the memory location currently stores the part of the page table entry according to the roster.

6. A method of protecting a host system from computer security threats, wherein the host system comprises a hardware processor and a memory, the hardware processor configured to execute a virtual machine (VM), an event filter, and an introspection engine, the event filter executing within the VM, the introspection engine executing outside of the VM, the method comprising:

configuring the hardware processor to generate an exception in response to detecting an attempt by a software entity executing within the VM to access a memory location in a manner which violates a memory access permission;

configuring the hardware processor to switch, in response to the first exception, from executing the software entity to executing the event filter, wherein executing the event filter comprises:

in response to the exception, determining whether the memory location currently stores a part of a page table entry used by the hardware processor to perform memory address translations for the VM, the part comprising a control bit or a memory address,

in response, when the memory location currently stores the part of the page table entry, triggering a VM exit event to cause the hardware processor to suspend execution of the VM and to switch to executing the introspection engine, and

when the memory location does not currently store the part of the page table entry, causing the hardware processor to resume executing the software entity without triggering the VM exit event; and

employing the introspection engine to determine whether the attempt is malicious in response to the VM exit event.

7. The method of claim 6, wherein determining whether the attempt is malicious comprises:
employing the introspection engine to determine whether the attempt was performed by a component of an operating system executing within the virtual machine; and
in response, when the attempt was performed by the component of the operating system, determining that the attempt is not malicious.

8. The method of claim 6, further comprising, in preparation for executing the event filter, employing the introspection engine to inject the event filter into another software entity executing within the VM, wherein injecting the event filter comprises writing at least a part of the event filter to a section of the memory currently allocated for the other software entity.

9. The method of claim 8, wherein the other software entity is a driver of an operating system of the VM.

10. The method of claim 6, wherein the event filter is further configured to:
assemble a roster of protected memory sections, the roster including, for each protected memory section, an address of the each protected memory section and an indicator of a type of asset stored within the each protected memory section; and
determine whether the memory location stores the part of the page table entry according to the roster.

11. A non-transitory computer-readable medium storing instructions which, when executed by a hardware processor of a host system exposing a virtual machine, cause the host system to form an event filter and an introspection engine, the event filter executing within the VM, the introspection engine executing outside of the VM, wherein:

the introspection engine is configured to:
configure the hardware processor to generate an exception in response to detecting an attempt by a software entity executing within the VM to access a memory location in a manner which violates a memory access permission,
configure the hardware processor to switch, in response to the exception, from

executing the software entity to executing the event filter, and

determine whether the attempt is malicious in response to a VM exit event triggered by the event filter; and

the event filter is configured to:

in response to the exception, determine whether the memory location currently stores a part of a page table entry used by the hardware processor to perform memory address translations for the VM, the part comprising a control bit or a memory address,

in response, when the memory location currently stores the part of the page table entry, trigger the VM exit event to cause the hardware processor to suspend execution of the VM and to switch to executing the introspection engine, and

when the memory location does not store the part of the page table entry, cause the hardware processor to resume executing the software entity without triggering the VM exit event;

12. The computer-readable medium of claim 11, wherein determining whether the attempt is malicious comprises:

employing the introspection engine to determine whether the attempt was performed by a component of an operating system executing within the virtual machine; and

in response, when the attempt was performed by the component of the operating system, determining that the attempt is not malicious.

13. The computer-readable medium of claim 11, wherein the instructions further cause the host system, in preparation for executing the event filter, to inject the event filter into another software entity executing within the VM, wherein injecting the event filter comprises writing at least a part of the event filter to a section of the memory currently allocated for the other software entity.

14. The computer-readable medium of claim 13, wherein the other software entity is a driver of an operating system of the VM.

15. The computer-readable medium of claim 11, wherein the event filter is further configured to:

assemble a roster of protected memory sections, the roster including, for each

protected memory section, an address of the each protected memory section and an indicator of a type of asset stored within the each protected memory section; and

determine whether the memory location stores the part of the page table entry according to the roster.