(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 639 500**

(51) Int. CI.:

G06F 9/455 (2006.01)
G06F 21/53 (2013.01)
G06F 21/54 (2013.01)
G06F 21/79 (2013.01)
G06F 12/08 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **02.07.2014 PCT/RO2014/000018**

(87) Fecha y número de publicación internacional: **08.10.2015 WO15152747**

(96) Fecha de presentación y número de la solicitud europea: **02.07.2014 E 14882801 (5)**

(97) Fecha y número de publicación de la concesión europea: **14.06.2017 EP 3022648**

(54) Título: **Inyección de error de página en máquinas virtuales para provocar mapeo de páginas de memoria intercambiada hacia fuera en memoria virtualizada de VM**

(30) Prioridad:

17.07.2013 US 201361847538 P
28.05.2014 US 201414289163

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
26.10.2017

(73) Titular/es:

BITDEFENDER IPR MANAGEMENT LTD. (100.0%)
Kreontos 12
1076 Nicosa, CY

(72) Inventor/es:

LUTAS, ANDREI-VLAD

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Inyección de error de página en máquinas virtuales para provocar mapeo de páginas de memoria intercambiada hacia fuera en memoria virtualizada de VM

Solicitudes relacionadas

Esta solicitud reivindica el beneficio de la fecha de presentación de la solicitud de patente provisional de Estados Unidos N.º 61/847.538, presentada el 17 de julio de 2013, titulada “Page Fault Injection In Virtual Machines”.

Antecedentes

La invención se refiere a sistemas y métodos para proteger sistemas informáticos de software maligno.

El software malicioso, también conocido como software maligno (software maligno), afecta a un gran número de sistemas informáticos por todo el mundo. En sus muchas formas, tales como virus informáticos, gusanos, rootkits y software espía, el software maligno presenta un riesgo serio a millones de usuarios informáticos, haciéndoles vulnerables a pérdida de datos e información sensible, robo de identidad y pérdida de productividad, entre otros.

La tecnología de virtualización de hardware permite la creación de entornos informáticos simulados comúnmente conocidos como máquinas virtuales, que se comportan de muchas maneras como los sistemas informáticos físicos. En aplicaciones típicas tales como consolidación de servidor e infraestructura como servicio (IAAS), varias máquinas virtuales pueden ejecutarse simultáneamente en la misma máquina física, compartiendo los recursos de hardware entre ellas, reduciendo por lo tanto los costes de inversión y de operación. Cada máquina virtual puede ejecutar su propio sistema operativo y/o aplicaciones de software, de manera separada de otras máquinas virtuales. Debido a la proliferación fija de software maligno, cada máquina virtual que opera en un entorno de este tipo potencialmente requiere protección de software maligno.

Una solución de virtualización comúnmente usada en la técnica comprende un hipervisor, también conocido como un monitor de una máquina virtual, que consiste en una capa de software que opera entre el hardware informático y el sistema operativo (SO) de una máquina virtual, y que tiene más privilegios de procesador que el respectivo SO. Las operaciones anti-software maligno pueden realizarse en el nivel de privilegio del hipervisor. Aunque tales configuraciones pueden aumentar la seguridad, introducen una capa adicional de complejidad y pueden llevar costes computacionales significativos.

Un sistema ejemplar diseñado para proteger una plataforma de virtualización contra software maligno se describe en el artículo “Detecting Past And Present Intrusions Through Vulnerability-Specific Predicates” por A. Joshi et al., Operating Systems Review vol. 39, n.º 5, páginas 91-104. El sistema descrito usa introspección de máquina virtual para monitorizar la ejecución del software de aplicación y sistema operativo en una máquina virtual.

Existe un interés considerable en desarrollar soluciones anti-software maligno eficaces, robustas y escalables para plataformas de virtualización de hardware.

Sumario

De acuerdo con un aspecto, un sistema anfitrión comprende un procesador de hardware configurado para operar un hipervisor y un motor de introspección de memoria. El hipervisor está configurado para exponer una máquina virtual que comprende un procesador virtualizado y una memoria virtualizada, la máquina virtual configurada para emplear el procesador virtualizado para ejecutar un proceso objetivo. El motor de introspección de memoria se ejecuta fuera de la máquina virtual y está configurado para determinar de acuerdo con una tabla de página de la máquina virtual si una página objetivo de un espacio de memoria virtual del proceso objetivo se intercambia fuera de la memoria virtualizada, y en respuesta, cuando la página objetivo se intercambia fuera de la memoria virtualizada, para inyectar directamente un error de página en la máquina virtual, provocando el error de página que un sistema operativo de la máquina virtual mapee la página objetivo a una página de la memoria virtualizada.

De acuerdo con otro aspecto, un método comprende emplear al menos un procesador de hardware de un sistema anfitrión para ejecutar un hipervisor, el hipervisor configurado para exponer una máquina virtual que comprende un procesador virtualizado y una memoria virtualizada, la máquina virtual configurada adicionalmente para emplear el procesador virtualizado para ejecutar un proceso objetivo. El método comprende adicionalmente emplear el al menos un procesador de hardware para ejecutar un motor de introspección de memoria fuera de la máquina virtual, emplear el motor de introspección de memoria para determinar de acuerdo con una tabla de página de la máquina virtual si una página objetivo de un espacio de memoria virtual del proceso objetivo se intercambia fuera de la memoria virtualizada, y en respuesta, cuando la página se intercambia fuera de la memoria virtualizada, emplear el motor de introspección de memoria para inyectar directamente un error de página en la máquina virtual, provocando el error de página que un sistema operativo de la máquina virtual mapee la página objetivo a una página de la memoria virtualizada.

De acuerdo con otro aspecto, un medio legible por ordenador no transitorio almacena instrucciones que, cuando se ejecutan mediante al menos un procesador de hardware de un sistema anfitrión, provocan que el sistema anfitrión forme un motor de introspección de memoria, en el que el sistema anfitrión está configurado adicionalmente para ejecutar un hipervisor que expone una máquina virtual que comprende un procesador virtualizado y una memoria virtualizada, la máquina virtual configurada para emplear un procesador virtualizado para ejecutar un proceso objetivo. El motor de introspección de memoria se ejecuta fuera de la máquina virtual y está configurado para determinar de acuerdo con una tabla de página de la máquina virtual si una página objetivo de un espacio de memoria virtual del proceso objetivo se intercambia fuera de la memoria virtualizada, y en respuesta, cuando la página objetivo se intercambia fuera de la memoria virtualizada, inyecta directamente un error de página en la máquina virtual, provocando el error de página que un sistema operativo de la máquina virtual mapee la página objetivo a una página de la memoria virtualizada.

Breve descripción de los dibujos

Los anteriores aspectos y ventajas de la presente invención se entenderán mejor tras leer la siguiente descripción detallada y tras la referencia a los dibujos donde:

La Figura 1 muestra un conjunto ejemplar de máquinas virtuales expuestas por un hipervisor que se ejecuta en un sistema anfitrión, y un motor de introspección de memoria que protege el conjunto de máquinas virtuales de software maligno de acuerdo con algunas realizaciones de la presente invención.

La Figura 2 muestra una configuración de hardware ejemplar del sistema anfitrión de acuerdo con algunas realizaciones de la presente invención.

La Figura 3 muestra una configuración ejemplar de hardware virtualizado expuesto a una máquina virtual invitada de acuerdo con algunas realizaciones de la presente invención.

La Figura 4 ilustra una jerarquía ejemplar de objetos de software que se ejecutan en el sistema anfitrión a diversos niveles de privilegio de procesador, de acuerdo con algunas realizaciones de la presente invención.

La Figura 5 muestra un mapeo de direcciones de memoria ejemplar y un intercambio ejemplar de una página de memoria page dentro y fuera de la memoria virtualizada de acuerdo con algunas realizaciones de la presente invención.

La Figura 6 muestra una secuencia ejemplar de etapas ejecutadas por el motor de introspección de memoria para proteger una máquina virtual de software maligno de acuerdo con algunas realizaciones de la presente invención.

La Figura 7 muestra una secuencia ejemplar de etapas realizadas por la introspección de memoria para llevar a cabo una inyección de error de página directa de acuerdo con algunas realizaciones de la presente invención.

La Figura 8 muestra una secuencia ejemplar de etapas que ilustra una aplicación de los métodos de las Figuras 6-7, de acuerdo con algunas realizaciones de la presente invención.

La Figura 9 muestra una secuencia ejemplar de etapas que ilustra otra aplicación de los métodos de las Figuras 6-7, de acuerdo con algunas realizaciones de la presente invención.

La Figura 10 ilustra una determinación ejemplar de un conjunto de direcciones virtuales de páginas de memoria que contienen datos de un proceso objetivo, de acuerdo con algunas realizaciones de la presente invención.

Descripción detallada de las realizaciones preferidas

En la siguiente descripción, se entiende que todas las conexiones indicadas entre las estructuras pueden ser conexiones operativas directas o conexiones operativas indirectas a través de estructuras intermedias. Un conjunto de elementos incluye uno o más elementos. Cualquier indicación de un elemento se entiende que hace referencia a al menos un elemento. Una pluralidad de elementos incluye al menos dos elementos. A menos que se requiera de otra manera, cualquier etapa del método descrita no necesita realizarse en un orden ilustrado particular. Un primer elemento (por ejemplo datos) obtenido a partir de un segundo elemento abarca un primer elemento igual al segundo elemento, así como un primer elemento generado procesando el segundo elemento y opcionalmente otros datos.

Tomar una determinación o decisión de acuerdo con un parámetro abarca tomar la determinación o decisión de acuerdo con el parámetro y opcionalmente de acuerdo con otros datos. A menos que se especifique de otra manera, un indicador de alguna cantidad/datos puede ser la misma cantidad/datos, o un indicador diferente de la misma cantidad/datos. A menos que se especifique de otra manera, un proceso es una instancia de un programa informático, tal como una aplicación o una parte de un sistema operativo, y está caracterizado por tener al menos un hilo de ejecución y una sección de memoria virtual asignada al mismo por el sistema operativo, comprendiendo la

respectiva sección código ejecutable. A menos que se especifique de otra manera, una página representa la unidad más pequeña de memoria virtual mapeada individualmente a una memoria física de un sistema anfitrión. A menos que se especifique de otra manera, inyectar directamente un error de página en una máquina virtual comprende inducir un evento de error de página en un procesador virtualizado de la respectiva máquina virtual sin asistencia desde un sistema operativo u otro software que se ejecute en la respectiva máquina virtual. Tal inyección directa no excluye el sistema operativo u otro software que tome acción en respuesta al error de página inyectado, por ejemplo para manejar el error de página. Medio legible por ordenador abarca medio no transitorio tal como medio de almacenamiento magnético, óptico y de semiconductores (por ejemplo discos duros, discos ópticos, memoria flash, DRAM), así como enlaces de comunicación tales como cables conductores y enlaces de fibra óptica. De acuerdo con algunas realizaciones, la presente invención proporciona, entre otros, sistemas informáticos que comprenden hardware (por ejemplo uno o más procesadores) programado para realizar los métodos descritos en el presente documento, así como medio legible por ordenador que codifica instrucciones para realizar los métodos descritos en el presente documento.

La siguiente descripción ilustra realizaciones de la invención a modo de ejemplo y no necesariamente por medio de limitación.

La Figura 1 muestra una configuración ejemplar de un sistema anfitrión 10 que emplea virtualización de hardware para protección de software maligno de acuerdo con algunas realizaciones de la presente invención. El sistema anfitrión 10 puede representar un dispositivo informático corporativo tal como un servidor empresarial, o un dispositivo de usuario final tal como un ordenador personal o un teléfono inteligente. Otros sistemas de anfitrión ejemplares incluyen dispositivos de entretenimiento tales como TV y consolas de juegos, o cualquier otro dispositivo que tenga una memoria y un procesador, y que requiera protección de software maligno. En el ejemplo de la Figura 1, el sistema anfitrión 10 ejecuta un conjunto de máquinas virtuales invitadas 32a-b, expuestas por un hipervisor 30. Una máquina virtual (VM) comprende una abstracción, por ejemplo, una emulación de software, de una máquina física/sistema informático real, pudiendo la VM ejecutar un sistema operativo y otras aplicaciones. El hipervisor 30 incluye software configurado para crear una pluralidad de dispositivos virtualizados, tal como un procesador virtual y un controlador de memoria virtual, y para presentar tales dispositivos virtualizados a software, en lugar de los dispositivos físicos reales del sistema anfitrión 10. En algunas realizaciones, el hipervisor 30 permite una multiplexación (compartición) por múltiples máquinas virtuales de recursos de hardware del sistema anfitrión 10. El hipervisor 30 puede gestionar adicionalmente tal multiplexación de modo que cada VM opere de manera independiente y no tenga conocimiento de otras VM que se ejecutan de manera concurrente en ejecución en el sistema anfitrión 10. Ejemplos de hipervisores conocidos incluyen el VMware vSphere™ de VMware Inc. y el hipervisor Xen de código abierto, entre otros.

Cada VM 32a-b puede ejecutar un sistema operativo (SO) invitado 34a-b, respectivamente. Un conjunto de aplicaciones ejemplares 42a-d representan de manera genérica cualquier aplicación de software, tal como aplicaciones de procesamiento de textos, procesamiento de imágenes, reproductor de medios, base de datos, calendario, gestión de contactos personal, explorador, juegos, comunicación por voz, comunicación de datos y aplicaciones anti-software maligno, entre otras. Los sistemas operativos 34a-b pueden comprender cualquier sistema operativo ampliamente disponible tal como Microsoft Windows®, MacOS®, Linux®, iOS® o Android™, entre otros. Cada SO proporciona una interfaz entre aplicaciones que se ejecutan en una máquina virtual y los dispositivos de hardware virtualizados de la respectiva VM. En la siguiente descripción, el software que se ejecuta en un procesador virtual de una máquina virtual se dice que se ejecuta en la respectiva máquina virtual. Por ejemplo, en el ejemplo de la Figura 1, las aplicaciones 42a-b se dice que se ejecutan en la VM invitada 32a, mientras que las aplicaciones 42c-d se dice que se ejecutan en la VM invitada 32b. En contraste, el hipervisor 30 se dice que se ejecuta fuera, o por debajo, de las VM invitadas 32a-b.

En algunas realizaciones, el hipervisor 30 incluye un motor de introspección de memoria 40, configurado para realizar operaciones anti-software maligno como se describe adicionalmente a continuación. El motor 40 puede incorporarse en el hipervisor 30, o puede suministrarse como un componente de software distinto e independiente del hipervisor 30, pero que se ejecuta en algún nivel de privilegio de procesador sustancialmente similar como el hipervisor 30. Un único motor 40 puede configurarse para proteger contra software maligno múltiples VM que se ejecutan en sistema anfitrión 10.

La Figura 2 muestra una configuración de hardware ejemplar de un sistema anfitrión 10. El sistema 10 comprende un conjunto de dispositivos físicos, que incluyen un procesador 12, una unidad de memoria 14, un conjunto de dispositivos de entrada 16, un conjunto de dispositivos de salida 18, un conjunto de dispositivos de almacenamiento 20 y un conjunto de adaptadores de red 22, todos conectados por un concentrador de controlador 24. En algunas realizaciones, el procesador 12 comprende un dispositivo físico (por ejemplo un circuito integrado de múltiples núcleos formado en un sustrato de semiconductores) configurado para ejecutar operaciones computacionales y/o lógicas con un conjunto de señales y/o datos. En algunas realizaciones, tales operaciones lógicas se suministran al procesador 12 en forma de una secuencia de instrucciones de procesador (por ejemplo código máquina u otro tipo de software). La unidad de memoria 14 puede comprender medio legible por ordenador volátil (por ejemplo RAM) que almacena datos/señales accedidos o generados por el procesador 12 en el transcurso de llevar a cabo las instrucciones.

Los dispositivos de entrada 16 pueden incluir teclados informáticos, ratones y micrófonos, entre otros, incluyendo las respectivas interfaces de hardware y/o adaptadores que permiten a un usuario introducir datos y/o instrucciones en el sistema anfitrión 10. Los dispositivos de salida 18 pueden incluir dispositivos de visualización tales como monitores y altavoces entre otros, así como interfaces/adaptadores de hardware tales como tarjetas gráficas, que permiten que el sistema anfitrión 10 comunique datos a un usuario. En algunas realizaciones, los dispositivos de entrada 16 y los dispositivos de salida 18 pueden compartir una pieza común de hardware, como en el caso de dispositivos de pantalla táctil. Los dispositivos de almacenamiento 20 incluyen medios legibles por ordenador que posibilitan el almacenamiento no volátil, la lectura y escritura de instrucciones y/o datos de software. Los dispositivos de almacenamiento 20 ejemplares incluyen discos magnéticos y ópticos y dispositivos de memoria flash, así como medio extraíble tal como discos y unidades de CD y/o DVD. El conjunto de adaptadores de red 22 posibilita que el sistema anfitrión 10 se conecte a una red informática y/o a otros dispositivos/sistemas informáticos. El concentrador de controlador 24 representa la pluralidad de sistemas, periféricos y/o buses de conjuntos de chips y/o toda la demás circuitería que posibilita la comunicación entre el procesador 12 y los dispositivos 14, 16, 18, 20 y 22. Por ejemplo, el concentrador de controlador 24 puede incluir un controlador de memoria, un controlador de entrada/salida (E/S) y un controlador de interrupción, entre otros. En otro ejemplo, el concentrador de controlador 24 puede comprender un procesador de conexión de puente norte 12 a la memoria 14 y/o un procesador de conexión de puente sur 12 a los dispositivos 16, 18, 20 y 22.

Para posibilitar las configuraciones como se muestra en la Figura 1, el hipervisor 30 puede crear una pluralidad de dispositivos virtualizados, emulando cada uno un dispositivo de hardware físico del sistema 10. El hipervisor 30 puede asignar adicionalmente un conjunto de dispositivos virtualizados a cada VM 32a-b, y planificación de control, señalización y comunicación de modo que las VM 32a-b pueden usar el procesador 12 y otros dispositivos de hardware de manera concurrente. La realización de tales operaciones es también conocida en la técnica como se expone las VM 32a-b.

La Figura 3 muestra una configuración ejemplar de una máquina virtual 32, como se expone por el hipervisor 30. La VM 32 puede representar, por ejemplo, cualquiera de las VM 32a-b de la Figura 1. La VM 32 incluye un procesador virtualizado 112, una unidad de memoria virtualizada 114, dispositivos de entrada virtualizados 116, dispositivos de salida virtualizados 118, almacenamiento virtualizado 120, adaptadores de red virtualizados 122 y un concentrador de controlador virtualizado 124. El procesador virtual 112 comprende una emulación de al menos alguna de la funcionalidad del procesador 12, y está configurado para recibir para ejecución instrucciones de procesador que forman parte del software, tal como el sistema operativo y otras aplicaciones. El software que usa el procesador 112 para ejecución se considera para ejecutarse en la máquina virtual 32. En algunas realizaciones, la unidad de memoria virtualizada 114 comprende espacios direccionables para almacenar y recuperar datos usados por el procesador virtual 112. Otros dispositivos virtualizados (por ejemplo, entrada, salida, almacenamiento virtualizados etc.) emulan al menos alguna de la funcionalidad de los respectivos dispositivos físicos del sistema anfitrión 10. El procesador virtual 112 puede configurarse para interactuar con tales dispositivos como se haría con los correspondientes dispositivos físicos. Por ejemplo, el software que se ejecuta en la VM 32 puede enviar y/o recibir tráfico de red mediante el adaptador o adaptadores de red virtualizados 122. En algunas realizaciones, el hipervisor 30 puede exponer únicamente un subconjunto de dispositivos virtualizados a la VM 32 (por ejemplo, únicamente el procesador virtual 112, la memoria virtualizada 114 y partes del concentrador 124). El hipervisor 30 puede proporcionar también un uso exclusivo de la VM seleccionada de algunos dispositivos de hardware del sistema anfitrión 10. En un ejemplo de este tipo, la VM 32a (Figura 1) puede tener uso exclusivo de los dispositivos de entrada 16 y los dispositivos de salida 18, pero carecer de un adaptador de red virtualizado. Mientras tanto, la VM 32b puede tener uso exclusivo del adaptador o adaptadores de red 22.

La Figura 4 ilustra una jerarquía de objetos de software que se ejecutan en sistema anfitrión 10 de acuerdo con algunas realizaciones de la presente invención. La Figura 4 se representa desde la perspectiva de niveles de privilegio del procesador, también conocido en la técnica como capas o anillos de protección. En algunas realizaciones, el hipervisor 30 toma el control del procesador 12 en el nivel más privilegiado (por ejemplo, VMXroot en las plataformas Intel® que soportan virtualización, también conocido como anillo -1, o modo raíz), creando de esta manera una plataforma de virtualización de hardware expuesta como la máquina virtual 32 a otro software que se ejecuta en el sistema anfitrión 10. Un sistema operativo 34, tal como los SO 34a-b en la Figura 2, se ejecuta en el entorno virtual de la VM 32, teniendo el SO 34 menos privilegios de procesador que el hipervisor 30 (por ejemplo, el anillo 0 en las plataformas Intel, o modo de núcleo). Un conjunto de aplicaciones 42e-f se ejecutan a privilegios de procesador menores que el SO 34 (por ejemplo, el anillo 3, o modo de usuario). Partes de las aplicaciones 42e-f pueden ejecutarse a nivel de privilegio de núcleo (por ejemplo, el controlador 36 instalado por la aplicación 42f; un controlador ejemplar 36 realiza operaciones anti-software maligno tales como detectar comportamiento indicativo de software maligno de objetos de software y/o detectar firmas indicativas de software maligno en objetos de software). De manera similar, partes del SO 34 pueden ejecutarse en modo de usuario (anillo 3).

En algunas realizaciones, el motor de introspección 40 se ejecuta sustancialmente en el mismo nivel de privilegio de procesador que el hipervisor 30, y está configurado para realizar introspección de máquinas virtuales que se ejecutan en el sistema anfitrión 10, tal como la VM 32. La introspección de una VM o de un objeto de software que se ejecuta en la respectiva VM puede comprender analizar un comportamiento del respectivo objeto de software, por ejemplo, identificar un conjunto de operaciones realizadas por el objeto (por ejemplo, emitir una llamada de sistema,

acceder a un registro del SO, descargar un fichero desde una localización remota, escribir datos a un fichero, etc.). La introspección puede comprender adicionalmente determinar direcciones de secciones de memoria que contienen partes del objeto de software, acceder a las respectivas secciones de memoria y analizar un contenido almacenado en las respectivas secciones de memoria. Otros ejemplos de introspección incluyen interceptar y/o restringir el acceso de ciertos procesos tales como secciones de memoria, por ejemplo, evitar que un proceso sobre-escriba código o datos usados por otro proceso. En algunas realizaciones, los objetos seleccionados para introspección por el motor 40 comprenden procesos, flujos de instrucciones, registros y estructuras de datos tales como tablas de página y objetos de controlador de la respectiva VM, entre otros.

Para realizar la introspección de la VM 32 en una configuración como se ilustra en la Figura 1 (es decir, desde fuera de la respectiva VM), algunas realizaciones del motor 40 emplean estructuras y mecanismos de mapeo de memoria del procesador 12. Las máquinas virtuales típicamente operan con una memoria física virtualizada, por ejemplo, la memoria 114 en la Figura 3, también conocida en la técnica como memoria física de invitado. La memoria física virtualizada comprende una representación abstracta de la memoria física real 14, por ejemplo un espacio contiguo de direcciones virtualizadas específicas para cada VM de invitado, con partes de dicho espacio mapeadas a direcciones en la memoria física 14 y/o dispositivos de almacenamiento físico 20. En sistemas configurados para soportar virtualización, tal mapeo se consigue típicamente mediante estructuras de datos especializadas controladas por el procesador 12, conocidas como traducción de dirección de segundo nivel (SLAT). Las implementaciones de SLAT conocidas incluyen tablas de página extendidas (EPT, en plataformas Intel®), y tablas de página anidadas (NPT, en plataformas AMD®). En tales sistemas, la memoria física virtualizada puede subdividirse en unidades conocidas en la técnica como páginas, representando una página la unidad más pequeña de memoria física virtualizada mapeada individualmente a memoria física mediante mecanismos tales como EPT y/o NPT, es decir, el mapeo entre memoria física y virtualizada se realiza con granularidad de página. Todas las páginas típicamente tienen un tamaño predeterminado, por ejemplo, 4 kilobytes, 2 megabytes, etc. El particionamiento de memoria física virtualizada en páginas se configura normalmente por el hipervisor 30. En algunas realizaciones, el hipervisor 30 también configura las EPT/NPT y por lo tanto el mapeo entre memoria física y memoria física virtualizada. El mapeo real (traducción) de una dirección de memoria física virtualizada a una dirección de memoria física puede comprender buscar la dirección de memoria física en una memoria intermedia de traducción adelantada (TLB) del sistema anfitrión 10. En algunas realizaciones, la traducción de dirección comprende realizar un recorrido de página, que incluye un conjunto de búsquedas de dirección sucesivas en un conjunto de tablas de página y/o directorios de página, y realizar cálculos tales como añadir un desplazamiento de una página a una dirección con relación a la respectiva página.

Algunas configuraciones de hardware permiten que el hipervisor 30 acceda con control de manera selectiva a datos almacenados en cada página de memoria física 14, por ejemplo, estableciendo los derechos de acceso de lectura, escritura y/o ejecución a la respectiva página. Tales derechos pueden establecerse, por ejemplo, modificando una entrada de la respectiva página en la EPT o NPT. El hipervisor 30 puede seleccionar por lo tanto qué objeto de software puede acceder a datos almacenados en las direcciones en cada página, y puede indicar qué operaciones están permitidas con los respectivos datos, por ejemplo, lectura, escritura, ejecución. Un intento por un objeto de software que se ejecuta en una VM para realizar una operación, tal como escribir datos en una página a la que el objeto no tiene el respectivo derecho, o ejecutar código desde una página marcada como no ejecutable, puede activar un evento de salida de máquina virtual (por ejemplo un evento VMExit en plataformas Intel). En algunas realizaciones, los eventos de salida de máquina virtual transfieren el control del procesador de la VM que ejecuta el respectivo objeto de software al hipervisor 30. Tales transferencias pueden permitir que el software que se ejecuta a nivel de privilegio del procesador del hipervisor 30 intercepte el intento de escritura o ejecución no autorizado. En algunas realizaciones, el motor de introspección 40 realiza tales intercepciones como parte de operaciones anti-software maligno.

En algunas realizaciones, el SO 34 configura un espacio de memoria virtual para un proceso tal como las aplicaciones 42e-f en la Figura 4, manteniendo un mapeo (traducción de dirección) entre el respectivo espacio de memoria virtual y la memoria física virtualizada de la VM 32, por ejemplo usando un mecanismo de tabla de página. En algunas realizaciones, el espacio de memoria virtual de proceso también está subdividido en páginas, representando tales páginas la unidad más pequeña de memoria virtual individualmente mapeada a la memoria física virtualizada por el SO 34, es decir, el mapeo de memoria física virtual a virtualizada se realiza con granularidad de página.

La Figura 5 ilustra un mapeo de direcciones de memoria ejemplar en una realización como se muestra en la Figura 4. Un objeto de software, tal como una aplicación, un proceso o una parte el sistema operativo que ejecuta la VM invitada 32, se le asigna un espacio de memoria virtual 214a por el SO invitado 34. Cuando el objeto de software intenta acceder a un contenido de una página de memoria ejemplar 60a del espacio 214a, una dirección de página 60a se traduce por el procesador virtualizado de la VM invitada 32 en una dirección de una página 60b del espacio de memoria física virtualizada 114 de la VM 32, de acuerdo con tablas de página configuradas y controladas por el SO invitado 34. El hipervisor 30, que configura y controla la memoria física virtualizada 114, a continuación mapea la dirección de la página 60b a una dirección de una página 60c en la memoria física 14 del sistema anfitrión 10, por ejemplo usando medios de SLAT como se ha analizado anteriormente.

En algunas realizaciones, el hipervisor 30 establece su propio espacio de memoria virtual 214b que comprende una representación de memoria física 14, y emplea un mecanismo de traducción (por ejemplo, tablas de página) para mapear direcciones en el espacio 214b en las direcciones en memoria física 14. En la Figura 5, un mapeo ejemplar de este tipo traduce una dirección de página 60c en una dirección de una página 60h. Tales mapeos permiten que el hipervisor 30 gestione (por ejemplo, lea desde, escriba en y acceda al control de) páginas de memoria que pertenecen a objetos de software que se ejecutan en diversas VM que se ejecutan en el sistema anfitrión 10.

La Figura 5 ilustra adicionalmente una operación de intercambio de página realizada por el SO invitado 34. El intercambio de página es una característica común de los sistemas operativos modernos, usado para gestionar de manera eficaz recursos de memoria disponibles. En algunas realizaciones, intercambiar una página fuera de memoria comprende que el SO mueva un contenido de la respectiva página de la memoria a un dispositivo de almacenamiento (por ejemplo, disco), de modo que la respectiva página pueda usarse para almacenar otros datos. En un momento más tarde, el SO puede realizar un intercambio de entrada de la página, moviendo el respectivo contenido de vuelta del almacenamiento en memoria, posiblemente en una dirección distinta de la dirección de la página original que almacena el contenido. Para completar el intercambio de entrada, el SO puede modificar una entrada de tabla de página de la respectiva página para reflejar el cambio de dirección. En el ejemplo ilustrado en la Figura 5, la página 60c se intercambia hacia fuera a una página 60d en un dispositivo de almacenamiento. Puesto que el SO 34 se ejecuta en una máquina virtual, el SO 34 ve la memoria física virtualizada 114 como su memoria física, y el dispositivo de almacenamiento virtualizado 120 como su almacenamiento físico. Por lo que intercambiar la página 60c fuera de la memoria comprende de manera eficaz mover un contenido de la página 60c al dispositivo de almacenamiento virtualizado 120. El dispositivo 120 puede comprender una abstracción creada por el hipervisor 30 del dispositivo de almacenamiento físico 20, por lo que el contenido de la página 60d puede realmente redirigirse a una página 60k en el dispositivo 20. En algunas realizaciones, el hipervisor 30 puede proporcionar a la VM invitada 32 acceso directo al dispositivo de almacenamiento 20, por ejemplo usando la tecnología VT-d de Intel®. En tales configuraciones, el dispositivo de almacenamiento virtualizado 120 puede coincidir con un dispositivo de almacenamiento del sistema anfitrión físico real 10. Para realizar un intercambio de entrada, el SO 34 puede mover el contenido de la página 60d a una página 60e de memoria física virtualizada 114. El SO anfitrión 34 puede modificar adicionalmente una entrada de tabla de página que corresponde a la página 60a para indicar una traducción de dirección desde la página 60a a la página 60e (flecha discontinua en la Figura 5). La página 60e puede mapearse a una página 60m en memoria física 14.

La Figura 6 muestra una secuencia ejemplar de etapas realizadas por el motor de introspección de memoria 40 para proteger una máquina virtual de software maligno de acuerdo con algunas realizaciones de la presente invención. Tal protección anti-software maligno incluye, por ejemplo, identificar una página (considerada en lo sucesivo la página objetivo) de un espacio de memoria de un proceso seleccionado (considerado en lo sucesivo el proceso objetivo) que se ejecuta en la respectiva VM, y proteger el contenido de la respectiva página de modificación no autorizada, por ejemplo, por una entidad de software maliciosa. En otro ejemplo, el motor de introspección 40 puede determinar si la página objetivo contiene código malicioso. El proceso objetivo puede pertenecer, por ejemplo, a una aplicación tal como las aplicaciones 42e-f, o al SO invitado 34 en la Figura 4. Cuando el proceso objetivo se ejecuta a los privilegios del procesador de nivel de usuario (por ejemplo, modo de usuario en Windows®), el contenido de la página objetivo puede no residir en memoria en ningún momento, pero en su lugar puede intercambiarse de manera ocasional dentro y fuera de la memoria por el SO. Ejecutándose fuera de la VM 32, el motor de introspección de memoria 40 puede no tener acceso directo al contenido de tales páginas de memoria intercambiadas hacia fuera.

En una secuencia de las etapas 302-304, el motor 40 espera hasta que el contexto de ejecución actual sea el del proceso objetivo, es decir, hasta que las instrucciones de ejecución actualmente pertenezcan al proceso objetivo. Determinar el contexto de ejecución actual puede comprender, por ejemplo, leer un contenido de un registro de CR3 del procesador virtual de la respectiva VM (el registro de CR3 de plataformas x86 almacena una dirección de una estructura de paginación, que identifica de manera inequívoca cada proceso de ejecución). Cuando el contexto de ejecución es el del proceso objetivo, en una secuencia de etapas 306-308, el motor 40 puede determinar si el contenido de la página objetivo se está intercambiando actualmente fuera de memoria. Cuando el contenido de la página objetivo está en memoria, en una etapa 316, el motor 40 puede continuar para realizar introspección en la página objetivo, por ejemplo, para analizar y/o proteger el contenido de la página objetivo. Cuando el contenido de la página objetivo se está intercambiando actualmente hacia fuera, en una etapa 310, el motor 40 inyecta directamente un error de página en la respectiva VM, para forzar un intercambio de entrada de la página objetivo como se describe en más detalle a continuación. A continuación, en una secuencia de etapas 312-314, el motor 40 espera hasta que la página objetivo se intercambie, es decir, hasta que el contenido de la respectiva página se mapee a la memoria física virtualizada de la respectiva VM, para realizar introspección.

Para determinar si la página objetivo reside en memoria (etapas 306-308), así como para determinar si la página objetivo se ha intercambiado (etapas 312-314), el motor de introspección de memoria 40 puede acceder a un contenido de una tabla de página establecido por el SO 34. En algunas realizaciones, un campo (por ejemplo, un bit especializado) de la entrada de tabla de página de la página objetivo indica si la respectiva página está actualmente en memoria.

La Figura 7 ilustra una secuencia ejemplar de etapas realizadas por el motor 40 para inyectar directamente un error de página, forzando de esta manera el intercambio de entrada de la página objetivo (etapa 310 en la Figura 6). En una secuencia de etapas 322-324, el motor 40 evalúa un estado actual o procesador virtual 112 para determinar si puede inyectarse una excepción de error de página de manera segura en la VM 32. La etapa 322 puede comprender evaluar la prioridad de solicitudes de interrupción actualmente bajo procesamiento. En un sistema Microsoft Windows®, una evaluación de este tipo puede comprender determinar un nivel de solicitud de interrupción actual (IRQL), por ejemplo buscando un contenido de un registro de segmento de la VM 32. Tales registros ejemplares incluyen los registros FS y/o GS de la arquitectura de procesador x86, almacenando un puntero a una estructura de datos que incluye la IRQL. En una realización ejemplar, cuando IRQL<2, inyectar un error de página se considera seguro. Cuando hay interrupciones de prioridad superior en espera (por ejemplo, IRQL≥2), las etapas 322-324 esperan a que se sirvan las solicitudes de alta prioridad.

En algunas realizaciones, la etapa 322 puede incluir determinar el nivel de privilegio (anillo) en el que se está ejecutando actualmente el procesador virtual 112. En sistemas de anfitrión que ejecutan Microsoft Windows®, mientras el procesador se está ejecutando en modo de usuario (anillo 3), IRQL es cero, por lo que la inyección de un error de página que corresponde a una página de modo de usuario puede considerarse seguro. Cuando el procesador 112 se ejecuta en modo de núcleo (anillo 0), pueden ser necesarias determinaciones adicionales para inferir si una inyección de error es segura.

Una secuencia de etapas 326-328 inyecta una excepción de error de página en la VM 32, la excepción configurada para activar un intercambio de entrada de la página objetivo. En una realización ejemplar, en la etapa 326, el motor 40 escribe una dirección virtual de la página objetivo en el registro de CR2 del procesador virtual de la respectiva VM, indicando al SO 34 qué página virtual intercambiar en memoria. A continuación, en la etapa 328, el motor 40 activa la excepción en el procesador virtual 112, por ejemplo escribiendo en un conjunto de bits de control de una estructura de control de máquina virtual (VMCS) de la VM 32, los respectivos bits de control configurables para activar un error de página en la respectiva VM. En procesadores Intel® configurados para soportar virtualización, tales bits de control son parte del campo de Inyección de Evento de Entrada de VM de la VMCS.

Las estructuras de control de VM son un tipo especial de estructuras de datos mantenidas por el hipervisor 30 para describir las VM invitadas que se ejecutan en sistema anfitrión 10. El formato de VMCS puede ser específico de la implementación y/o de la plataforma. Para las VM que comprenden múltiples procesadores virtuales 112, el hipervisor 30 puede mantener un VMCS distinto para cada procesador virtual. En algunas realizaciones, cada VMCS puede comprender un área de estado invitada y un área de estado anfitrión, almacenando el área de estado invitada datos tales como el estado de CPU y/o contenido de registros de control del respectivo procesador virtual, y almacenando el área de estado de anfitrión datos similares para el hipervisor 30. En algunas realizaciones, el procesador 12 asocia una región en memoria con cada VMCS, denominada la región de VMCS. El software puede hacer referencia a una VMCS específica usando una dirección de la región (por ejemplo, un puntero de VMCS). En cualquier momento dado, como máximo puede cargarse una VMCS en el procesador 12, que representa la VM que actualmente tiene el control del procesador.

Las Figuras 8-9 muestran aplicaciones ejemplares de algunas realizaciones de la presente invención en un entorno Windows®. La Figura 8 ilustra una secuencia de etapas realizadas por el motor 40 para determinar una dirección de memoria virtual de un principal ejecutable de un proceso objetivo. En una etapa 332, el motor 40 puede detectar un lanzamiento del proceso objetivo. La etapa 332 puede emplear cualquier método conocido en la técnica, tal como interceptar un mecanismo del SO que gestiona la lista de procesos activos. Por ejemplo, en Windows®, cada vez que se crea un proceso, un indicador del respectivo proceso se inserta en la lista de procesos activos; el indicador se elimina de la lista tras la terminación del respectivo proceso. En algunas realizaciones, tras lanzar un proceso, el SO 34 también establece una estructura de datos específica de proceso conocida como un bloque de entorno de proceso (PEB) que comprende datos usados por el SO 34 para gestionar recursos asociados con el respectivo proceso. Interceptando (por ejemplo, colocando un gancho en) una instrucción de SO para insertar el proceso objetivo en la lista de procesos activos, el motor 40 puede obtener información tal como una dirección de memoria del respectivo PEB, que el motor 40 puede extraer en una etapa 334. En Windows, la dirección virtual del PEB se almacena en una estructura de datos del SO, conocido como el bloque de proceso ejecutivo (EPROCESS). La Figura 10 muestra un diagrama ilustrativo de tales estructuras de datos específicas de proceso.

Siendo una estructura de datos de nivel de usuario, la página de memoria virtual que contiene datos de PEB puede o puede no residir actualmente en memoria. En una etapa 336 el motor 40 determina si se intercambia hacia fuera la respectiva página de memoria virtual, y si no, en una etapa 340 el motor 40 continúa para determinar una dirección virtual del principal ejecutable del proceso objetivo, por ejemplo analizando los datos de PEB. Cuando los datos de PEB se intercambian actualmente fuera de la memoria, una etapa 338 fuerza un intercambio de entrada de la respectiva página que contiene datos de PEB, usando por ejemplo, un mecanismo como se ha descrito anteriormente en relación con la Figura 7.

La Figura 9 muestra una secuencia ejemplar de etapas llevadas a cabo por el motor 40 para realizar introspección de memoria de un módulo ejecutable (tal como una biblioteca) cargada por el proceso objetivo. El software maligno

a menudo usa las DLL como vectores para llevar a cabo código malicioso, por lo que analizar el contenido de tales bibliotecas puede ser importante para operaciones anti-software maligno. Después de acceder a la página virtual que contiene datos de PEB en una etapa 342 (véase por ejemplo, las etapas 336-338 anteriores), en una secuencia de etapas 344-346-348 el motor 40 identifica un módulo objetivo, por ejemplo, una biblioteca de enlace dinámico (DLL), usada por el proceso objetivo, y determina si se ha cargado el respectivo módulo. Cuando se ha cargado el módulo objetivo, el motor 40 puede determinar una dirección virtual del respectivo módulo en una etapa 350, por ejemplo de acuerdo con un campo de datos específico del PEB (véase por ejemplo, la Figura 10). En una etapa 352, el motor 40 determina si la página virtual que contiene los datos de módulo y que residen en la dirección determinada en la etapa 350 se está intercambiando actualmente fuera de la memoria, y cuando no, en una etapa 356, continúa para realizar introspección de memoria del respectivo módulo. Cuando la página virtual del respectivo módulo se está intercambiando hacia fuera actualmente, en una etapa 354, el motor 40 fuerza un intercambio de entrada de la respectiva página virtual usando, por ejemplo, el mecanismo anteriormente descrito en relación con la Figura 7.

La Figura 10 muestra una determinación ejemplar de direcciones virtuales en un entorno Windows®. En algunas realizaciones, el SO 34 mantiene un espacio de memoria virtual de núcleo 214d, en el que una página localizada en una dirección virtual 60p contiene una parte de la estructura de EPROCESS usada por el SO 34 para gestionar la ejecución del proceso objetivo. La dirección 60p puede determinarse, por ejemplo, interceptando el lanzamiento del proceso objetivo (véase, por ejemplo, la etapa 332 en la Figura 8). Un campo de la estructura de datos de EPROCESS mantiene un indicador (por ejemplo, un puntero) del bloque de entorno de proceso (PEB) del proceso objetivo. El puntero indica una dirección virtual 60q en una memoria virtual de proceso 214e asignada al proceso objetivo por el SO 34. La estructura de PEB incluye adicionalmente un puntero a una estructura (datos de LDR) que contiene información acerca de módulos ejecutables (por ejemplo bibliotecas) cargadas por el proceso objetivo. Los datos de LDR se localizan en una dirección 60r en el espacio 214e. Recorriendo la jerarquía de estructuras de datos de gestión de procesos establecidas por el SO 34, el motor de introspección 40 puede determinar por lo tanto una pluralidad de direcciones virtuales de objetos dirigidos para introspección. Cuando el contenido de las páginas de memoria localizadas en tales direcciones se intercambia fuera de la RAM, el motor 40 puede forzar al SO 34 para intercambiar las respectivas páginas al usar los métodos descritos en el presente documento.

Los sistemas y métodos ejemplares anteriormente descritos permiten proteger un sistema anfitrión de software maligno que usa tecnología de virtualización. En algunas realizaciones, un motor de introspección de memoria opera por debajo de las máquinas virtuales que se ejecutan en el sistema anfitrión. El motor de introspección de memoria puede proteger una máquina virtual analizando los contenidos de una página de memoria usada por un proceso que se ejecuta en la respectiva máquina virtual. El motor de introspección puede por lo tanto determinar, desde fuera de la respectiva VM, si el código del respectivo proceso contiene software maligno.

En algunas realizaciones, el motor de introspección puede evitar también una modificación no autorizada (por ejemplo, por software maligno) de algunos objetos críticos, tal como ciertos controladores y tablas de página, entre otros. Para proteger un objeto de este tipo, algunas realizaciones pueden evitar cambios interceptando un intento de escritura de una página de memoria asignada al respectivo objeto. Tales intercepciones pueden realizarse desde el nivel del hipervisor.

En sistemas anti-software maligno convencionales, las aplicaciones de seguridad se ejecutan en un nivel de privilegio de procesador similar al del sistema operativo o de aplicaciones comunes. Tales sistemas pueden ser vulnerables a software maligno avanzado que también opera en el nivel de privilegio del sistema operativo. En contraste, en algunas realizaciones de la presente invención, un hipervisor se ejecuta en el nivel más privilegiado (por ejemplo, modo de raíz o anillo -1), desplazando el sistema operativo a una máquina virtual. El motor de introspección de memoria puede ejecutarse en el mismo nivel de privilegio de procesador que el hipervisor. Las operaciones anti-software maligno pueden por lo tanto realizarse desde un nivel de privilegio de procesador más alto que el del sistema operativo. En algunas realizaciones, un único motor de introspección de memoria puede proteger múltiples máquinas virtuales que se ejecutan concurrentemente en el respectivo sistema informático.

Aunque el motor de introspección de memoria se ejecuta fuera de la máquina virtual dirigida para protección, el motor puede determinar direcciones virtuales usadas por objetos de software que se ejecutan en la VM protegida. Sin embargo, cuando tales direcciones virtuales apuntan a contenido de páginas que se están intercambiando actualmente fuera de la memoria por el sistema operativo, el motor de introspección de memoria no tiene acceso al respectivo contenido. En algunas realizaciones de la presente invención, cuando una página se está intercambiando actualmente hacia fuera, el motor de introspección de memoria puede forzar al SO a que se intercambie la respectiva página de entrada, poniendo el contenido de la respectiva página a disposición para análisis y/o protección. Para forzar el intercambio de entrada, el motor de introspección de memoria puede activar un evento de procesador, tal como una excepción de error de página, en el procesador virtualizado de la respectiva máquina virtual, el evento de procesador configurado para provocar que el sistema operativo proporcione la página intercambiada de vuelta a memoria. Activar el evento de procesador puede comprender, por ejemplo, escribir en un conjunto de bits de control de una estructura de control de máquina virtual usada por la respectiva máquina virtual.

El motor de introspección de memoria puede inyectar por lo tanto un error de página en la respectiva máquina virtual sin asistencia del SO o desde otro software que se ejecuta en la respectiva VM.

Será evidente para un experto en la materia que las realizaciones anteriores pueden modificarse de muchas maneras sin alejarse del alcance de la invención. Por consiguiente, el alcance de la invención debería determinarse por las siguientes reivindicaciones y sus equivalentes legales.

REIVINDICACIONES

1. Un sistema anfitrión [10] que comprende un procesador de hardware [12] configurado para operar:

un hipervisor [30] configurado para exponer una máquina virtual [32] que comprende un procesador virtualizado [112] y una memoria virtualizada [114], la máquina virtual [32] configurada para emplear el procesador virtualizado [112] para ejecutar un proceso objetivo, en el que exponer la máquina virtual [32] comprende configurar una estructura de datos para almacenar un estado actual del procesador virtualizado [112], comprendiendo la estructura de datos un campo de inyección de evento que, cuando se establece a un valor predeterminado, provoca que el procesador virtualizado [112] genere un error de página; y
un motor de introspección de memoria [40] que se ejecuta fuera de la máquina virtual [32],
en el que el sistema anfitrión [10] está caracterizado por tener el motor de introspección de memoria [40] configurado para:

determinar de acuerdo con una tabla de página de la máquina virtual [32] si una página objetivo [60a] de un espacio de memoria virtual del proceso objetivo se intercambia fuera de la memoria virtualizada [114]; y
en respuesta, cuando la página objetivo [60a] se intercambia fuera de la memoria virtualizada [114], inyecta directamente el error de página en la máquina virtual [32], provocando el error de página que un sistema operativo [34] de la máquina virtual [32] mapee la página objetivo [60a] a una página [60c] de la memoria virtualizada [114], en el que la inyección de manera directa del error de página comprende que el motor de introspección de memoria [40] escriba el valor predeterminado al campo de inyección de evento.

2. El sistema anfitrión de la reivindicación 1, en el que inyectar directamente el error de página comprende que el motor de introspección de memoria [40] escriba una dirección virtual de la página objetivo [60a] a un registro del procesador virtualizado [112].

3. El sistema anfitrión de la reivindicación 1, en el que el motor de introspección de memoria [40] está configurado adicionalmente para:

en la preparación para inyectar directamente el error de página, determinar si se satisface una condición de inyección de evento de acuerdo con el estado actual del procesador virtualizado [112]; y
en respuesta, inyectar directamente el error de página cuando se satisface la condición de inyección de evento.

4. El sistema anfitrión de la reivindicación 3, en el que determinar si se satisface la condición de inyección de evento comprende determinar un nivel de solicitud de interrupción actual (IRQL) del procesador virtualizado [112].

5. El sistema anfitrión de la reivindicación 3, en el que determinar si se satisface la condición de inyección de evento comprende determinar un nivel de privilegio en el que se está ejecutando actualmente el procesador virtualizado [112].

6. El sistema anfitrión de la reivindicación 3, en el que determinar si se satisface la condición de inyección de evento comprende determinar un contexto de ejecución actual del procesador virtualizado [112].

7. El sistema anfitrión de la reivindicación 1, en el que el motor de introspección de memoria [40] está configurado adicionalmente, en respuesta a inyectar directamente el error de página, para:

detectar la modificación de una entrada de tabla de página de la página objetivo [60a]; y
en respuesta, determinar si la página objetivo [60a] se mapeó a la página [60c] de la memoria virtualizada [114] de acuerdo con la modificación.

8. El sistema anfitrión de la reivindicación 1, en el que el motor de introspección de memoria [40] está configurado adicionalmente para determinar si el proceso objetivo es malicioso de acuerdo con un contenido de la página objetivo [60a].

9. El sistema anfitrión de la reivindicación 1, en el que el motor de introspección de memoria [40] está configurado adicionalmente para interceptar un intento de modificar un contenido de la página objetivo [60a].

10. El sistema anfitrión de la reivindicación 1, en el que el motor de introspección de memoria [40] está configurado adicionalmente, en la preparación para determinar si la página objetivo [60a] se intercambia fuera de la memoria virtualizada [114], para:

detectar un evento del procesador virtualizado [112], el evento indicativo de un lanzamiento del proceso objetivo en la máquina virtual [32]; y
en respuesta, determinar una dirección virtual de la página objetivo [60a] de acuerdo con el evento.

11. Un método que comprende:

emplear al menos un procesador de hardware [12] de un sistema anfitrión [10] para ejecutar un hipervisor [30], el hipervisor [30] configurado para exponer una máquina virtual [32] que comprende un procesador virtualizado [112] y una memoria virtualizada [114], la máquina virtual [32] configurada para emplear el procesador virtualizado [112] para ejecutar un proceso objetivo, en el que exponer la máquina virtual [32] comprende configurar una estructura de datos para almacenar un estado actual del procesador virtualizado [112], comprendiendo la estructura de datos un campo de inyección de evento que, cuando se establece a un valor predeterminado, provoca que el procesador virtualizado [112] genere un error de página, y emplear el al menos un procesador de hardware [12] para ejecutar un motor de introspección de memoria [40] fuera de la máquina virtual [32]

en el que el método está caracterizado por las etapas de:

emplear el motor de introspección de memoria [40] para determinar de acuerdo con una tabla de página de la máquina virtual [32] si una página objetivo [60a] de un espacio de memoria virtual del proceso objetivo se intercambia fuera de la memoria virtualizada [114]; y
en respuesta, cuando la página [60a] se intercambia fuera de la memoria virtualizada [114], emplear el motor de introspección de memoria [40] - para inyectar directamente
un error de página en la máquina virtual [32], provocando el error de página que un sistema operativo [34] de la máquina virtual [32] mapee la página objetivo [60a] a una página [60c] de la memoria virtualizada [114], en el que inyectar directamente el error de página comprende que el motor de introspección de memoria [40] escriba el valor predeterminado al campo de inyección de evento.

12. El método de la reivindicación 11, en el que inyectar directamente el error de página comprende escribir una dirección virtual de la página objetivo [60a] a un registro del procesador virtualizado [112].

13. El método de la reivindicación 11, que comprende adicionalmente, en la preparación para inyectar directamente el error de página:

Emplear el motor de introspección de memoria [40] para determinar si se satisface una condición de inyección de evento de acuerdo con el estado actual del procesador virtualizado [112]; y
en respuesta, emplear el motor de introspección de memoria [40] para inyectar directamente el error de página cuando se satisface la condición de inyección de evento.

14. El método de la reivindicación 13, en el que determinar si se satisface la condición de inyección de evento comprende determinar un nivel de solicitud de interrupción actual (IRQL) del procesador virtualizado [112].

15. El método de la reivindicación 13, en el que determinar si se satisface la condición de inyección de evento comprende determinar un nivel de privilegio en el que se está ejecutando actualmente el procesador virtualizado [112].

16. El método de la reivindicación 13, en el que determinar si se satisface la condición de inyección de evento comprende determinar un contexto de ejecución actual del procesador virtualizado [112].

17. El método de la reivindicación 11, que comprende adicionalmente, en respuesta a inyectar directamente el error de página:

emplear el motor de introspección de memoria [40] para detectar una modificación de una entrada de tabla de página de la página objetivo [60a]; y
en respuesta, emplear el motor de introspección de memoria [40] para determinar si la página objetivo [60a] se mapeó a la página de la memoria virtualizada [114] de acuerdo con la modificación.

18. El método de la reivindicación 11, que comprende adicionalmente emplear el motor de introspección de memoria [40] para determinar si el proceso objetivo es malicioso de acuerdo con un contenido de la página objetivo [60a].

19. El método de la reivindicación 11, que comprende adicionalmente emplear el motor de introspección de memoria [40] para interceptar un intento de modificar un contenido de la página objetivo [60a].

20. El método de la reivindicación 11, que comprende adicionalmente, en la preparación para determinar si la página objetivo [60a] se intercambia fuera de la memoria virtualizada [114]:

emplear el motor de introspección de memoria [40] para detectar un evento del procesador virtualizado [112], el evento indicativo de un lanzamiento del proceso objetivo en la máquina virtual [32]; y
en respuesta, emplear el motor de introspección de memoria [40] para determinar una dirección virtual de la página objetivo [60a] de acuerdo con el evento.

21. Un medio legible por ordenador no transitorio que almacena instrucciones que, cuando se ejecutan mediante al menos un procesador de hardware [12] de un sistema anfitrión [10], provoca que el sistema anfitrión [10] forme un motor de introspección de memoria [40], en el que el sistema anfitrión ejecuta un hipervisor [30] que expone una máquina virtual [32] que comprende un procesador virtualizado [112] y una memoria virtualizada [114], la máquina virtual [32] configurada para emplear el procesador virtualizado [112] para ejecutar un proceso objetivo, en el que exponer la máquina virtual [32] comprende configurar una estructura de datos para almacenar un estado actual del procesador virtualizado [112], comprendiendo la estructura de datos un campo de inyección de evento que, cuando se establece a un valor predeterminado, provoca que el procesador virtualizado [112] genere un error de página, en el que el motor de introspección de memoria [40] se ejecuta fuera de la máquina virtual [32],
y en el que el medio legible por ordenador está caracterizado por tener el motor de introspección de memoria [40] configurado para:

determinar de acuerdo con una tabla de página de la máquina virtual [32] si una página objetivo [60a] de un espacio de memoria virtual del proceso objetivo se intercambia fuera de la memoria virtualizada [114]; y

en respuesta, cuando la página objetivo [60a] se intercambia fuera de la memoria virtualizada [114], inyectar directamente el error de página en la máquina virtual [32], provocando el error de página que un sistema operativo [34] de la máquina virtual [32] mapee la página objetivo [60a] a una página [60c] de la memoria virtualizada [114], en el que inyectar directamente el error de página comprende que el motor de introspección de memoria [40] escriba el valor predeterminado al campo de inyección de evento.

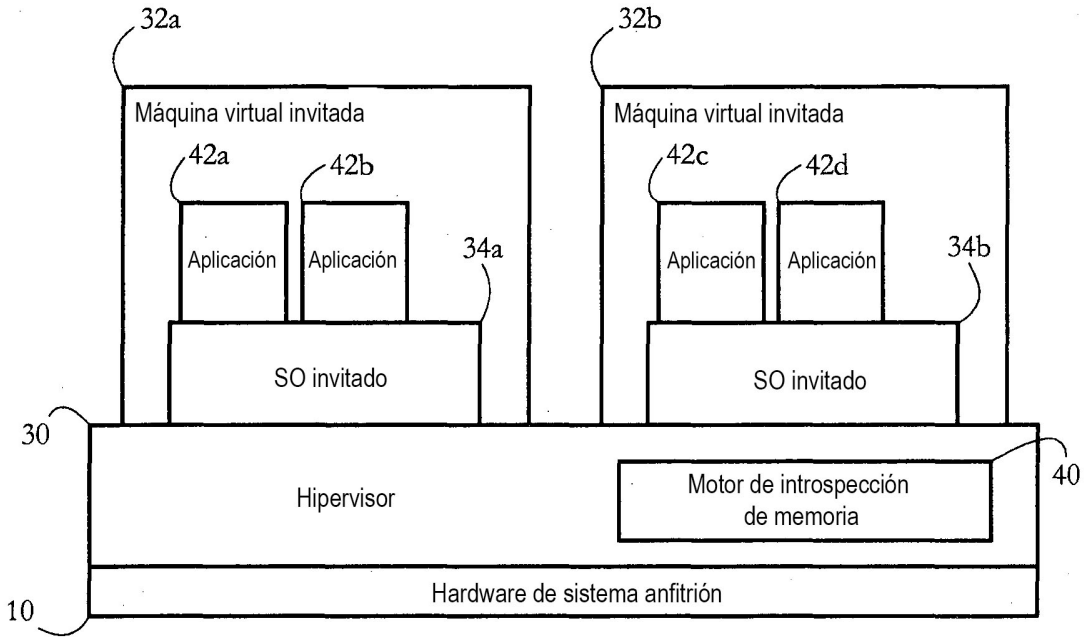
32a
32b
Máquina virtual invitada
Máquina virtual invitada
42a
42b
42c
42d
Aplicación
Aplicación
Aplicación
Aplicación
34a
34b
SO invitado
SO invitado
30
Hipervisor
Motor de introspección de memoria
40
Hardware de sistema anfitrión
10

FIG. 1

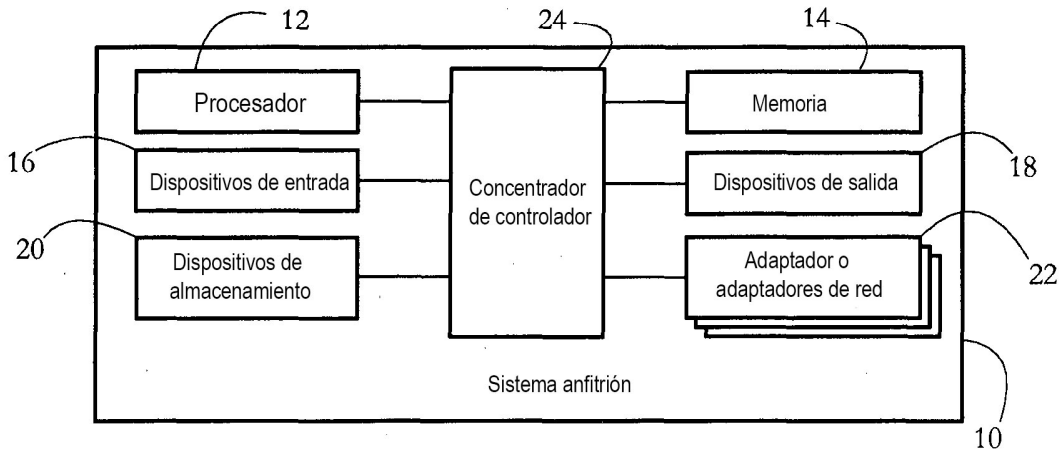
12
24
14
Procesador
Memoria
16
Dispositivos de entrada
Concentrador de controlador
Dispositivos de salida
18
20
Dispositivos de almacenamiento
Adaptador o adaptadores de red
22
Sistema anfitrión
10

FIG. 2

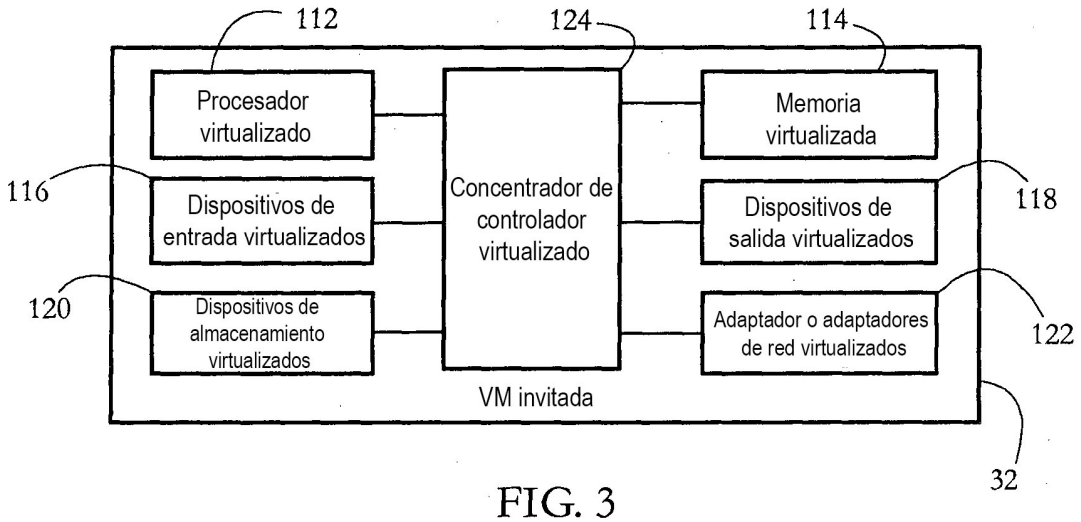
112
124
114
Procesador virtualizado
Memoria virtualizada
116
Dispositivos de entrada virtualizados
Concentrador de controlador virtualizado
Dispositivos de salida virtualizados
118
120
Dispositivos de almacenamiento virtualizados
Adaptador o adaptadores de red virtualizados
122
VM invitada
32

FIG. 3

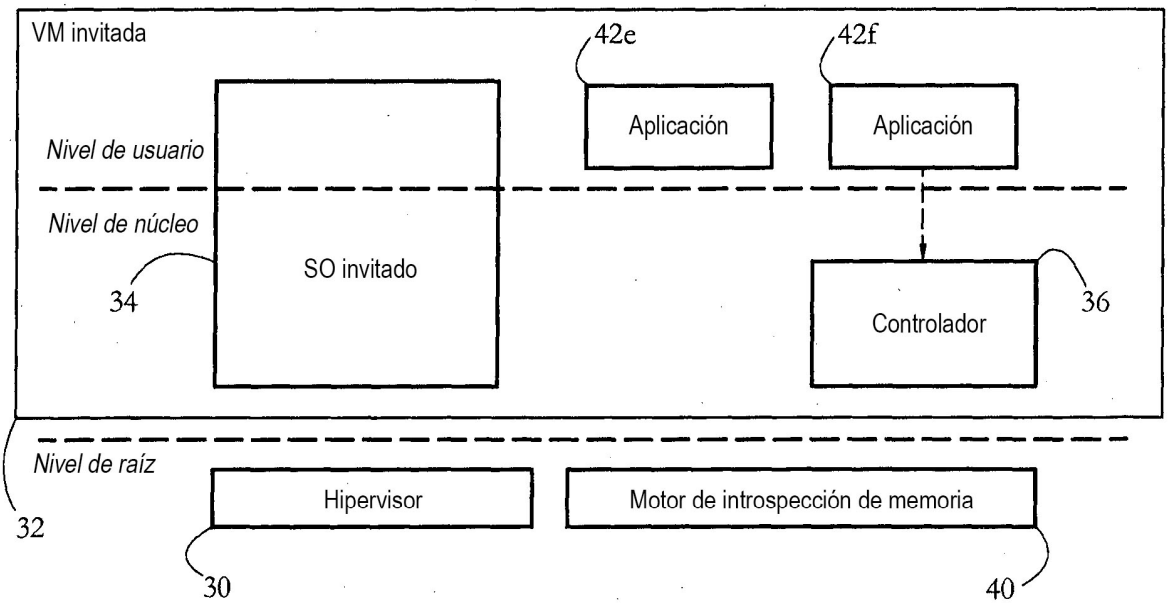
VM invitada
42e
42f
Aplicación
Aplicación
Nivel de usuario
Nivel de núcleo
SO invitado
34
Controlador
36
Nivel de raíz
32
Hipervisor
Motor de introspección de memoria
30
40

FIG. 4

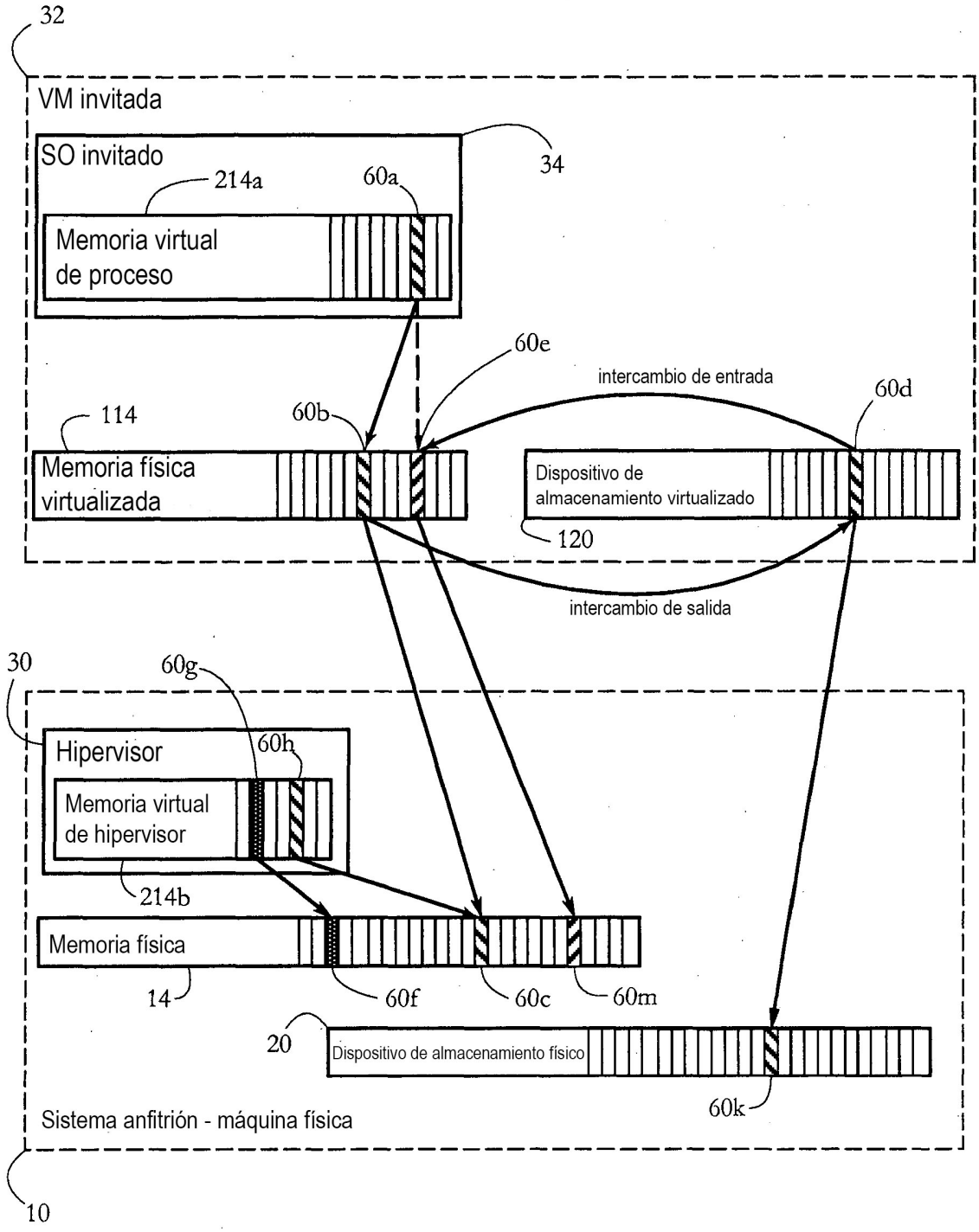
32
VM invitada
SO invitado
214a
60a
34
Memoria virtual
de proceso
60e
intercambio de entrada
60d
114
60b
Memoria física
virtualizada
Dispositivo de
almacenamiento virtualizado
120
intercambio de salida
30
60g
Hipervisor
60h
Memoria virtual
de hipervisor
214b
Memoria física
14
60f
60c
60m
20
Dispositivo de almacenamiento físico
60k
Sistema anfitrión - máquina física
10

FIG. 5

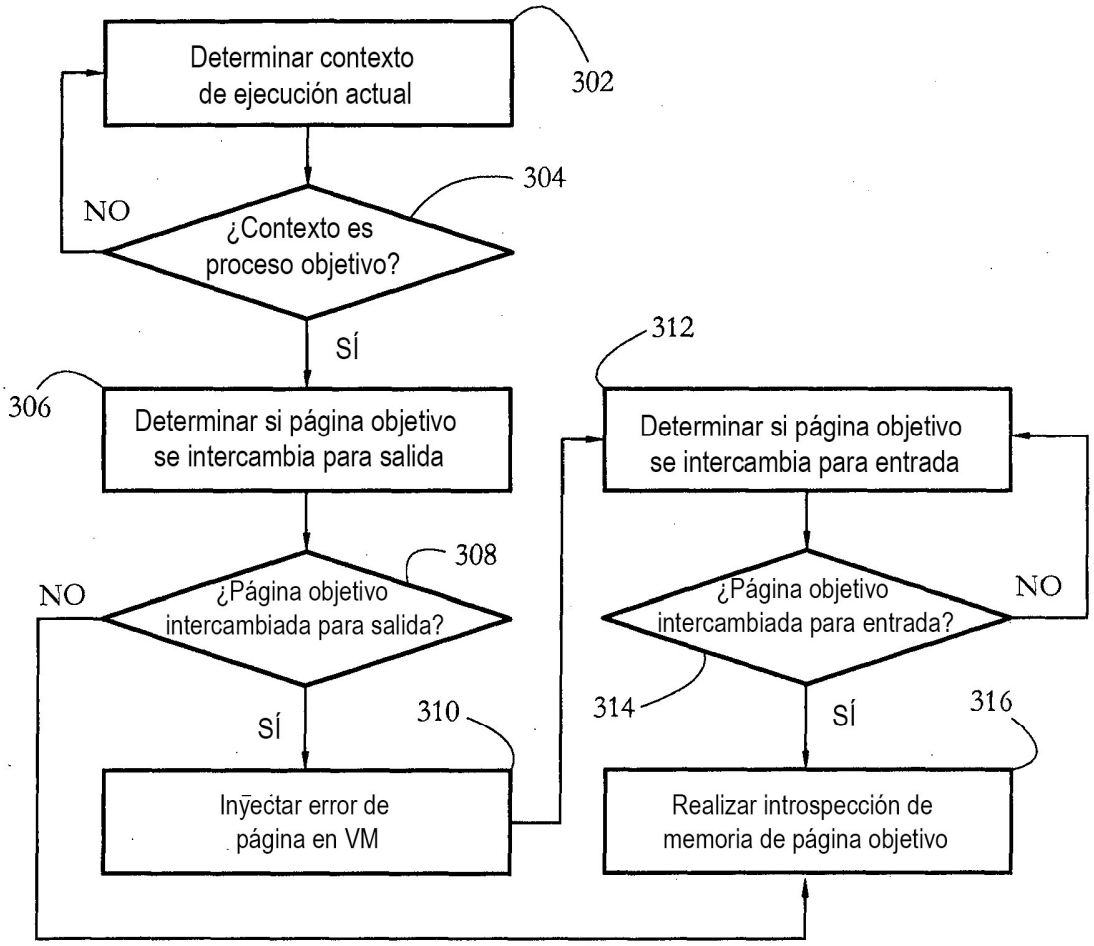
Determinar contexto de ejecución actual
302
304
NO
¿Contexto es proceso objetivo?
SÍ
306
Determinar si página objetivo se intercambia para salida
308
NO
¿Página objetivo intercambiada para salida?
SÍ
310
Inyectar error de página en VM
312
Determinar si página objetivo se intercambia para entrada
¿Página objetivo intercambiada para entrada?
NO
314
SÍ
316
Realizar introspección de memoria de página objetivo

FIG. 6

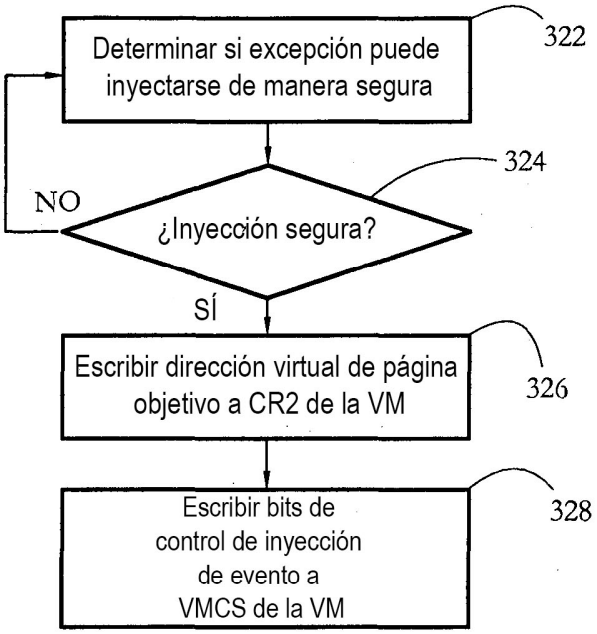
Determinar si excepción puede inyectarse de manera segura
322
¿Inyección segura?
324
NO
SÍ
Escribir dirección virtual de página objetivo a CR2 de la VM
326
Escribir bits de control de inyección de evento a VMCS de la VM
328

FIG. 7

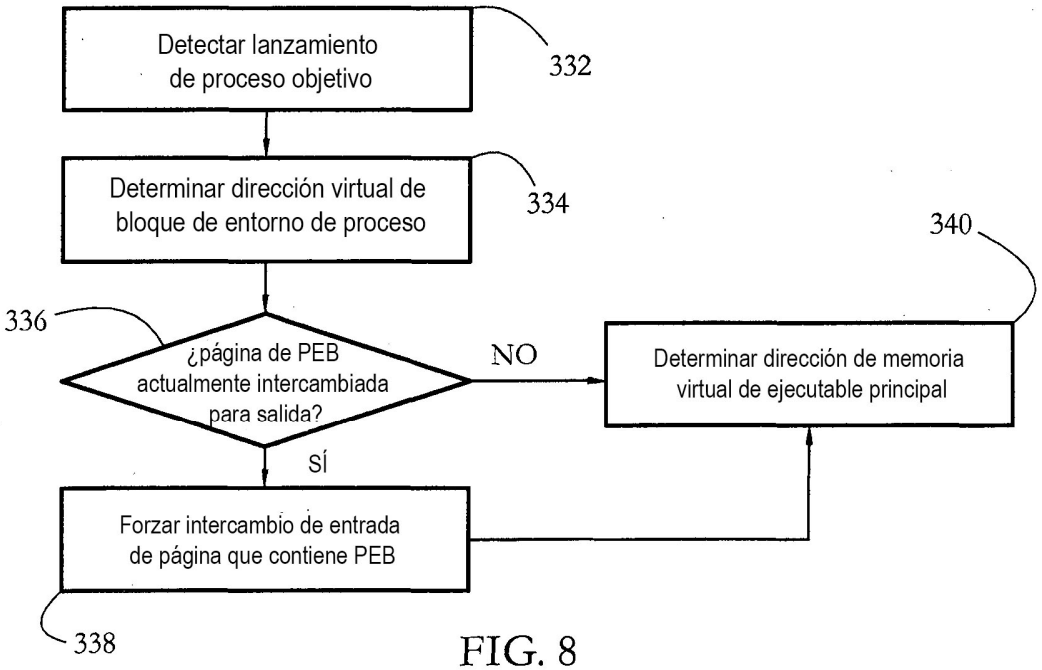
Detectar lanzamiento de proceso objetivo
332
Determinar dirección virtual de bloque de entorno de proceso
334
336
¿página de PEB actualmente intercambiada para salida?
NO
SÍ
340
Determinar dirección de memoria virtual de ejecutable principal
Forzar intercambio de entrada de página que contiene PEB
338

FIG. 8

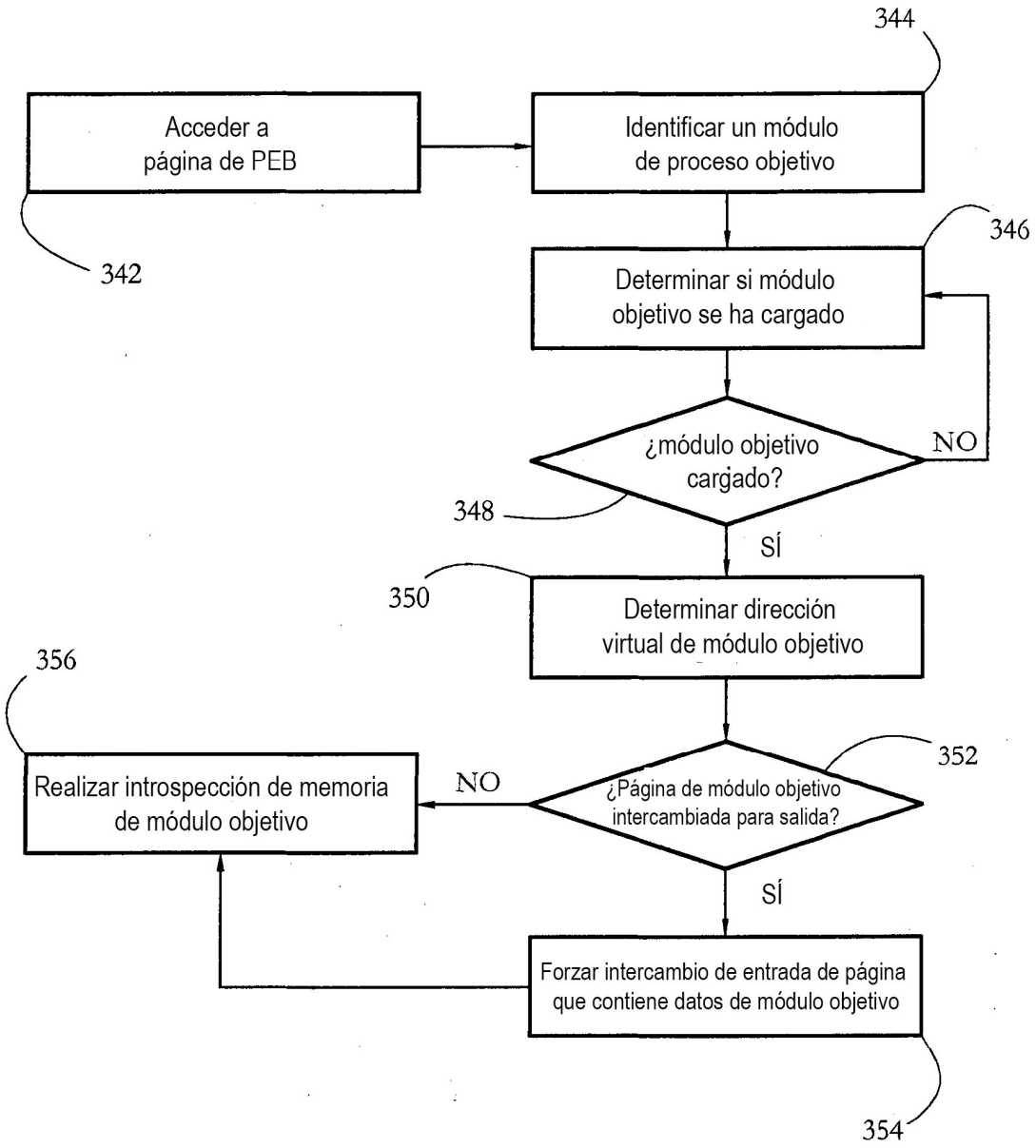
344
Acceder a página de PEB
Identificar un módulo de proceso objetivo
342
346
Determinar si módulo objetivo se ha cargado
¿módulo objetivo cargado?
NO
348
SÍ
350
Determinar dirección virtual de módulo objetivo
356
352
Realizar introspección de memoria de módulo objetivo
NO
¿Página de módulo objetivo intercambiada para salida?
SÍ
Forzar intercambio de entrada de página que contiene datos de módulo objetivo
354

FIG. 9

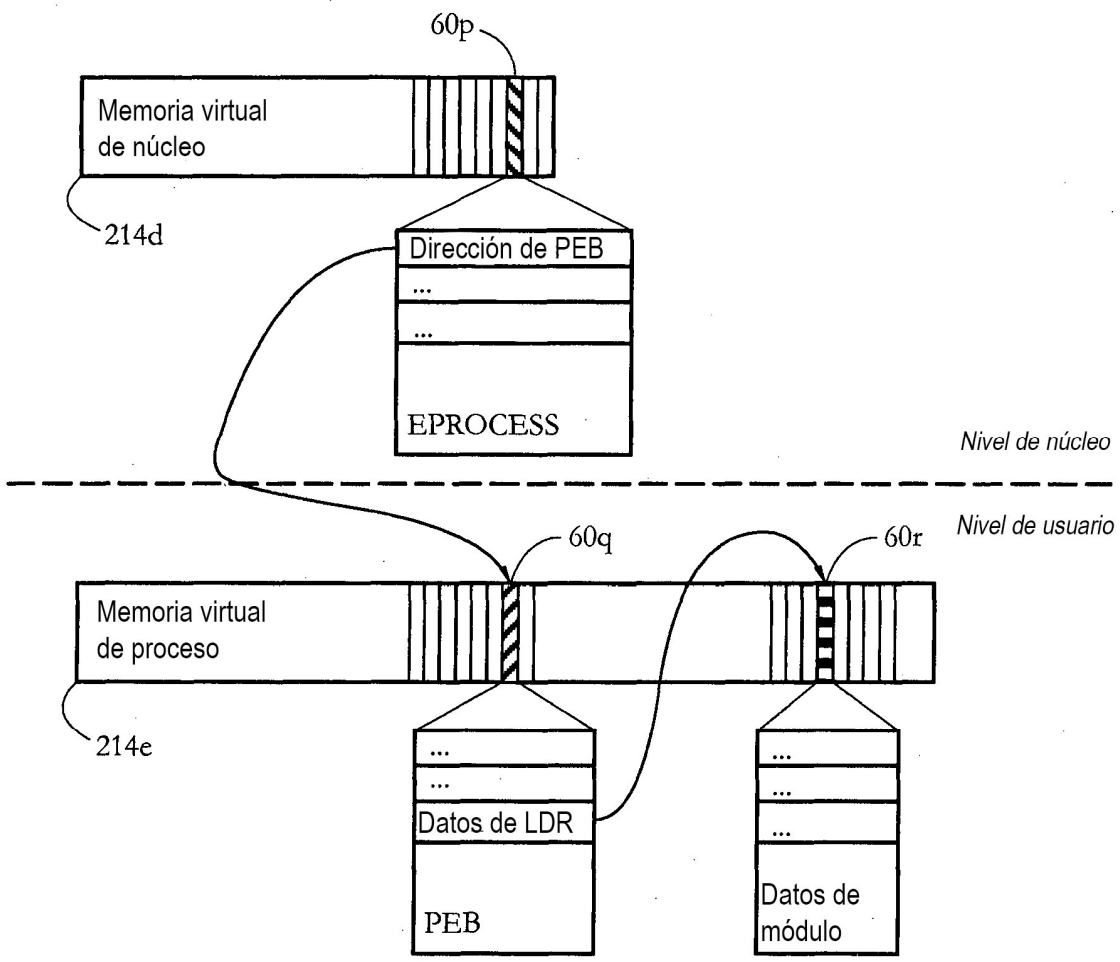
60p
Memoria virtual de núcleo
214d
Dirección de PEB
...
...
EPROCESS
Nivel de núcleo
Nivel de usuario
60q
60r
Memoria virtual de proceso
214e
...
...
Datos de LDR
PEB
...
...
...
Datos de módulo

FIG. 10