(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 960 375**

(51) Int. Cl.:

G06F 21/52 (2013.01)
G06F 21/56 (2013.01)
G06F 21/55 (2013.01)
G06F 21/57 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **05.11.2017 PCT/IL2017/051206**

(87) Fecha y número de publicación internacional: **11.05.2018 WO18083702**

(96) Fecha de presentación y número de la solicitud europea: **05.11.2017 E 17867010 (5)**

(97) Fecha y número de publicación de la concesión europea: **26.07.2023 EP 3535681**

(54) Título: **Sistema y método para detectar y para alertar de aprovechamientos en sistemas informatizados**

(30) Prioridad:

07.11.2016 US 201662418294 P
07.06.2017 US 201762516126 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
04.03.2024

(73) Titular/es:

PERCEPTION POINT LTD (100.0%)
9 Ahad Ha'am St.
6525101 Tel Aviv, IL

(72) Inventor/es:

LEVIN, SHLOMI y
AMINOV, MICHAEL

(74) Agente/Representante:

LINAGE GONZÁLEZ, Rafael

DESCRIPCIÓN

Sistema y método para detectar y para alertar de aprovechamientos en sistemas informatizados

Campo de la invención

La presente invención se refiere en general a la detección de aprovechamientos en sistemas informatizados. Más particularmente, la presente invención se refiere a la emisión de alertas tras la detección de ataques que se aprovechan de vulnerabilidades para tomar el control de la unidad central de procesamiento (CPU).

Antecedentes de la invención

En los últimos años, varios servicios informáticos administrados se han convertido en el estándar cuando se trata de que las organizaciones elijan dónde construir su infraestructura de TI. Algunas empresas pueden optar por usar un servicio de correo electrónico en la nube en lugar de administrar sus propios servidores y software de correo electrónico. La razón principal de esto es que es más barato y seguro dejar que otra persona se ocupe de asuntos de infraestructura y mantenimiento. Por lo tanto, empresas de todo el mundo consumen servicios administrados como IaaS, PaaS y SaaS. En consecuencia, la seguridad de estos servicios administrados se convierte en un problema real una vez que muchos IP de organizaciones residen en estos servicios.

La vulnerabilidad de día cero, por ejemplo, momento de identificar errores en el sistema, se define como una vulnerabilidad de software no divulgada que partes malintencionadas (o piratas informáticos) pueden aprovecharse para afectar negativamente a programas informáticos, datos, ordenadores adicionales o una red. Tales vulnerabilidades se han vuelto comunes en ataques dirigidos contra organizaciones, por ejemplo, por espionaje corporativo y/o motivos económicos. Los proveedores de la nube ejecutan cientos de servicios y programas diferentes para atender a sus clientes, lo que los sitúa como el objetivo principal de dichos ataques dirigidos. Hay muchos vectores de ataque posibles diferentes que pueden afectar al centro de datos. Por ejemplo, un atacante puede alquilar parte de la infraestructura como cualquier otra organización y realizar un ataque llamado escape de máquina virtual que se aprovecha de vulnerabilidades en el código de la infraestructura subyacente, dándole al atacante control sobre la infraestructura y permitiéndole acceder a todos los datos que están presentes en el sistema. En un escenario diferente, un atacante puede atacar a un solo cliente violando los servicios que se ejecutan dentro de las máquinas virtuales de ese cliente específico, por ejemplo, un servicio Secure-Shell (SSH) que contiene una vulnerabilidad que solo el atacante conoce.

Los centros de datos (por ejemplo, los servidores de FACEBOOKO) ejecutan cientos de servicios de software y, por lo tanto, corren el riesgo de sufrir una vulnerabilidad que pueda existir en al menos uno de esos servicios. Aquí es cuando tiene lugar la primera etapa del ataque mediante el aprovechamiento de vulnerabilidades en un servicio de software o dentro de una máquina virtual. El aprovechamiento logra hacerse con el control de la CPU y ejecutar un código de bajo nivel llamado 'shellcode', que se basa en la infraestructura subyacente e inicia la segunda etapa del ataque. La segunda etapa generalmente incluye una lógica maliciosa más avanzada, como la instalación de un programa de puerta trasera que brinda al atacante acceso persistente al sistema, dando por ello a los atacantes el poder de operar de forma independiente y obtener el control del sistema. Además de los centros de datos, también se pueden atacar de forma similar diversas unidades de monitorización.

Las soluciones de seguridad se pueden implementar a nivel de hardware o de software. Las implementaciones a nivel de hardware (como NX-bit, TPM y SGX) tienen la ventaja de una visibilidad total del sistema y no afectan el rendimiento del sistema. Sin embargo, sus largos ciclos de integración significan que los atacantes tienen tiempo suficiente para crear nuevas técnicas de explotación que los mantengan en un punto de ventaja. Las soluciones de seguridad a nivel de software tienen ciclos de integración cortos y se actualizan con frecuencia, pero tienen una visibilidad limitada del sistema. Además, todo el software de seguridad se basa esencialmente en "conectarse" a las aplicaciones para ganar visibilidad, lo que afecta en gran medida al rendimiento, la estabilidad y la integridad de la aplicación, por ejemplo, cuando se ejecuta en el centro de datos.

Hay dos posibles métodos de despliegue de productos de seguridad: despliegues basados en red y basadas en anfitrión (es decir, punto final/servidor). El despliegue basado en anfitrión usa una variedad de técnicas para detectar malware. Las más comunes son las firmas estáticas que se comparan continuamente con el sistema de archivos; sin embargo, el escaneo continuo provoca una reducción del rendimiento y puede sortearse fácilmente simplemente realizando pequeños cambios en el malware. Se desarrollaron firmas basadas en comportamiento dinámico, pero debido a la naturaleza de la implementación (es decir, enlaces de memoria, etc.), la estabilidad del sistema se ve comprometida y genera altas tasas de falsos positivos. En consecuencia, los operadores de centros de datos prefieren desplegar soluciones de seguridad basadas en redes.

De manera similar a los despliegues basados en anfitrión, las soluciones de seguridad de red escanean los flujos de red en busca de firmas de malware conocidas, lo que elimina la penalización de alto rendimiento del escaneo basado en anfitrión, aunque todavía es fácilmente sorteable por las mismas razones. Las firmas basadas en comportamiento dinámico tampoco son aplicables a las redes, ya que requieren la ejecución de código binario

que no cumple con los requisitos de rendimiento de una red de centro de datos.

Independientemente del vector de ataque y del método de explotación empleados, el objetivo final de un atacante es realizar computación maliciosa en el sistema de destino mediante la ejecución de instrucciones de la máquina que están bajo el control del atacante. Por lo general, la computación maliciosa es causada por código ilegítimo que no fue proporcionado ni destinado a ser ejecutado por el desarrollador del servicio del que se aprovechan (por ejemplo, SSH). El código malicioso generalmente se introduce en el sistema de destino usando datos de red externos o archivos de aplicaciones.

Para contrarrestar los ataques de inyección de código, los proveedores de hardware introdujeron una característica que evita que las regiones de memoria designadas sean ejecutadas por el procesador. De esta manera, el código que introdujo (o inyectó) en regiones de memoria que sirven como algo más que regiones de código no se ejecutará cuando el atacante active una vulnerabilidad que desvíe el flujo de ejecución a su código inyectado. Para sortear esta mitigación impuesta por hardware, los atacantes se valen de ataques de reutilización de código donde, en lugar de inyectar código, simplemente juntan fragmentos de código existentes de la aplicación atacada para realizar la misma lógica que habría hecho su código inyectado. Un ataque de reutilización de código abusa de los códigos de operación de la máquina relacionados con el flujo de control para reconstruir la lógica, por ejemplo, un método que abusa del código de operación de retorno (por ejemplo, programación orientada al retorno). Para detener este tipo de ataques que se aprovechan de la arquitectura informática, una solución de seguridad debe tener acceso a lo que realmente sucede a ese nivel. La mayoría de las tecnologías actuales buscan malware con firmas de muestras previamente descubiertas de ese mismo malware; sin embargo, diariamente se introducen miles de nuevas variantes simplemente permutando el antecesor original. Por lo tanto, los proveedores de seguridad no pueden mantenerse al día para brindar la seguridad adecuada.

Las soluciones de seguridad a nivel de hardware poseen la visibilidad necesaria para frustrar tales ataques; sin embargo, los atacantes se mantienen a la vanguardia debido a los largos ciclos de integración y al requisito de recompilar el código fuente del software para admitir nuevas características basadas en hardware. Cada vez que se desarrolla un nuevo componente de seguridad a nivel de hardware, los atacantes logran desarrollar una manera de sortearlo, por lo que, cuando el hardware se implementa a nivel mundial, ya no es efectivo. Por lo tanto, existe la necesidad de una solución de seguridad que tenga ciclos de actualización de software con la visibilidad de las soluciones de hardware.

El documento US 2016/0283714 describe un dispositivo informático de autocontrol con mitigación de aprovechamientos de flujo de control mediante el análisis de datos de trazado indicativos del flujo de control de un proceso ejecutado, para identificar supuestos aprovechamientos de flujo de control.

Yubin Xia et al. "*CFIMon: Detecting violation of control flow integrity using performance counters*", SISTEMAS Y REDES DEPENDABLES (DSN), 42.ª CONFERENCIA INTERNACIONAL ANUAL IEEE/IFIP ON, IEEE (2012), páginas 1-12, XP032220335, describe la detección de violaciones de la integridad del flujo de control basada en soporte de hardware para el monitoreo del rendimiento en procesadores modernos.

Sumario de la invención

La presente invención está definida por la reivindicación independiente 1 adjunta. Las reivindicaciones dependientes constituyen realizaciones de la invención.

Breve descripción de los dibujos

La materia considerada como la invención se señala particularmente y se reivindica claramente en la porción final de la memoria descriptiva. Sin embargo, la invención, tanto en lo que respecta a la organización como al método de funcionamiento, junto con objetivos, características y ventajas de la misma, puede entenderse mejor haciendo referencia a la siguiente descripción detallada cuando se lee con los dibujos que se acompañan en los que:

la figura 1 muestra un diagrama de bloques de un dispositivo informático de ejemplo, de acuerdo con algunas realizaciones de la invención;

la figura 2 ilustra esquemáticamente un diagrama de bloques de un sistema de detección de aprovechamientos, de acuerdo con algunas realizaciones de la invención;

la figura 3 muestra un diagrama de flujo para un algoritmo de detección de aprovechamientos de inyección de pila, de acuerdo con algunas realizaciones de la invención;

la figura 4A muestra un diagrama de flujo para un algoritmo de detección de aprovechamiento de gestión estructurada de excepciones (SEH), de acuerdo con algunas realizaciones de la invención;

la figura 4B muestra una continuación del diagrama de flujo de la figura 4A, de acuerdo con algunas realizaciones de la invención;

la figura 5A muestra un diagrama de flujo para el uso del algoritmo de detección de aprovechamientos de rastreo de asignación de memoria, de acuerdo con algunas realizaciones de la invención;

la figura 5B muestra una continuación del diagrama de flujo de la figura 5A, de acuerdo con algunas realizaciones de la invención;

la figura 6A muestra un diagrama de flujo para un primer algoritmo de detección de aprovechamientos de programación orientada al retorno, de acuerdo con algunas realizaciones de la invención;

la figura 6B muestra una continuación del diagrama de flujo de la figura 6A, de acuerdo con algunas realizaciones de la invención;

la figura 6C muestra un diagrama de flujo para un segundo algoritmo de detección de aprovechamientos de programación orientada al retorno, de acuerdo con algunas realizaciones de la invención;

la figura 6D muestra un diagrama de flujo para una continuación del diagrama de flujo de la figura 6C, de acuerdo con algunas realizaciones de la invención;

la figura 7 muestra un diagrama de flujo para el algoritmo de detección de transferencia de control indirecto ilegal, de acuerdo con algunas realizaciones de la invención;

la figura 8 muestra un diagrama de flujo para un método de detección de un aprovechamiento de un dispositivo informático, de acuerdo con algunas realizaciones de la invención;

la figura 9A muestra un diagrama de flujo para la operación de algoritmo combinado, de acuerdo con algunas realizaciones de la invención; y

la figura 9B muestra un diagrama de flujo para una continuación del diagrama de flujo de la figura 9A, de acuerdo con algunas realizaciones de la invención.

Se apreciará que, por simplicidad y claridad de la ilustración, los elementos mostrados en las figuras no necesariamente se han dibujado a escala. Por ejemplo, las dimensiones de algunos de los elementos pueden exagerarse con respecto a otros elementos para mayor claridad. Además, cuando se considere apropiado, los números de referencia podrán repetirse entre las figuras para indicar elementos correspondientes o análogos.

Descripción detallada de realizaciones de la invención

En la siguiente descripción detallada, se exponen numerosos detalles específicos para proporcionar una comprensión profunda de la invención. Sin embargo, los expertos en la técnica entenderán que la presente invención se puede practicar sin estos detalles específicos. En otros casos, métodos, procedimientos y componentes bien conocidos no se han descrito en detalle para no complicar la presente invención.

Aunque las realizaciones de la invención no se limitan a este respecto, las discusiones que utilizan términos tales como, por ejemplo, "procesar", "computar", "calcular", "determinar", "establecer", "analizar", "comprobar" o similares, pueden referirse a operaciones y/o procesos de un ordenador, una plataforma informática, un sistema informático u otro dispositivo informático electrónico, que manipula y/o transforma datos representados como cantidades físicas (por ejemplo, electrónicos) dentro de los registros y/o memorias del ordenador en otros datos representados de manera similar como cantidades físicas dentro de los registros y/o memorias del ordenador u otro medio de almacenamiento no transitorio de información que pueda almacenar instrucciones para realizar operaciones y/o procesos. Aunque las realizaciones de la invención no se limitan a este respecto, los términos "pluralidad" y "una pluralidad" tal como se usan en el presente documento pueden incluir, por ejemplo, "múltiples" o "dos o más". Los términos "pluralidad" o "una pluralidad" pueden usarse en toda la especificación para describir dos o más componentes, dispositivos, elementos, unidades, parámetros o similares. A menos que se indique explícitamente, las realizaciones del método descritas en el presente documento no están limitadas a un orden o secuencia particular. Además, algunas de las realizaciones del método descritas o elementos de las mismas pueden ocurrir o realizarse simultáneamente, en el mismo momento o al mismo tiempo.

Se hace referencia a la figura 1, que muestra un diagrama de bloques de un dispositivo informático de ejemplo de acuerdo con realizaciones de la presente invención. El dispositivo informático 100 puede incluir un controlador 105 que puede ser, por ejemplo, un procesador de unidad central de procesamiento (CPU), un chip o cualquier dispositivo informático o computacional adecuado, un sistema operativo 115, una memoria 120, un almacenamiento 130, un dispositivo 135 de entrada y uno dispositivo 140 de salida.

El sistema operativo 115 puede ser o puede incluir cualquier segmento de código diseñado y/o configurado para realizar tareas que implican coordinación, planificación, arbitraje, monitoreo, control o administrar de otro modo del funcionamiento del dispositivo informático 100, por ejemplo, planificación de la ejecución de programas. El sistema operativo 115 puede ser un sistema operativo comercial. La memoria 120 puede ser o puede incluir, por ejemplo, una memoria de acceso aleatorio (RAM), una memoria de sólo lectura (ROM), una RAM dinámica (DRAM), una DRAM síncrona (SD-RAM), un chip de memoria de velocidad de datos doble (DDR), una memoria Flash, una memoria volátil, una memoria no volátil, una memoria caché, un búfer, una unidad de memoria a corto plazo, una unidad de memoria a largo plazo u otras unidades de memoria o unidades de almacenamiento adecuadas. La memoria 120 puede ser o puede incluir una pluralidad de unidades de memoria posiblemente diferentes.

El código ejecutable 125 puede ser cualquier código ejecutable, por ejemplo, una aplicación, un programa, un proceso, tarea o script. El código ejecutable 125 puede ser ejecutado por el controlador 105 posiblemente bajo el control del sistema operativo 115. Por ejemplo, el código ejecutable 125 puede ser una aplicación para administrar datos de consumo de energía. Cuando corresponda, el código ejecutable 125 puede llevar a cabo las operaciones descritas en el presente documento en tiempo real. El dispositivo informático 100 y el código ejecutable 125 pueden configurarse para actualizar, procesar y/o actuar sobre la información al mismo ritmo que se recibe la información o un evento relevante. En algunas realizaciones, se puede usar más de un dispositivo informático 100. Por ejemplo, una pluralidad de dispositivos informáticos que incluyen componentes similares a los incluidos en el dispositivo informático 100 pueden conectarse a una red y usarse como un sistema. Por ejemplo, la administración de datos de consumo de energía se puede realizar en tiempo real mediante el código ejecutable 125 cuando se ejecuta en uno o más dispositivos informáticos, como el dispositivo informático 100.

El almacenamiento 130 puede ser o puede incluir, por ejemplo, una unidad de disco duro, una unidad de disquete, una unidad de disco compacto (CD), una unidad de CD grabable (CD-R), un dispositivo de bus universal en serie (USB) u otra unidad de almacenamiento adecuada extraíble y/o fija. El contenido puede almacenarse en el almacenamiento 130 y puede cargarse desde el almacenamiento 130 en la memoria 120 donde puede ser procesado por el controlador 105. En algunas realizaciones, se pueden omitir algunos de los componentes mostrados en la figura 1. Por ejemplo, la memoria 120 puede ser una memoria no volátil que tiene la capacidad de almacenamiento del almacenamiento 130. En consecuencia, aunque se muestra como un componente separado, el almacenamiento 130 puede estar incrustado o incluido en la memoria 120.

Los dispositivos 135 de entrada pueden ser o pueden incluir un ratón, un teclado, una pantalla o panel táctil o cualquier dispositivo de entrada adecuado. Se reconocerá que cualquier número adecuado de dispositivos de entrada puede conectarse operativamente al dispositivo informático 100 como se muestra en el bloque 135. Los dispositivos 140 de salida pueden incluir uno o más visualizadores, altavoces y/o cualquier otro dispositivo de salida adecuado. Se reconocerá que cualquier número adecuado de dispositivos de salida puede conectarse operativamente al dispositivo informático 100 como se muestra en el bloque 140. Cualquier dispositivo de entrada/salida (E/S) aplicable puede conectarse al dispositivo informático 100 como se muestra en los bloques 135 y 140. Por ejemplo, una tarjeta de interfaz de red (NIC) cableada o inalámbrica, un módem, una impresora o una máquina de fax, un dispositivo de bus universal en serie (USB) o un disco duro externo pueden incluirse en los dispositivos 135 de entrada y/o dispositivos 140 de salida.

Las realizaciones de la invención pueden incluir un artículo tal como un medio legible no transitorio por ordenador o procesador, o un medio de almacenamiento no transitorio de ordenador o procesador, tal como por ejemplo una memoria, una unidad de disco o una memoria flash USB, codificación, incluir o almacenar instrucciones, por ejemplo, instrucciones ejecutables por ordenador, que, cuando las ejecuta un procesador o controlador, llevan a cabo los métodos divulgados en el presente documento. Por ejemplo, un medio de almacenamiento tal como la memoria 120, instrucciones ejecutables por ordenador tal como el código ejecutable 125 y un controlador tal como el controlador 105.

Algunas realizaciones pueden proporcionarse en un producto de programa informático que puede incluir un medio no transitorio legible por máquina, con instrucciones almacenadas en el mismo, que pueden usarse para programar un ordenador u otros dispositivos programables para realizar métodos como se divulga en el presente documento. Las realizaciones de la invención pueden incluir un artículo tal como un medio legible no transitorio por ordenador o procesador, o un medio de almacenamiento no transitorio de ordenador o procesador, tal como por ejemplo una memoria, una unidad de disco o una memoria flash USB, codificación, incluir o almacenar instrucciones, por ejemplo, instrucciones ejecutables por ordenador, que cuando las ejecuta un procesador o controlador, llevan a cabo los métodos divulgados en el presente documento. El medio de almacenamiento puede incluir, entre otros, cualquier tipo de disco, incluidos disquetes, discos ópticos, memorias de sólo lectura en discos compactos (CD-ROM), discos compactos regrabables (CD-RW) y discos magnetoópticos. dispositivos semiconductores tales como memorias de solo lectura (ROM), memorias de acceso aleatorio (RAM), tales como una RAM dinámica (DRAM), memorias de solo lectura programables y borrables (EPROM), memorias flash, memorias de solo lectura programables y borrables eléctricamente (EEPROM)), tarjetas magnéticas u ópticas, o cualquier tipo de soporte adecuado para almacenar instrucciones electrónicas, incluidos los dispositivos de

almacenamiento programables.

Un sistema de acuerdo con realizaciones de la invención puede incluir componentes tales como, entre otros, una pluralidad de unidades centrales de procesamiento (CPU) o cualquier otro procesador o controlador multipropósito o específico adecuado, una pluralidad de unidades de entrada, una pluralidad de unidades de salida, una pluralidad de unidades de memoria y una pluralidad de unidades de almacenamiento. Un sistema puede incluir adicionalmente otros componentes de hardware y/o componentes de software adecuados. En algunas realizaciones, un sistema puede incluir o puede ser, por ejemplo, un ordenador personal, un ordenador de escritorio, un ordenador móvil, un ordenador portátil, un ordenador portátil, un terminal, una estación de trabajo, un ordenador de servidor, un asistente digital personal (PDA), una tableta, un dispositivo de red o cualquier otro dispositivo informático adecuado. A menos que se indique explícitamente, las realizaciones del método descritas en el presente documento no están limitadas a un orden o secuencia particular. Además, algunas de las realizaciones del método descritas o elementos de las mismas pueden ocurrir o realizarse en el mismo momento.

Ahora se hace referencia a la figura 2, que ilustra esquemáticamente un diagrama de bloques de un sistema 200 de detección de aprovechamientos, de acuerdo con algunas realizaciones de la invención. Cabe señalar que la dirección de las flechas en la figura 2 puede indicar la dirección del flujo de información. El sistema 200 de detección de aprovechamientos puede utilizarse para emitir una alerta tras la detección de aprovechamientos en un dispositivo computarizado 210 tal como un dispositivo informático 100 (por ejemplo, como se muestra en la figura 1), que tiene un procesador 201 acoplado operativamente a una unidad 202 de memoria.

Cabe apreciar que el dispositivo computarizado 210 como se usa en lo sucesivo también puede referirse a un punto final de un centro de datos de un sistema computarizado. En algunas realizaciones, el dispositivo computarizado 210, por ejemplo un ordenador personal (PC) o un teléfono inteligente, puede incluir además una unidad 204 de monitoreo del rendimiento (PMU) que puede configurarse para monitorear el funcionamiento del procesador 201.

Cabe apreciar que para prevenir ataques que se aprovechen de la arquitectura del ordenador, puede ser necesario tener acceso a procesos reales en el nivel informático de hardware y/o software (por ejemplo, detectar aprovechamientos debajo de la capa del sistema operativo). En algunas realizaciones, la recopilación de datos de procesos informáticos reales (por ejemplo, de la PMU) puede permitir el acceso a registros de ejecución de código, de modo que se pueda permitir la detección de aprovechamientos o infracciones, como se describe con más detalle a continuación.

De acuerdo con algunas realizaciones, el sistema 200 de detección de aprovechamientos puede incluir una CPU observadora 203 configurada para recibir datos correspondientes al flujo de ejecución de procesamiento desde el dispositivo computarizado 210. La CPU observadora 203 puede recibir datos de flujo de ejecución desde la PMU 204 y/o el procesador 201. En algunas realizaciones, la CPU observadora 203 puede recibir además registros de memoria (por ejemplo, RAM) del procesador 201 y/o la unidad 202 de memoria. Cabe señalar que mientras el procesador 201 ejecuta código y/o programas específicos, la CPU observadora 203 puede configurarse para reconstruir la ejecución y comprobar constantemente las instrucciones que se ejecutan. Entonces será posible actuar según estas instrucciones y detectar violaciones.

De acuerdo con algunas realizaciones, la CPU observadora 203 puede implementarse en un componente de hardware dedicado (por ejemplo, en un chip de ordenador separado) y/o como un algoritmo de software. En algunas realizaciones, la CPU observadora 203 puede recibir datos de trazado (con información sobre las instrucciones ejecutadas) desde el procesador 201, por ejemplo, usando técnicas de trazado de rama y/o trazado del procesador.

Cabe apreciar que los registros de memoria como se usan en lo sucesivo pueden hacer referencia a páginas de memoria y/o imágenes de memoria. En algunas realizaciones, los datos recibidos por la CPU observadora 203 pueden almacenarse al menos temporalmente en una base de datos 205.

En algunas realizaciones, la CPU observadora 203 puede reconstruir el flujo de ejecución a partir de los datos recibidos y las páginas de memoria para permitir la comparación (por ejemplo, con datos almacenados en la base de datos 205) con el comportamiento esperado y mediante ello detectar aprovechamientos en el mismo. En algunas realizaciones, la CPU observadora 203 puede configurarse además para verificar que la secuencia de códigos de operación no incluye contenido malicioso que intenta aprovecharse de las vulnerabilidades del proceso. En algunas realizaciones, la CPU observadora 203 puede monitorear la ejecución de código desde la pila que indica un posible ataque.

De acuerdo con algunas realizaciones, en caso de que la CPU observadora 203 sea un punto final del centro de datos y se detecte un ataque, el sistema 200 de detección de aprovechamientos puede emitir automáticamente una alerta y/o desactivar ese punto final (o nodo) para mantener el funcionamiento normal del sistema restante y/o evitar la propagación del ataque a otros puntos finales. En algunas realizaciones, un fallo en el sistema puede

indicar que el aprovechamiento detectado no es un ataque, ya que los atacantes desean evitar fallos.

En algunas realizaciones, al menos dos algoritmos de detección de aprovechamientos diferentes (por ejemplo, quince algoritmos) pueden implementarse en la CPU observadora 203 (y almacenarse en la base de datos 205), donde cada algoritmo de detección de aprovechamientos puede dedicarse a detectar un aprovechamiento diferente. Los al menos dos algoritmos de detección de aprovechamientos pueden ejecutarse simultáneamente hasta que al menos un algoritmo de detección de aprovechamientos detecte un aprovechamiento y, por ejemplo, emita una alerta.

Ahora se hace referencia a las figuras 3, 4A-4B, 5A-5B y 6A-6B que muestran diagramas de flujo con varios ejemplos de algoritmos de detección de aprovechamientos. Cabe apreciar que, tras la detección de un aprovechamiento por al menos uno de dichos algoritmos, el procesador 201 y/o la CPU observadora 203 pueden emitir una alerta, por ejemplo al usuario, para evitar daños al sistema.

La figura 3 muestra un diagrama de flujo para un algoritmo de detección de aprovechamientos de inyección de pila, de acuerdo con algunas realizaciones de la invención. El algoritmo de detección de aprovechamientos de inyección de pila puede incluir la CPU observadora 203 para mantener 301 direcciones de pila en la memoria (por ejemplo, en la base de datos 205) y recibir instrucciones 302. Cabe apreciar que, tal como se usa en el presente documento, la memoria de pila también puede referirse a la memoria dinámica (o de montón).

La CPU observadora 203 puede recibir un puntero de instrucción (IP) desde la PMU 204, comprobar 303 que la IP incluya una memoria de valores que sea una pila, de modo que la CPU observadora 203 pueda emitir 304 una alerta. En caso de que esa memoria de valores no sea una pila 303, la CPU observadora 203 puede continuar la iteración hasta que se detecte una pila.

Las figuras 4A-4B muestran un diagrama de flujo para un algoritmo de detección de aprovechamientos de gestión estructurada de excepciones (SEH), de acuerdo con algunas realizaciones de la invención. El algoritmo de detección de aprovechamientos SEH puede incluir la CPU observadora 203 para asignar 401 el gestor SEH del sistema operativo, por ejemplo, del dispositivo computarizado 210, y recibir instrucciones 402. La CPU observadora 203 puede comprobar 403 que, si se registra un nuevo gestor SEH, la dirección del gestor correspondiente puede añadirse a una lista enlazada SEH oculta 404. Cabe señalar que el SEH oculto realiza un rastreo del registro y eliminación del gestor SEH para detectar un intento de explotación.

En caso de que no se registre 403 ningún SEH, la CPU observadora 203 puede continuar comprobando 405 si se ha eliminado el gestor SEH. En caso de que el gestor SEH haya sido eliminado 405, la CPU observadora 203 puede eliminar 406 la dirección de la lista enlazada SEH oculta. Cabe señalar que, si bien en el presente documento se discuten los gestores SEH, también pueden ser aplicables otros sistemas de rastreo, y en las figuras 5A-5B.

En caso de que el gestor SEH no haya sido eliminado 405, la CPU observadora 203 puede comprobar 407 si se ha invocado el gestor de excepciones del sistema operativo. En caso de que ese gestor de excepciones del sistema operativo no se haya ejecutado 407, la CPU observadora 203 puede volver a comprobar 402 la siguiente instrucción a procesar. Cabe señalar que debido a la estructura de flujo (circular) del algoritmo, para cada instrucción ejecutada el algoritmo puede realizar comprobaciones 403 y/o 405 y/o 407 repetidamente. En algunas realizaciones, sólo cuando la comprobación 407 es negativa se puede procesar la siguiente instrucción.

En caso de que se haya ejecutado el gestor de excepciones del sistema operativo 407, la CPU observadora 203 puede continuar comprobando 408 si la dirección del gestor invocado está en la lista enlazada SEH oculta. En caso de que la dirección del gestor invocada esté en la lista enlazada SEH oculta 408, la CPU observadora 203 puede regresar para comprobar 402 la siguiente instrucción a procesar. En caso de que la dirección del gestor invocada no esté en la lista enlazada SEH oculta 408, la CPU observadora 203 puede emitir 409 una alerta.

Las figuras 5A-5B muestran un diagrama de flujo para el uso del algoritmo de detección de aprovechamientos de rastreo de asignación de memoria, de acuerdo con algunas realizaciones de la invención. El algoritmo de detección de aprovechamientos de rastreo de asignación de memoria puede incluir la CPU observadora 203 para asignar 501 las funciones de asignación de memoria y desasignación de memoria, recibir instrucciones 502 y también puede comprobar 503 que una función de asignación es llamada.

En caso de que una función de asignación haya sido llamada 503, la CPU observadora 203 puede añadir 1 al contador 504 de asignación. En caso de que una función de asignación no haya sido llamada 503, la CPU observadora 203 puede comprobar 505 si una desasignación ha sido llamada. En caso de que una función de desasignación haya sido llamada 505, la CPU observadora 203 puede sumar 1 al contador 506 de desasignación.

En caso de que una función de desasignación no haya sido llamada 505, la CPU observadora 203 puede comprobar 507 si la IP contiene un valor que está mapeado como memoria de montón. En caso de que la IP no

contenga un valor que esté mapeado como memoria 507 de montón, la CPU observadora 203 puede regresar para comprobar también 502 la siguiente instrucción a procesar. Cabe señalar que debido a la estructura de flujo (circular) del algoritmo, para cada instrucción ejecutada el algoritmo puede realizar comprobaciones 503 y/o 505 y/o 507 repetidamente. En algunas realizaciones, sólo cuando la comprobación 507 es negativa se puede procesar la siguiente instrucción.

En caso de que la IP contenga un valor que esté asignado como memoria 507 de montón, la CPU observadora 203 puede comprobar 508 si la diferencia entre los contadores de asignación y desasignación es mayor que un valor predefinido (por ejemplo, mayor que diez). En caso de que la diferencia no sea mayor que el valor predefinido 508, la CPU observadora 203 puede regresar para comprobar también 502 la siguiente instrucción a procesar. En caso de que la diferencia sea mayor que el valor predefinido 508, la CPU observadora 203 puede emitir 509 una alerta.

Las figuras 6A-6B muestran un diagrama de flujo para un primer algoritmo de detección de aprovechamientos de programación orientada al retorno (ROP), de acuerdo con algunas realizaciones de la invención. El primer algoritmo de detección de aprovechamientos orientado al retorno puede incluir la CPU observadora 203 para mantener una pila oculta 601 y recibir instrucciones 602. La CPU observadora 203 puede entonces comprobar 603 si se ha ejecutado una instrucción de "llamada" (o devolución) en el flujo de ejecución para verificación.

En caso de que se haya ejecutado 603 una instrucción de llamada, la CPU observadora 203 puede enviar 604 la dirección de retorno esperada a la pila oculta. En caso de que no se haya ejecutado 603 una instrucción de llamada, la CPU observadora 203 puede comprobar 605 si se ha ejecutado una instrucción de retorno (RET). En caso de que no se haya ejecutado 605 una instrucción de retorno, la CPU observadora 203 puede regresar para comprobar 602 la siguiente instrucción a procesar.

En caso de que se haya ejecutado 605 una instrucción de retorno, la CPU observadora 203 puede comprobar 606 si la dirección de destino es diferente de la dirección superior en la pila oculta. En caso de que la dirección de destino sea diferente de la dirección superior en la pila oculta 606, la CPU observadora 203 puede emitir 607 una alerta. Cabe señalar que debido a la estructura de flujo (circular) del algoritmo, para cada instrucción ejecutada el algoritmo puede realizar comprobaciones 603 y/o 605 y/o 606 repetidamente. En algunas realizaciones, sólo cuando la comprobación 606 es negativa se puede procesar la siguiente instrucción.

En caso de que la dirección de destino no sea diferente de la dirección superior en la pila oculta 606, la CPU observadora 203 puede extraer 608 una dirección (por ejemplo, la dirección más reciente que coincida correctamente con la dirección anticipada) de la pila y luego regresar para comprobar 602 la siguiente instrucción a procesar.

Las figuras 6C-6D muestran un diagrama de flujo para un segundo algoritmo de detección de aprovechamientos de programación orientada al retorno (ROP), de acuerdo con algunas realizaciones de la invención. El segundo algoritmo de detección de aprovechamientos orientado al retorno puede incluir la CPU observadora 203 para mantener una pila oculta 621 y recibir instrucciones 622. La CPU observadora 203 puede entonces comprobar 623 si se ha ejecutado una instrucción de "llamada" (o devolución) en el flujo de ejecución para verificación.

En caso de que se haya ejecutado 623 una instrucción de llamada, la CPU observadora 203 puede añadir 624 la dirección de retorno esperada al conjunto oculto y actualizar el contador (por ejemplo, inicializar el contador a '1') y/o añadir '1' al contador existente. Después de añadir la dirección de retorno esperada al conjunto oculto 624, la CPU 203 puede comprobar si la dirección de retorno esperada ya existe en el conjunto oculto 625. En caso de que la dirección ya exista en el conjunto oculto 625, la CPU observadora 203 puede actualizar el contador (por ejemplo, añadir '1' al contador existente) 626a. En caso de que la dirección no esté en el conjunto oculto 625, la CPU observadora 203 puede actualizar el contador (por ejemplo, inicializar el contador a "1") 626b.

En caso de que no se haya ejecutado 623 una instrucción de llamada, la CPU observadora 203 puede comprobar 627 si se ha ejecutado una instrucción de retorno (RET). En caso de que no se haya ejecutado 625 una instrucción de retorno, la CPU observadora 203 puede regresar para comprobar 623 la siguiente instrucción a procesar.

En caso de que se haya ejecutado 627 una instrucción de retorno, la CPU observadora 203 puede comprobar 628 si la dirección de destino está presente en el conjunto oculto. En caso de que la dirección de destino no esté presente en el conjunto oculto 628, la CPU observadora 203 puede emitir 629 una alerta. Cabe señalar que debido a la estructura de flujo (circular) del algoritmo, para cada instrucción ejecutada el algoritmo puede realizar comprobaciones 623 y/o 627 y/o 628 repetidamente. En algunas realizaciones, sólo cuando la comprobación 628 es positiva entonces se puede procesar la siguiente instrucción.

En caso de que esa dirección de destino esté presente en el conjunto oculto 628, la CPU observadora 203 puede comprobar 630 si el contador es '0'. En caso de que el contador sea '0' 629, la CPU observadora 203 puede emitir 629 una alerta. En caso de que el contador no sea '0' 629, la CPU observadora 203 puede actualizar el

contador 631 (por ejemplo, reducir el contador en '1'), y luego volver a comprobar 622 la siguiente instrucción a procesar.

Ahora se hace referencia a la figura 7, que muestra un diagrama de flujo para el algoritmo de detección de transferencia de control indirecto ilegal, de acuerdo con algunas realizaciones de la invención. Las instrucciones de control de programa indirecto pueden calcular la dirección de destino basándose en un argumento o variable que reside en un registro o ubicación de memoria. Para cada rama indirecta en el flujo de ejecución reconstruido, la dirección de destino puede validarse con una base de datos preprocesada. La base de datos preprocesada puede incluir direcciones de destino legítimas a las que el procesador puede transferir el control y/o direcciones de origen legítimas del sitio de llamada a "funciones críticas" desde las que el procesador puede transferir el control. En algunas realizaciones, una función crítica puede ser cualquier función (por ejemplo, del sistema operativo) que un atacante tiene interés en invocar para llevar a cabo con éxito el ataque. Por ejemplo, las funciones que controlan los permisos de la memoria son "críticas" porque el atacante puede trabajar más libremente cuando los permisos de la memoria están bajo su control. En algunas realizaciones, un atacante puede habilitar el indicador de ejecución a través de dicha función crítica.

De acuerdo con algunas realizaciones, una dirección de destino (por ejemplo, recopilada analizando imágenes de memoria de código ejecutable) puede clasificarse como segura si cumple al menos uno de los siguientes: la dirección puede estar en una tabla de reubicación de un módulo, de modo que el puntero puede fijarse en el momento de la carga y, por lo tanto, apunta a una dirección a la que es legal transferir el control, y/o la dirección puede ser el punto de entrada de un módulo, y/o la dirección puede estar en la tabla de importación/tabla de símbolos dinámicos. de un módulo para que el control pueda transferirse a estas direcciones desde módulos externos durante el tiempo de ejecución, y/o la dirección puede estar en la tabla de exportación/tabla de símbolos de un módulo para que el control pueda transferirse a estas direcciones desde el módulo actual, y/o la dirección puede ser una llamada relativa de modo que un módulo pueda transferir el flujo de control dentro de sí mismo, y/o la dirección puede estar precedida por una instrucción de 'LLAMADA' en los módulos.

El algoritmo de detección de transferencia de control indirecto ilegal puede incluir el mantenimiento 701 de una base de datos de direcciones legales a las que transferir el control indirectamente. En algunas realizaciones, el algoritmo de detección de transferencia de control indirecto ilegal puede incluir además recibir 702 instrucciones para la siguiente rama indirecta, por ejemplo, recibir instrucciones del flujo de ejecución reconstruido. Si la dirección no está en la base 703 de datos, se puede emitir una alarma 704. Si la dirección está en la base 703 de datos, entonces el algoritmo de detección de transferencia de control indirecto ilegal puede esperar hasta que se puedan recibir instrucciones para la siguiente rama indirecta. En algunas realizaciones, se puede emitir una alarma si la fuente de la dirección de la instrucción de rama indirecta está en la base de datos de direcciones legales a las que transferir el control indirectamente.

Ahora se hace referencia a la figura 8, que muestra un diagrama de flujo para un método para detectar un aprovechamiento de una vulnerabilidad de un dispositivo informático 210, de acuerdo con algunas realizaciones de la invención. El método para detectar un aprovechamiento de una vulnerabilidad de un dispositivo informático puede incluir recibir 801 un flujo de ejecución de al menos un proceso que se ejecuta en un procesador 201 del dispositivo informático 210, en el que el flujo de ejecución se recibe desde una unidad de monitoreo del rendimiento (PMU). 204 del procesador 201.

En algunas realizaciones, el método puede incluir además recibir 802 páginas de memoria desde una memoria del dispositivo informático 210. En algunas realizaciones, el método puede incluir además la reconstrucción 803 del proceso en otro procesador basándose en el flujo de ejecución y las páginas de memoria.

En algunas realizaciones, el método puede incluir además ejecutar 804 al menos un algoritmo de detección de aprovechamientos en el proceso reconstruido para identificar un intento de aprovechamiento, y emitir 805 y una alerta, por ejemplo, tras la detección de un aprovechamiento. La alerta (por ejemplo, al usuario) puede ser emitida por el procesador 201 y/o por la CPU observadora 203.

En algunas realizaciones, el método puede incluir además interrumpir el proceso que se ejecuta en el procesador del dispositivo informático cuando se detecta un aprovechamiento.

De acuerdo con algunas realizaciones, un decodificador (para reconstruir la ejecución) puede incluir varias capas abstractas tales como capas de paquetes, capas de eventos, capas de flujo de instrucciones y capas de bloques. Si bien los datos de trazado que puede generar la CPU incluyen paquetes con unidades lógicas de datos binarios que representan diferentes tipos de eventos que ocurren a lo largo del ciclo de vida de ejecución, el proceso de decodificación puede añadir información al contexto en cada capa hasta que se logre la reconstrucción final. Por lo tanto, hacer coincidir el algoritmo de detección con la capa correcta puede permitir maximizar el rendimiento y la visibilidad del sistema. En algunas realizaciones, el orden de rendimiento de las capas puede ser paquete, evento, flujo de instrucciones y capa de bloque, donde cada capa puede tener una visibilidad diferente.

En algunas realizaciones, los algoritmos descritos en las figuras 3 y 7 pueden corresponder a un decodificador

con una capa de eventos. En algunas realizaciones, los algoritmos descritos en las figuras 4A-4B y 6A-6D pueden corresponder a un decodificador con una capa de bloque. Por ejemplo, los algoritmos de conjuntos ocultos pueden detectar instrucciones de devolución abusadas que transfieren el control a las direcciones elegidas por el atacante. Para poder tener la visibilidad que necesita, el conjunto oculto puede coincidir con la capa de bloque. Esto puede deberse al hecho de que solo desde la capa de flujo de instrucciones y superiores (por ejemplo, la capa de bloque) el contexto puede incluir el tipo de instrucción ejecutada por la CPU.

Ahora se hace referencia a las figuras 9A-9B, que muestran un diagrama de flujo para la operación de algoritmo combinado, de acuerdo con algunas realizaciones de la invención. De acuerdo con algunas realizaciones, al menos dos algoritmos diferentes pueden operarse simultáneamente y/o en combinación (por ejemplo, en modo híbrido) de modo que cada algoritmo pueda activarse y/o cambiarse cuando sea necesario. Por ejemplo, detectar una dirección de destino de rama indirecta de una 'función crítica' y cambiar para escanear la dirección de origen de rama indirecta de la función correspondiente para llamadas indirectas sospechosas.

El algoritmo combinado puede incluir mantener 901 una base de datos de direcciones legales para transferir indirectamente el control y/o mantener 901 una base de datos de funciones críticas. En algunas realizaciones, el algoritmo combinado puede incluir además recibir 902 instrucciones para la siguiente rama indirecta, por ejemplo recibir instrucciones del flujo de ejecución reconstruido. Si la dirección de destino de la rama indirecta no está en la base 903 de datos (por ejemplo, base de datos de direcciones legales a las que transferir indirectamente el control), el algoritmo combinado puede emitir una alarma 904. Si la dirección de destino de la rama indirecta está en la base 903 de datos (por ejemplo, base de datos de direcciones legales a las que transferir indirectamente el control), el algoritmo combinado puede comprobar 905 si la dirección de destino indirecta es una dirección de función crítica.

En caso de que la dirección de destino indirecta no sea una dirección 905 de función crítica, el algoritmo combinado puede volver a recibir 902 instrucciones para la siguiente rama indirecta. En caso de que la dirección de destino indirecta sea una dirección de función crítica 905, el algoritmo combinado puede volver a escanear 906 la rama indirecta para obtener la dirección de origen de la rama indirecta. Después de volver a escanear 906, el algoritmo combinado puede comprobar 907 si la dirección de origen corresponde a la dirección de función crítica (por ejemplo, en la base de datos de direcciones de función crítica). En caso de que la dirección de origen no corresponda con la dirección 907 de función crítica, se puede emitir una alarma 908. En caso de que la dirección de origen corresponda a la dirección 907 de función crítica, el algoritmo combinado puede volver a recibir 902 instrucciones para la siguiente rama indirecta.

A menos que se indique explícitamente, las realizaciones del método descritas en el presente documento no están limitadas a un orden particular en el tiempo o secuencia cronológica. Además, algunos de los elementos del método descritos se pueden omitir, o se pueden repetir, durante una secuencia de operaciones de un método.

Se han presentado diversas realizaciones. Por supuesto, cada una de estas realizaciones puede incluir características de otras realizaciones presentadas, y las realizaciones no descritas específicamente pueden incluir diversas características descritas en el presente documento.

REIVINDICACIONES

1.- Un método para detectar un aprovechamiento de una vulnerabilidad de un dispositivo informático (210), comprendiendo el método:

recibir un flujo de ejecución, mediante una CPU observadora (203), de al menos un proceso que se ejecuta en un procesador (201) del dispositivo informático (210), en el que el flujo de ejecución se recibe desde una unidad de monitoreo del rendimiento, PMU, (204) del procesador (201);

recibir páginas de memoria, mediante la CPU observadora (203), desde una memoria (202) del dispositivo informático (210);

reconstruir el flujo de ejecución del proceso en la CPU observadora (203) basándose en los datos de la PMU (204) y las páginas de memoria;

ejecutar un algoritmo de detección de aprovechamientos en el proceso reconstruido para identificar un intento de aprovechamiento, el método caracterizado porque el algoritmo de detección de aprovechamientos comprende:

mapear (501) una función de asignación de memoria y una función de desasignación de memoria;

comprobar (503, 505), mediante la CPU observadora (203), si al menos una de la función de asignación de memoria y la función de desasignación de memoria es llamada;

aumentar (504, 506), mediante la CPU observadora (203), al menos uno de entre el contador de asignación y el contador de desasignación en los casos en los que al menos una función correspondiente es llamada;

comprobar (508), mediante la CPU observadora (203), si la diferencia entre el contador de asignación y el contador de desasignación es mayor que un valor predefinido; y

emitir una alerta (509), mediante la CPU observadora (203), cuando la diferencia es mayor que el valor predefinido.

2.- El método de la reivindicación 1, que comprende además interrumpir el proceso que se ejecuta en el procesador (201) del dispositivo informático (210) cuando se detecta un aprovechamiento.

3.- El método de la reivindicación 1, que comprende además mantener el mapa de direcciones en la memoria (202) del procesador (201).

4.- El método de la reivindicación 1, que comprende además mapear, mediante la CPU observadora (203), el gestor de excepciones estructurado, SEH, del sistema operativo del procesador (201).

5.- El método de la reivindicación 4, que comprende además añadir, mediante la CPU observadora (203), la dirección de memoria de SEH.

6.- El método de la reivindicación 4, que comprende además comprobar, mediante la CPU observadora (203), si está registrado un gestor de excepciones estructurado, SEH.

7.- El método de la reivindicación 1, que comprende además recibir, mediante la CPU observadora (203), al menos una instrucción.

8.- El método de la reivindicación 1, que comprende además mantener, mediante la CPU observadora (203), una pila oculta.

9.- El método de la reivindicación 8, que comprende además enviar, mediante la CPU observadora (203), la dirección de retorno esperada a la pila oculta.

10.- El método de la reivindicación 1, que comprende además:

mantener una base de datos (205) de direcciones legales a las que transferir el control indirectamente; y

recibir instrucciones para la rama indirecta del flujo de ejecución reconstruido.

11.- El método de la reivindicación 10, que comprende además comprobar si la instrucción recibida corresponde a la base de datos (205) de direcciones legales a las que transferir el control indirectamente.

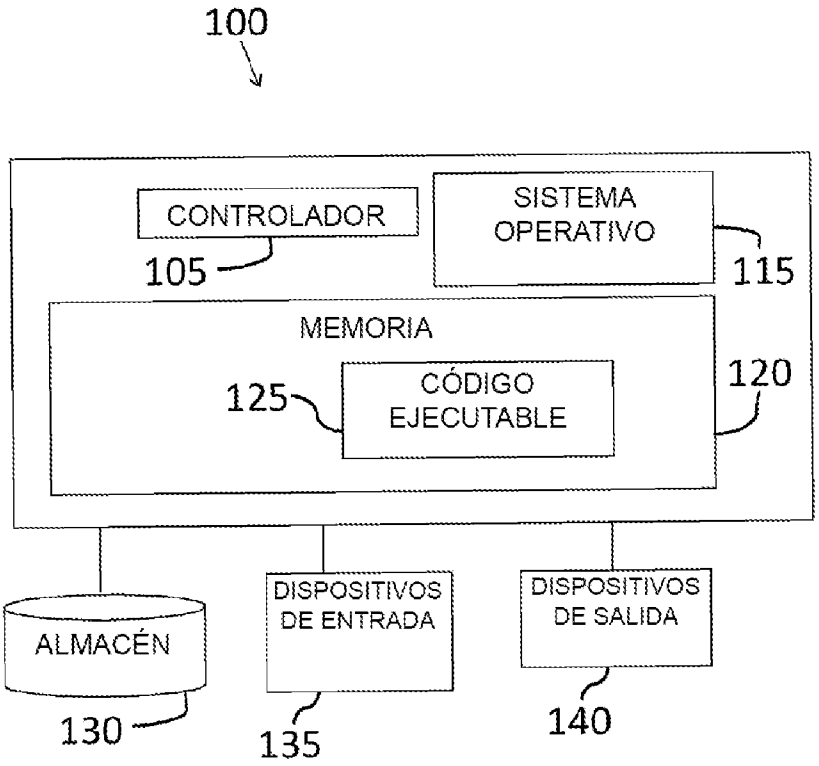
100
CONTROLADOR
105
SISTEMA OPERATIVO
115
MEMORIA
125
CÓDIGO EJECUTABLE
120
ALMACÉN
130
DISPOSITIVOS DE ENTRADA
135
DISPOSITIVOS DE SALIDA
140

Fig. 1

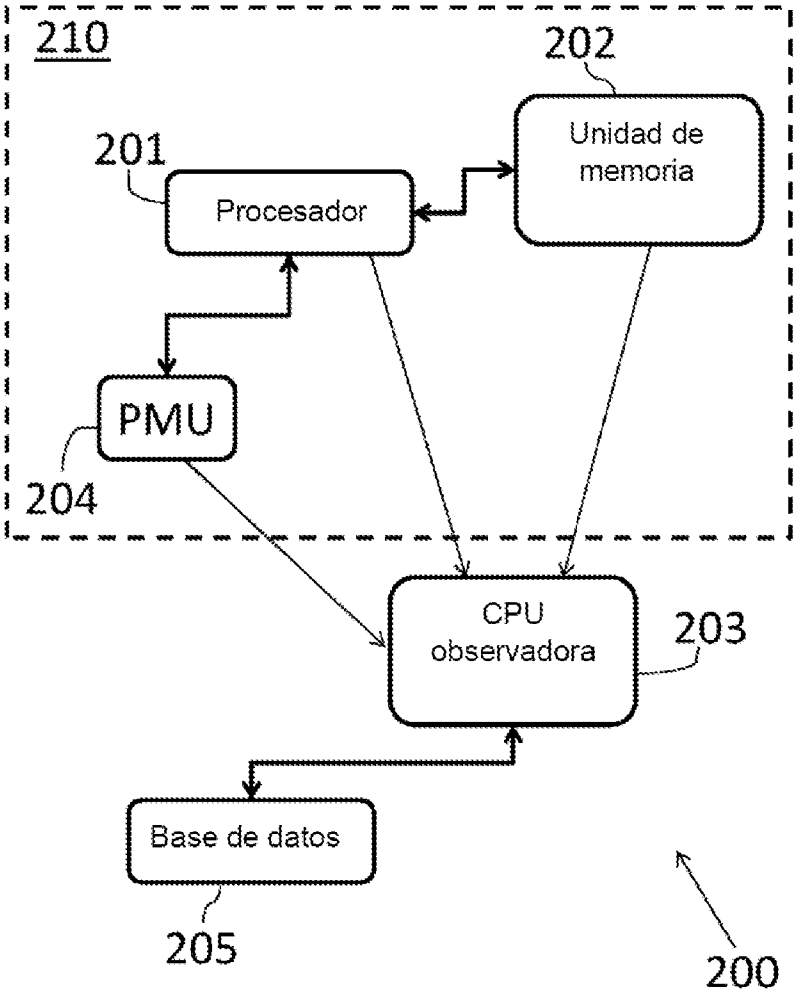
210
202
Unidad de memoria
201
Procesador
PMU
204
CPU observadora
203
Base de datos
205
200

Fig. 2

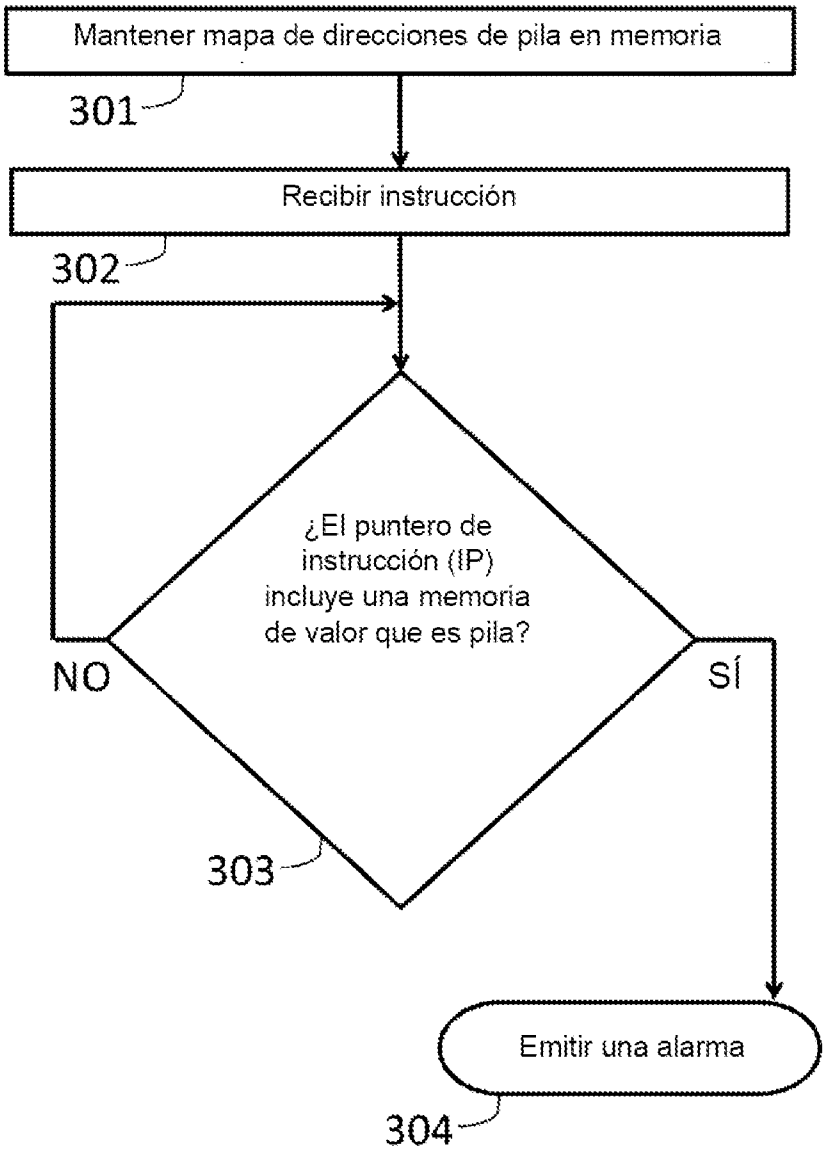
Mantener mapa de direcciones de pila en memoria
301
Recibir instrucción
302
¿El puntero de instrucción (IP) incluye una memoria de valor que es pila?
NO
SÍ
303
Emitir una alarma
304

Fig. 3

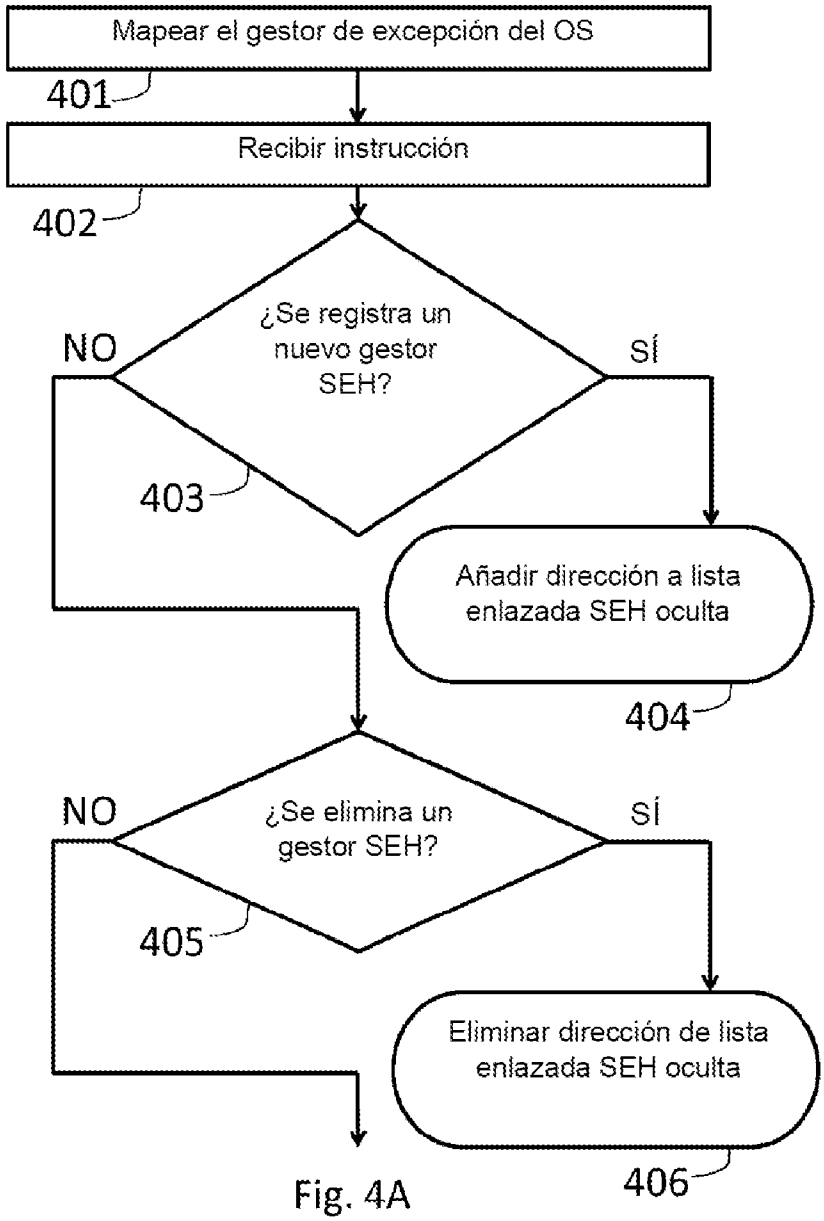
Mapear el gestor de excepción del OS
401
Recibir instrucción
402
¿Se registra un nuevo gestor SEH?
NO
SÍ
403
Añadir dirección a lista enlazada SEH oculta
404
¿Se elimina un gestor SEH?
NO
SÍ
405
Eliminar dirección de lista enlazada SEH oculta
406

Fig. 4A

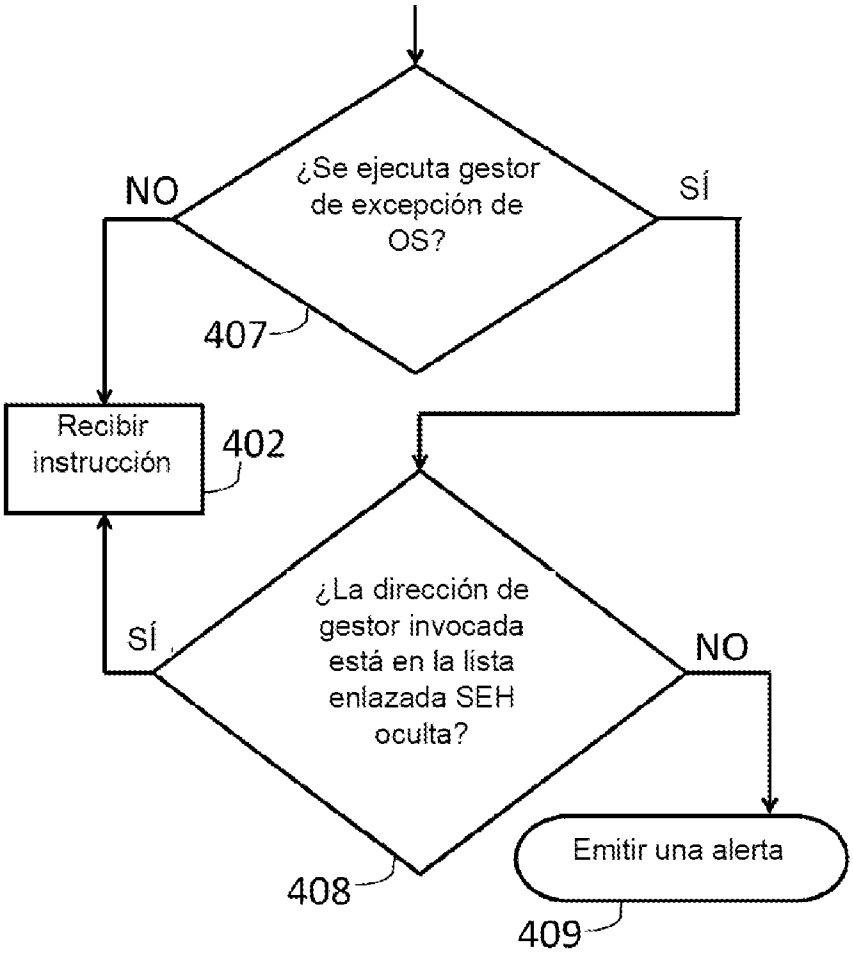
¿Se ejecuta gestor de excepción de OS?
NO
SÍ
407
Recibir instrucción
402
¿La dirección de gestor invocada está en la lista enlazada SEH oculta?
SÍ
NO
408
Emitir una alerta
409

Fig. 4B

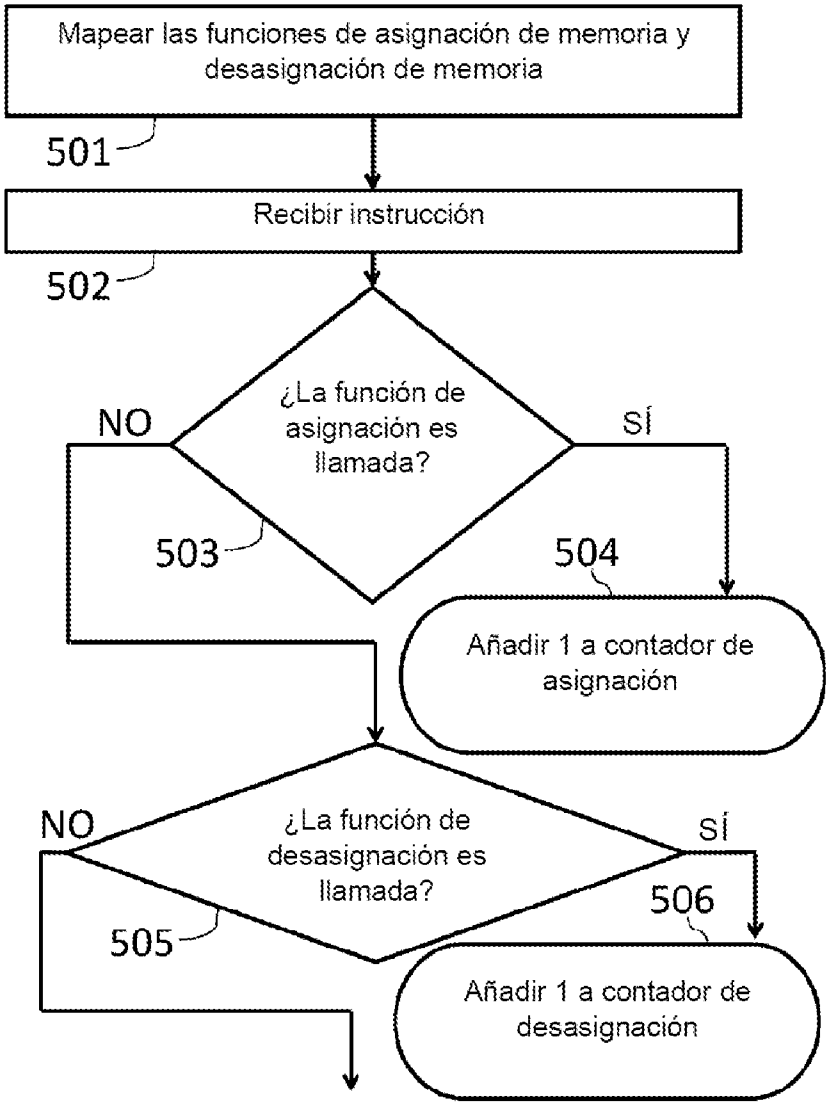
Mapear las funciones de asignación de memoria y desasignación de memoria
501
Recibir instrucción
502
¿La función de asignación es llamada?
NO
SÍ
503
504
Añadir 1 a contador de asignación
¿La función de desasignación es llamada?
NO
SÍ
505
506
Añadir 1 a contador de desasignación

Fig. 5A

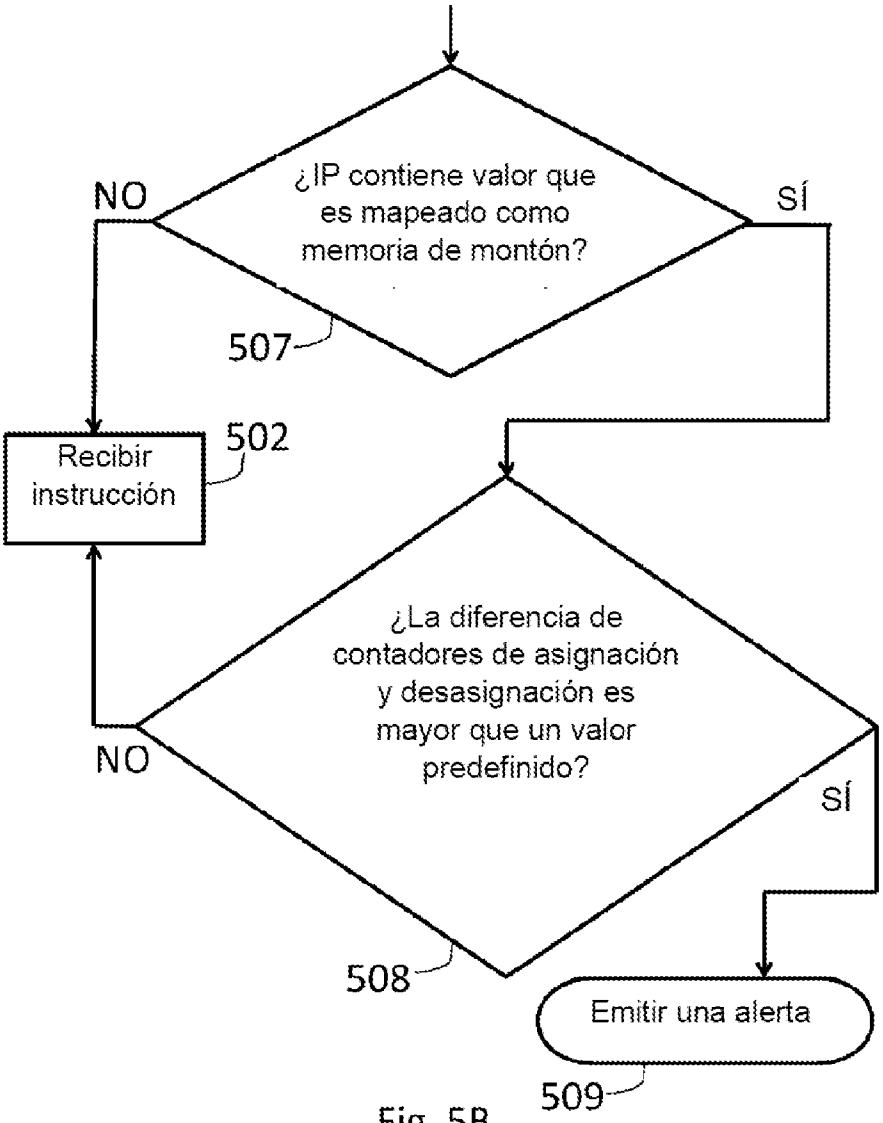
¿IP contiene valor que es mapeado como memoria de montón?
NO
SÍ
507
Recibir instrucción
502
¿La diferencia de contadores de asignación y desasignación es mayor que un valor predefinido?
NO
SÍ
508
Emitir una alerta
509

Fig. 5B

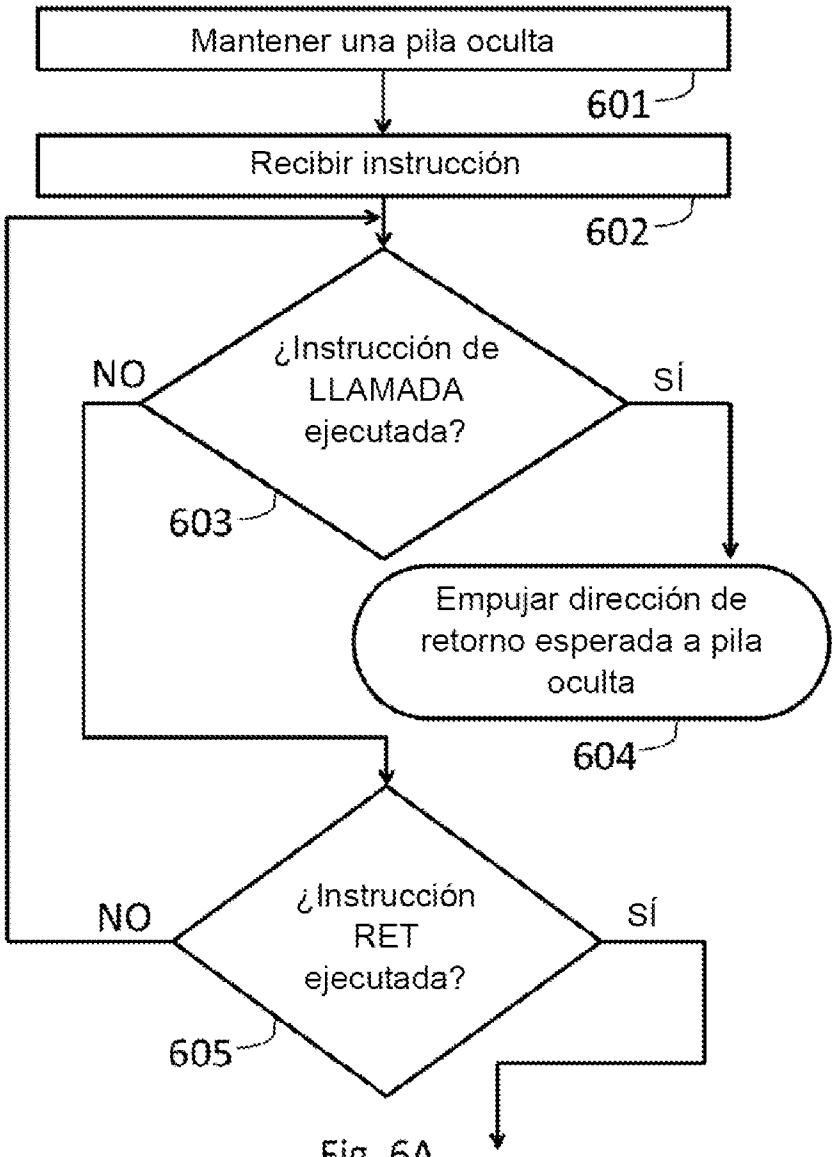
Mantener una pila oculta
601
Recibir instrucción
602
¿Instrucción de LLAMADA ejecutada?
NO
SÍ
603
Empujar dirección de retorno esperada a pila oculta
604
¿Instrucción RET ejecutada?
NO
SÍ
605

Fig. 6A

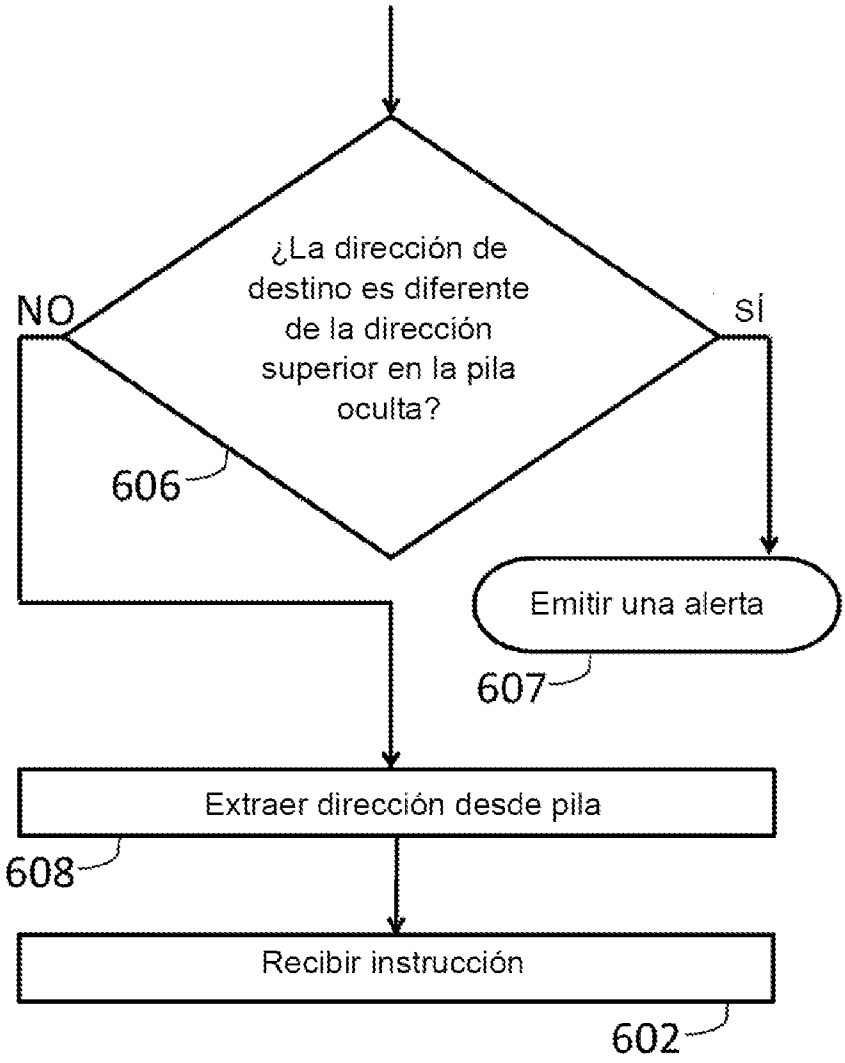
¿La dirección de destino es diferente de la dirección superior en la pila oculta?
NO
SÍ
606
Emitir una alerta
607
Extraer dirección desde pila
608
Recibir instrucción
602

Fig. 6B

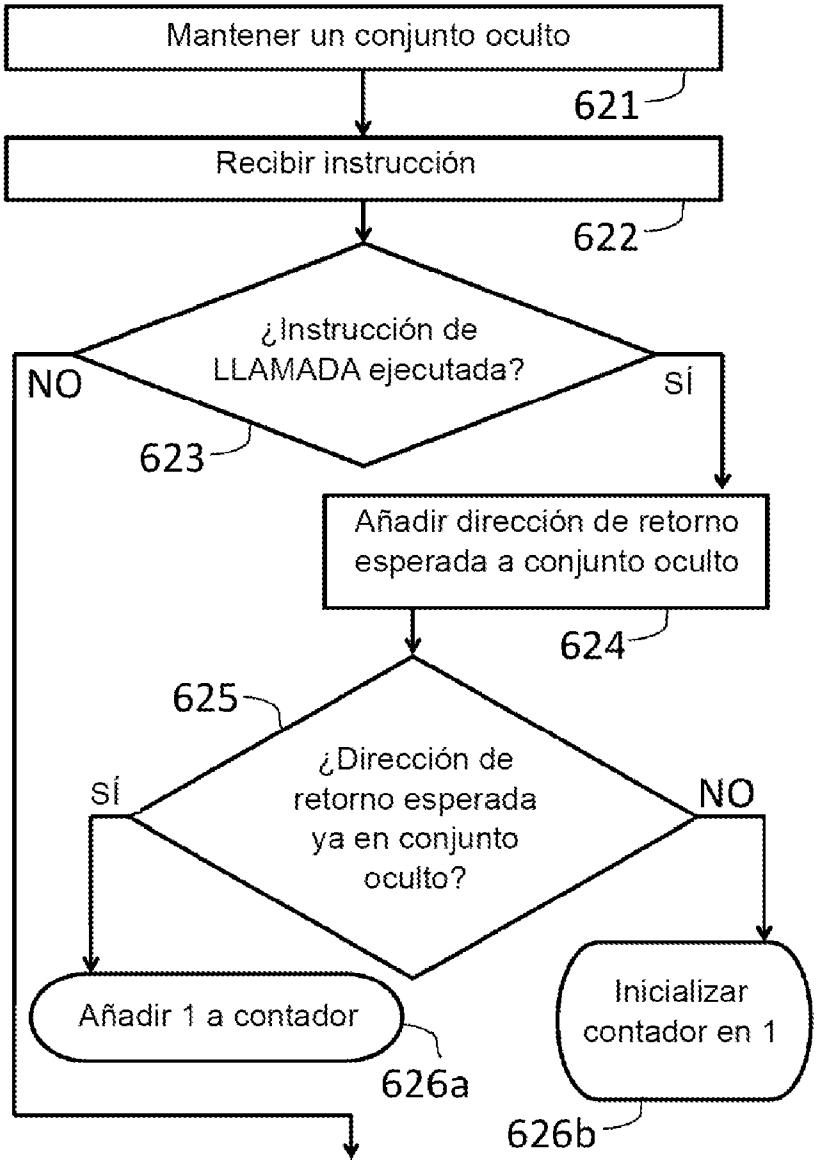
Mantener un conjunto oculto
621
Recibir instrucción
622
¿Instrucción de LLAMADA ejecutada?
NO
SÍ
623
Añadir dirección de retorno esperada a conjunto oculto
624
625
¿Dirección de retorno esperada ya en conjunto oculto?
SÍ
NO
Añadir 1 a contador
626a
Inicializar contador en 1
626b

Fig. 6C

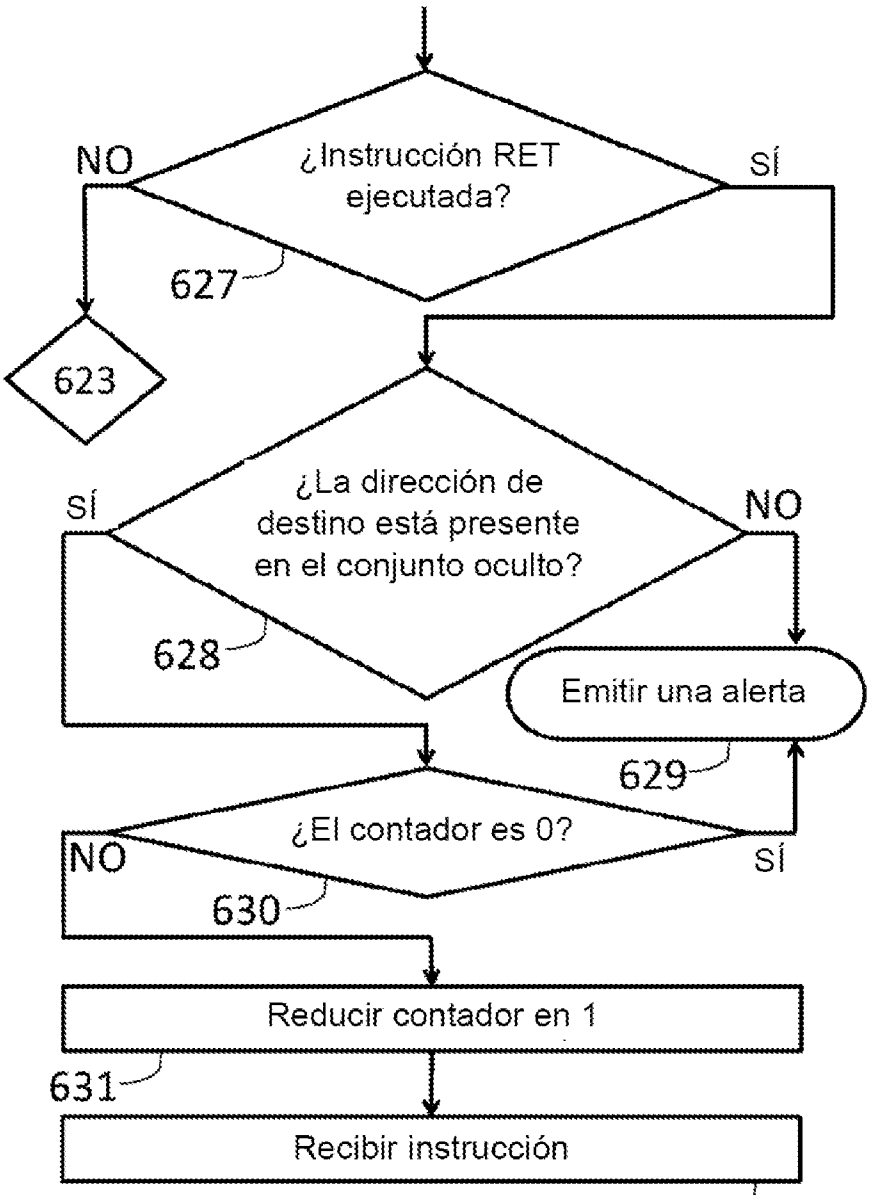
¿Instrucción RET ejecutada?
NO
SÍ
627
623
¿La dirección de destino está presente en el conjunto oculto?
SÍ
NO
628
Emitir una alerta
629
¿El contador es 0?
NO
SÍ
630
Reducir contador en 1
631
Recibir instrucción
622

Fig. 6D

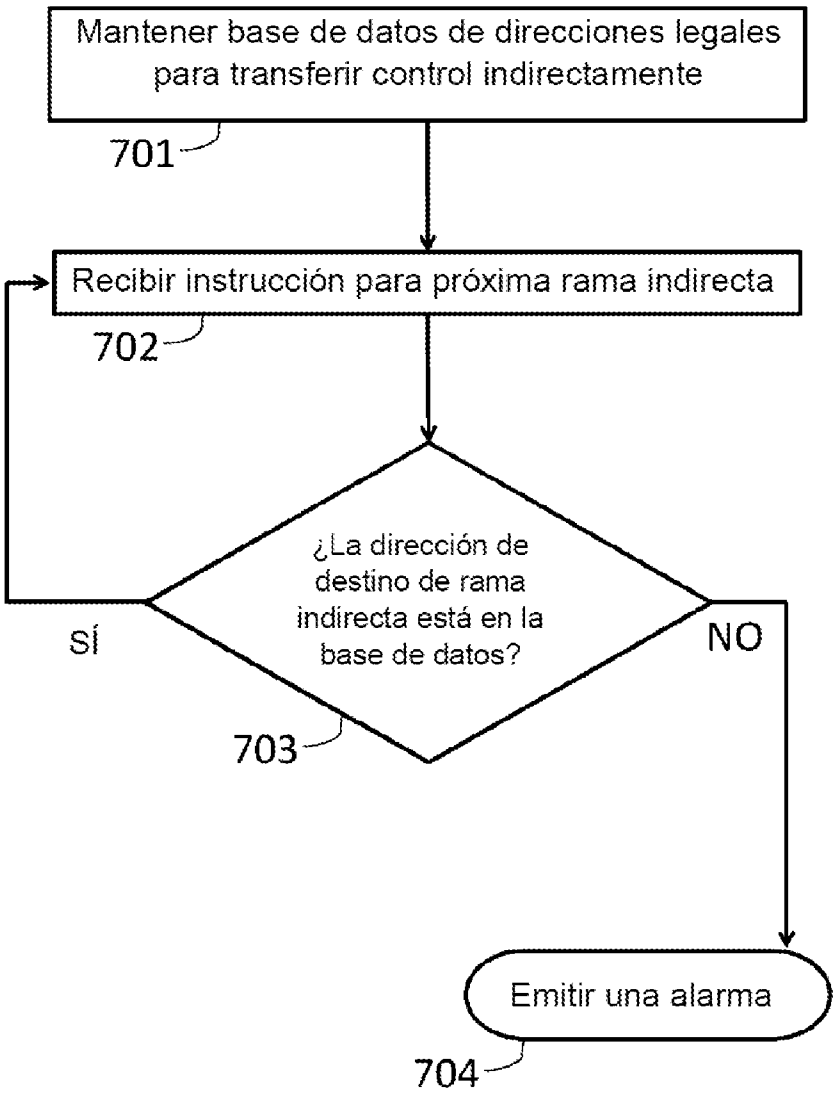
Mantener base de datos de direcciones legales para transferir control indirectamente
701
Recibir instrucción para próxima rama indirecta
702
¿La dirección de destino de rama indirecta está en la base de datos?
SÍ
NO
703
Emitir una alarma
704

Fig. 7

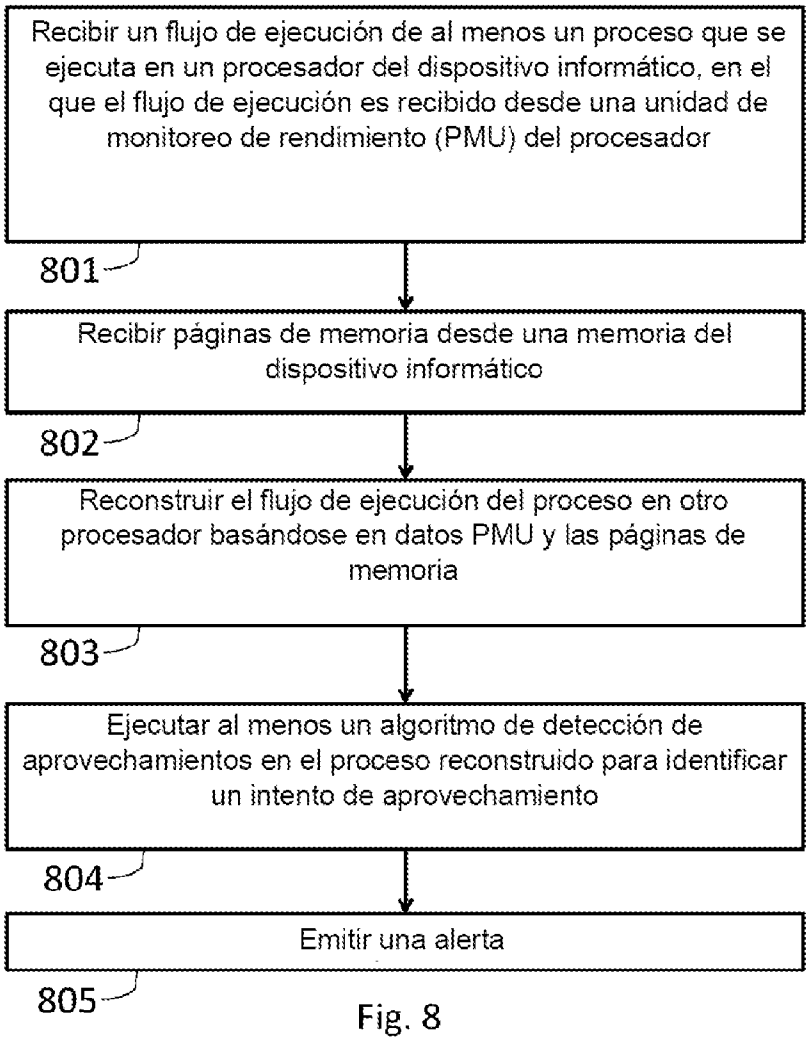
Recibir un flujo de ejecución de al menos un proceso que se ejecuta en un procesador del dispositivo informático, en el que el flujo de ejecución es recibido desde una unidad de monitoreo de rendimiento (PMU) del procesador
801
Recibir páginas de memoria desde una memoria del dispositivo informático
802
Reconstruir el flujo de ejecución del proceso en otro procesador basándose en datos PMU y las páginas de memoria
803
Ejecutar al menos un algoritmo de detección de aprovechamientos en el proceso reconstruido para identificar un intento de aprovechamiento
804
Emitir una alerta
805

Fig. 8

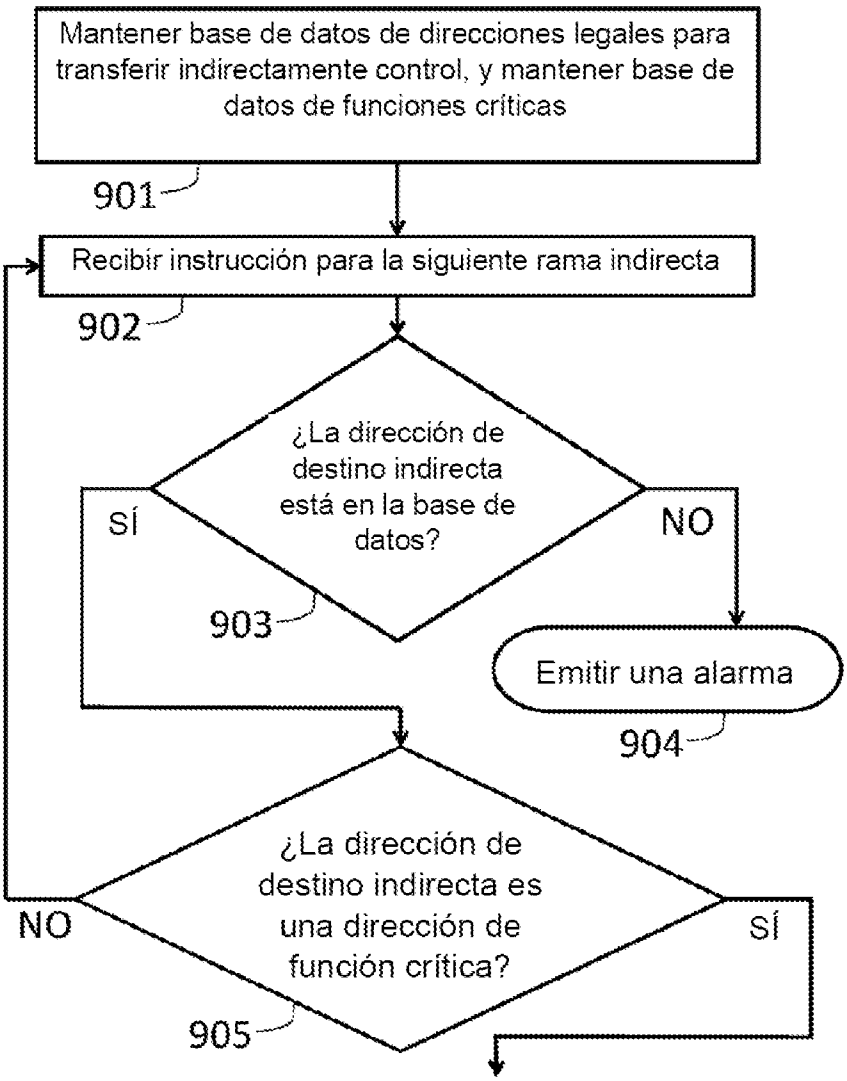
Mantener base de datos de direcciones legales para transferir indirectamente control, y mantener base de datos de funciones críticas
901
Recibir instrucción para la siguiente rama indirecta
902
¿La dirección de destino indirecta está en la base de datos?
SÍ
NO
903
Emitir una alarma
904
¿La dirección de destino indirecta es una dirección de función crítica?
NO
SÍ
905

Fig. 9A

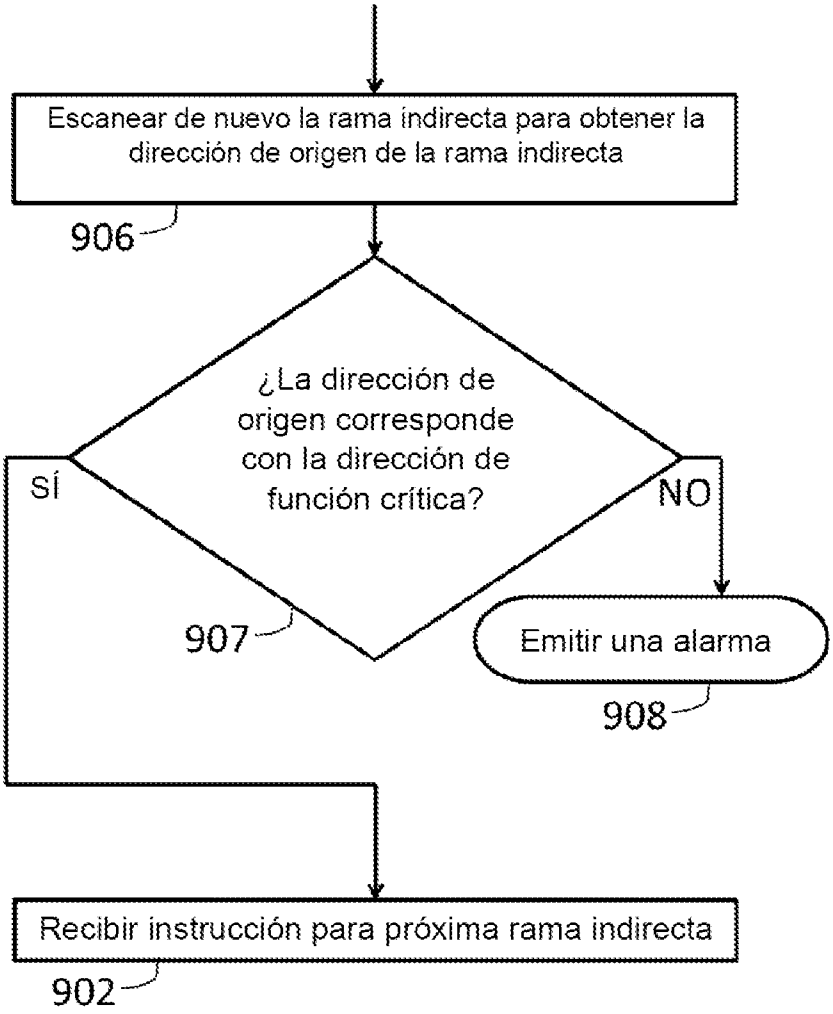
Escanear de nuevo la rama indirecta para obtener la dirección de origen de la rama indirecta
906
¿La dirección de origen corresponde con la dirección de función crítica?
SÍ
NO
907
Emitir una alarma
908
Recibir instrucción para próxima rama indirecta
902

Fig. 9B