(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 660 538**

(51) Int. CI.:

G06F 11/07 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **29.07.2010 PCT/US2010/043657**

(87) Fecha y número de publicación internacional: **03.02.2011 WO11014620**

(96) Fecha de presentación y número de la solicitud europea: **29.07.2010 E 10742940 (9)**

(97) Fecha y número de publicación de la concesión europea: **22.11.2017 EP 2460075**

(54) Título: **Reparación de archivos ejecutables portátiles**

(30) Prioridad:

29.07.2009 US 229497 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
22.03.2018

(73) Titular/es:

REVERSINGLABS CORPORATION (100.0%)
169 Msgr. O'Brian Highway Apt. 802
Cambridge, MA 02141, US

(72) Inventor/es:

PERICIN, TOMISLAV

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Reparación de archivos ejecutables portátiles

Campo técnico

Esta solicitud se refiere de manera general a software de ingeniería inversa, y más particularmente se refiere a software de desempaquetado y análisis de validez de archivos de software.

Antecedentes de la descripción

El formato de archivo ejecutable portátil (formato de archivo PE), como se define por Microsoft Corporation en la "Microsoft Portable Executable and Common Object File Format Specification" es un formato de archivo para ejecutables, código objeto y DLL (bibliotecas de enlace dinámico). Los archivos PE se usan en versiones de 32 bits y 64 bits de los sistemas operativos Windows de Microsoft. El formato de archivo PE es un formato altamente versátil que se puede usar en numerosos entornos de sistemas operativos y soporta varias soluciones de procesador.

Los desarrolladores de software pueden usar diversos esquemas para proteger el software, incluyendo archivos PE. Por ejemplo, los empaquetadores de software se pueden utilizar para comprimir binarios, que pueden disminuir el uso de ancho de banda asociado con la transferencia de los binarios y volumen de almacenamiento. De manera similar, los empaquetadores se pueden utilizar para proteger la propiedad intelectual incorporada dentro del software y para evitar un robo de código. El empaquetado puede implicar diversos esquemas de compresión y/o cifrado que pueden ofuscar los contenidos del código ejecutable. La ejecución del archivo ejecutable empaquetado puede desempaquetar el código ejecutable original (por ejemplo, que puede incluir descomprimir y/o descifrar) y entonces transferir el control al código ejecutable original. Por tanto, la naturaleza del código ejecutable puede no ser conocida hasta que esté ejecutándose realmente el software. Esto puede ser problemático, por ejemplo, si el código ejecutable es un programa de ordenador malintencionado u otro software indeseable, ya que la naturaleza del software no puede ser conocida hasta que sea demasiado tarde. El documento de patente de EE.UU. número 7367056 a nombre de Szor y otros divulga métodos, aparatos y medios legibles por ordenador para hacer frente a infecciones de códigos maliciosos en archivos informáticos. El documento de patente de EE.UU. número 7478431 a nombre de Nachenberg divulga métodos implementados en ordenador, aparatos y medios legibles por ordenador para detectar la presencia de infecciones de virus en archivos de destino localizados dentro de un ordenador.

Compendio de la descripción

Según una implementación, un método implementado por ordenador incluye identificar, mediante un dispositivo informático, un campo no válido de un archivo ejecutable portátil y determinar, mediante el dispositivo informático, la probabilidad de reparar el campo no válido de archivo ejecutable portátil. Un modelo de reparación se genera por el dispositivo informático para reparar el campo no válido del archivo ejecutable portátil. El campo no válido del archivo ejecutable portátil se repara por el dispositivo informático basándose, al menos en parte, en el modelo de reparación.

Se pueden incluir una o más de las siguientes características. Determinar una probabilidad de reparar el campo no válido del archivo ejecutable portátil puede incluir determinar el número y características de atributos del campo no válido que no se ajustan a una característica válida de un campo correspondiente válido basándose, al menos en parte, en una especificación de formato de archivo ejecutable portátil.

Generar un modelo de reparación puede incluir generar un modelo de reparación basándose en una o más reglas derivadas empíricamente. Reparar el campo no válido puede incluir reparar estáticamente el campo no válido. Reparar estáticamente el campo no válido puede incluir modificar, mediante el dispositivo informático, una imagen del archivo ejecutable portátil sobre un medio de almacenamiento. Reparar estáticamente el campo no válido puede incluir, también, almacenar, por el dispositivo informático, la imagen modificada en el medio de almacenamiento.

Reparar el campo no válido puede incluir reparar dinámicamente el campo no válido. Reparar dinámicamente el campo no válido puede incluir ejecutar, por el dispositivo informático, el archivo ejecutable portátil. Reparar dinámicamente el campo no válido puede incluir, también, modificar, mediante el dispositivo informático, un archivo almacenado en memoria durante la ejecución del archivo ejecutable portátil, en donde el archivo almacenado en memoria está basado en el archivo ejecutable portátil.

Reparar el campo no válido puede incluir deshabilitar, mediante el dispositivo informático, el campo no válido. Deshabilitar el campo no válido puede incluir eliminar, mediante el dispositivo informático, el campo no válido de una imagen del archivo ejecutable portátil almacenada en un medio de almacenamiento antes de la ejecución del archivo ejecutable portátil.

Según otra implementación, un producto de programa de ordenador incluye un medio legible por ordenador que tiene una pluralidad de instrucciones almacenadas en él. Cuando se ejecutan por un procesador, las instrucciones hacen al procesador realizar operaciones que incluyen identificar un campo no válido de un archivo ejecutable portátil y determinar una probabilidad de reparar el campo no válido del archivo ejecutable portátil. Un modelo de reparación se genera para reparar el campo no válido del archivo ejecutable portátil y el campo no válido del archivo

ejecutable portátil se repara basándose, al menos parte, en el modelo de reparación.

Se pueden incluir una o más de las características siguientes. Determinar una probabilidad de reparar el campo no válido del archivo ejecutable portátil puede incluir determinar el número y características de atributos del campo no válido que no se ajustan a una característica válida de un campo correspondiente válido basándose, al menos en parte, en una especificación de formato de archivo ejecutable portátil. Generar un modelo de reparación puede incluir generar un modelo de reparación basándose en una o más reglas derivadas empíricamente.

Reparar el campo no válido puede incluir reparar estáticamente el campo no válido. Reparar estáticamente el campo no válido puede incluir modificar una imagen del archivo ejecutable portátil sobre un medio de almacenamiento y almacenar la imagen modificada en el medio de almacenamiento.

Reparar el campo no válido puede incluir reparar dinámicamente el campo no válido. Reparar dinámicamente el campo no válido puede incluir ejecutar el archivo ejecutable portátil y modificar un archivo almacenado en memoria durante la ejecución del archivo ejecutable portátil, en donde el archivo almacenado en memoria está basado en el archivo ejecutable portátil.

Reparar el campo no válido puede incluir deshabilitar el campo no válido. Deshabilitar el campo no válido puede incluir eliminar el campo no válido de una imagen del archivo ejecutable portátil almacenada en un medio de almacenamiento antes de la ejecución del archivo ejecutable portátil.

Según otra implementación, un sistema incluye un procesador, y una memoria acoplada con el procesador. Un primer módulo de software es ejecutable por el procesador y la memoria. El primer módulo de software está configurado para identificar un campo no válido de un archivo ejecutable portátil. Un segundo módulo de software es ejecutable por el procesador y la memoria. El segundo módulo de software está configurado para determinar una probabilidad de reparar el campo no válido del archivo ejecutable portátil. Un tercer módulo de software es ejecutable por el procesador y la memoria. El tercer módulo de software está configurado para generar un modelo de reparación para reparar el campo no válido del archivo ejecutable portátil. Un cuarto módulo de software es ejecutable por el procesador y la memoria. El cuarto módulo de software está configurado para reparar el campo no válido del archivo ejecutable portátil basándose, al menos en parte, en el modelo de reparación.

Se pueden incluir una o más de las características siguientes. El segundo módulo de software, el cual está configurado para determinar una probabilidad de reparar el campo no válido del archivo ejecutable portátil, puede estar configurado para determinar el número y características de los atributos del campo no válido que no se ajustan a una característica válida de un campo correspondiente válido basándose, al menos en parte, en una especificación de formato de archivo ejecutable portátil.

El tercer módulo de software, el cual está configurado para generar un modelo de reparación, puede estar configurado para generar un modelo de reparación basándose en una o más reglas derivadas empíricamente.

El cuarto módulo de software, el cual está configurado para reparar el campo no válido, puede estar configurado para reparar estáticamente el campo no válido. Reparar estáticamente el campo no válido puede incluir modificar una imagen del archivo ejecutable portátil sobre un medio de almacenamiento y almacenar la imagen modificada en el medio de almacenamiento.

El cuarto módulo de software, el cual está configurado para reparar el campo no válido, puede estar configurado para reparar dinámicamente el campo no válido. Reparar dinámicamente el campo no válido puede incluir ejecutar el archivo ejecutable portátil y modificar un archivo almacenado en memoria durante la ejecución del archivo ejecutable portátil, en donde el archivo almacenado en memoria está basado en el archivo ejecutable portátil.

El cuarto módulo de software, el cual está configurado para reparar el campo no válido, puede estar configurado para deshabilitar el campo no válido. El cuarto módulo de software, el cual puede estar configurado para deshabilitar el campo no válido, puede estar configurado para eliminar el campo no válido de una imagen del archivo ejecutable portátil almacenada en un medio de almacenamiento antes de la ejecución del archivo ejecutable portátil.

Los detalles de una o más implementaciones se exponen en los dibujos anexos y la descripción a continuación. Otras características y ventajas llegarán a ser evidentes a partir de la descripción, los dibujos, y las reivindicaciones.

Breve descripción de los dibujos

La FIG. 1 representa esquemáticamente un dispositivo informático que puede ejecutar uno o más de un proceso de comprobación de validez, un proceso de reparación y un proceso de desempaquetado automatizado.

La FIG. 2 es un diagrama de flujo de un proceso realizado por el proceso de comprobación de validez de la FIG. 1.

La FIG. 3 es un diagrama de flujo de un proceso realizado por el proceso de reparación de la FIG. 1.

La FIG. 4 es un diagrama de flujo de un proceso realizado por el proceso de desempaquetado automatizado de la FIG. 1.

Descripción detallada de las realizaciones ejemplares

Como se apreciará por un experto en la técnica, la presente invención se puede incorporar como un sistema, método o producto de programa de ordenador. Por consiguiente, la presente invención puede tomar la forma de una realización enteramente hardware, una realización enteramente software (incluyendo microprogramas, software residente, microcódigo, etc.) o una realización que combina aspectos software y hardware que puede ser referida en la presente memoria de manera general como un "circuito", "módulo" o "sistema". Además, la presente invención puede tomar la forma de un producto de programa de ordenador incorporado en uno o más medios legibles por ordenador (es decir, utilizables en ordenador) que tienen un código de programa utilizable en ordenador incorporado en el mismo.

Se puede utilizar cualquier combinación de uno o más medios legibles por ordenador. El medio legible por ordenador incluye un medio de almacenamiento legible por ordenador, que puede ser, por ejemplo, pero no está limitado a, un sistema, un aparato, un dispositivo electrónico, magnético, óptico, electromagnético, de infrarrojos, o de semiconductores, o cualquier combinación adecuada de los precedentes. Un medio de almacenamiento legible por ordenador ejemplar puede incluir, pero no se limita a, un disquete de ordenador portátil, un disco duro, una unidad de disco de estado sólido, una memoria de acceso aleatorio (RAM), una memoria de sólo lectura (ROM), una memoria de sólo lectura programable y borrable (memoria EPROM o rápida), una fibra óptica, un disco compacto portátil de memoria de sólo lectura (CD-ROM), un dispositivo de almacenamiento óptico, un dispositivo de almacenamiento magnético, o cualquier combinación adecuada de los precedentes. En el contexto de este documento, un medio de almacenamiento legible por ordenador puede ser cualquier medio que pueda contener, o almacenar un programa para uso por o en conexión con un sistema, aparato o dispositivo de ejecución de instrucciones.

El código de programa de ordenador para llevar a cabo las operaciones de la presente invención puede estar escrito en un lenguaje de programación orientado a objetos tal como Java, Smalltalk, C++ o similares. No obstante, el código de programa de ordenador para llevar a cabo las operaciones de la presente invención también puede estar escrito en lenguajes de programación de procedimiento convencional, tales como el lenguaje de programación "C" o lenguajes de programación similares. El código de programa puede ejecutarse totalmente en un único dispositivo informático, por ejemplo, un paquete de software autónomo, y o puede ser al menos ejecutado parcialmente en múltiples dispositivos informáticos que pueden estar remotos unos de otros. En este último escenario, los dispositivos informáticos remotos se pueden conectar entre sí a través de una red de área local (LAN) o red de área extensa (WAN), o la conexión se puede hacer a uno o más dispositivos informáticos remotos (por ejemplo, a través de Internet usando un Proveedor de Servicios de Internet).

La presente invención se describe a continuación con referencia a ilustraciones de diagrama de flujo y/o diagramas de bloques de métodos, aparatos (sistemas) y productos de programas de ordenador según realizaciones de la invención. Se entenderá que cada bloque de las ilustraciones de diagrama de flujo y/o de los diagramas de bloques, y combinaciones de bloques en las ilustraciones de diagrama de flujo y/o diagramas de bloques, se pueden implementar mediante instrucciones de programa de ordenador. Estas instrucciones de programa de ordenador se pueden proporcionar a un procesador de un ordenador de propósito general, un ordenador de propósito especial, u otro aparato de procesamiento de datos programable para producir una máquina, de manera que las instrucciones, que se ejecutan a través del procesador del ordenador u otro aparato de procesamiento de datos programable, creen medios para implementar las funciones/actos especificados en el bloque o bloques del diagrama de flujo y/o diagrama de bloques.

Estas instrucciones de programa de ordenador también se pueden almacenar en una memoria legible por ordenador que pueden dirigir a un ordenador o a otro aparato de procesamiento de datos programable a funcionar de una manera particular, de manera que las instrucciones almacenadas en la memoria legible por ordenador producen un artículo de fabricación que incluye medios de instrucciones que implementan la función/acto especificado en el bloque o bloques del diagrama de flujo y/o diagrama de bloques.

Las instrucciones de programa de ordenador también se pueden cargar en un ordenador u otro aparato de procesamiento de datos programable para hacer que se realicen una serie de pasos operativos en el ordenador u otro aparato programable para producir un proceso implementado por ordenador de manera que las instrucciones que se ejecutan en el ordenador u otro aparato programable proporcionan los pasos para implementar las funciones/actos especificados en el bloque o bloques del diagrama de flujo y/o diagrama de bloques.

Con referencia a la FIG. 1, se muestra el proceso de comprobación de validez 10, el proceso de reparación 12, y el proceso de desempaquetado automatizado 14 que pueden residir cada uno en y se pueden ejecutar por el dispositivo informático 16. Mientras que cada uno del proceso de comprobación de validez 10, el proceso de reparación 12, y el proceso de desempaquetado automatizado 14 se muestran residiendo en el dispositivo informático 12, esto está previsto solamente con propósitos ilustrativos, ya que uno o más del proceso de comprobación de validez 10, el proceso de reparación 12, y el proceso de desempaquetado automatizado 14 pueden residir en un dispositivo informático separado.

Ejemplos de dispositivo informático 16 pueden incluir, pero no se limitan a: un ordenador personal, un ordenador servidor, una serie de ordenadores servidores, un miniordenador, y un ordenador central. El dispositivo informático 16 puede ejecutar un sistema operativo, como por ejemplo, Microsoft® Windows® XP o Red Hat® Linux®, por ejemplo. Varios dispositivos informáticos y sistemas operativos adicionales/alternativos pueden ser utilizados igualmente. Por ejemplo, el dispositivo informático 16 puede ser parte de una red informática distribuida con uno o más del proceso de comprobación de validez 10, proceso de reparación 12 y proceso de desempaquetado automatizado 14 que se ejecuta, en todo o en parte, en otro dispositivo informático acoplado con el dispositivo informático 16 a través de una red de datos (por ejemplo, una LAN, una WAN, Internet, etc.).

Como se debatirá a continuación en mayor detalle, el proceso de comprobación de validez 10 puede analizar sintácticamente una imagen binaria de un archivo ejecutable portátil para generar un campo analizado sintácticamente. El proceso de comprobación de validez 10 también puede determinar un atributo del campo analizado sintácticamente. Además el proceso comprobación de validez 10 puede comparar el atributo del campo analizado sintácticamente con una característica válida de un campo correspondiente válido basado, al menos en parte, en una especificación de formato de archivo ejecutable portátil. El proceso de comprobación de validez 10 también puede determinar si el atributo del campo analizado sintácticamente coincide con la característica válida del campo correspondiente válido.

Además, y como también se debatirá a continuación en mayor detalle, el proceso de reparación 12 puede identificar un campo no válido de un archivo ejecutable portátil. El proceso de reparación 12 puede determinar también la probabilidad de reparar el campo no válido del archivo ejecutable portátil. El proceso de reparación 12 puede generar un modelo de reparación para reparar el campo no válido del archivo ejecutable portátil. El proceso de reparación 12 puede reparar el campo no válido del archivo ejecutable portátil que se repara basado, al menos en parte, en el modelo de reparación.

De manera similar, y como también se debatirá a continuación en mayor detalle, el proceso de desempaquetado automatizado 14 puede establecer un punto de interrupción de depuración en una dirección de punto de entrada original de un archivo ejecutable portátil empaquetado. El proceso de desempaquetado automatizado 14 puede ejecutar también un proceso de depuración para el archivo ejecutable portátil empaquetado para obtener un archivo ejecutable portátil depurado en memoria. El proceso de desempaquetado automatizado 14 puede recoger también uno o más de los datos de la tabla de direcciones de importación y de los datos de la tabla de reubicación durante la ejecución del proceso de depuración para el archivo ejecutable portátil empaquetado. El proceso de desempaquetado automatizado 14 puede copiar el archivo ejecutable portátil depurado en memoria a un medio de almacenamiento, y puede terminar el proceso de depuración.

Los conjuntos de instrucciones y subrutinas del proceso de comprobación de validez 10, el proceso de reparación 12, y el proceso de desempaquetado automatizado 14, que pueden incluir uno o más módulos de software, y que se pueden almacenar en el dispositivo de almacenamiento 18 acoplado al dispositivo informático 16, se pueden ejecutar por uno o más procesadores (no mostrados) y uno o más módulos de memoria (no mostrados) incorporados en el dispositivo informático 16. El dispositivo de almacenamiento 18 puede incluir, pero no está limitado a: una unidad de disco duro; una unidad de estado sólido, una unidad de cinta; una unidad óptica; una agrupación RAID; una memoria de acceso aleatorio (RAM); y una memoria de sólo lectura (ROM).

Debido al hecho de que los archivos PE (ejecutables portátiles) contienen código ejecutable, puede ser deseable realizar una validación de archivo anterior a la ejecución del objeto binario (por ejemplo, la imagen binaria del archivo PE). El proceso de comprobación de validez 10 puede analizar una imagen binaria PE anterior a la ejecución para determinar si el archivo PE es una imagen binaria válida. Una imagen binaria válida puede referirse a un archivo que se puede usar por un sistema operativo dado, o bien como una imagen que contiene código ejecutable o bien otro tipo de información multimedia.

Como se ha debatido anteriormente, y con referencia también a la FIG. 2, el proceso de comprobación de validez 10 puede analizar sintácticamente 50 una imagen binaria de un archivo ejecutable portátil (por ejemplo, una imagen binaria PE 20, que reside en el dispositivo de almacenamiento 18, mostrado en la FIG. 1) para generar 52 un campo analizado sintácticamente. El proceso de comprobación de validez 10 también puede determinar 54 un atributo del campo analizado sintácticamente. Además, el proceso de comprobación de validez 10 puede comparar 56 el atributo del campo analizado sintácticamente con una característica válida de un campo correspondiente válido basada, al menos en parte, en una especificación de formato de archivo ejecutable portátil. El proceso de comprobación de validez 10 puede determinar 58 si el atributo del campo analizado sintácticamente coincide con la característica válida del campo correspondiente válido.

El proceso de comprobación de validez 10 puede analizar sintácticamente 50 la imagen binaria PE 20 para generar 52 un campo analizado sintácticamente. El proceso de comprobación de validez 10 puede analizar sintácticamente 50 la imagen binaria PE 20 para generar 52 una pluralidad de campos coherentes con el formato de archivo PE 100. Por ejemplo, el proceso de comprobación de validez 10 puede analizar sintácticamente 50 de manera general una imagen binaria PE en una firma de formato ejecutable portátil, un campo ImageBase, un campo SizeOfImage, un campo FileAlignment, un campo SectionAlignment, una dirección EntryPoint, una tabla de importación, una tabla de direcciones de importación, una tabla de exportación, una tabla de reubicación, una tabla de recursos, una tabla de

almacenamiento local de subprocesos, una tabla de configuración de carga, una tabla de importación vinculada, una tabla COM y una tabla de sección ejecutable portátil.

Aunque se han indicado varios campos, éstos se prevén solamente con propósitos ilustrativos, en la medida que el proceso de comprobación de validez 10 puede analizar sintácticamente 50 la imagen binaria PE 20 en varios campos adicionales/alternativos seleccionados basada en criterios de diseño y necesidad del usuario. Adicionalmente, el análisis sintáctico 50 de la imagen binaria PE 20 para generar 52 uno o más campos analizados sintácticamente, puede incluir, pero no se limita a, aislar físicamente cada campo (por ejemplo, copiar cada campo en un archivo separado, un campo de base de datos o similar), leer individualmente cada campo, asociar un desplazamiento con el principio (y/o final) de cada campo, o similares. Por tanto, el análisis sintáctico 50 de la imagen binaria PE 20 para generar 52 uno o más campos analizados sintácticamente puede permitir un examen individual de cada campo.

Como se ha debatido anteriormente, el proceso de comprobación de validez 10 también puede determinar 54 un atributo del campo analizado sintácticamente. El atributo determinado 54 mediante el proceso de comprobación de validez 10 puede incluir uno o más de un identificador de campo, una longitud de campo, y un contenido de campo. Por ejemplo, el proceso de comprobación de validez 10 puede determinar 54 que la imagen binaria PE 20 incluye un campo ImageBase que tiene un valor de 0x00400000, un campo SectionAlignment que tiene un valor de 0x1000, y un campo FileAlignment que tiene un valor de 0x200.

El proceso de comprobación de validez 10 puede comparar 56 uno o más atributos determinados 54 del campo analizado sintácticamente con una característica válida de un campo correspondiente válido basada, al menos en parte, en una especificación de formato de archivo ejecutable portátil. Un campo correspondiente válido puede incluir un campo que se requiere o permite por la “Microsoft Portable Executable and Common Object File Format Specification” publicada por Microsoft Corporation (PECOFF), y que corresponde a un campo analizado sintácticamente. Por ejemplo, un campo correspondiente válido para el campo ImageBase analizado sintácticamente puede ser el campo ImageBase permitido como un campo específico de Windows opcional por PECOFF. Una característica válida de un campo correspondiente válido puede incluir una característica que es admisible por PECOFF. Por ejemplo, PECOFF puede especificar identificadores de campo, longitudes de campo y contenidos de campo aceptables de una imagen binaria PE aceptada. Por ejemplo, PECOFF puede definir un valor de ImageBase por defecto de 0x00400000, y puede requerir que el valor sea un múltiplo de 64 K. De manera similar, PECOFF puede especificar que el campo SectionAlignment tenga un valor que sea mayor o igual al FileAlignment. Además, PECOFF puede especificar que el FileAlignment tenga un valor que sea una potencia de 2 entre 512 y 64 K, con un valor por defecto de 512. Por consiguiente, el precedente puede ser un ejemplo de características válidas para los campos identificados.

El proceso de comprobación de validez 10 puede determinar 58 si el atributo del campo analizado sintácticamente coincide con la característica válida del campo correspondiente válido basada, al menos en parte, en la comparación 56 entre el atributo del campo analizado sintácticamente y una característica válida de un campo correspondiente válido. Continuando con el ejemplo indicado anterior, el atributo determinado 54 para el campo FileAlignment de la imagen binaria PE 20 era 512. Como también se ha debatido anteriormente, PECOFF puede especificar que el campo FileAlignment tenga un valor que sea una potencia de 2 entre 512 y 64 K. Por consiguiente, el proceso de comprobación de validez 10 puede determinar 58 que el atributo del campo FileAlignment analizado sintácticamente (por ejemplo, que tiene un valor de 512) coincide con una característica válida de un campo FileAlignment válido.

La determinación 58 de si el atributo del campo analizado sintácticamente coincide con la característica válida del campo correspondiente válido puede incluir determinar 60 si el atributo del campo analizado sintácticamente es válido para un sistema operativo predeterminado. De nuevo, continuando con el ejemplo indicado anteriormente, el proceso de comprobación de validez 10 puede haber determinado 54 un valor de campo ImageBase de 0x00400000 para la imagen binaria PE 20. Este valor determinado puede ser el valor por defecto para los sistemas operativos Windows NT, Windows 2000, Windows XP, Windows 95, Windows 98 y Windows Me. No obstante, el valor por defecto del campo ImageBase para Windows CE es 0x00010000, por PECOFF. Por consiguiente, el proceso de comprobación de validez 10 puede determinar 60 que el campo ImageBase analizado sintácticamente para una imagen binaria PE 20 no es válido para Windows CE.

El proceso de comprobación de validez 10 puede determinar 58 si el campo analizado sintácticamente no coincide con la característica válida del campo correspondiente válido. Continuando con el ejemplo indicado anteriormente, PECOFF especifica que el campo SectionAlignment tiene un valor que es mayor o igual que el FileAlignment. Además, el proceso de comprobación de validez 10 puede haber determinado 54 un atributo de campo SectionAlignment de 256 y un atributo de campo FileAlignment de 512 para la imagen binaria PE 20. Por consiguiente, como el atributo de campo SectionAlignment determinado 54 (es decir, 256) no es mayor o igual que el atributo de campo FileAlignment determinado 54 (es decir, 512) para la imagen binaria PE 20, el proceso de comprobación de validez 10 puede determinar 58 que el campo SectionAlignment analizado sintácticamente no coincide con una característica válida de un campo correspondiente válido (es decir, el atributo de campo SectionAlignment determinado 54 no es mayor o igual que el atributo de campo FileAlignment determinado 54). Por consiguiente, el proceso de comprobación de validez 10 puede proporcionar un indicador (por ejemplo, proporcionar un indicador en una interfaz gráfica de usuario, no mostrada).

Si el campo analizado sintácticamente no coincide con la característica válida del campo correspondiente válido, el proceso de comprobación de validez 10 puede determinar 62 una probabilidad de modificar el campo analizado sintácticamente que no coincide con la característica válida del campo correspondiente válido para generar un campo válido. El proceso de comprobación de validez 10 puede determinar 62 la probabilidad de modificar el campo analizado sintácticamente para generar un campo válido basado, al menos en parte, en el número y la naturaleza de los errores en un campo que no coincide con la característica válida de un campo correspondiente válido. Por ejemplo, y continuando con el ejemplo debatido anteriormente, el campo SectionAlignment analizado sintácticamente de la imagen binaria PE 20 no coincide con una característica válida de un campo SectionAlignment válido porque el valor es menor que el valor del campo FileAlignment. El proceso de comprobación de validez 10 puede determinar 62 una probabilidad relativamente fuerte de ser capaz de modificar el campo SectionAlignment de la imagen binaria PE 20 para generar una característica válida en la medida que el campo SectionAlignment analizado sintácticamente de la imagen binaria PE 20 incluye un error bien definido único (esto es, el valor es menor que el campo FileAlignment). Por ejemplo, puede ser posible modificar el campo SectionAlignment para incluir un valor que es mayor o igual que el campo FileAlignment. Aunque pueden ser necesarias algunas pruebas recursivas para modificar el campo SectionAlignment de la imagen binaria PE 20 para lograr un campo válido, puede ser razonablemente probable que se pueda lograr tal modificación.

La probabilidad de modificación de un campo para generar un campo válido se puede determinar 62 basada, al menos en parte, en una o más reglas determinadas empíricamente. La una o más reglas determinadas empíricamente se pueden basar, al menos en parte, en varios tipos posibles de errores que pueden ocurrir en diversos campos, y las posibles modificaciones que se pueden implementar para corregir los errores. Por tanto, un tipo de error en un campo dado para el cual puede haber relativamente pocas modificaciones posibles que pueden dar como resultado generalmente un campo válido, el proceso de comprobación de validez 10 puede determinar 62 una probabilidad relativamente alta de modificación del campo para generar un campo válido. Por el contrario, para un tipo de error que tenga muchas modificaciones posibles, muchas de las cuales pueden no dar como resultado un campo válido, el proceso de comprobación de validez 10 puede determinar 62 una probabilidad relativamente baja de modificación del campo para generar un campo válido. De manera similar, si el número de los errores detectados entre el campo analizado sintácticamente y un campo correspondiente válido son relativamente grandes, el proceso de comprobación de validez 10 también determina 62 una probabilidad relativamente baja de modificación del campo para generar un campo válido.

El proceso de comprobación de validez 10 también puede determinar 64 si la imagen binaria del archivo ejecutable portátil incluye una biblioteca de vínculos dinámicos, un controlador de núcleo, o un objeto ejecutable. El proceso de comprobación de validez 10 puede determinar 64 la naturaleza de la imagen binaria PE 20 basada, por ejemplo, en uno o más de los campos incluidos, el contenido de los campos incluidos, o similares, evaluando los campos analizados sintácticamente relativos a posibles características válidas y posibles campos válidos. Varias características adicionales/alternativas de la imagen binaria PE 20 se pueden determinar de manera similar. Por ejemplo, el proceso de comprobación de validez 10 puede determinar uno o más de un entorno en el que puede ejecutarse el archivo PE, si el archivo PE es una consola u otra aplicación con una interfaz gráfica de usuario, si el archivo PE incluye dependencias y si las dependencias existen en el sistema de destino, si el archivo PE incluye funciones dependientes y si la función dependiente existe en bibliotecas disponibles en el sistema de destino, etc.

Como se ha mencionado brevemente anteriormente, el proceso de comprobación de validez 10 puede proporcionar una salida indicando los diversos campos analizados sintácticamente, los atributos de campo, la validez de los campos, la naturaleza del archivo PE, etc. En una realización, el proceso de comprobación de validez 10 puede proporcionar una interfaz gráfica de usuario, a través de la cual se pueden reproducir las diversas salidas. El proceso de comprobación de validez 10 adicional/alternativamente puede proporcionar una salida a una base de datos, archivo, etc., que puede ser consumida por un usuario a través de un programa adecuado, tal como una aplicación de base de datos. Otras diversas salidas adecuadas se apreciarán por los expertos en la técnica.

Las imágenes binarias PE pueden llegar a ser dañadas a través de varios mecanismos. Por ejemplo, los archivos PE pueden llegar a ser dañados cuando los archivos se transfieren de un medio a otro. De manera similar, se pueden introducir errores por los empaquetadores de software (por ejemplo, UPX, PECompact, ASPack, etc.). El error introducido por los empaquetadores de software puede reproducir algunos archivos válidos solamente para ciertas versiones de sistemas operativos que soportan formatos de archivos PE. Por consiguiente, el proceso de reparación 12 se puede implementar para reparar los archivos PE dañados.

Con referencia también a la FIG. 3, el proceso de reparación 12 puede identificar 100 un campo no válido de un archivo ejecutable portátil (por ejemplo, la imagen binaria PE 20 mostrada en la FIG. 1). Además, el proceso de reparación 12 puede determinar 102 una probabilidad de reparar el campo no válido del archivo ejecutable portátil. El proceso de reparación 12 puede generar 104 un modelo de reparación para reparar el campo no válido del archivo ejecutable portátil. El proceso de reparación 12 puede reparar 106 el campo no válido del archivo ejecutable portátil basado, al menos en parte, en el modelo de reparación generado 104 por el proceso de reparación 12.

El proceso de reparación 12 puede identificar 100 un campo no válido de imagen binaria PE 20 utilizando una variedad de mecanismos. Por ejemplo, el proceso de reparación 12 puede recibir 108 un indicador del proceso de comprobación de validez 10 (descrito en detalle anteriormente) indicando la validez de la imagen binaria PE 20 y/o

partes del subconjunto de la imagen binaria PE 20 (es decir, la validez de los diversos campos de la imagen binaria PE 20). El proceso de reparación 12 puede recibir 108 el indicador directamente del proceso de comprobación de validez 10. Adicional/alternativamente, en una realización en la que el proceso de comprobación de validez 10 puede generar un informe de validez (por ejemplo, en forma de un archivo, entradas de base de datos, o similares), el proceso de reparación 12 puede identificar 100 un campo no válido (y/o una pluralidad de campos no válidos) accediendo 110 al informe de validez e interpretando los contenidos del mismo.

En realizaciones adicionales, el proceso de reparación 12 puede identificar 100 un campo no válido de la imagen binaria PE 20 realizando 112 una o más comprobaciones de validez en la imagen binaria PE 20. Por ejemplo, el proceso de reparación 12 puede realizar una o más comprobaciones de validez en la imagen binaria PE 20 de una manera similar a la debatida anteriormente con referencia al proceso de comprobación de validez 10. Por ejemplo, el proceso de reparación 12 puede analizar sintácticamente de manera general la imagen binaria PE 20 en una pluralidad de campos, y puede comparar atributos de la pluralidad de campos con características válidas de los campos correspondientes válidos. Como se ha debatido anteriormente, las características válidas de los campos correspondientes válidos se pueden especificar por PECOFF. Por consiguiente, la validez de un campo (y/o de la imagen binaria PE 20 como un todo) se puede determinar basada, al menos en parte, en si los diversos campos y atributos cumplen con PECOFF. Por lo tanto, el proceso de reparación 12 puede identificar 100 un campo no válido como un campo que tiene un atributo que no cumple con una característica válida de unos campos correspondientes válidos como se especifica por PECOFF.

Cuando se identifica 100 un campo no válido, el proceso de reparación 12 puede examinar todos los campos de la imagen binaria PE 20, y/o puede prestar especial atención a los campos más cruciales de la imagen binaria PE 20. Ejemplos de campos que pueden ser particularmente importantes (por ejemplo, que pueden tener el impacto más grande sobre la capacidad de ejecución de la imagen binaria PE 20) pueden incluir, pero no se limitan a, firmas de formato PE, campos específicos PE (por ejemplo, ImageBase, SizeOfImage, FileAlignment, SectionAlignment, y la dirección de EntryPoint), y tablas específicas PE (por ejemplo, tabla de importación, tabla de direcciones de importación, tabla de exportación, tabla de reubicación, tabla de recursos, tabla de almacenamiento local de subprocesos, tabla de configuración de carga, tabla de importación vinculada, tabla COM y tablas de sección PE).

Como se ha debatido anteriormente, el proceso de reparación 12 puede determinar 102 una probabilidad de reparación del campo no válido (o múltiples campos no válidos) de la imagen binaria PE 20. El proceso de reparación 12 puede determinar 102 una probabilidad o reparar el campo no válido de imagen binaria PE 20 basado, al menos en parte, en la determinación 114 del número y las características de los atributos del campo no válido que no coinciden con una característica válida de un campo correspondiente válido basado, al menos en parte, en una especificación de formato de archivo ejecutable portátil (por ejemplo, PECOFF). Por ejemplo, se apreciará que diversos errores pueden tener una mayor probabilidad de ser reparables que otros errores. De manera similar, un archivo PE que tiene relativamente pocos errores puede tener una mayor probabilidad de ser reparable que un archivo PE que tiene un número relativamente grande de errores.

El proceso de reparación 12 puede determinar 102 la probabilidad de reparación de un campo no válido que incluye la comparación del campo o de los campos no válidos identificados 100 (incluyendo los atributos de los campos no válidos que no cumplen con PECOFF) contra una biblioteca de posibles errores y probabilidad de reparación del error. La biblioteca (por ejemplo, la biblioteca 22 que reside en el dispositivo de almacenamiento 18), puede incluir datos derivados empíricamente de varios errores que se hayan encontrado previamente y si fue posible reparar el error para obtener un archivo ejecutable.

Como se ha mencionado anteriormente, el proceso de reparación 12 puede generar 104 un modelo de reparación para reparar el campo o los campos no válidos identificados 100. El modelo de reparación generado 104 por el proceso de reparación 12 puede incluir uno o más algoritmos para reparar uno o más campos no válidos identificados 100. Similar a determinar 102 una probabilidad de reparación del campo no válido, el proceso de reparación 12 puede generar 104 el modelo de reparación basado, al menos en parte, en una o más reglas derivadas empíricamente (por ejemplo, que se pueden incluir en la biblioteca 22). Por ejemplo, el proceso de reparación 12 puede generar 104 un modelo de reparación para reparar un campo SizeOfImage no valido que tiene un valor anormal, en el que la regla puede incluir el nuevo cálculo de un valor SizeOfImage correcto. De manera similar, el proceso de reparación 12 puede generar 104 un modelo de reparación para reparar una sección de punto de entrada no válida que no incluye un atributo ejecutable, en el que la regla puede incluir la corrección de los atributos de sección. En un ejemplo adicional, el proceso de reparación 12 puede generar 104 un modelo de reparación para reparar datos de una tabla de recursos no válidos que no se pueden situar físicamente, en el que la regla puede incluir la eliminación temporal de los valores de la tabla de recursos no válidos en la cabecera PE. Adicionalmente, la biblioteca 22 puede incluir reglas que se derivan empíricamente basadas en una comparación entre diferentes versiones de sistema operativo y la forma en que las diferentes versiones de sistema operativo procesan el formato de archivo PE.

El proceso de reparación 12 puede reparar 106 el campo no válido del archivo ejecutable portátil basado, al menos en parte, en el modelo de reparación generado 104 por el proceso de reparación 12. Algunos errores (es decir, campos no válidos) se pueden reparar “en disco” modificando la imagen binaria PE 20 residente en el dispositivo de almacenamiento 18. Por consiguiente, el proceso de reparación 12 puede reparar 106 el campo no válido mediante

la reparación 116 estática del campo no válido. La reparación 116 estática del campo no válido puede incluir modificar 118 la imagen del archivo ejecutable portátil (por ejemplo, imagen binaria PE 20) en el dispositivo de almacenamiento 18. El proceso de reparación 12 puede almacenar 120 la imagen PE modificada en el dispositivo de almacenamiento 18.

Por ejemplo, y continuando con el ejemplo anterior, en el que el campo PE SizeOfImage fue identificado 100 como que es no válido para tener un valor anormal, el proceso de reparación 12 puede reparar estáticamente el campo SizeOfImage de la imagen binaria PE 20. Por ejemplo, el proceso de reparación puede volver a calcular un valor SizeOfImage correcto. El proceso de reparación 12 puede modificar la imagen binaria PE 20 para incluir el valor de SizeOfImage correcto. El proceso de reparación 12 puede almacenar la imagen binaria PE 20 modificada en el dispositivo de almacenamiento 18.

Además de los errores que se pueden reparar “en disco”, otros errores se pueden reparar en memoria. La determinación en cuanto a qué errores se pueden reparar “en disco” sobre qué errores se pueden reparar en memoria se puede basar, al menos en parte, en las reglas derivadas empíricamente (por ejemplo, que pueden residir en la biblioteca 22). Para errores que se pueden reparar en memoria, el proceso de reparación 12 puede reparar 122 dinámicamente el campo no válido. Para reparar 122 dinámicamente un campo no válido, el proceso de reparación 12 puede ejecutar 124 el archivo ejecutable portátil (por ejemplo, la imagen binaria PE 20). El proceso de reparación 12 además puede modificar 126 el archivo ejecutable portátil que reside en memoria (por ejemplo, en RAM) durante la ejecución, en la que el archivo ejecutable portátil que reside en memoria durante la ejecución se basa en el archivo ejecutable portátil (por ejemplo, basado en la imagen binaria PE 20).

Además, el proceso de reparación puede reparar 106 el campo no válido deshabilitando 128 el campo no válido. Por ejemplo, el proceso de reparación puede deshabilitar 128 temporalmente un campo no válido eliminando 130 el campo no válido de una imagen del archivo ejecutable portátil (por ejemplo, la imagen binaria PE 20) almacenada en el dispositivo de almacenamiento 18 anterior a la ejecución del archivo ejecutable portátil.

En algunas realizaciones, el proceso de reparación 12 puede reparar 106 un campo no válido deshabilitando 128 el campo no válido y reparando 122 dinámicamente el campo no válido. Por ejemplo, y con referencia al ejemplo anterior en el que los datos de la tabla de recursos podrían no estar situados físicamente, el proceso de reparación 12 puede eliminar 130 temporalmente los valores de la tabla de recursos no validos en la cabecera PE. El proceso de reparación 12 entonces puede ejecutar 124 la imagen binaria PE 20 (por ejemplo, el proceso de reparación 12 puede ejecutar un desempaquetador de la imagen binaria PE 20) hasta el punto de entrada original del archivo ejecutable portátil. El punto de entrada original puede incluir la primera instrucción de código del archivo ejecutable portátil antes de que el archivo ejecutable portátil fuera protegido (por ejemplo, empaquetado). Una vez que la ejecución de la imagen binaria PE 20 alcanza el punto de entrada original, la memoria de proceso se puede volcar 132 al dispositivo de almacenamiento 18. Es decir, la memoria de proceso asociada con la ejecución de la imagen binaria PE 20 que reside en RAM se puede guardar en el dispositivo de almacenamiento 18. Los datos de la tabla de recursos adquiridos de la memoria durante el desempaquetado de la imagen binaria PE 20 se pueden revertir a un estado original, y se puede almacenar un nuevo archivo PE basado, al menos en parte, en la memoria de proceso volcada. Por consiguiente, se puede lograr un archivo PE válido (es decir, un archivo PE en cumplimiento con PECOFF).

En una realización, una imagen binaria PE 24 puede incluir un archivo ejecutable portátil empaquetado. Un archivo ejecutable portátil empaquetado puede incluir un archivo ejecutable portátil (coherente con PECOFF, debatido en la presente memoria anteriormente) que puede incluir una o más protecciones de software, tales como compresión, cifrado, combinaciones de compresión y cifrado, etc. El proceso de desempaquetado automatizado 14, de manera general, puede ejecutar un proceso de depuración para el archivo ejecutable portátil empaquetado, y puede utilizar varios puntos de interrupción y rellamada para recoger datos de llenado de la tabla de direcciones de importación, así como otros diversos datos que se pueden usar para construir un archivo ejecutable portátil válido, desprotegido basado en la imagen binaria PE 24 (por ejemplo, protegida) empaquetada.

Con referencia también a la FIG. 4, el proceso de desempaquetado automatizado 14 en general puede fijar 150 un punto de interrupción de depuración en una dirección de punto de entrada original de un archivo ejecutable portátil empaquetado (por ejemplo, una imagen binaria de PE empaquetada 24, mostrada en la FIG. 1). El proceso de desempaquetado automatizado 14 también puede ejecutar 152 un proceso de depuración para el archivo ejecutable portátil empaquetado para obtener un archivo ejecutable portátil depurado en memoria (por ejemplo, en RAM). El proceso de desempaquetado automatizado 14 puede recoger 154 uno o más de los datos de la tabla de direcciones de importación y de los datos de la tabla de reubicación durante la ejecución 152 del proceso de depuración para el archivo ejecutable portátil empaquetado. El proceso de desempaquetado automatizado 14 puede copiar 156 el archivo ejecutable portátil depurado almacenado en memoria a un medio de almacenamiento (por ejemplo, el dispositivo de almacenamiento 18). El proceso de desempaquetado automatizado 14 puede terminar 158 el proceso de depuración en el punto de entrada original del archivo ejecutable portátil.

Como se ha debatido, el proceso de desempaquetado automatizado 14 puede establecer 150 un punto de interrupción de depuración en una dirección de punto de entrada original de la imagen binaria PE empaquetada 24. El punto de entrada original de la imagen binaria PE empaquetada 24 puede ser la primera instrucción del código

ejecutable antes de que el archivo fuera protegido. El establecimiento 150 de un punto de interrupción de depuración en la dirección del punto de entrada original de la imagen binaria PE empaquetada 24 puede permitir que la ejecución de la imagen binaria PE empaquetada 24 sea suspendida anterior al control que se pasa al archivo ejecutable incorporado dentro de la imagen binaria PE empaquetada 24. Tal como se usa en la presente memoria, "ejecución de la imagen binaria PE empaquetada" y "ejecutar la imagen binaria PE empaquetada" se puede referir a la ejecución del archivo incorporado por la imagen binaria PE empaquetada y ejecutar el archivo PE incorporado por la imagen binaria PE empaquetada. El proceso de desempaquetado automatizado 14 puede determinar 160 los datos del campo ImageBase del archivo ejecutable portátil empaquetado y los datos de AddressOfEntryPoint del archivo ejecutable portátil empaquetado. Los datos del campo ImageBase y los datos de AddressOfEntryPoint se pueden cargar desde la imagen binaria PE 24. La dirección del punto de entrada original de la imagen binaria PE empaquetada 24 puede ser la suma de los datos de ImageBase y de los datos de AddressOfEntryPoint. La determinación del punto de entrada original puede incluir adicionalmente otros cálculos numéricos, que se pueden basar, al menos en parte, en la disposición del empaquetador de software en sí mismo. El proceso de desempaquetado automatizado 14 puede cargar varios datos adicionales de la imagen binaria PE empaquetada 24, tales como, pero no limitados a, datos de ImageBase, datos de SizeOfImage, y datos de sección PE.

El proceso de desempaquetado automatizado 14 puede inicializar 162 el proceso de depuración. La inicialización 162 del proceso de depuración puede incluir crear un proceso de depuración basado, al menos en parte, en la imagen binaria PE empaquetada 24. Es decir, la inicialización 162 del proceso de depuración puede establecer un entorno de depuración en el que se puede ejecutar la imagen binaria PE empaquetada 24. En el proceso de depuración inicializado, el proceso de desempaquetado automatizado 14 puede establecer 150 un punto de interrupción de depuración en el punto de entrada original. El punto de interrupción de depuración establecido en el punto de entrada original se llamará una vez que el proceso de depuración finaliza la carga, antes de la ejecución de la primera instrucción del archivo ejecutable incorporado dentro de la imagen binaria PE empaquetada 24.

El proceso de desempaquetado automatizado 14 puede ejecutar 152 el proceso de depuración inicializado. Es decir, la imagen binaria PE empaquetada 24 se puede ejecutar dentro del entorno de depuración establecido. El proceso de desempaquetado automatizado 14 puede recoger 154 uno o más de los datos de la tabla de direcciones de importación y de los datos de la tabla de reubicación. La recogida 154 de uno o más de los datos de la tabla de direcciones de importación y de los datos de la tabla de reubicación puede incluir ejecutar el proceso de depuración hasta que alcance el código de llenado de la tabla de direcciones de importación. En parte, el proceso de desempaquetado automatizado 14 puede recoger 154 uno o más de los datos de la tabla de direcciones de importación y de los datos de la tabla de reubicación estableciendo 164 uno o más puntos de interrupción de depuración asociados con una llamada LoadLibrary, una llamada GetModuleHandle, y una llamada GetProcAddress. También se pueden asociar puntos de interrupción adicionales con una parte del empaquetador de software que reubica el archivo en memoria. Los puntos de interrupción asociados con una llamada LoadLibrary, una llamada GetModuleHandle, y una llamada GetProcAddress se pueden establecer, en algunas realizaciones, durante la inicialización 162 del proceso de depuración.

La imagen binaria PE empaquetada 24 que se ejecuta dentro del proceso de depuración puede utilizar una llamada de API LoadLibrary o una llamada de API GetModuleHandle con el fin de cargar una biblioteca de vínculos dinámicos dependientes. El ajuste 164 de uno o más puntos de interrupción asociados con una llamada LoadLibrary o una llamada GetModuleHandle puede dar como resultado una rellamada al proceso de desempaquetado automatizado 14 cuando se ejecuta la imagen binaria PE empaquetada 24 carga una biblioteca de vínculos dinámicos. En respuesta a la rellamada del punto de interrupción asociada con una llamada LoadLibrary o una llamada GetModuleHandle, el proceso de desempaquetado automatizado 14 puede recoger 154 el nombre de la biblioteca de vínculos dinámicos que se carga ejecutando la imagen binaria PE empaquetada 24.

De manera similar, la imagen binaria PE empaquetada 24 que se ejecuta dentro del proceso de depuración puede utilizar una llamada de API GetProcAddress para encontrar las ubicaciones de las API (interfaces de programación de aplicaciones) necesarias. El ajuste 164 de uno o más puntos de interrupción asociados con una llamada de API GetProcAddress puede dar como resultado una rellamada al proceso de desempaquetado automatizado 14 cuando se ejecuta la imagen binaria PE empaquetada 24 carga las direcciones de las API necesarias. En respuesta a la rellamada de punto de interrupción asociado con una llamada de API GetProcAddress, el proceso de desempaquetado automatizado 14 puede recoger 154 las direcciones de las API que se sitúan ejecutando la imagen binaria PE empaquetada 24. La ejecución de la imagen binaria PE empaquetada 24 puede llamar a la API GetProcAddress en dos ubicaciones, por ejemplo, para la ubicación de la API de cadena y la ubicación de la API ordinal. El proceso de desempaquetado automatizado 14 puede establecer 164 un punto de interrupción asociado con cada llamada de API GetProcAddress. El proceso de desempaquetado automatizado 14 puede añadir las ubicaciones de las API ubicadas por las llamadas de API GetProcAddress a la última biblioteca de vínculos dinámicos recogida.

El proceso de desempaquetado automatizado 14 puede copiar 156 un archivo PE depurado de memoria (por ejemplo, RAM) a un medio legible por ordenador, tal como el dispositivo de almacenamiento 18. Una vez que la imagen binaria PE empaquetada 24, que se ejecuta dentro del entorno de depuración, alcanza el punto de entrada original del archivo ejecutable incorporado en el mismo, el desempaquetado del archivo puede ser sustancialmente completo. Es decir, el archivo se puede descomprimir y/o descifrar, o similar (dependiendo de la naturaleza de las

protecciones asociadas con la imagen binaria PE empaquetada 24). Por tanto, en este punto un archivo PE desempaquetado puede residir en memoria asociada con el dispositivo informático 16 (por ejemplo, PE en memoria 26, mostrada en la FIG. 1). El proceso desempaquetado automatizado 14 puede copiar el PE desempaquetado en memoria 26, por ejemplo, a un archivo que reside en el medio de almacenamiento 18. Por tanto, el proceso de desempaquetado automatizado 14 puede crear el PE almacenado 28, que puede ser al menos una parte de un archivo ejecutable portátil desempaquetado basado, al menos en parte, en la imagen binaria PE empaquetada 24.

El proceso de desempaquetado automatizado 14 puede pegar 166 una o más de una tabla de direcciones de importación, basada, al menos en parte, en los datos de la tabla de direcciones de importación recogidos 154, y una tabla de reubicación, basada, al menos en parte, en los datos de la tabla de reubicación recogidos 154 en el archivo ejecutable portátil depurado (por ejemplo, PE 28 almacenado). Por ejemplo, el proceso de desempaquetado automatizado 14 puede construir una o más de una tabla de direcciones de importación y una tabla de reubicación basada, al menos en parte, en los datos de la tabla de direcciones de importación y los datos de la tabla de reubicación recogidos 154 por el proceso de desempaquetado automatizado 14 durante la ejecución 152 del proceso de depuración (por ejemplo, que puede incluir la ejecución de la imagen binaria PE empaquetada 24 dentro de un entorno de depuración).

El pegado 166 de una o más de una tabla de direcciones de importación, basada, al menos en parte, en los datos de la tabla de direcciones de importación 154 recogidos, y una tabla de reubicación, basada, al menos en parte, en los datos de la tabla de reubicación 154 recogidos en el archivo ejecutable portátil depurado (por ejemplo, PE 28 almacenado) puede incluir añadir 168 una nueva sección al archivo ejecutable portátil depurado. Por ejemplo, el PE 28 almacenado puede no incluir una sección para una tabla de direcciones de importación y/o una sección para una tabla de reubicación. Por consiguiente, el proceso de desempaquetado automatizado 14 puede hacer espacio para una tabla de direcciones de importación y/o una tabla de reubicación dentro del PE 28 almacenado (por ejemplo, añadiendo 168 una sección adecuada dentro del PE 28 almacenado para una tabla de direcciones de importación y/o una tabla de reubicación). El proceso de desempaquetado automatizado 14 entonces puede pegar 166 la tabla de direcciones de importación y/o la tabla de reubicación en las ubicaciones adecuadas del PE 28 almacenado.

Una vez que se han pegado 166 la tabla de direcciones de importación y/o la tabla de reubicación en el PE 28 almacenado, el proceso de desempaquetado automatizado 14 puede realinear 170 el archivo PE depurado (por ejemplo, PE 28 almacenado). Generalmente, realinear 170 el archivo PE depurado puede incluir compactar el archivo y verificar que el archivo es una imagen válida, por ejemplo, que puede incluir verificar que los tamaños físicos de las secciones PE individuales del archivo son correctas y tan pequeñas como sea posible. Adicionalmente, el proceso de desempaquetado automatizado 14 puede hacer leer, escribir y ejecutar todos los atributos de sección del archivo PE depurado (por ejemplo, PE 28 almacenado). Por tanto, el proceso de desempaquetado automatizado 14 puede crear un archivo PE válido que puede parecerse sustancialmente a la imagen binaria PE empaquetada 24 anterior a empaquetar (es decir, anterior a modificar el archivo con protecciones y/o compresión de software).

Con el proceso de desempaquetado completo, el proceso de desempaquetado automatizado 14 puede terminar 158 la depuración de la imagen binaria PE empaquetada 25 en el punto de entrada original.

Aunque se han debatido en la presente memoria anteriormente diversos procesos discretos, tal discusión se pretende por facilidad de explicación. Los diversos procesos discretos (y/o partes de los mismos) pueden incluir módulos de una mayor aplicación que pueden interactuar unos con otros. Adicionalmente, las diversas características y pasos de los procesos se pueden utilizar en combinación con las características y pasos de otros procesos descritos en la presente memoria. Por consiguiente, la presente descripción no se debería interpretar como que está limitada a los procesos discretos que se han descrito anteriormente.

Se han descrito una serie de implementaciones. Sin embargo, se entenderá que se pueden hacer diversas modificaciones. Por consiguiente, otras implementaciones están dentro del alcance de las siguientes reivindicaciones.

REIVINDICACIONES

1. Un método implementado en un ordenador que comprende:

identificar (100), mediante un dispositivo informático (16), un campo no válido de un archivo ejecutable portátil comparando atributos de los campos de dicho archivo ejecutable portátil con características válidas de campos correspondientes válidos según se especifica en la especificación de formato de archivo ejecutable portátil;

determinar (102), mediante el dispositivo informático (16), una probabilidad de reparar el campo no válido del archivo ejecutable portátil determinando (114) el número y características de atributos de los atributos de los campos no válidos que no se ajustan a una característica válida de un campo correspondiente válido basándose, al menos en parte, en la especificación de formato de archivo ejecutable portátil;

generar (104), mediante el dispositivo informático (16), un modelo de reparación para reparar el campo no válido del archivo ejecutable portátil generando el modelo de reparación basándose en una o más reglas derivadas empíricamente, en donde la una o más reglas derivadas empíricamente incluyen cálculos para corregir el campo no válido basándose, al menos en parte, en errores previos y si los errores previos fueron reparados para obtener un archivo ejecutable; y

reparar (106), mediante el dispositivo informático (16), el campo no válido del archivo ejecutable portátil basándose, al menos parte, en el modelo de reparación.

2. El método implementado en un ordenador de la reivindicación 1, en el que reparar (106) el campo no válido incluye reparar estáticamente (116) el campo no válido incluyendo:

modificar (118), mediante el dispositivo informático (16), una imagen del archivo ejecutable portátil sobre un medio de almacenamiento; y

almacenar (120), por el dispositivo informático (16), la imagen modificada en el medio de almacenamiento.

3. El método implementado en un ordenador de la reivindicación 1, en el que reparar (106) el campo no válido incluye reparar dinámicamente (122) el campo no válido incluyendo:

ejecutar (124), por el dispositivo informático (16), el archivo ejecutable portátil; y

modificar (126), mediante el dispositivo informático (16), un archivo almacenado en memoria durante la ejecución del archivo ejecutable portátil, en donde el archivo almacenado en memoria están basado en el archivo ejecutable portátil.

4. El método implementado en un ordenador de la reivindicación 1, en el que reparar (106) el campo no válido incluye deshabilitar (128), mediante el dispositivo informático (16), el campo no válido.

5. Un producto de programa de ordenador que comprende un medio legible por ordenador que tiene una pluralidad de instrucciones almacenadas en el mismo, que, cuando se ejecutan por un procesador, hacen al procesador realizar operaciones que comprenden:

identificar (100) un campo no válido de un archivo ejecutable portátil comparando atributos de los campos de dicho archivo ejecutable portátil con características válidas de campos correspondientes válidos según se especifica en la especificación de formato de archivo ejecutable portátil;

determinar (102) una probabilidad de reparar el campo no válido de archivo ejecutable portátil determinando (114) el número y características de atributos de los atributos de los campos no válidos que no se ajustan a una característica válida de un campo correspondiente válido basándose, al menos en parte, en la especificación de formato de archivo ejecutable portátil;

generar (104) un modelo de reparación para reparar el campo no válido del archivo ejecutable portátil generando el modelo de reparación basándose en una o más reglas derivadas empíricamente, en donde la una o más reglas derivadas empíricamente incluyen cálculos para corregir el campo no válido basándose, al menos en parte, en errores previos y si los errores previos fueron reparados para obtener un archivo ejecutable; y

reparar (106) el campo no válido del archivo ejecutable portátil basándose, al menos parte, en el modelo de reparación.

6. El producto de programa informático de la reivindicación 5, en el que reparar (106) el campo no válido incluye reparar estáticamente (116) el campo no válido incluyendo:

modificar (118) una imagen del archivo ejecutable portátil sobre un medio de almacenamiento; y almacenar (120) la imagen modificada en el medio de almacenamiento.

7. El producto de programa informático de la reivindicación 5, en el que reparar (106) el campo no válido incluye

reparar dinámicamente (122) el campo no válido incluyendo:

ejecutar (124) el archivo ejecutable portátil ; y

modificar (126) un archivo almacenado en memoria durante la ejecución del archivo ejecutable portátil, en donde el archivo almacenado en memoria está basado en el archivo ejecutable portátil.

8. El producto de programa informático de la reivindicación 5, en el que reparar (106) el campo no válido incluye deshabilitar (128) el campo no válido.

9. Un sistema que comprende:

un procesador;

una memoria acoplada con el procesador;

un primer módulo de software ejecutable por el procesador y la memoria, configurado el primer módulo de software para identificar (100) un campo no válido de un archivo ejecutable portátil comparando atributos de los campos de dicho archivo ejecutable portátil con características válidas de campos correspondientes válidos según se especifica en la especificación de formato de archivo ejecutable portátil;

un segundo módulo de software ejecutable por el procesador y la memoria, configurado el segundo módulo de software para determinar (102) una probabilidad de reparar el campo no válido del archivo ejecutable portátil determinando (114) el número y características de atributos de los atributos del campo no válido que no se ajustan a una característica válida de un campo correspondiente válido basándose, al menos en parte, en la especificación de formato de archivo ejecutable portátil;

un tercer módulo de software ejecutable por el procesador y la memoria, configurado el tercer módulo de software para generar (104) un modelo de reparación para reparar el campo no válido del archivo ejecutable portátil generando un modelo de reparación basándose en una o más reglas derivadas empíricamente, en donde la una o más reglas derivadas empíricamente incluyen cálculos para corregir el campo no válido basándose, al menos en parte, en errores previos y si los errores previos fueron reparados para obtener un archivo ejecutable; y

un cuarto módulo de software ejecutable por el procesador y la memoria, configurado el cuarto módulo de software para reparar (106) el campo no válido del archivo ejecutable portátil basándose, al menos parte, en el modelo de reparación.

10. El sistema de la reivindicación 9, en el que el cuarto módulo de software, configurado para reparar (106) el campo no válido, está configurado para reparar estáticamente (116) el campo no válido incluyendo:

modificar (118) una imagen del archivo ejecutable portátil sobre un medio de almacenamiento; y almacenar (120) la imagen modificada en el medio de almacenamiento.

11. El sistema de la reivindicación 9, en el que el cuarto módulo de software, configurado para reparar (106) el campo no válido, está configurado para reparar dinámicamente (122) el campo no válido incluyendo:

ejecutar (124) el archivo ejecutable portátil; y modificar (126) un archivo almacenado en memoria durante la ejecución del archivo ejecutable portátil, en donde el archivo almacenado en memoria están basado en el archivo ejecutable portátil.

12. El sistema de la reivindicación 9, en el que el cuarto módulo de software, configurado para reparar (106) el campo no válido, está configurado para deshabilitar (128) el campo no válido.

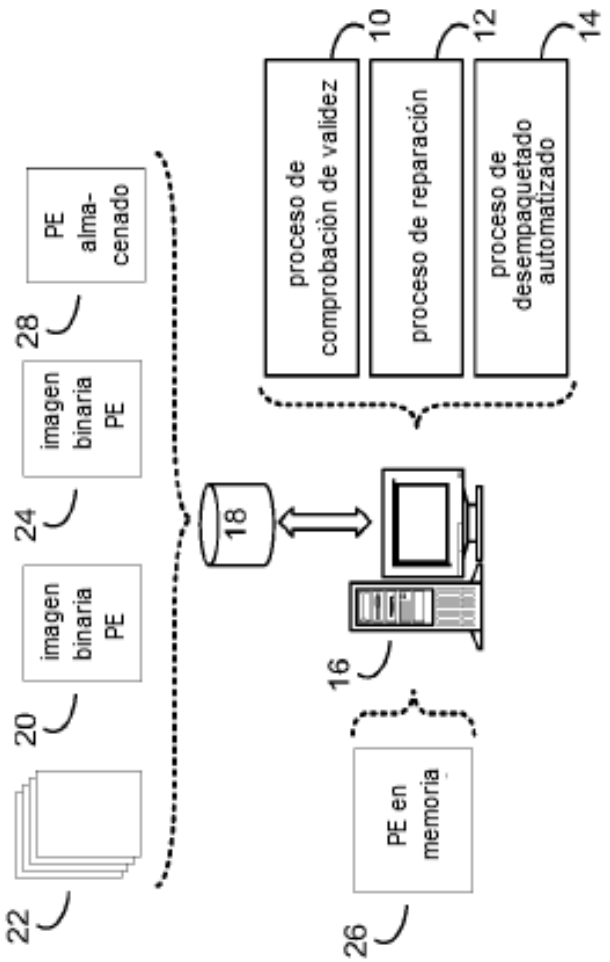
proceso de comprobaciòn de validez
10
proceso de reparación
12
proceso de desempaquetado automatizado
14
18
16
PE en memoria
26
22
20
imagen binaria PE
24
imagen binaria PE
28
PE alma-cenado

FIG. 1

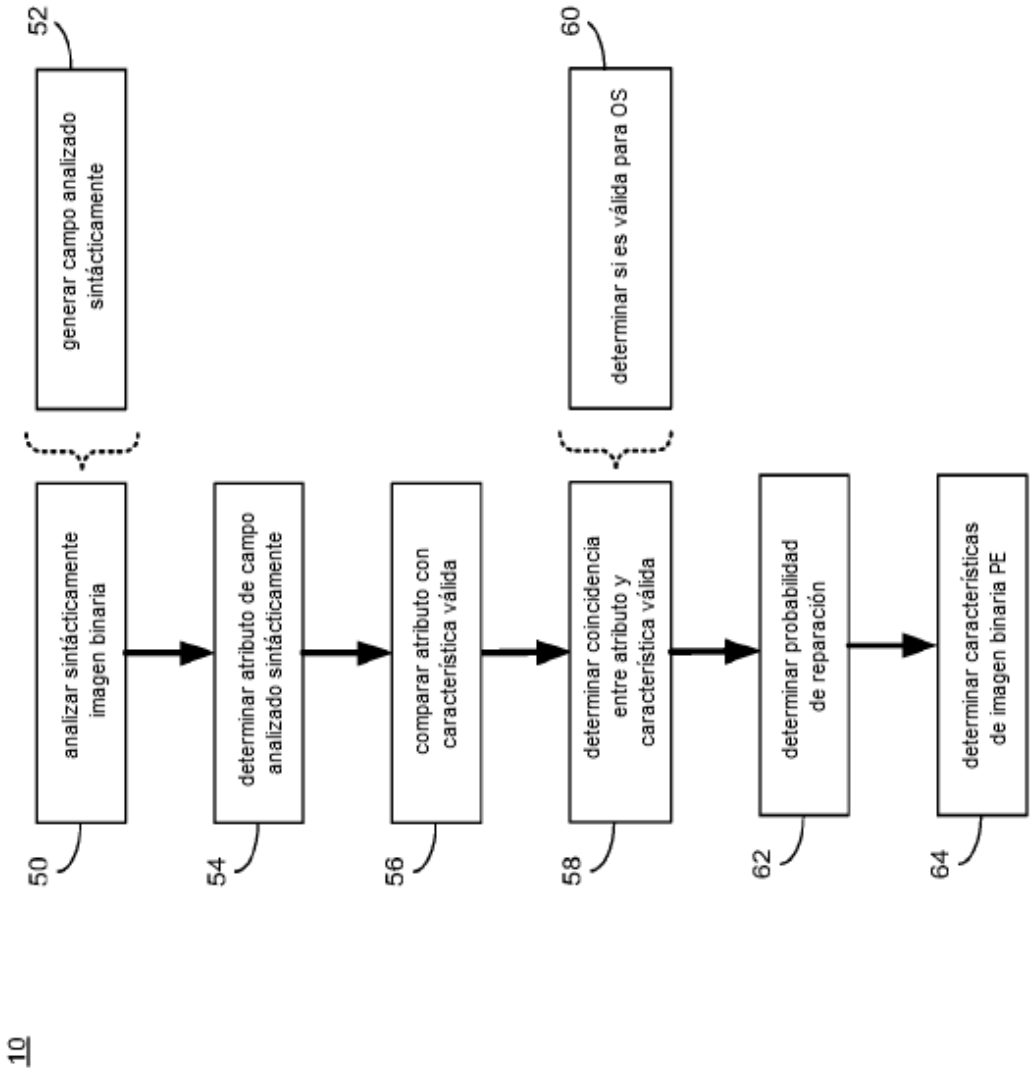
10
50
analizar sintácticamente imagen binaria
52
generar campo analizado sintácticamente
54
determinar atributo de campo analizado sintácticamente
56
comparar atributo con característica válida
58
determinar coincidencia entre atributo y característica válida
60
determinar si es válida para OS
62
determinar probabilidad de reparación
64
determinar características de imagen binaria PE

FIG. 2

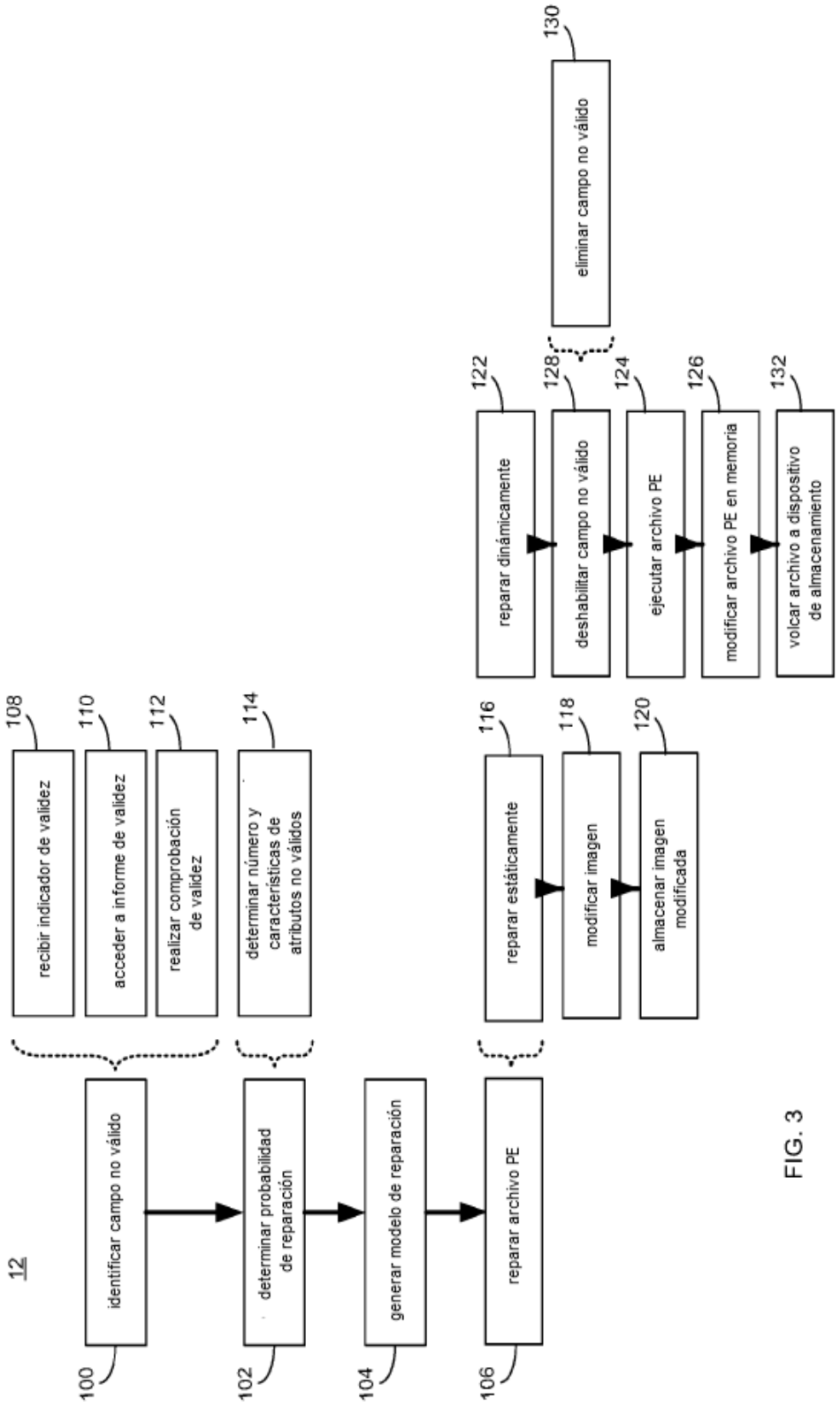
12
identificar campo no válido
100
determinar probabilidad de reparación
102
generar modelo de reparación
104
reparar archivo PE
106
recibir indicador de validez
108
acceder a informe de validez
110
realizar comprobación de validez
112
determinar número y características de atributos no válidos
114
reparar estáticamente
116
modificar imagen
118
almacenar imagen modificada
120
reparar dinámicamente
122
deshabilitar campo no válido
128
eliminar campo no válido
130
ejecutar archivo PE
124
modificar archivo PE en memoria
126
volcar archivo a dispositivo de almacenamiento
132

FIG. 3

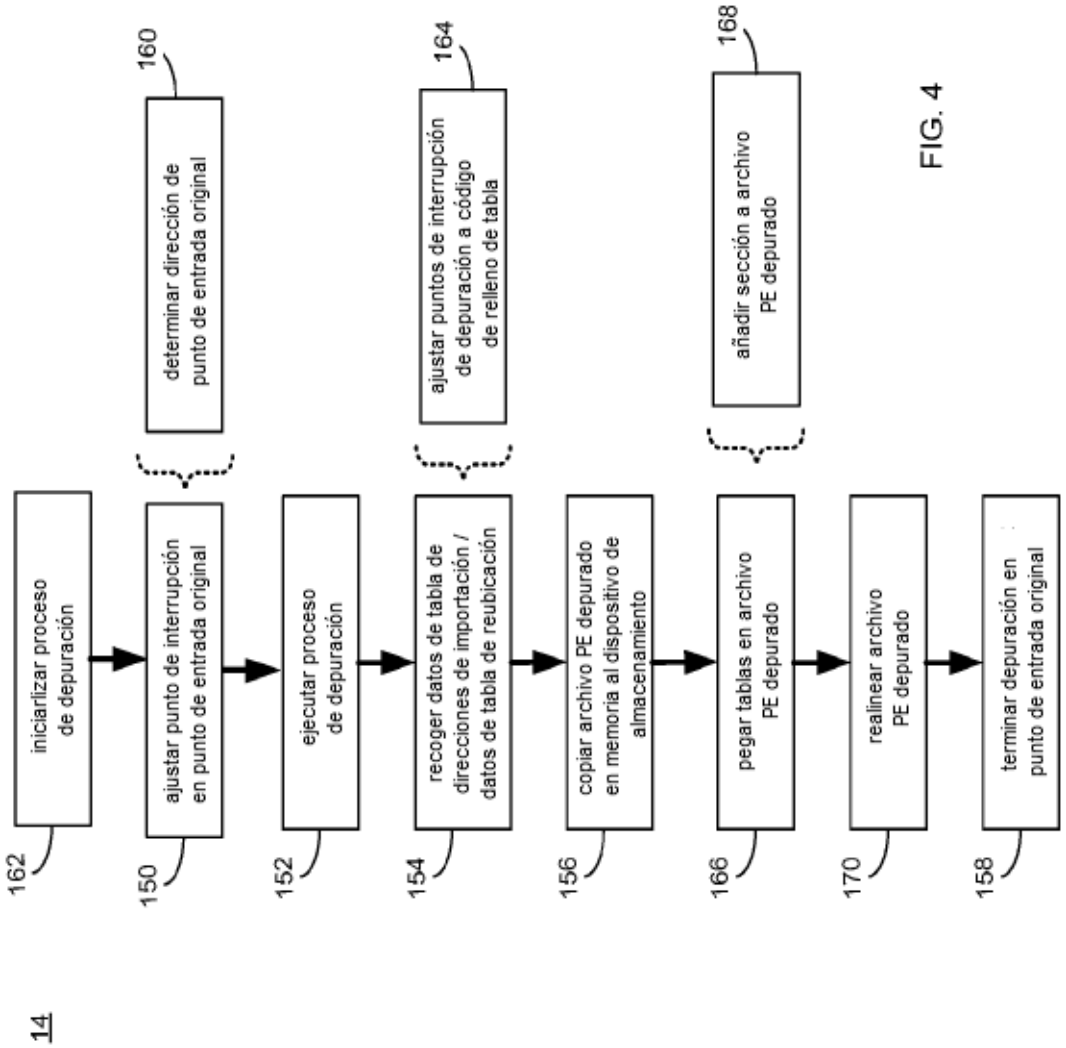
14
iniciarlizar proceso de depuración
162
ajustar punto de interrupción en punto de entrada original
150
160
determinar dirección de punto de entrada original
ejecutar proceso de depuración
152
recoger datos de tabla de direcciones de importación / datos de tabla de reubicación
154
164
ajustar puntos de interrupción de depuración a código de relleno de tabla
copiar archivo PE depurado en memoria al dispositivo de almacenamiento
156
pegar tablas en archivo PE depurado
166
168
añadir sección a archivo PE depurado
realinear archivo PE depurado
170
terminar depuración en punto de entrada original
158

FIG. 4