(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 524 788**

(51) Int. CI.:

G06F 21/56 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **16.11.2009 E 09014289 (4)**

(97) Fecha y número de publicación de la concesión europea: **13.08.2014 EP 2189920**

(54) Título: **Generador de firmas de software malicioso y detección de código ejecutable**

(30) Prioridad:

17.11.2008 IL 19534008

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
12.12.2014

(73) Titular/es:

DEUTSCHE TELEKOM AG (100.0%)
FRIEDRICH-EBERT-ALLEE 140
53113 BONN, DE

(72) Inventor/es:

DOLEV SHLOMO y
TZACHAR NIR

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Generador de firmas de software malicioso y detección de código ejecutable

Campo de la invención

La presente invención se refiere al campo de la detección de software malicioso. Más en particular, la invención se refiere a la generación y detección de firmas de software malicioso.

Antecedentes de la invención

La detección de software malicioso (en lo sucesivo "*malware*", que es cualquier programa o archivo que es dañino para un ordenador) mediante una firma es una práctica habitual en cualquier herramienta de seguridad de software. Generando una firma que identifica de manera unívoca a un malware, puede evitarse la propagación de este malware a través de las redes e impedir que llegue a otros ordenadores.

Una firma de un ejecutable (un archivo que contiene un programa y que puede ejecutarse o hacerse funcionar como un programa en el ordenador) puede generarse mediante diversas técnicas. Por ejemplo, puede usarse una secuencia binaria intrínseca en el ejecutable y aplicar un algoritmo *hash* a esta secuencia usando una función *hash* criptográfica, tal como MD5 (un algoritmo que se usa para verificar la integridad de los datos a través de la creación de un compendio de mensajes de 128 bits a partir de datos únicos introducidos, como la huella dactilar de una persona) o SHA1 (una función *hash* criptográfica). Los datos *hash* generados pueden usarse como una firma. Otra técnica habitual es incluir detalles que residen en las cabeceras de los ejecutables. Por ejemplo, las librerías con las que están relacionados.

La firma generada debe ser lo más fiable posible. La tasa de falsos positivos y de falsos negativos debe ser igual a cero (o ser lo más próxima posible a cero). Para conseguir este objetivo, la generación de firmas es llevada a cabo habitualmente por expertos e implica la inspección rigurosa del código sospechoso. Sólo tras identificar un código común (un código generado por compiladores, librerías relacionadas, etc.) el experto puede extraer los fragmentos de código que identifican de manera unívoca al propio malware.

Una de las desventajas más evidentes de la detección basada en firmas es el hacer frente a nuevo malware aún sin tratar. Una firma específicamente vinculada a un ejecutable dado no está diseñada para adecuarse a otros ejecutables, lo cual deja una puerta abierta a malware que se haya generado recientemente.

Otro problema evidente es la gran cantidad de firmas que deben mantenerse y comprobarse cuando se inspeccionan ejecutables sospechosos.

Las técnicas actuales para la generación de firmas son o bien manuales (llevadas a cabos por expertos) o automáticas. Los expertos confían en su capacidad de desensamblar el ejecutable sospechoso, analizar el flujo del código e identificar secciones de código que son únicas del ejecutable sospechoso. Los sistemas automatizados no pueden basarse en el análisis de los flujos y utilizan técnicas de extracción de datos (o similares). Por ejemplo, el sistema del polígrafo ("*Polygraph: Automatically Generating Signatures for Polymorphic Worms*," de James Newsome, Brad Karp, Dawn Song, S&P05, páginas 226 a 241, 2005) trata de encontrar patrones coincidentes en un flujo de tráfico de red sospechoso. Tales patrones comunes son convertidos después en firmas.

El documento US 5.452.442 da a conocer un procedimiento estadístico para extraer y/o evaluar automáticamente firmas de un virus informático. Una firma candidata es aceptada como una firma válida si la probabilidad estimada de la aparición de la firma de virus candidata es menor que una probabilidad umbral.

El documento US 2007/0094734 da a conocer un procedimiento para clasificar software informático polimórfico extrayendo características de un archivo sospechoso y comparando las características extraídas con características de tipos de software conocidos.

El documento EP 0 896 285 da a conocer un procedimiento para la detección eficaz de virus informáticos aplicando un primer mapeo para obtener un subconjunto de firmas estándar y aplicando un segundo mapeo para identificar un conjunto de virus informáticos que podrían estar presentes en una cadena de datos.

El documento US 2007/0152854 da a conocer un procedimiento para determinar si un archivo informático sospechoso es malicioso modelando una secuencia de código de octetos extraída usando al menos una prueba de modelación de entropía, comparando cada resultado de entropía con una tabla para determinar un valor de probabilidad y sumando los valores de probabilidad para determinar la probabilidad de que la secuencia de código de octetos sea maliciosa.

El documento US 2007/0240221 da a conocer un sistema y un procedimiento para detectar malware en una plataforma móvil. Se calcula una suma de control y esa suma de control se compara con una suma de control obtenida a partir de una copia libre de malware del ejecutable.

El documento WO 01/69356 da a conocer un sistema de detección de virus basado en histogramas que presenta un archivo de datos de código P, un archivo de definiciones de virus para guardar firmas de virus conocidos, y un motor que contiene un intérprete de código P, un módulo de exploración para explorar regiones del archivo de definiciones de virus y un módulo de emulación para emular instrucciones del archivo.

El documento US 2006/0053295 da a conocer un procedimiento y un sistema para detectar una firma de gusano en un tráfico de red. Se calcula una función *hash* para al menos una parte del flujo de datos. Al menos un contador se incrementa en respuesta al resultado calculado de la función *hash*. Cada contador corresponde a un resultado respectivo calculado de una función *hash*. El contenido repetido se identifica cuando al menos un contador supera un valor de cómputo. Después se comprueba si el contenido repetido identificado no es una cadena benigna.

Por lo tanto, un objeto de la presente invención es reducir el número de firmas usadas para detectar malware, acelerando así el proceso de detección de malware.

Otro objeto de la presente invención es poder detectar nuevo malware desconocido.

Otros objetos y ventajas de la invención resultarán evidentes a medida que se avance en la descripción.

Resumen de la invención

La presente invención se refiere a un procedimiento para implementar un generador de firmas de malware y detectar códigos ejecutables, según el cual se inspeccionan los códigos de operación de los ejecutables, descartándose al mismo tiempo otros parámetros de los ejecutables. Las firmas se generan a partir de motores comunes de códigos ejecutables, y una gran variedad de malware de la misma familia se representa mediante un pequeño número de firmas. De este modo se identifica malware conocido y desconocido usando el pequeño número de firmas.

En una realización, los ejecutables benignos son desensamblados por un desensamblador, y las secuencias de código de operación del ejecutable benigno son almacenadas en una primera base de datos y cada ejecutable malicioso es desensamblado por el desensamblador en una secuencia de códigos de operación. Un comparador de cadenas obtiene todas las subcadenas de códigos de operación comunes (por ejemplo, usando un proceso LCS) para cada par de ejecutables maliciosos. Un conjunto de subcadenas de códigos de operación comunes entre dos ejecutables maliciosos cualesquiera está compuesto por el comparador de cadenas, que también comprueba cuál de las subcadenas de códigos de operación aparece en al menos una secuencia de códigos de operación de ejecutable benigno almacenada en la primera base de datos. Las subcadenas de códigos de operación comunes son ignoradas y un conjunto de subcadenas de códigos de operación comunes, que no aparecen en ningún código de operación de ejecutable benigno para cada par de ejecutables maliciosos, es identificado por el comparador de cadenas. Todos los conjuntos de subcadenas de códigos de operación comunes son almacenados en una segunda base de datos como firmas candidatas.

El comparador de cadenas puede usarse para encontrar, para cada firma candidata, los malwares en los que aparece la firma candidata, así como el conjunto mínimo de firmas candidatas que identifican a todos los ejecutables maliciosos. Después, el conjunto mínimo es almacenado en la segunda base de datos.

Las secuencias de códigos de operación pueden convertirse en expresiones regulares, las cuales pueden emparejarse usando secuencias binarias.

Las firmas pueden convertirse en expresiones regulares extrayendo, mediante el convertidor de cadenas, subcadenas de códigos de operación (firmas) de la segunda base de datos y convirtiendo las subcadenas de códigos de operación en expresiones regulares usando una tabla de consulta; reduciendo la longitud de las expresiones regulares resultantes mediante el convertidor de cadenas añadiendo “máscaras binarias” como elementos emparejables; y almacenando las expresiones regulares en la segunda base de datos.

Breve descripción de los dibujos

Estas y otras características y ventajas de la invención se entenderán mejor a través de la siguiente descripción detallada, ilustrativa y no limitativa de realizaciones preferidas de la misma, con referencia a los dibujos adjuntos, en los que:

- la Fig. 1 ilustra un sistema para reducir el número de firmas usadas para detectar malware, según una realización de la invención;
- la Fig. 2 es un diagrama de flujo que ilustra la generación de firmas para una familia de malware; y
- la Fig. 3 es un diagrama de flujo que ilustra la generación de un pequeño conjunto de firmas para una familia de malware.

Descripción detallada de realizaciones preferidas

La presente invención da a conocer un procedimiento y el uso de extracción de firmas a partir de malware. Por consiguiente, se identifican familias de malware, tal como dos miembros de la misma familia que comparten un "motor" común.

Es bien sabido que existen herramientas de generación de malware que usan un motor común para crear nuevo malware. La presente invención propone generar firmas a partir de dicho motor, permitiendo identificar así una gran variedad de malware de la misma familia, e incluso miembros desconocidos de la misma familia, mediante un pequeño número de firmas. Usar un pequeño número de firmas para representar una gran variedad de malware aligera el proceso de detección de malware, haciéndolo más eficiente en lo que respecta a los recursos asignados, el uso de la CPU y la complejidad en el tiempo. Además, los futuros diseños de malware que usen el mismo motor compartido también podrán ser detectados por las firmas generadas propuestas.

Según la presente invención, solo se inspecciona la parte de una instrucción en lenguaje máquina que especifica la operación que va a llevarse a cabo (en lo sucesivo, "códigos de operación") de un ejecutable, descartando al mismo tiempo cualquier parámetro dado. Ésta es una técnica nueva que no se ha considerado anteriormente. El motivo es que hay que darse cuenta de que los diseñadores de malware tratarán de ocultar tales motores comunes usando una gran variedad de técnicas. Por ejemplo, estos motores comunes pueden residir en diferentes ubicaciones dentro de diferentes ejecutables, pueden estar relacionados con diferentes direcciones de memoria o incluso pueden estar ligeramente modificados. Usando los códigos de operación pueden vencerse los procedimientos usados por los diseñadores de malware.

Generación de firmas para múltiples archivos

Un sistema para reducir el número de firmas usadas para detectar malware, según una realización de la invención, se muestra en la Fig. 1. El sistema comprende un ordenador 100, conectado a una primera base de datos 10 (DB 10) y a una segunda base de datos 20 (DB 20). El ordenador 100 incluye un desensamblador 200 (usado para convertir un programa que está en su forma ejecutable, lista para usarse, o código objeto en alguna forma de representación de lenguaje ensamblador que sea legible), un comparador de cadenas 300 y un convertidor de cadenas 400.

La Fig. 2 ilustra un diagrama de flujo del proceso que se lleva a cabo según una realización de la presente invención. Por consiguiente, la generación de firmas comienza en una etapa de preprocesamiento. En la etapa 21, una DB 10 que almacena muestras de ejecutables benignos se utiliza para desensamblar cada ejecutable benigno mediante el desensamblador 200, y las secuencias de códigos de operación de dicho ejecutable benigno se almacenan en la DB 10. Según una realización de la presente invención, se usa el desensamblador IDA-Pro (DataRescue, Liège – Bélgica). Una gran y diversa base de datos es esencial, ya que el tamaño y la variedad de la base de datos afecta directamente a la tasa de falsos positivos (es decir, identificar un ejecutable como malware aunque éste sea totalmente seguro). Según una realización de la presente invención, se establece una base de datos de 23.000 ejecutables benignos que abarcan la mayoría de los sistemas operativos más comunes.

La etapa 22 ilustra que dada una familia de malware (la generación de firmas para malware autónomo se describirá posteriormente), cada ejecutable malicioso es desensamblado por el desensamblador 200 en una secuencia de códigos de operación. Según una realización de la presente invención, se usa el desensamblador IDA-Pro. La etapa 23 ilustra que para cada par de ejecutables maliciosos, el comparador de cadenas 300 obtiene todas las subcadenas de códigos de operación comunes usando cualquier algoritmo LCS (subcadena común más grande) de tiempo lineal. La etapa 23 ilustra además que después de que el comparador de cadenas 300 componga el conjunto de subcadenas de códigos de operación comunes entre dos ejecutables maliciosos cualesquiera, comprueba cuál de dichas subcadenas de códigos de operación aparece en al menos una secuencia de códigos de operación de ejecutable benigno en la DB 10, y estas subcadenas de códigos de operación comunes son ignoradas. El resultado final de esta etapa es que para cada par de ejecutables maliciosos existe un conjunto de subcadenas de códigos de operación comunes que no aparecen en ningún código de operación de ejecutable benigno. Estos conjuntos de subcadenas de códigos de operación comunes son almacenados por el comparador de cadenas 300 en la DB 20. Cada una de estas subcadenas de códigos de operación, que son las firmas candidatas, puede usarse como una firma.

Según una realización de la presente invención, se usan todas las firmas candidatas almacenadas en la DB 20.

Según otra realización de la presente invención, se desea un pequeño número de firmas (por ejemplo, en un filtrado en tiempo real) y se usa el número más pequeño posible de firmas candidatas, ya que cada subcadena de códigos de operación puede cubrir potencialmente más de un par de ejecutables maliciosos. Las firmas obtenidas usando este proceso pueden contener los segmentos de código compartidos de los ejecutables maliciosos que, a su vez, definen el motor "común" de dichos ejecutables maliciosos. Además, es muy probable que tales motores comunes coincidan con versiones futuras de estos ejecutables maliciosos, ya que las versiones futuras utilizarán normalmente dicho motor común. Haciendo referencia a la Fig. 3, las etapas 31 a 33 son idénticas a las etapas 21 a 23 especificadas en la Fig. 2. Sin embargo, con el fin de obtener un pequeño número de firmas a partir del conjunto de firmas candidatas, como se

ilustra en la etapa 34, el comparador de cadenas 300 halla para cada firma candidata el malware en el que aparece (ya que puede aparecer en más de dos). Después, el comparador de cadenas 300 obtiene el conjunto mínimo de firmas candidatas de modo que todos los ejecutables maliciosos sean identificados mediante dicho conjunto. Esto se realiza convirtiendo el problema del conjunto mínimo de firmas candidatas en el problema de colorear el conjunto de archivos de malware.

Cada firma candidata se convierte en un color y, de este modo, se colorean todos los ejecutables maliciosos en los que aparece. El problema de combinatoria que consiste en obtener un conjunto mínimo de colores de modo que todos los ejecutables maliciosos se coloreen al menos una vez, se resuelve mediante el comparador de cadenas 300. Puesto que este problema es de tipo NP completo (un problema se denomina NP (polinomio no determinista) si su solución puede averiguarse y verificarse con polinomios; si un problema es NP y el resto de problemas NP pueden reducirse al mismo con polinomios, el problema es NP completo), según una realización de la presente invención, se usa un algoritmo voraz sencillo, el cual proporciona una buena aproximación de las firmas a usar. Por ejemplo, en una familia de 80 códigos maliciosos examinados, bastó con 15 firmas para cubrir toda la familia. El conjunto obtenido de firmas candidatas es almacenado después por el comparador de cadenas 300 en la DB 20.

El resultado de esta fase es un conjunto de firmas compuestas por una secuencia de subcadenas de códigos de operación, como las descritas anteriormente. Según una realización de la presente invención, el convertidor de cadenas 400, como se describirá posteriormente, es responsable de convertir dichas secuencias de códigos de operación en expresiones regulares que pueden emparejarse usando secuencias binarias.

Conversión de firmas en expresiones regulares

Según una realización de la presente invención, el convertidor de cadenas 400 extrae subcadenas de códigos de operación (firmas) de la DB 20 y convierte dichas subcadenas de códigos de operación en expresiones regulares. Esto se realiza con el fin de obtener una firma compacta, densa y fácil de usar. Puesto que cada código de operación tiene un conjunto finito de representaciones binarias, la generación de una expresión regular se facilita mediante una tabla de consulta sencilla. Sin embargo, las expresiones regulares resultantes son bastante largas ya que el conjunto de posibles representaciones binarias puede ser muy grande. Por ejemplo, el código de operación “sumar” tiene 9 representaciones binarias diferentes. Además, la longitud de las expresiones regulares es bastante larga, ya que el elemento más pequeño emparejable de la mayoría de implementaciones de expresiones regulares es un octeto; algunos códigos de operación son más cortos que un octeto y codifican información de registros en este mismo octeto.

En una realización de la presente invención, la longitud de las expresiones regulares resultantes se reduce por el convertidor de cadenas 400 usando el aumento siguiente en el operador de expresión regular. En concreto, el convertidor de cadenas 400 añade “máscaras binarias” (una criba de números que indica qué números situados por debajo hay que mirar. En una máscara binaria, un “1” sobre un número significa “mirar el número situado debajo”; un “0” significa “no mirar”) como elementos emparejables. Por ejemplo se emparejan los siguientes valores binarios, “0xa1, 0xa2, 0va3,…, 0xaf”, usando una única expresión, “&0xaf”, que implica que un octeto X puede emparejarse si y solo si la representación en forma de bits (es decir, los datos a nivel de bits) de X y 0xaf difiere de 0. Por tanto, el convertidor de cadenas 400 reduce considerablemente la longitud de las expresiones regulares generadas. Después de crear dichas expresiones regulares a partir de cada una de las firmas, el convertidor de cadenas 400 almacena dichas expresiones regulares en la DB 20.

Resultados a modo de ejemplo

Se estudió una familia de 99 gusanos. Bastó un total de 15 firmas para cubrir 79 gusanos de la familia. Los otros 20 gusanos no tenían una subcadena común con otros gusanos y necesitaron una firma propia.

Evidentemente, los ejemplos y la descripción anteriores se han proporcionado solamente con fines ilustrativos y no pretenden limitar la invención de manera alguna. Como apreciarán los expertos en la técnica, la invención puede llevarse a cabo de muchas maneras diferentes, utilizando más de una técnica de las descritas anteriormente, estando todo ello dentro del alcance de la invención.

REIVINDICACIONES

1.- Procedimiento para implementar un generador de firmas de malware y detectar códigos ejecutables, que comprende:

a) desensamblar ejecutables benignos mediante un desensamblador (200) en una secuencia de códigos de operación de instrucciones en lenguaje máquina descartando al mismo tiempo otros parámetros;
b) almacenar (21, 31) secuencias de códigos de operación de cada uno de dichos ejecutables benignos en una primera base de datos (10);
c) desensamblar (22, 32) cada ejecutable malicioso de una familia de malware dada mediante dicho desensamblador en una secuencia de códigos de operación de instrucciones en lenguaje máquina descartando al mismo tiempo otros parámetros;
d) para cada par de ejecutables maliciosos, encontrar (23, 33) todas las subcadenas de códigos de operación comunes mediante un comparador de cadenas (300);
e) componer un conjunto de subcadenas de códigos de operación comunes entre dos códigos ejecutables maliciosos cualesquiera mediante dicho comparador de cadenas;
f) comprobar mediante dicho comparador de cadenas cuál de dichas subcadenas de códigos de operación comunes aparecen en al menos una secuencia de códigos de operación de un ejecutable benigno almacenado en dicha primera base de datos;
g) ignorar dichas subcadenas de códigos de operación comunes que aparecen en al menos una secuencia de códigos de operación de un ejecutable benigno;
h) para cada par de ejecutables maliciosos, identificar (23, 33), mediante dicho comparador de cadenas, un conjunto de subcadenas de códigos de operación comunes que no aparezcan en ninguna secuencia de códigos de operación de un ejecutable benigno; e
i) almacenar mediante dicho comparador de cadenas en una segunda base de datos (20) todos los conjuntos de subcadenas de códigos de operación comunes que no aparecen en ninguna secuencia de códigos de operación de un ejecutable benigno, como firmas candidatas.

2.- Procedimiento según la reivindicación 1, en el que el comparador de cadenas obtiene todas las subcadenas de códigos de operación comunes utilizando un proceso de subcadena común más grande.

3.- Procedimiento según la reivindicación 1, que comprende además:

a) encontrar, para cada firma candidata, mediante el comparador de cadenas, malware en el que aparece dicha firma candidata;
b) encontrar (34), mediante el comparador de cadenas y un algoritmo voraz, el conjunto mínimo de firmas candidatas que identifican a todos los ejecutables maliciosos; y
c) almacenar dicho conjunto mínimo en la segunda base de datos (20).

4.- Procedimiento según la reivindicación 3, que comprende además convertir las secuencias de códigos de operación en expresiones regulares que pueden emparejarse usando secuencias binarias.

5.- Procedimiento según la reivindicación 4, en el que las firmas se convierten en expresiones regulares:

a) extrayendo mediante un convertidor de cadenas (400) subcadenas de códigos de operación de la segunda base de datos (20) y convirtiendo dichas subcadenas de códigos de operación extraídas en expresiones regulares usando una tabla de consulta;
b) reduciendo la longitud de las expresiones regulares resultantes convertidas por dicho convertidor de cadenas añadiendo máscaras binarias como elementos emparejables; y
c) almacenando dichas expresiones regulares en la segunda base de datos.

6.- Sistema para reducir el número de firmas usadas para detectar malware, que comprende:

a) un ordenador (100);
b) una primera base de datos (10) y una segunda base de datos (20) conectadas a dicho ordenador;
c) un desensamblador (200) incluido en dicho ordenador; y
d) un comparador de cadenas (300) incluido en dicho ordenador,

caracterizado porque

- dicho desensamblador puede hacerse funcionar para desensamblar ejecutables benignos mediante una secuencia de códigos de operación de instrucciones en lenguaje máquina descartando al mismo tiempo otros parámetros, almacenar secuencias de código de operación de cada uno de dichos ejecutables benignos en dicha primera base de datos, y desensamblar cada ejecutable malicioso de una familia de malware dada en una secuencia de códigos de operación de instrucciones en lenguaje máquina descartando al mismo tiempo otros parámetros,

- dicho comparador de cadenas puede hacerse funcionar para encontrar todas las subcadenas de códigos de operación comunes para cada par de ejecutables maliciosos, componer un conjunto de subcadenas de códigos de operación comunes entre dos códigos ejecutables maliciosos cualesquiera, comprobar cuál de dichas subcadenas de códigos de operación comunes aparecen en al menos una secuencia de códigos de operación de un ejecutable benigno almacenado en dicha primera base de datos, ignorar dichas subcadenas de códigos de operación comunes que aparecen en al menos una secuencia de códigos de operación de un ejecutable benigno, identificar para cada par de ejecutables maliciosos un conjunto de subcadenas de códigos de operación comunes que no aparezcan en ninguna secuencia de códigos de operación de un ejecutable benigno, y almacenar en dicha segunda base de datos todos los conjuntos de subcadenas de códigos de operación comunes que no aparecen en ninguna secuencia de códigos de operación de un ejecutable benigno, como firmas candidatas.

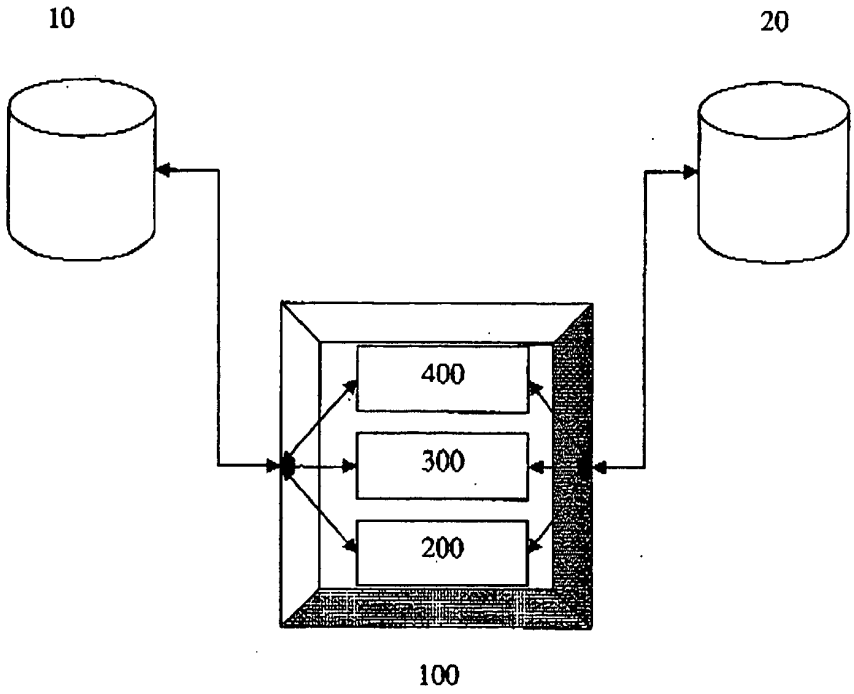

Fig. 1

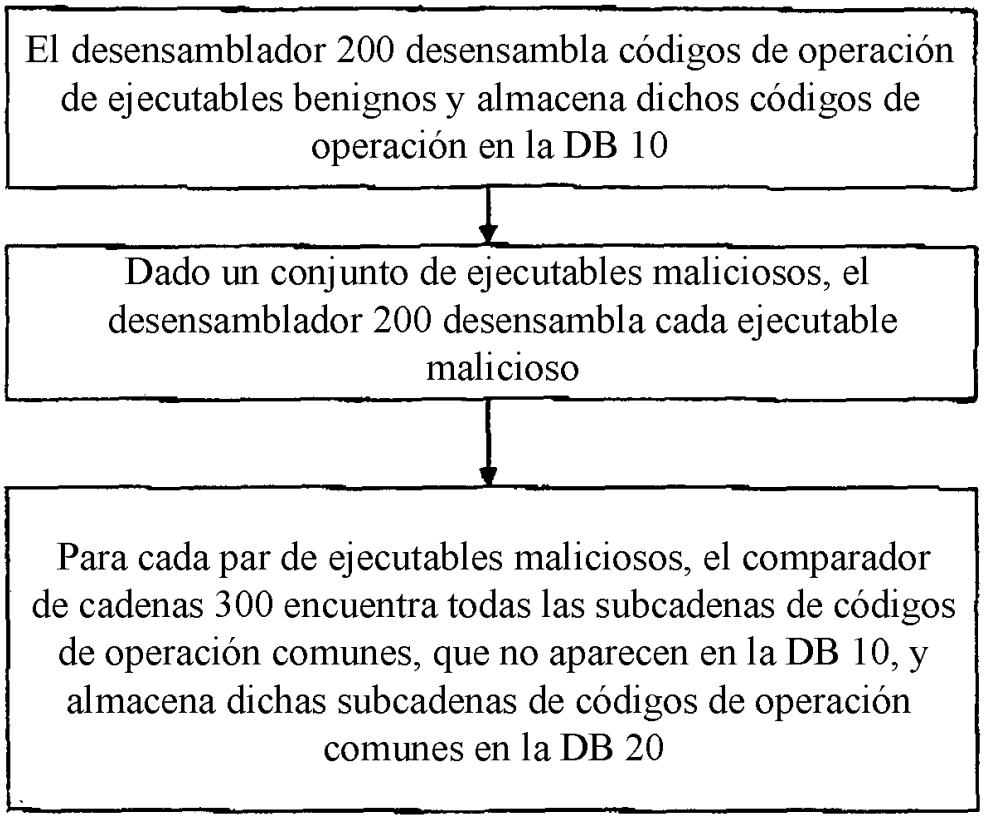
El desensamblador 200 desensambla códigos de operación de ejecutables benignos y almacena dichos códigos de operación en la DB 10
21
Dado un conjunto de ejecutables maliciosos, el desensamblador 200 desensambla cada ejecutable malicioso
22
Para cada par de ejecutables maliciosos, el comparador de cadenas 300 encuentra todas las subcadenas de códigos de operación comunes, que no aparecen en la DB 10, y almacena dichas subcadenas de códigos de operación comunes en la DB 20
23

Fig. 2

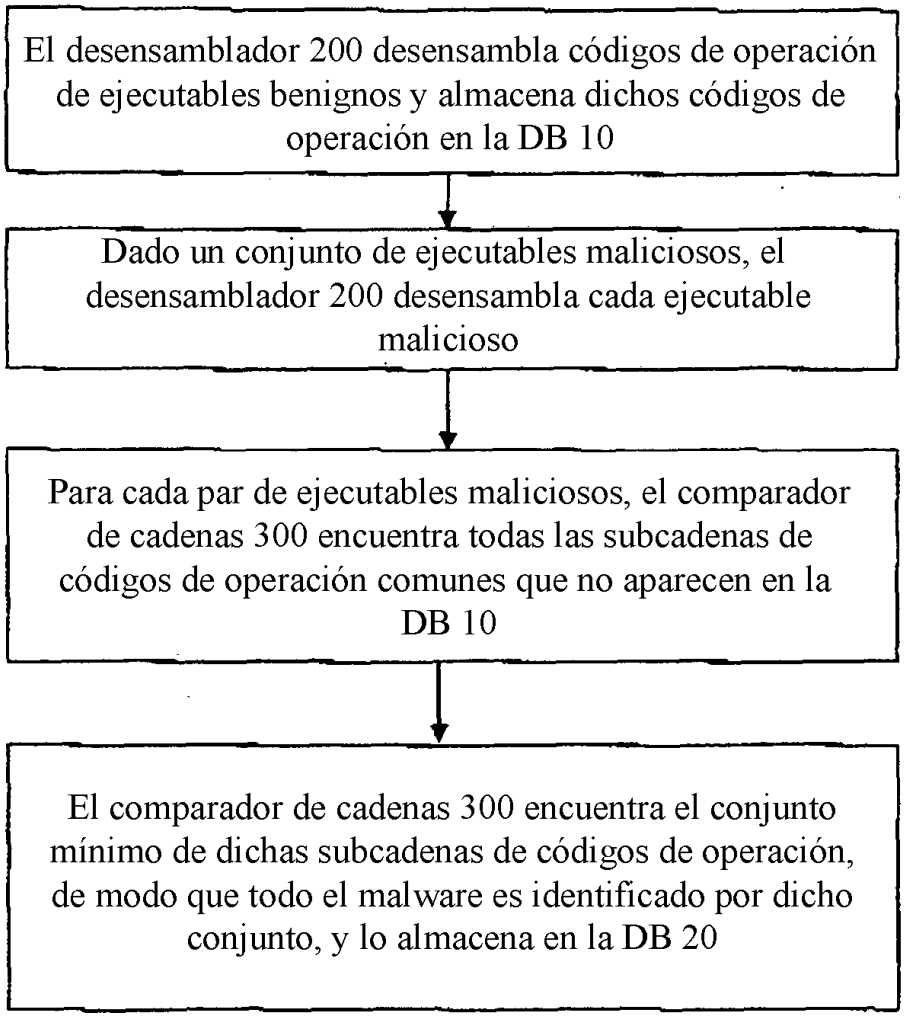
El desensamblador 200 desensambla códigos de operación de ejecutables benignos y almacena dichos códigos de operación en la DB 10
31
Dado un conjunto de ejecutables maliciosos, el desensamblador 200 desensambla cada ejecutable malicioso
32
Para cada par de ejecutables maliciosos, el comparador de cadenas 300 encuentra todas las subcadenas de códigos de operación comunes que no aparecen en la DB 10
33
El comparador de cadenas 300 encuentra el conjunto mínimo de dichas subcadenas de códigos de operación, de modo que todo el malware es identificado por dicho conjunto, y lo almacena en la DB 20
34

Fig. 3