(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 698 563**

(51) Int. CI.:

H04L 29/06 (2006.01)
H04L 29/08 (2006.01)
G06F 21/55 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **30.10.2015 PCT/EP2015/075223**

(87) Fecha y número de publicación internacional: **19.05.2016 WO16074947**

(96) Fecha de presentación y número de la solicitud europea: **30.10.2015 E 15787597 (2)**

(97) Fecha y número de publicación de la concesión europea: **10.10.2018 EP 3219068**

(54) Título: **Método de identificar y contrarrestar ataques de Internet**

(30) Prioridad:

13.11.2014 US 201462079337 P
30.04.2015 US 201514701115

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
05.02.2019

(73) Titular/es:

CLEAFY S.R.L. (100.0%)
Via della Malvasia 30
38122 Trento, IT

(72) Inventor/es:

PASTORE, NICOLÒ;
PARRINELLO, EMANUELE y
GIANGREGORIO, CARMINE

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Método de identificar y contrarrestar ataques de Internet

Campo de la invención

La presente descripción está relacionada con un método de detectar y contrarrestar ataques de Internet.

En concreto, la presente descripción está relacionada con un método de detectar y contrarrestar ataques Man-in-the-Browser (Hombre en el Navegador) y/o Man-in-the-Middle (Hombre en el Medio) y/o Bot. Dicho de otra manera, la presente invención permite monitorización y protección de una aplicación Web o de un navegador Web contra ataques dirigidos al navegador Web de un cliente.

Exposición de la técnica relacionada

Se sabe que en la técnica se utiliza software antivirus para contrarrestar ataques de seguridad informática, incluidos ataques Man-in-the-Browser y/o Man-in-the-Middle y/o Bot.

Por ejemplo, Man-in-the-Browser es un tipo de ataque que consiste en manipulación directa del navegador Web para cambiar los contenidos que se muestran normalmente al usuario cuando él/ella visita un sitio Web (véase la figura 1). Los ataques Man-in-the-Browser se llevan a cabo utilizando software malicioso (*malware*) instalado en el ordenador sin el conocimiento del usuario. Dicho software malicioso (p. ej. Troyanos Proxy) interactúa con la memoria de procesos del navegador Web, para redirigir el flujo normal de llamadas del sistema (utilizadas por el navegador Web) a ciertas funciones del software malicioso, las cuales tienen el objetivo, por ejemplo, de inyectar código HTML adicional en la página Web descargada. Se debería observar que, en el caso del ataque Man-in-the-Browser, se establece una conexión con el servidor Web original del sitio web que ha sido atacado, lo cual hace difícil la detección del ataque. Por lo tanto, el navegador Web y la aplicación Web son incapaces de identificar los contenidos que han sido añadidos por el software malicioso a los contenidos que han sido descargados realmente por el navegador Web. Se han reconocido diferentes ataques Man-in-the-Browser, incluyendo robo de datos de tarjetas de crédito a sitios web de banca electrónica y de comercio electrónico y transacciones fraudulentas que a menudo se inician automáticamente sin ninguna interacción con el usuario.

Más en detalle, cuando un usuario solicita una página Web (es decir, una aplicación Web) a través de un navegador Web, el servidor Web que aloja a la página Web envía un código fuente HTML (un Modelo de Objetos del Documento, DOM, del inglés Document Object Model) al navegador Web. El código DOM se transfiere al motor de renderizado del navegador Web para ser mostrado al usuario. Por ejemplo, en un PC infectado por software malicioso, el código DOM recibido por el navegador Web procedente del servidor Web es cambiado por el software malicioso antes de que sea procesado por el motor de renderizado del navegador Web. Para ello, el software malicioso inyecta un código adicional (p. ej. un script) en el código DOM que ha recibido procedente del servidor Web para modificar los contenidos mostrados al usuario. Los cambios hechos por el software malicioso al código DOM descargado del servidor Web son cambios en los códigos HTML y/o javascript y/o cualquier otro contenido o recurso Web. Dicho de otra manera, el navegador Web está conectado al servidor Web original mientras el software malicioso hace cambios en el código DOM descargado. Estos cambios pueden incluir alteraciones gráficas y/o de comportamiento. Por lo tanto, se muestra al usuario una página Web, la cual ha sido cambiada en su comportamiento y/o representación gráfica, a partir de la página Web que fue solicitada originalmente por el cliente. El cliente permite acceso contra su voluntad a sus propios datos personales o autoriza transacciones fraudulentas en su cuenta.

Por ejemplo, en el campo de los servicios bancarios, un ordenador infectado con software malicioso típicamente inicia sesión en el sitio web de banca en línea utilizando un protocolo HTTPS, y descarga los datos de la página Web. Sin embargo, el software malicioso altera estos datos en tiempo real, añadiendo scripts de manipulación de transacciones, y realizando, por ejemplo, transferencias automáticas de datos. El script también puede redirigir transferencias de dinero que fueron ordenadas realmente por el usuario a otros destinatarios, o más simplemente solicitar datos de la tarjeta de crédito y/o añadir campos adicionales para que sean rellenados por el usuario con datos adicionales.

Un ejemplo adicional son los ataques Bot, como se muestra en la Figura 1. Estos ataques consisten en solicitudes de página que proceden de un sistema automático en vez de una persona. Esto puede implicar para el proveedor de servicios un enorme consumo de ancho de banda. Además, sistemas automáticos pueden utilizar el servicio de maneras indeseadas e ilegales. Ejemplos conocidos en la técnica son el Web scraping (es decir, extracción de información del servicio Web), el Carding (es decir, el proceso de validación de tarjetas de crédito robadas) o el ataque por Fuerza Bruta (es decir, el intento de buscar la credencial de un usuario en la página de inicio de sesión de una aplicación Web).

El documento US2002166051 describe una función de cifrado realizada sobre el código DOM de una aplicación Web. El código DOM está disponible en el servidor Web en combinación con un programa de descifrado. Cuando un usuario solicita la aplicación Web, se proporciona un código DOM cifrado en respuesta a dicha solicitud. Este código DOM cifrado no puede ser renderizado por el cliente que lo solicitó. Esto se debe a que sólo un cliente autorizado

puede acceder al programa de descifrado disponible en el navegador Web, el cual le permite descifrar el código DOM para acceder a la aplicación Web.

En el documento US 2009/327411 A1 se describe un ejemplo de un método adaptado para detectar ataques de inyección SQL (Lenguaje de Consulta Estructurado, del inglés Structured Query Language) y ataques XSS (Ataques por Secuencias de comandos entre páginas web, en inglés Cross-Site Scripting) realizados por un hacker (es decir, ataques humanos) en el lado del cliente. Este documento describe un método que implica inyectar un script en la respuesta de la aplicación web y enviarlo al navegador web del cliente para capturar acciones de usuario sobre una función objetivo específica que se quiere monitorizar. A continuación, estas acciones de usuario capturadas se realizan sobre un documento de respuesta de la página web “original” del que se ha hecho copia de seguridad, y finalmente el resultado se compara con la solicitud presentada por el cliente usuario. Si las dos solicitudes no coinciden, entonces se detecta un ataque de un hacker.

En el documento US 2002/166051 A1 se proporciona un ejemplo de un método para proteger un script de inspección no autorizada e ingeniería inversa mediante uso de un método de cifrado/descifrado.

En el documento US 2011/247075 A1 se proporciona un ejemplo de un sistema para proteger tráfico de datos digitales entre un servidor web y un cliente usuario comprobando el entorno de la máquina del cliente usuario.

Problema de la técnica anterior

Los softwares antivirus, instalados en PCs o en dispositivos de usuario del cliente (p. ej. teléfono inteligente, tabletas, etc.) son poco efectivos contra este tipo de amenaza de seguridad informática. Los softwares antivirus sólo pueden identificar parte de los ataques Man-in-the-Browser que ocurren en Internet. También se conocen navegadores Web que cumplen altos estándares de seguridad o que tienen software de seguridad de Internet. Sin embargo, ninguna de las soluciones de la técnica anterior puede contrarrestar de forma efectiva ataques Man-in-the-Browser y/o Man-in-the-Middle y/o Bot.

Por ejemplo, incluso cuando los códigos DOM estén cifrados, todavía se pueden obtener códigos de descifrado por medio de ataques dirigidos al servidor Web que contiene el programa de descifrado. Además, aunque el código esté cifrado, todavía se proporciona de manera inmediata al cliente que lo solicita. Por lo tanto, todavía son posibles ataques, porque existe un gran riesgo de que las claves de descifrado puedan ser identificadas por individuos que realizan dichos ataques.

Resumen de la invención

Es un objeto de la presente invención proporcionar un método de impedir ataques de Internet.

Un objeto adicional de la presente invención es proporcionar un método de identificar y contrarrestar ataques Man-in-the-Browser y/o Man-in-the-Middle y/o Bot.

Otro objeto de esta invención es proporcionar un método de detectar los cambios hechos por un software malicioso a los códigos HTML y/o javascript del DOM de la página Web y/o del recurso Web que ha sido descargado por un usuario y certificar que los contenidos y/o el DOM del recurso y/o de la página Web transmitidos a un cliente dado son realmente los que se le muestran a él o los que son utilizados por él.

Ventajas de la invención

Una realización proporciona un método que permite monitorización y manipulación del flujo de solicitudes HTTP y/o HTTPS hechas por un usuario y transferidas entre el navegador Web y la aplicación Web que está siendo monitorizada.

Una realización adicional proporciona un método que protege el código DOM descargado e impide que el software malicioso acceda a él.

Otra realización adicional proporciona un método de identificar cualquier alteración al código DOM que ha sido realmente descargado del servidor Web. Esto permitirá identificación de un ataque Man-in-the-Browser y/o Man-in-the-Middle y/o Bot, para garantizar que la página Web solicitada se mostrará correctamente al usuario.

Breve descripción de los dibujos

Las características y ventajas de la presente descripción se pondrán de manifiesto a partir de la siguiente descripción detallada de una posible realización práctica, ilustrada como un ejemplo no limitativo en el conjunto de dibujos, en los cuales:

- La Figura 1 muestra un diagrama de flujo de un ejemplo de un ataque de Internet, no estando incluido este ejemplo en el método de la presente invención;

- La Figura 2 muestra un diagrama de flujo del método de identificar y contrarrestar ataques de Internet de la Figura 1;

- La Figura 3 muestra un diagrama de flujo adicional del método de identificar y contrarrestar ataques de Internet, de acuerdo con la presente descripción;

- La Figura 4 muestra un diagrama de flujo de un paso concreto del método de identificar y contrarrestar ataques de Internet, de acuerdo con la presente descripción;

Descripción detallada

Incluso cuando esto no se indique de forma expresa, los rasgos individuales descritos con referencia a las realizaciones concretas estarán concebidos como auxiliares a y/o intercambiables con otros rasgos descritos con referencia a otras realizaciones ejemplares.

La presente invención está relacionada con un método de identificar y contrarrestar ataques de Internet, en concreto ataques Man-in-the-Browser y/o Man-in-the-Middle y/o Bot.

Por ejemplo, la Figura 1 muestra un diagrama de flujo de un ejemplo, que no forma parte de la presente invención, de un ataque de Internet del tipo de ataque Man-in-the-Browser y/o Bot.

En las figuras, el número 1 designa un sistema 1 en el cual se puede implementar el método de la presente invención. Dicho de otra manera, el sistema 1 se designa como entorno de red que permite implementación del método de la presente invención.

Haciendo referencia a la Figura 2, el método de identificar y contrarrestar ataques de Internet de los tipos de ataque Man-in-the-Browser y/o Man-in-the-Middle y/o Bot comprende un paso de generar una solicitud (p. ej. una solicitud GET, una solicitud POST), mediante un navegador Web 4, relativa a una aplicación Web 6 (p. ej. una página Web) que reside en un servidor Web 5. Preferiblemente, la solicitud se genera cuando un usuario del navegador Web 4 introduce una URI o URL para la aplicación Web 6. Más preferiblemente, el método incluye el paso de enviar la solicitud utilizando un protocolo HTTP o HTTPS.

El método también incluye el paso de enviar la solicitud mediante el navegador Web 4 a un servidor de caja 2 que está en comunicación de señal con el servidor Web 5.

Por ejemplo, el método utiliza un servidor de caja 2 en comunicación de señal con al menos un ordenador cliente 3 en cuyo interior reside un navegador Web para navegación por Internet. El servidor de caja 2 está en comunicación de señal con un servidor Web 5 en cuyo interior reside una aplicación Web 6. En un aspecto, un usuario puede utilizar el navegador Web 4 en el ordenador cliente 3 para solicitar una aplicación Web 6 (p. ej. una página Web) que reside en un servidor Web 5. Dicho de otra manera, un usuario (o cliente) utiliza el navegador Web 4 instalado en el ordenador cliente 3 para acceder a una página Web. Obviamente, durante el uso, el ordenador cliente 3 deberá estar conectado a una red de Internet a través de equipos telefónicos con cable o inalámbricos o cualquier otro método de comunicación conocido. El servidor de caja 2 está configurado para recibir al menos una solicitud asociada con la aplicación Web 6 procedente del navegador Web 4 y para enviar dicha solicitud al servidor Web 5.

El método también incluye el paso de recibir un código DOM del servidor mediante el servidor de caja 2, habiendo sido generado automáticamente dicho código por el servidor Web 5 de acuerdo con la solicitud.

Por ejemplo, el servidor de caja 2 está configurado para recibir un código DOM del servidor relacionado con la solicitud procedente del servidor Web 5. A saber, la solicitud es generada por el navegador Web 4 cuando el usuario utiliza el navegador Web 4 para solicitar una URL (Localizador Uniforme de Recursos, del inglés Uniform Resource Locator) que identifica unívocamente la dirección de un recurso de Internet (es decir, la aplicación Web 6) que reside en el servidor Web 5. Por ejemplo, el servidor de caja 2 está configurado para recibir al menos la solicitud asociada con la aplicación Web 6 procedente del navegador Web 4 utilizando el protocolo HTTP o HTTPS y para enviar dicha solicitud al servidor Web 5.

A modo de ejemplo, cuando el usuario solicita una página del sitio Web de la aplicación Web 6, con una configuración de red concreta (p. ej. haciendo cambios a las claves DNS para el dominio del sitio Web con el que se quiere contactar) o balanceadores de carga (p. ej. introduciendo reglas para cambiar el flujo de tráfico), establece contacto con el servidor de caja 2 en vez de establecer contacto con el servidor Web 5. Este último reenvía la solicitud al servidor Web 5 de la aplicación Web 6 (p. ej. en modo estándar, de proxy inverso).

El método comprende el paso de enviar un código de página de servicio mediante el servidor de caja 2 al servidor Web 4 en respuesta a la solicitud. Preferiblemente, el código de página de servicio comprende un código javascript ofuscado o polimórfico y/o códigos HTML. Más preferiblemente, el código de página de servicio puede ser diferente para cada usuario o para cada solicitud HTTP/HTTPS.

De acuerdo con una realización preferida, el servidor de caja 2 está diseñado para ser instalado como un componente software en la aplicación Web 6 y/o como un módulo software firewall y/o balanceador de carga y/o aparato de red y/o un dispositivo hardware y/o un módulo software en el servidor Web 5 que aloja a la aplicación Web 6.

Preferiblemente, el servidor de caja 2 está instalado en la misma red que la aplicación Web 6 (p. ej. instalada en los servidores del usuario, *on-premises*) o se proporciona como un servicio externo (p. ej. de tipo SaaS, Software como Servicio, o en la nube, *Cloud*).

El método comprende además el paso de recibir o ejecutar los códigos javascript y/o HTML, mediante el servidor Web 4, para generar automáticamente una solicitud asíncrona, de tal manera que los datos de entorno del navegador Web 4 se puedan transmitir al servidor de caja 2.

El método también comprende el paso de procesar los datos de entorno del navegador Web, mediante el box 2, para identificar ataques de Internet de los tipos de ataque Man-in-the-Browser y/o Man-in-the-Middle y/o Bot.

El método comprende el paso de realizar una función de cifrado sobre el código DOM del servidor mediante el servidor de caja 2 para generar un código DOM ofuscado, así como el paso de enviar el código DOM ofuscado al navegador Web en respuesta a la solicitud asíncrona.

De acuerdo con una realización preferida, el método comprende el paso de realizar funciones de cifrado y/o de ofuscación y/o de compresión y/o de codificación sobre el código DOM del servidor mediante el servidor de caja 2, para obtener el código DOM ofuscado. Preferiblemente, la función de cifrado implica el uso de claves simétricas o asimétricas. Por ejemplo, los métodos de ofuscación que se utilizan para el código de página de servicio pueden incluir sustitución de nombres de variables y funciones, introducción de código no utilizado, cifrado, codificación de números y cadenas (*strings*).

El método también comprende el paso de realizar una función de descifrado sobre el código DOM ofuscado mediante el código de página de servicio, para obtener el código DOM del servidor. Dicho de otra manera, el código de página de servicio que está siendo renderizado por el navegador Web comprende una porción de código que está configurada para descifrar el código DOM ofuscado por medio de claves de descifrado compartidas con el servidor de caja 2.

El método comprende el paso de renderizar el código DOM del servidor mediante el navegador Web 4.

Haciendo referencia a lo anterior, el código DOM del servidor (p. ej. el código HTML) es recibido y procesado por el motor de renderizado del navegador Web 4 de tal manera que los contenidos de la aplicación Web 6 se pueden mostrar al usuario como hipertexto (p. ej. una página Web).

Ventajosamente, con el método de la presente invención, no se puede hacer ningún Ataque Bot, ya que la página recibida procedente del sistema automático que envía la solicitud es el código de página de servicio, el cual está relacionado con una página que no tiene ninguno de los contenidos del código DOM del servidor original obtenido del servidor Web 5.

Ventajosamente, el código de página de servicio se ofusca en modo avanzado para obstaculizar la extracción de información por el software malicioso que se podría utilizar para cambiar el código DOM del servidor ofuscado.

Ventajosamente, con el método de la presente invención no se puede hacer ningún ataque Man-in-the-Browser, ya que el software malicioso sólo puede interceptar el código DOM ofuscado, el cual se modifica y se cifra y no se puede alterar o sustituir.

De acuerdo con una realización preferida, el método comprende los pasos de generar la solicitud asociada con la aplicación Web 6 mediante un sistema automático, y el paso de enviar la solicitud y de un código de autorización único al servidor de caja 2 mediante el mismo sistema automático. El método comprende el paso de recibir el código DOM del servidor que ha sigo generado automáticamente por el servidor Web 5 de acuerdo con la solicitud, mediante el servidor de caja 2. El método comprende además el paso de enviar el código DOM del servidor al sistema automático mediante el servidor de caja 2, de acuerdo con el código de autorización único. De forma alternativa, el código de autorización único se puede sustituir por una lista blanca (*whitelist*), es decir, una lista de sistemas automáticos que están autorizados para solicitar el código DOM del servidor (véase la Figura 3). Preferiblemente, un sistema automático comprende sistemas autorizados que pueden recibir los contenidos del servicio proporcionado por el servidor Web 5. Por ejemplo, los motores de búsqueda necesitan contenidos de las páginas para indexación de las páginas. Ventajosamente, el método de la presente invención no bloquea a estos sistemas automáticos, pero les proporciona el código DOM del servidor si están en una lista blanca autorizada y/o si tienen un código de autorización único (o token de seguridad).

De acuerdo con una realización preferida, el método comprende el paso de proporcionar una clave criptográfica de un solo uso, mediante un dispositivo externo 7a, 7b, al servidor de caja 2 y al navegador Web 4. Asimismo, el método comprende el paso de realizar la función de cifrado sobre el código DOM del servidor utilizando la clave

criptográfica de un solo uso, para generar un código DOM ofuscado. Además, el método comprende el paso de realizar descifrado del código DOM ofuscado de acuerdo con la clave criptográfica de un solo uso, para obtener el código DOM del servidor. Dicho de otra manera, la clave criptográfica de un solo uso necesaria para cifrado y descifrado es proporcionada por el dispositivo externo 7a, 7b en cada solicitud. Esto incrementará aún más la seguridad del sistema, ya que las claves de cifrado proceden de un sistema externo que es inmune a alteración por software malicioso. Por ejemplo, sistemas OTK (clave de un solo uso, del inglés one-time-key) o sistemas OTP (contraseña de un solo uso, del inglés one-time-password) se utilizan como claves criptográficas de un solo uso.

De acuerdo con un sistema preferido, el método comprende el paso de generar un código de usuario asociado con un usuario del navegador Web 4. Además, el método comprende el paso de proporcionar la clave criptográfica de un solo uso de acuerdo con el código del usuario.

De acuerdo con una realización preferida de la presente invención, el método comprende el paso de recibir y renderizar el código de página de servicio mediante el navegador Web 4. Dicho de otra manera, el código de página de servicio renderizado se genera como resultado del procesamiento del código de página de servicio por el motor de renderizado del navegador Web 4. El método también comprende el paso de recibir o procesar los códigos javascript y/o HTML, mediante el navegador Web 4, para generar de manera automática la solicitud asíncrona, de tal manera que el código de página de servicio renderizado se pueda transmitir al servidor de caja 2. Además, el método comprende el paso de procesar y comparar el código de página de servicio renderizado y el código de página de servicio mediante una aplicación de algoritmo 8 que reside en el servidor de caja 2, de tal manera que se puede identificar al menos una diferencia en el código.

Por lo tanto, el código de página de servicio es el código que puede ser alterado potencialmente por software malicioso para ataques Man-in-the-Browser y/o Man-in-the-Middle. Como se ha mencionado anteriormente, el software malicioso altera las funciones internas del navegador (lo cual también se conoce como *hooking*) cambiando el código de página de servicio antes de que sea transferido al motor de renderizado del navegador Web 4. Por lo tanto, si el código de página de servicio es alterado por software malicioso, el código de página de servicio renderizado también es modificado.

De acuerdo con una realización preferida, el método comprende el paso de generar una señal indicativa de ataque, mediante el servidor de caja 2, cuando la aplicación de algoritmo 8 identifica al menos una diferencia de código entre el código de página de servicio y el código de página de servicio renderizado. El método incluye el paso de enviar la señal indicativa de ataque al navegador Web 4 y/o el paso de guardar la señal indicativa de ataque en una base de datos.

De acuerdo con una disposición preferida, el método comprende el paso de procesar el código de página de servicio renderizado para compararlo con el código de página de servicio, mediante una función de comparación de la aplicación de algoritmo 8, para generar de ese modo al menos una señal indicativa de ataque cuando el código de página de servicio es incompatible con el código de página de servicio renderizado.

Se observará que el código de página de servicio es un código pequeño, el cual sólo tiene objetivos funcionales y de seguridad. Este código no se utiliza para mostrar una página Web asociada con la aplicación Web 6, sino que sólo actúa para proteger la aplicación Web 6.

Ventajosamente, el método permite comparación entre el código de página de servicio y el código de página de servicio renderizado, para identificar cualquier incompatibilidad entre los dos códigos, asociada con la presencia de software malicioso en el ordenador cliente. Gracias al método de la presente invención, se pueden comparar dos códigos pequeños, lo cual reduce considerablemente las necesidades de potencia y tiempo de cálculo.

De acuerdo con una realización preferente, el método comprende el paso de procesar la solicitud mediante el servidor de caja 2, para identificar y contrarrestar ataques Bot.

De acuerdo con una realización preferente, el servidor de caja 2 comprende un analizador de tráfico asociado con él, en el cual reside la aplicación de algoritmo 8.

De acuerdo con una realización preferente, el servidor de caja 2 y/o el analizador de tráfico con componentes software. Más preferiblemente, el servidor de caja y/o el analizador de tráfico son componentes de un servidor dedicado.

Preferiblemente, el servidor de caja 2 y/o el analizador de tráfico se comunican enviando tareas (jobs). Estas tareas se transmiten por medio de uno o más protocolos de comunicación conocidos, tales como TCP, UDP, HTTP(S) e IMAP.

Ventajosamente, en el método el analizador de tráfico es externo al flujo de datos de solicitudes HTTP/HTTPS, y puede actuar con independencia de dicho flujo de datos.

Preferiblemente el código DOM del servidor es un código HTML y/o un código javascript asociado con la solicitud.

Preferiblemente, el código de página de servicio es un código HTML y/o un código javascript.

De acuerdo con una realización preferida, el código de página de servicio es un código preestablecido, el cual preferiblemente está configurado para proporcionar al menos una instrucción al navegador Web para enviar el código de página de servicio renderizado al servidor de caja 2.

Preferiblemente, la aplicación de algoritmo 8 está configurada para procesar el código de página de servicio renderizado y compararlo con el código de página de servicio para identificar al menos una diferencia de código. Preferiblemente, la aplicación de algoritmo 8 está configurada para generar una señal indicativa de ataque (p. ej. Alerta de MitB, alerta de BOT), cuando identifica al menos una diferencia de código que puede estar relacionada con un ataque de Internet, tal como un ataque Man-in-the-Browser (MitB).

De acuerdo con un sistema preferido, la aplicación de algoritmo 8 procesa el código de página de servicio renderizado para proporcionar una estimación del código de página de servicio esperado. Más en detalle, la aplicación de algoritmo 8 está configurada para proporcionar una estimación del código de página de servicio esperado que ha sido procesado por el motor de renderizado del navegador Web 4 para generar el código de página de servicio renderizado. El código de página de servicio esperado se compara con el código de página de servicio original (es decir, el que fue recibido inicialmente por el ordenador cliente 3) para identificar la compatibilidad entre los dos códigos. Dicho de otra manera, los dos códigos de página de servicio esperado/original son idénticos y coincidentes, si no se ha hecho ningún cambio al código antes del renderizado, o similares y compatibles si la presencia de software malicioso no provoca diferencias de código.

De acuerdo con la presente invención, los dos códigos de página de servicio esperado/original estarán concebidos para que sean incompatibles cuando la aplicación de algoritmo 8 identificó al menos una diferencia de código que puede estar relacionada con un ataque de Internet (tal como un ataque MitB).

Preferiblemente, la aplicación de algoritmo 8 es implementada manualmente por un programador o por medio de un sistema de aprendizaje, y por tanto es variable con el tiempo (polimorfismo). Más preferiblemente, el sistema de aprendizaje mediante el cual se implementa la aplicación de algoritmo 8 está basado, por ejemplo, en análisis estadístico del comportamiento concreto del navegador Web 4 (p. ej. indicando Suplantación de Identidad del Agente de Usuario, del inglés User Agent Spoofing).

Tal como se usa en la presente invención, el término aplicación de algoritmo 8 está concebido para designar un programa o una serie de programas que están siendo ejecutados en el servidor de caja 2 (o en el analizador de tráfico) para permitir comparación del código de página de servicio con el código de página de servicio renderizado para comprobar si existe incompatibilidad entre ellos. En concreto, la aplicación de algoritmo 8 es un programa o una serie de programas que pueden procesar el código de página de servicio para hacerlo comparable con el código de página de servicio renderizado. Por ejemplo, el algoritmo utiliza una función preestablecida y apropiadamente configurada para proporcionar una estimación del código de página de servicio esperado (es decir, el código que ha sido potencialmente cambiado por el software malicioso) que ha sido recibido y procesado por el motor de renderizado del navegador Web 4. Esta función preestablecida cambia con el tiempo (polimorfismo) debido a un mecanismo de aprendizaje que tiene en cuenta el comportamiento del navegador Web concreto para el cual ha sido implementada. Esto significa que la función preestablecida es específica para cada navegador Web 4, ya que cada navegador Web 4 renderiza el código de página de servicio a su manera específica. Dicho de otra manera, la función preestablecida de la aplicación de algoritmo 8 realiza una “función inversa” del código de página de servicio renderizado. De esta manera, una vez que ha recibido el código de servicio renderizado, puede proporcionar una estimación del código de página de servicio esperado que ha sido procesado realmente por el navegador Web 4. Esto es posible porque la función preestablecida tiene en cuenta el comportamiento del navegador Web 4.

De acuerdo con una realización preferida, el método comprende el paso de procesar el código de página de servicio renderizado para compararlo con el código de página de servicio, mediante una función de comparación de la aplicación de algoritmo 8, para generar de ese modo al menos una señal indicativa de ataque cuando el código de página de servicio es incompatible con el código de página de servicio renderizado.

De acuerdo con una realización preferida, el método comprende un paso en el cual el analizador de tráfico obtiene datos y comienza un análisis basado en la aplicación de algoritmo 8.

De acuerdo con una realización alternativa de la presente invención, la aplicación de algoritmo 8 puede utilizar técnicas conocidas para comprobar si la página de código de servicio renderizada que ha sido recibida por el navegador Web 4 coincide o no con la esperada.

Un ejemplo de una técnica conocida es el análisis heurístico del código recibido en el navegador Web, como se describe en “Detecting client-side e-banking fraud using a heuristic model” a nombre de T. Timmermans y J.Kloosterman, Universidad de Amsterdam – 11.07.2013.

Un ejemplo adicional es el método que se describe en el documento US 2011/0239300, en el cual se distribuye un código de comprobación con la página de servicio. Este código de comprobación comprueba la presencia de cierto software malicioso conocido dentro del código en el navegador Web.

De acuerdo con una realización preferida proporcionada por la presente invención, se puede utilizar el método que se describe en la Solicitud de Patente Provisional de EE.UU. Nº 62/079.337 por el Solicitante de la misma, en el cual el código de página de servicio renderizado es enviado al analizador de tráfico. El analizador de tráfico comprueba, utilizando la aplicación de algoritmo 8, si el código de página de servicio original que se ha recibido es compatible o no con el código de página de servicio renderizado que ha sido enviado por el navegador Web 4 para identificación de software malicioso.

Ventajosamente el método proporciona una visión global de lo que sucede durante una sesión de navegación web, porque analiza las solicitudes HTTP/HTTPS individuales.

Ventajosamente, el método de la presente invención permite identificación y contrarrestación efectivas y seguras de ataques Man-in-the-Browser y/o Man-in-the-Middle y/o Bot, permitiendo de este modo monitorización total de las solicitudes de usuario.

De acuerdo con una realización preferida, el paso de recibir y procesar el código de página de servicio para generación automática del código de página de servicio renderizado comprende los pasos de:

- recibir el código de página de servicio mediante el navegador Web 4,

- procesar los códigos javascript y/o HTML ofuscados y polimórficos existentes en el código de página de servicio, mediante el navegador Web 4, para generar de forma automática el código de página de servicio renderizado,

- procesar los códigos javascript y/o HTML ofuscados y polimórficos existentes en el código de página de servicio, mediante el navegador Web 4, para enviar el código de página de servicio renderizado al servidor de caja 2.

A continuación se describen unos pocos ejemplos de aplicación del método de la presente invención.

EJEMPLO 1 – Figura 2

Véase la Figura 2 para el siguiente ejemplo de aplicación del método de la presente invención. Este método comprende los siguientes pasos:

a. El usuario solicita una página del sitio Web de interés (es decir la aplicación Web 6), y de esta forma una solicitud HTTP o HTTPS es generada por el navegador Web 4 del usuario, la cual es enviada a un servidor de caja 2 (instalado en las instalaciones del propietario de la aplicación Web o disponible en un entorno en la Nube);

b. El servidor de caja 2 actúa como un sistema proxy inverso, lee el nombre de máquina (*hostname*), y comprueba el nombre de máquina original de la ubicación de la aplicación Web 6 comparándolo con sus claves de configuración. Reenvía la solicitud HTTP o HTTPS y obtiene el código DOM del servidor (/incluidos los encabezados http/https y las cookies) de la página solicitada;

c. El servidor de caja 2 genera de forma aleatoria un UID (ID de usuario, del inglés user ID). Si el usuario ya ha hecho solicitudes, lee el UID que ha sido enviado por el cliente, por ejemplo a través de una cookie contenida y pre-registrada en el navegador del usuario, o genera y envía uno nuevo;

d. El servidor de caja 2 asigna de forma aleatoria un HID único (código de solicitud) a cada solicitud HTTP o HTTPS individual;

e. El servidor de caja 2 envía al usuario una página de servicio que contiene los códigos javascript y/o HTML ofuscados y polimórficos para incrementar el nivel de seguridad de la página;

f. El servidor de caja 2 aplica funciones de cifrado y/u ofuscación y/o compresión y/o codificación al código DOM del servidor original que ha sido transmitido por el servidor Web 5 de la aplicación Web 6;

g. La página de servicio que está siendo renderizada por el navegador Web 4 del usuario 4 hace una solicitud asíncrona al servidor de caja 2 y transmite información de entorno del navegador Web 4 y/o el código de página de servicio renderizado;

h. El servidor de caja recibe la solicitud asíncrona procedente de la página de servicio y realiza una función algoritmo 4 para comprobar la integridad y seguridad del entorno del navegador Web 4 del usuario;

i. Basándose en reglas de seguridad, el servidor de caja 2 transmite el código DOM del servidor, ofuscado bajo f), en respuesta a la llamada asíncrona de la página de servicio;

l. La página de servicio recibe el código DOM del servidor ofuscado (es decir cambiado bajo f)), realiza una función inversa para obtener el código DOM del servidor original, y sustituye la página de servicio por la página original asociada con el código DOM del servidor.

EJEMPLO 2 – Figura 3

Haciendo referencia a la Figura 3, en caso de solicitudes enviadas por sistemas automáticos autorizados, el método incluye los siguientes pasos:

a. El sistema automático autorizado solicita una página del sitio Web de interés, se hace una solicitud HTTP p HTTPS a un servidor de caja 2 y al mismo tiempo se transmite un código (token) de autorización único, por ejemplo dentro de un encabezamiento HTTP de la solicitud;

b. El servidor de caja 2 comprueba si el token es válido y bloquea la comunicación si no lo es. Si el token es válido, reenvía la solicitud HTTP o HTTPS y obtiene el código DOM del servidor original de la página solicitada:

c. El servidor de caja 2, transmite el código DOM del servidor original al sistema automático que lo había solicitado;

d. El sistema automático recibe el código DOM del servidor original y lo utiliza según sea necesario.

Por ejemplo, otro método de aceptar solicitudes por sistemas automáticos (p. ej. rastreadores de indexación) consiste en comprobar si la dirección IP del peticionario está o no en una lista blanca o realizar una comprobación inversa del nombre de máquina a partir de la IP.

EJEMPLO 3 – Figura 4

Haciendo referencia a la Figura 4, claves de codificación y/o cifrado también pueden ser intercambiadas por dispositivos externos 7a, 7b, los cuales están conectados por un secreto compartido entre el servidor de caja 2 y los dispositivos externos 7a, 7b. Por ejemplo, se utilizan sistemas OTK (clave de un solo uso, del inglés one-time-key) u OTP (contraseña de un solo uso, del inglés one-time-password). El método del ejemplo de la Figura 4 comprende los siguientes pasos:

- Se llevan a cabo los pasos a) a e) del Ejemplo 1;

f. El servidor de caja 2 aplica funciones de cifrado y/u ofuscación y/o compresión y/o codificación al código DOM del servidor original que ha sido transmitido por el servidor Web 5 de la aplicación Web 6. Este paso requiere una clave de un solo uso, la cual se genera de acuerdo con solicitud de la página al usuario.

- Se llevan a cabo los pasos g) a i) del Ejemplo 1;

La página de servicio recibe el código DOM del servidor ofuscado (es decir cambiado bajo f)) realiza una función inversa para obtener el código DOM del servidor original, y sustituye la página de servicio por la página original asociada con el código DOM del servidor. Para realizar la función inversa, la página de servicio utiliza una clave de un solo uso generada por un dispositivo externo 7b. Dicha clave es igual que o está relacionada con la clave que se utilizó para hacer los cambios bajo f).

Por ejemplo, el dispositivo externo 7b que se utilizó para generar la clave de descifrado interacciona con la página de servicio mediante un programa adicional (plug-in) instalado en el navegador Web 4 del usuario o utilizando la interfaz de entrada de audio.

Ventajosamente el sistema 1 proporciona una visión global de lo que sucede durante una sesión web, porque analiza las solicitudes individuales.

Por ejemplo, el método puede detectar:

- Si la solicitud procede de un usuario humano o de un BOT;
- Si la solicitud tiene temporizaciones “razonables” o “no razonables”;
- El Agente de Usuario y la IP de los cuales procede la solicitud;
- El UID del cual procede la solicitud.

Ventajosamente, el método de la presente invención permite seguimiento completo del usuario (es decir, el usuario cliente), y monitorización de sesión.

Ventajosamente, el método de la presente invención permite identificación y contrarrestación efectivas y seguras de ataques Man-in-the-Browser y/o Man-in-the-Middle, permitiendo de ese modo monitorización total de solicitudes de usuario.

Obviamente los expertos en la técnica apreciarán que varios cambios y variantes como los descritos anteriormente se pueden hacer para cumplir requisitos concretos, sin apartarse del alcance de la invención, como se define en las siguientes reivindicaciones.

REIVINDICACIONES

1. Un método de identificar y contrarrestar ataques de Internet, de tipos de ataque Man-in-the-Browser y/o Man-in-the-Middle y/o Bot, que comprende los pasos de:

- generar una solicitud mediante un navegador Web (4), relativa a una aplicación Web (6) que reside en un servidor Web (5),

- enviar dicha solicitud mediante dicho navegador Web (4) a un servidor de caja (2), el cual está en comunicación de señal con dicho servidor Web (5),

- recibir un código DOM del servidor mediante dicho servidor de caja (2), habiendo sido dicho código generado por dicho servidor Web (5) de acuerdo con dicha solicitud,

caracterizado por que comprende los pasos de:

- enviar un código de página de servicio mediante dicho servidor de caja (2) a dicho navegador Web (4), en respuesta a dicha solicitud, comprendiendo dicho código de página de servicio un código javascript y/o HTML ofuscado y polimórfico,

- recibir y renderizar dicho código de página de servicio mediante dicho navegador Web (4), - recibir y procesar dicho código javascript y/o HTML, mediante dicho navegador Web (4), para generar automáticamente una solicitud asíncrona, de tal manera que datos de entorno de dicho navegador Web (4) se pueden transmitir a dicho servidor de caja (2) y dicho código de página de servicio renderizado se puede transmitir a dicho servidor de caja (2),

- procesar dichos datos de entorno de dicho navegador Web (4), mediante dicho servidor de caja (2), para identificar ataques de Internet de los tipos de ataque Man-in-the-Browser y/o Man-in-the-Middle y/o Bot,

- procesar y comparar dicho código de página de servicio renderizado y dicho código de página de servicio mediante una aplicación de algoritmo (8) que reside en dicho servidor de caja (2), de tal manera que se pueda identificar al menos una diferencia de código, y procesar dicho código de página de servicio renderizado para compararlo con dicho código de página de servicio, mediante una función de comparación de dicha aplicación de algoritmo (8), para de ese modo generar al menos una señal indicativa de ataque Man-in-the-Browser y/o Man-in-the-Middle y/o Bot cuando dicho código de página de servicio es incompatible con dicho código de página de servicio renderizado,

- realizar una función de cifrado sobre dicho código DOM del servidor mediante dicho servidor de caja (2) para generar un código DOM ofuscado, y enviar dicho código DOM ofuscado a dicho navegador Web (4) en respuesta a dicha solicitud asíncrona,

- realizar una función de descifrado sobre dicho código DOM ofuscado mediante dicho código de página de servicio, para obtener dicho código DOM del servidor,

- renderizar dicho código DOM del servidor mediante dicho navegador Web (4).

2. Un método de acuerdo con la reivindicación 1, que comprende los pasos de:

- generar una solicitud asociada con dicha aplicación Web (6) mediante un sistema automático,

- enviar dicha solicitud y un código de autorización único a dicho servidor de caja mediante dicho mismo sistema automático,

- recibir dicho código DOM del servidor mediante dicho servidor de caja (2), habiendo sido dicho código generado por dicho servidor Web (5) de acuerdo con dicha solicitud,

- enviar dicho código DOM del servidor a dicho sistema automático mediante dicho servidor de caja (2), de acuerdo con dicho código de autorización único.

3. Un método de acuerdo con la reivindicación 1, que comprende los pasos de:

- proporcionar una clave criptográfica de un solo uso, mediante un dispositivo externo (7a, 7b), a dicho servidor de caja (2) y a dicho navegador Web (4),

- realizar dicha función de cifrado sobre dicho código DOM del servidor utilizando dicha clave criptográfica de un solo uso, para generar un código DOM ofuscado,

- realizar dicha función de descifrado sobre dicho código DOM ofuscado de acuerdo con dicho clave criptográfica de un solo uso, para generar dicho código DOM del servidor.

4. Un método de acuerdo con la reivindicación 3, que comprende los pasos de:

- generar un código de usuario mediante dicho servidor de caja (2), estando dicho código asociado con un usuario de dicho navegador Web (4),

- proporcionar dicha clave criptográfica de un solo uso de acuerdo a dicho código de usuario.

5. Un método de acuerdo con la reivindicación 1, que comprende los pasos de:

- generar una señal indicativa de ataque, mediante dicho servidor de caja (2), cuando dicha aplicación de algoritmo (8) identifica dicha al menos una diferencia de código, y enviar dicha señal indicativa de ataque a dicho navegador Web (4) y/o guardar dicha señal indicativa de ataque en una base de datos.

6. Un método de acuerdo con la reivindicación 1, que comprende los pasos de:

- procesar dicha solicitud mediante dicho servidor de caja (2), para identificar y contrarrestar ataques de Bot.

7. Un método de acuerdo con la reivindicación 1, que comprende los pasos de:

- realizar funciones de cifrado y/o ofuscación y/o compresión y/o codificación sobre dicho código DOM del servidor, para generar dicho código DOM ofuscado.

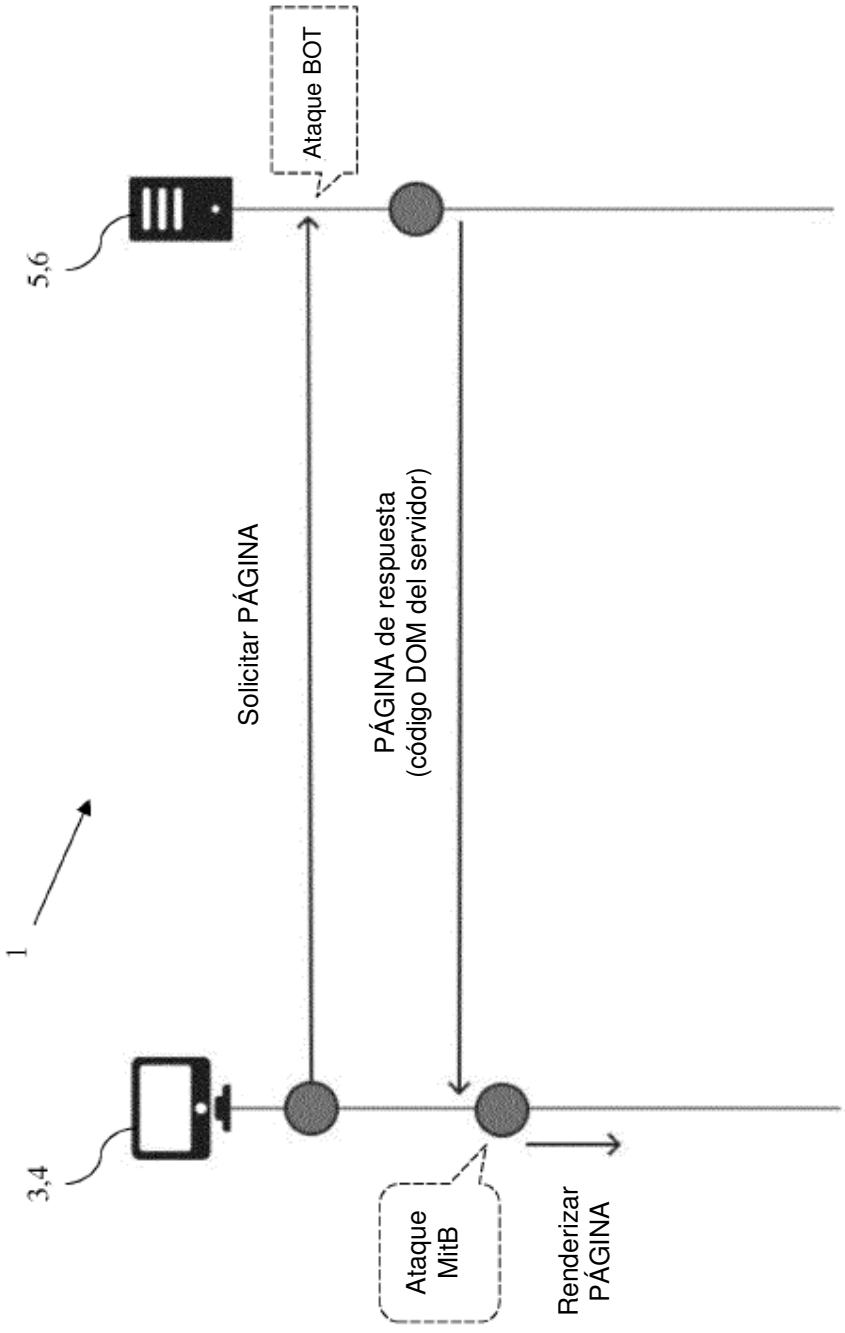
Ataque BOT
5,6
Solicitar PÁGINA
PÁGINA de respuesta
(código DOM del servidor)
1
3,4
Ataque
MitB
Renderizar
PÁGINA

Fig. 1

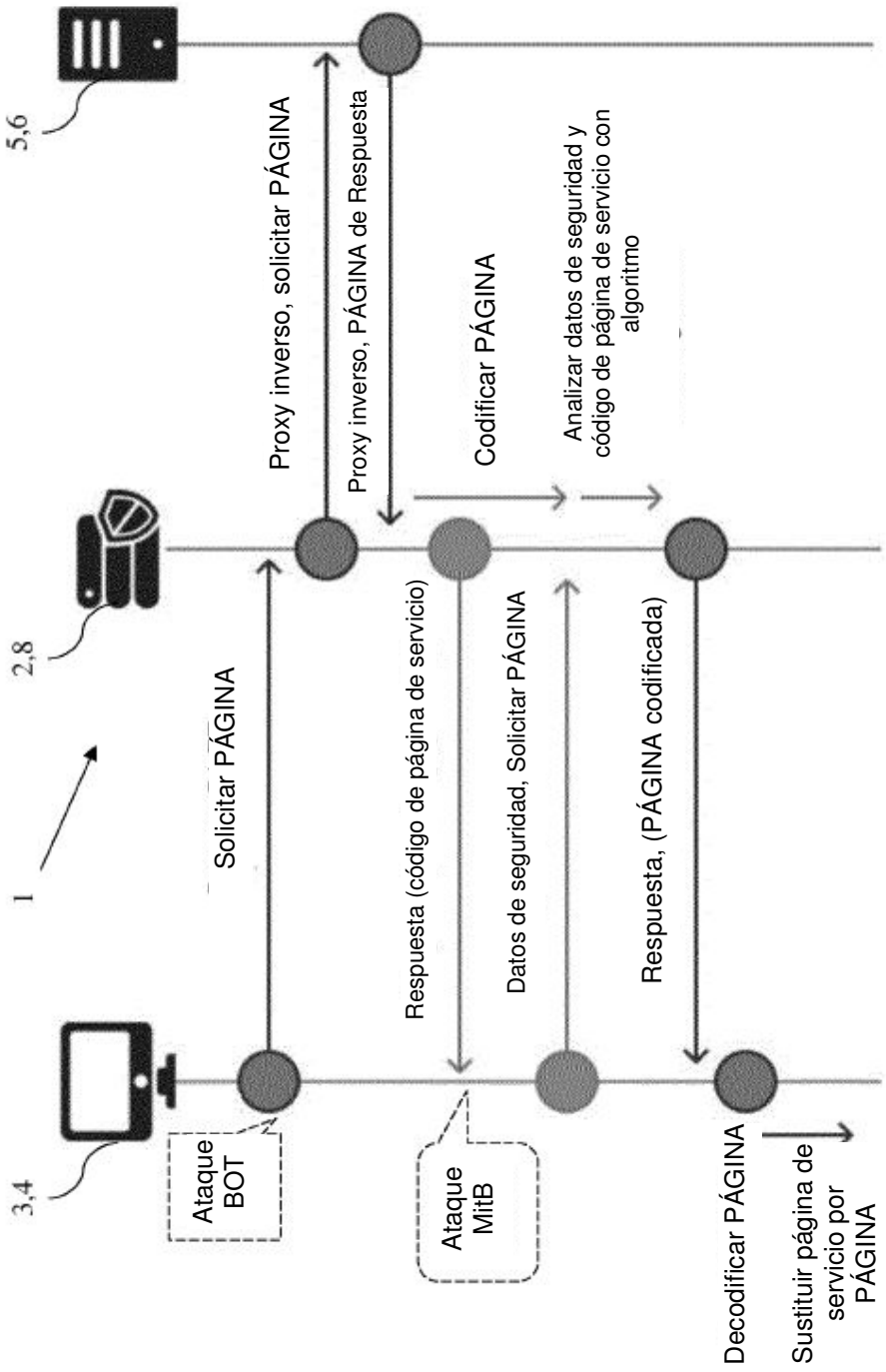
5,6
2,8
3,4
1
Solicitar PÁGINA
Proxy inverso, solicitar PÁGINA
Proxy inverso, PÁGINA de Respuesta
Codificar PÁGINA
Analizar datos de seguridad y código de página de servicio con algoritmo
Respuesta (código de página de servicio)
Datos de seguridad, Solicitar PÁGINA
Respuesta, (PÁGINA codificada)
Ataque BOT
Ataque MitB
Decodificar PÁGINA
Sustituir página de servicio por PÁGINA

Fig. 2

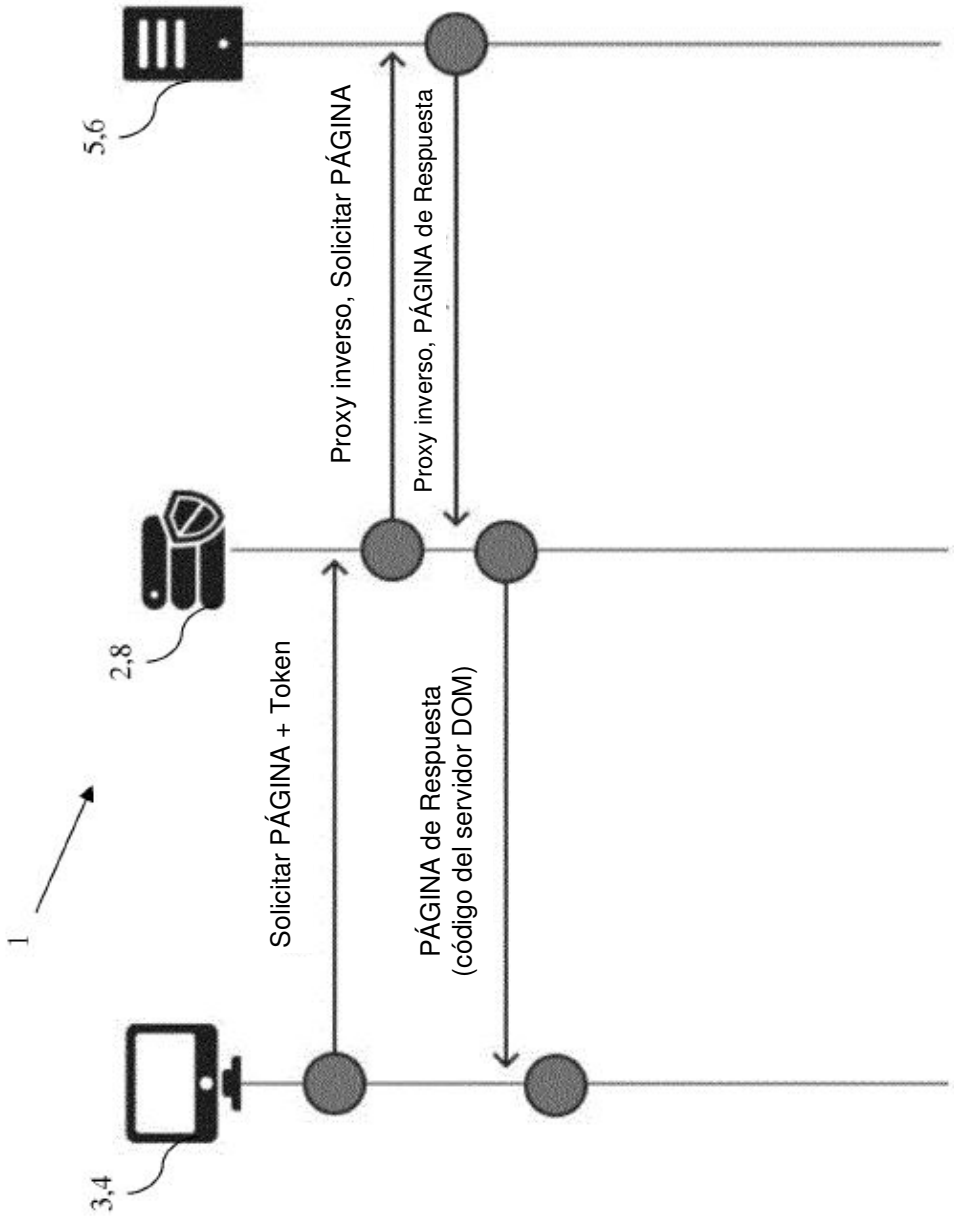
5,6
Proxy inverso, Solicitar PÁGINA
Proxy inverso, PÁGINA de Respuesta
2,8
Solicitar PÁGINA + Token
PÁGINA de Respuesta
(código del servidor DOM)
1
3,4

Fig. 3

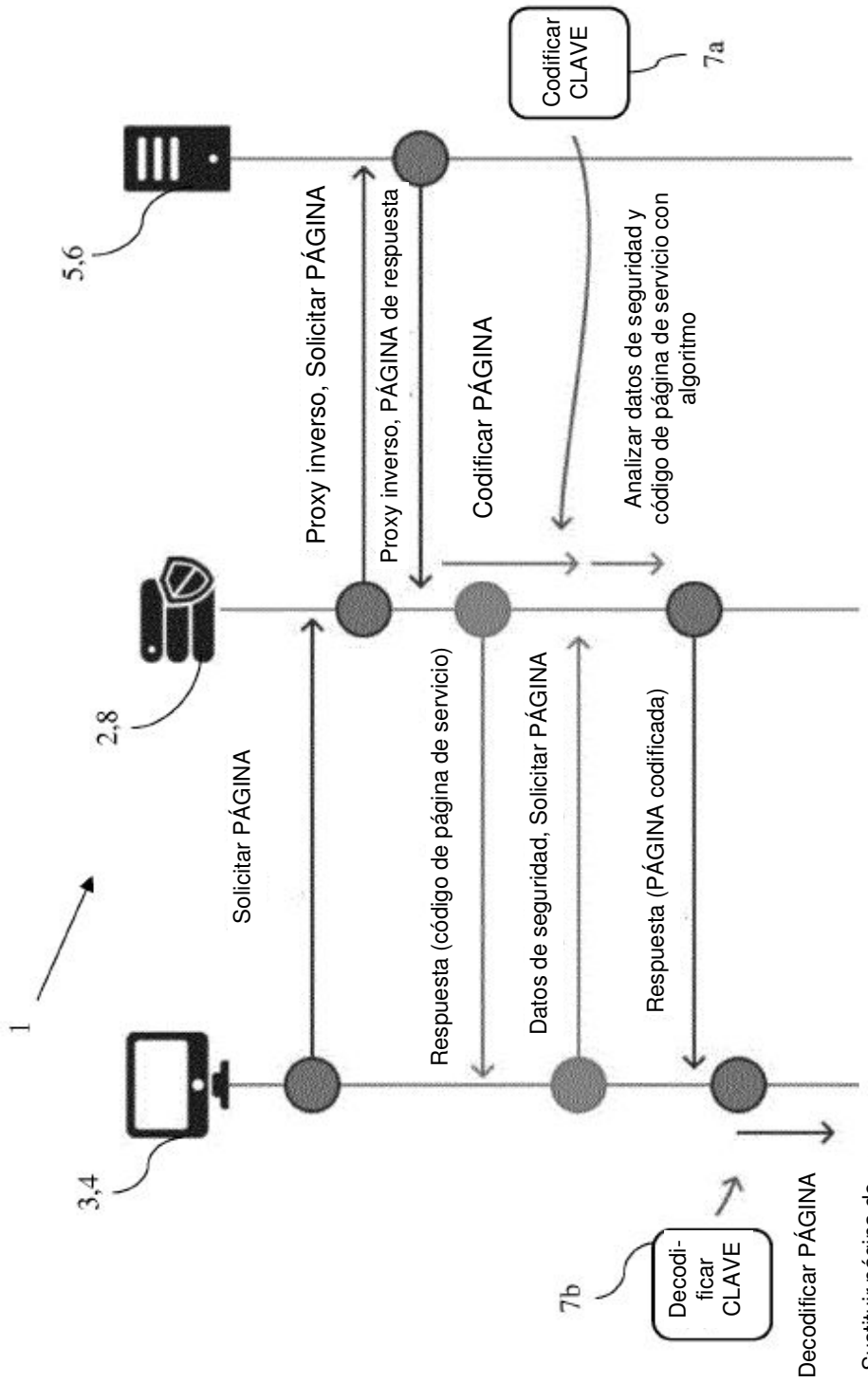
1
3,4
2,8
5,6
7a
7b
Solicitar PÁGINA
Proxy inverso, Solicitar PÁGINA
Proxy inverso, PÁGINA de respuesta
Codificar PÁGINA
Analizar datos de seguridad y código de página de servicio con algoritmo
Codificar CLAVE
Respuesta (código de página de servicio)
Datos de seguridad, Solicitar PÁGINA
Respuesta (PÁGINA codificada)
Decodi-ficar CLAVE
Decodificar PÁGINA
Sustituir página de servicio por PÁGINA

Fig. 4