SEGURIDAD EN INTERNET: WEB SPOOFING

José L. Núñez[1], Alejandro Millán[2], Paulino Ruiz de Clavijo[2], David Guerrero[2], Enrique Ostúa[2], Manuel J. Bellido[2] y Jorge Juan[2]

[1]Nuevas Profesiones - Universidad de Sevilla
http://www.nuevasprofesiones.org
Escuela Técnica Superior de Ingeniería Informática - Universidad de Sevilla
http://www.informatica.us.es
jnunez@nuevasprofesiones.org

[2]Departamento de Tecnología Electrónica - Universidad de Sevilla
http://www.dte.us.es
Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica
http://www.imse.cnm.es
{amillan, paulino, guerre, ostua, bellido, jjchico}@dte.us.es

RESUMEN

En este trabajo se estudia la técnica *Web Spoofing* como método de ataque a través de Internet. Se trata de una variante del clásico ataque *man-in-the-middle* en el que un ordenador intermedio analiza y registra información sensible. Además, se analizan los diferentes aspectos a considerar a la hora de aplicar esta técnica así como los detalles de la implementación realizada para mostrar su funcionamiento. Por último, se presentan una serie de contramedidas que el usuario puede adoptar con objeto de protegerse de este tipo de ataques. Así, el trabajo en su conjunto se plantea como una forma atractiva de mostrar al alumno algunos de los aspectos más importantes de la seguridad en Internet.

1. INTRODUCCIÓN

La capacidad que ofrecen las redes de comunicaciones y en especial Internet ha propiciado que numerosos expertos en el ámbito de la informática y de las telecomunicaciones dediquen su esfuerzo a la seguridad en la red. Parte de ellos se dedican a mantener seguros sus sistemas mientras que el resto intenta explotar las vulnerabilidades que dejan los primeros. El objetivo de este trabajo es explicar y mostrar uno de los ataques más recientes descubiertos en Internet, la técnica *Web Spoofing*. Para ello, se ha procedido a la implementación de esta técnica demostrando la vulnerabilidad de los dos navegadores más populares: *Microsoft Internet Explorer* y *Netscape Navigator*.

El término *spoof* significa engaño, truco, trampa o burla. Existen diversas técnicas informáticas de *Spoofing* basadas en falsear direcciones de correo electrónico (*Mail Spoofing*), direcciones IP (IP *Spoofing*), direcciones MAC (MAC *Spoofing*), servidores de nombres de dominio (DNS *Spoofing*), etc. La técnica *Web Spoofing* se realiza en el entorno *World Wide Web* y su principal objetivo es falsear un sitio web de confianza para la víctima (por ejemplo, el sitio web de su empresa, su banco, etc.) La técnica en sí es una variante del clásico ataque *man-in-the-middle* [1] en el que un ordenador intermedio analiza y registra información sensible (ataque pasivo) o incluso la modifica en ambos sentidos (ataque activo).

Aunque en 1995 Cohen ya describe un método de ataque parecido a éste basado en la reescritura de URI (*Uniform Resource Identifiers*) [2], el término *Web Spoofing* fue acuñado por Felten *et al.* en 1996 [3]. En su informe contemplaban la posibilidad de recrear una copia

falsa de Internet (lógicamente generada bajo demanda) mediante reescritura de URI utilizando *JavaScript* [4, 5]. Trabajos posteriores han analizado estas posibilidades [6, 7] y en la actualidad Ye *et al.* [8] han logrado mostrar el funcionamiento de este ataque realizando una implementación mediante *JavaScript* y DHTML (*Dynamic HyperText Markup Language*) [9] de algunos sitios concretos que además consigue falsear una conexión TLS/SSL (*Transport Layer Security*/*Secure Sockets Layer*) [10].

En este trabajo, se han fusionado la idea inicial de Felten *et al.* de recrear una copia falsa de Internet al completo y la implementación de Ye *et al.* para desarrollar un *Servidor Web Spoofing* (SWS) genérico. La organización de este artículo es la siguiente: en la sección 2 se presenta la técnica *Web Spoofing* detallando las fases que la componen así como la forma de actuar del SWS, en la sección 3, se explica cómo se ha implementado el SWS incluyendo su estructura y su funcionamiento, en la sección 4 se presentan una serie de contramedidas que el usuario puede emplear para protegerse de este tipo de ataques y, finalmente, en la sección 5 se detallan las principales conclusiones de este trabajo.

2. WEB SPOOFING

El ataque mediante *Web Spoofing* consiste en hacer creer a la víctima que todo lo que ve es genuino cuando no lo es. Es decir, la víctima entra en contacto con un SWS camuflado como el sitio web que realmente pretendía visitar. Desde ese momento, la interacción con el sitio web es la esperada aunque puede estar siendo registrada y manipulada.

Los peligros de este ataque son claros ya que atentan contra los principios de: privacidad (el SWS analiza y registra toda la información que circula entre la víctima y el servidor real—dirección IP, nombre de la máquina, localización geográfica, datos personales, claves de acceso, etc.), autenticidad e integridad (la información en ambos sentidos puede ser alterada—por ejemplo, podría modificarse el número de cuenta de destino en una transferencia bancaria) y réplica (las peticiones realizadas por la víctima podrían ser replicadas por el SWS—por ejemplo repitiendo una acción de compra por Internet).

Además de todo esto, es importante considerar que las conexiones seguras (por ejemplo a través de TLS/SSL) no evitan este tipo de ataques ya que aunque la víctima haya establecido una conexión segura, lo ha hecho con el SWS no con el servidor real. Por último, dentro de la técnica podemos distinguir dos fases: una primera fase de contacto y una segunda fase en la que actúa el propio SWS.

2.1. Fase de contacto

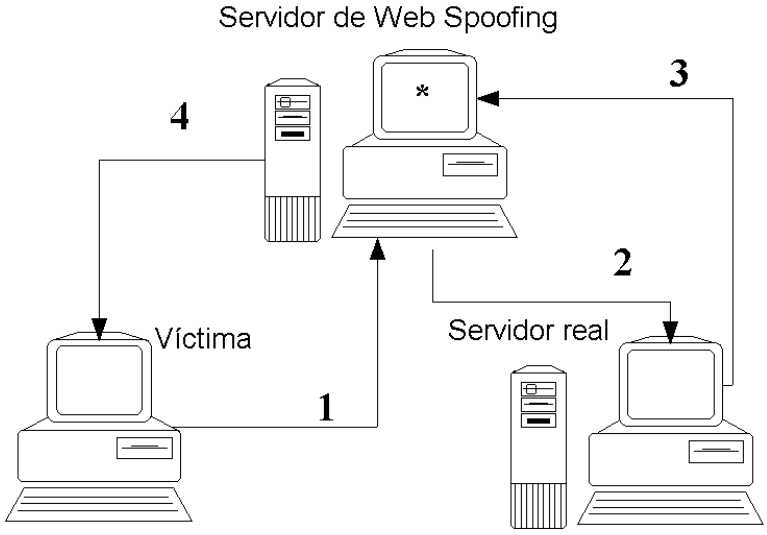
La actuación del SWS comienza cuando la víctima accede mediante su navegador a alguna página alojada en el mismo. Esto puede conseguirse de varias formas:

- Buscador o enlace: el atacante incluye una página del SWS en algún buscador asociada a palabras clave que la víctima relacione con el sitio web real. En dicha página puede colocarse un enlace del tipo “Clic aquí para ir a www.mibanco.com”. Por otra parte, el atacante puede insertar un enlace en algún sitio web muy visitado que engañe a la víctima conduciéndole al SWS.
- *Typejacking*: esta técnica se basa en imitar el nombre de un sitio web de confianza para la víctima utilizando un nombre de dominio parecido (por ejemplo, www.microsof.com) o bien insertando en un buscador un nombre similar visualmente (por ejemplo, www.e1monte.es—sustituyendo la letra “l” minúscula por el número “1”)

- Combinación con otras técnicas de *Spoofing*: el atacante puede enviar el enlace falso por correo electrónico mediante *Mail Spoofing* o bien conducir a la víctima al SWS mediante DNS *Spoofing*.

2.2. Actuación del SWS

Con objeto de engañar a la víctima, el SWS le muestra una copia del sitio web falseado. Esta copia es generada dinámicamente y no es en ningún caso una copia exacta. Aunque visualmente se trata de la misma página, el SWS se encarga de modificar la información necesaria para conseguir: (a) que la víctima no abandone el SWS (asegurándose de que todos los enlaces conducen al propio SWS) y (b) que la información que consulte o introduzca la víctima sea debidamente registrada (Fig. 1). Así, el SWS necesita tratar dos aspectos básicos: el aspecto que presenta el navegador de la víctima y el código fuente de la propia página web.



Funcionamiento del Web Spoofing

1.- La víctima solicita cualquier recurso mediante una dirección o URL
2.- El servidor malicioso procesa la petición de la víctima y solicita el recurso al Web real utilizando para ello la URL proporcionada por la víctima
3.- El servidor real proporciona el recurso al servidor de Web Spoofing
4.- El servidor atacante realiza las modificaciones del recurso real y se lo envía a la víctima

*: En todo momento el tráfico de información está siendo observado

Fig. 1. Funcionamiento del SWS.

En primer lugar, para recrear adecuadamente el aspecto del navegador de la víctima, es necesario falsear fundamentalmente: el título de la ventana, la barra de direcciones (con objeto de alterar cualquier URI que pudiera teclear la víctima para conducirla de nuevo al SWS) y la barra de estado (con objeto de que la víctima no vea el URI real al que le conduce cada enlace de la página falsa). Todo esto puede llevarse a cabo utilizando por ejemplo *JavaScript* o DHTML.

En segundo lugar, para tratar el código fuente de la propia página web, el SWS necesita procesarlo antes de transferirlo a la víctima. En este proceso, el SWS se encarga de reescribir dos elementos fundamentales: enlaces (todos los URI del código deben conducir ahora al propio SWS—incluidos los marcos) y formularios (si se trata de un ataque activo, es

necesario que el campo ACTION de los formularios conduzca a un *script* intermedio que se encargue de registrar los datos introducidos por la víctima). Todo esto puede llevarse a cabo utilizando por ejemplo PHP (PHP: *Hypertext Preprocessor*) [11, 12, 13] o CGI (*Common Gateway Interface*) [14].

3. IMPLEMENTACIÓN

Para mostrar todo lo expuesto anteriormente, se ha implementado una versión de este ataque. Para ello se ha creado un SWS genérico, basado en *JavaScript* y PHP. El SWS utiliza PHP para realizar todo el tratamiento de los URI que solicita la víctima y reescribir las páginas web falseadas. Por otro lado, utilizamos las posibilidades que nos ofrece *JavaScript* para falsear el navegador del usuario. El desarrollo se ha centrado en los aspectos más interesantes de dos de los navegadores más utilizados hoy en día, *Microsoft Internet Explorer* y *Netscape Navigator*.

La implementación se ha realizado sobre una máquina con sistema operativo GNU/Linux sobre el que se ejecuta un servidor web. Además, es necesario que este servidor tenga soporte para PHP. En este sentido, un servidor típico en entornos de libre distribución es Apache [15, 16] ya que su estructura modular permite incluir la interfaz necesaria para interactuar con PHP. Por último, cabe indicar que el sistema en conjunto puede ejecutarse sobre cualquier máquina con sistema operativo GNU/Linux y que cuente con conexión a Internet.

3.1. Estructura

Para la realización práctica del ataque se ha creado una página que contiene tres marcos. El marco superior (*top*) contiene la parte superior del navegador falseado, la parte principal (*main*) muestra la web falseada en cada momento y la parte inferior (*footer*) mantiene la apariencia de la barra inferior del navegador. El ataque se ha recreado para *Explorer* 6.0 y *Navigator* 4.78 con sus apariencias estándar en español (sistema de menús, posicionamiento de ventanas, etc.).

3.2. Funcionamiento

En primer lugar, se produce la toma de contacto entre el servidor atacante y la víctima. Para la demostración se ha supuesto que la víctima, por alguno de los métodos explicados con anterioridad, ya ha caído en manos del SWS. En nuestro caso, existe una página web expresamente creada con los enlaces necesarios para que comience la demostración.

Por tanto, la primera vez que entramos en contacto con el SWS lo haremos pinchando en algún enlace “falseado”. Una vez que pinchamos en tal enlace se producirán una serie de acciones ajenas al usuario que permitirán continuar con el engaño. Lo primero será determinar el tipo de navegador que éste posee. Dependiendo de si se trata de *Explorer* o *Navigator*, el SWS tendrá que ofrecernos una página de marcos u otra. De hecho, la página de marcos será la que determinará la apariencia del navegador falseado.

Así, al pinchar en este primer enlace se abrirá una nueva ventana del navegador que no posee los elementos a los cuales estamos habituados; es decir, barra de estado, barra de herramientas, barra de direcciones, etc. La nueva ventana se abre totalmente vacía pero contiene tres marcos que sirven para simular el navegador. La rapidez con la que se cargan estos marcos sirve para que el usuario crea que sigue con su navegador. Estos marcos, como explicábamos anteriormente dependerán del navegador de la víctima.

3.3. Falsificación de la apariencia del navegador

Para conseguir engañar al usuario crearemos en los marcos superior e inferior de la ventana del SWS tanto la parte superior como la inferior del navegador de la víctima. Para la realización práctica de todo esto hemos utilizado *JavaScript* y un programa para capturar las imágenes de un navegador real. Con el fin de evitar que el usuario vea la carga de las imágenes una por una, utilizamos un método de cacheado de imágenes en *JavaScript* que permite mostrar el conjunto al completo cuando se haya terminado la carga (Fig. 2). Según el nivel de detalle deseado puede añadirse más o menos realismo a las partes falseadas. En nuestra implementación, nos hemos limitado a falsear los aspectos más habituales en los que se fija el usuario.

3.3.1. Título de la ventana

Por cada acción que realice el usuario solicitando una nueva página, ya sea a través de un enlace o bien mediante la barra de direcciones, el SWS actúa en primer lugar leyendo la página real y estableciendo el título de ésta en la nueva ventana falseada. La forma más sencilla de realizar esto es utilizando el contenido de una variable global (*titulo*) tras la marca <TITLE> de la página que define los marcos.

3.3.2. Barra de menús

Para el caso de *Explorer* hemos creado una tabla con tantas columnas como posibles opciones existen en la barra de menús del navegador. El usuario puede mover el puntero del ratón sobre estas imágenes comprobando el efecto de relieve típico en estos casos (evento *onmouseover*) y también pulsando sobre ellas; lo que hace aparecer el menú correspondiente (evento *onclick*). Para que las imágenes falsas se comporten así hemos utilizado funciones en *JavaScript* para cambiar su estado. No obstante, hemos optado por falsear tan sólo el primer nivel de la jerarquía. Sin embargo, es posible una falsificación mediante mapas de imágenes en *JavaScript* que permitieran mayor nivel de interacción.

Por otra parte, la versión para *Netscape* se ha diseñado utilizando la barra de menú genuina. Esta barra podemos mostrarla abriendo la nueva ventana con el atributo correspondiente establecido a cierto (argumento de la función *winopen* de *JavaScript*). La diferencia nos ahorra tener que copiar toda la barra de menú y controlar los eventos, pero al mismo tiempo facilitaría que el usuario pudiera abandonar el control del atacante o darse cuenta del fraude.

3.3.3. Barra de herramientas

En este caso, el tratamiento que hemos seguido para *Explorer* sigue siendo distinto que para *Navigator*. Así pues, en la versión para el primero actuamos de forma totalmente análoga a como lo hacemos con la barra de menús, es decir, precacheado de imágenes para cada botón y control de los eventos de ratón para cuando el usuario pase el puntero por encima o pinche. En la versión para *Navigator* hemos retirado esta funcionalidad y hemos expuesto la barra de herramientas y de direcciones como el fondo de la página que la contiene (atributo BACKGROUND).

Fig. 2.a. *Microsoft Internet Explorer* 6.0 falsificado mediante *JavaScript* a una resolución de 800x600.

Fig. 2.b. *Netscape Navigator* 4.78 falsificado mediante *JavaScript* a una resolución de 800x600.

3.3.4. Barra de direcciones

Para conseguir la barra de direcciones hemos creado un formulario con un campo de texto, posicionado en su sitio, que recoge las entradas del usuario. Tenemos pues, una barra de direcciones falsa y totalmente editable que envía la petición de la página al SWS al presionar la tecla *Enter* (realizando la función SUBMIT del formulario). Uno de los problemas que encontramos al realizar la falsificación es el hecho de posicionar el campo de texto en el lugar adecuado y establecer el tipo de letra que contendría el formulario. Para solucionar ambos problemas utilizamos el atributo STYLE que ofrecen las hojas de estilo.

3.3.5. Barra de estado

La barra de estado se encuentra falseada en ambos casos como una tabla que contiene las imágenes propias de la misma. Así pues, mediante *JavaScript*, esta tabla permite la escritura de los mensajes de información que se obtienen normalmente al pasar por ejemplo el puntero sobre un enlace y muestra los iconos que nos alertan de la seguridad de la página (si existe conexión segura o no).

3.3.6. Barra de progreso

Para ambos navegadores hemos utilizado un *script* que pretende imitar la carga de la página mostrando una barra de progreso ficticia.

3.4. Acción del SWS paso a paso

Una vez que hemos determinado el tipo de navegador de la víctima y lo hemos falseado en pantalla se produce la acción del servidor atacante sobre el URI proporcionado por la víctima. Lo primero que hace la aplicación desde el marco principal es establecer los valores apropiados para las variables globales. Dentro de éstas, se encuentra la variable que determina el tipo de navegador que utiliza el cliente y las variables que especifican las opciones de configuración activas en ese momento. Las opciones de configuración son leídas de un fichero en cada ejecución y permiten, además de especificar las acciones del SWS, realizar algunas funciones de borrado sobre directorios y ficheros en el servidor.

Lo siguiente a realizar por el SWS es un tratamiento inicial del URI que desea la víctima. Este tratamiento tiene como finalidad saber qué sitio web es el que vamos a falsear (establecer el valor de ciertas variables globales) y adaptar a un formato común la cadena que representa el URI para su posterior procesado. Una vez que tenemos un formato apropiado para la dirección del recurso, la aplicación descompone el URI en tres partes: el dominio falseado, la ruta hasta el recurso y el propio nombre del recurso. El objetivo en este caso es crear la misma estructura de directorios en el disco duro del SWS para poder falsear la web y acceder al recurso posteriormente. La especificación del recurso y su localización real en Internet vienen determinadas por su URI. En concreto, el SWS siempre precisará el URI absoluto, de ahí que necesite variables globales para conocer el sitio falseado y los directorios que contiene. No obstante, conociendo la estructura de directorios del sitio falseado, el SWS también podrá tratar enlaces relativos al documento.

Con la estructura de directorios creada en nuestro servidor atacante procedemos a obtener el recurso pedido. Para ello tendremos que tener en cuenta un par de cosas: el tipo de conexión y el tipo de recurso. El tipo de conexión lo determina el protocolo del URI introducido por el usuario:

- HTTP: las conexiones no seguras que siguen el protocolo de transferencia de hipertexto las podemos tratar directamente con las funciones específicas que nos ofrece PHP: *fopen*, *fread*, *fgets*, etc. que permiten establecer conexiones HTTP.
- HTTPS: para manejar conexiones seguras tendremos que hacer uso de la librería CURL [17]. Esta librería, al no ser estándar, precisa instalarse en el servidor atacante.

Una vez que sabemos el tipo de conexión que podemos establecer entre el SWS y el sitio web real tenemos que tener en cuenta el tipo de recurso que nos pide la víctima para poder realizar así su tratamiento en el servidor. Así pues, el servidor no modificará aquellos recursos que por su estructura binaria no interese, por ejemplo, imágenes o animaciones *Flash* [18]. De esta forma, la acción principal del SWS se centrará en recursos de texto que contengan código fuente que nos interese modificar. El caso general será que la víctima solicite alguna página web, es aquí donde se centra la mayor funcionalidad de la aplicación. Si el recurso como hemos dicho es de texto y contiene marcas interesantes de analizar, el SWS lo analizará detenidamente para realizar la reescritura del código.

3.5. Reescritura del código

La reescritura del URI se produce de forma distinta según sea el tipo de conexión que abrimos con el servidor real. Los recursos accesibles mediante el protocolo HTTP pueden ser leídos carácter a carácter y modificados dinámicamente, sin embargo, los recursos que precisan de una conexión TLS/SSL tienen que ser previamente guardados como locales en el atacante y posteriormente darles el tratamiento normal. Por tanto, la particularidad de las conexiones TLS/SSL provoca que tengamos dos copias locales del recurso: una original y otra modificada que ofreceremos a la víctima.

El tratamiento carácter a carácter consiste en ir obteniendo las posibles etiquetas HTML que pueda contener el fichero y en analizarlas. Cada vez que obtengamos una cadena que contiene <...> la pasaremos a una función que se encargará de analizarla y realizar las modificaciones pertinentes en base a expresiones regulares. Las modificaciones que realiza el SWS son las siguientes:

3.5.1. Etiquetas que modifican su URI

El siguiente conjunto de etiquetas HTML ven modificadas por el SWS su URI. El nuevo URI hace siempre referencia a la dirección falseada en el servidor atacante. El SWS actúa de tal forma que primero obtiene el recurso al cual hace referencia la etiqueta, lo hace propio, y después reescribe la etiqueta para que enlace a su dirección. De esta forma se consigue que todo lo que finalmente se ofrece a la víctima lo tenga el atacante localmente. Las etiquetas que siguen este tratamiento son las siguientes: <AREA HREF = “URI” ...>, <APPLET CODEBASE = “URI” ...>, <BODY BACKGROUND = “URI” ...>, <EMBED SRC = “URI” ...>, <FRAME SRC = “URI” ...>, <IMG SRC = “URI” ...>, <META URI = “URI” ...>, <LINK HREF = “URI” ...>, <SCRIPT SRC = “URI” ...> y <PARAM VALUE = “URI” ...>

3.5.2. Formularios: <FORM ACTION = “URI” ...>

Uno de los objetivos del atacante es averiguar los datos que la víctima envía al sitio web real. Es por ello necesario que el SWS incluya alguna forma de capturar estos datos. La manera más sencilla de conseguirlo es falsificando el formulario que la víctima rellena. Para ello,

cuando el usuario solicita un URI que contiene un elemento FORM se modifica el formulario original de forma que permite que un programa PHP se encargue de registrar los datos que el usuario envía cuando realiza la acción SUBMIT (ese programa además se genera dinámicamente).

3.5.3. Entradas de formulario: <INPUT TYPE = "..." NAME = "..." ...>

En un formulario común, básicamente podemos encontrar los siguientes elementos de entrada para datos del usuario: INPUT, SELECT y TEXTAREA. Para nuestra demostración, hemos tenido especialmente en cuenta los elementos de tipo INPUT. Estos controles pueden ser de varias clases según se especifique en su atributo TYPE. Así, tenemos: BUTTON, CHECKBOX, FILE, HIDDEN, IMAGE, PASSWORD, RADIO, RESET, SUBMIT y TEXT.

Para poder hacer referencia a cada uno de estos controles, HTML dispone del atributo NAME. Por otra parte, el valor que contienen se establece en el atributo VALUE del control. Por tanto, el par NAME/VALUE es lo que se envía al programa PHP que procesa el formulario. La forma en que el SWS trata una etiqueta INPUT de este estilo es la siguiente:

- Comprueba si el tipo de entrada es SUBMIT. En tal caso, puede ocurrir que la etiqueta incluya el evento *onclick* o no. Si lo incluye probablemente llamará a alguna función propia del CGI real para procesar los datos. En este caso, al haber modificado ya el atributo ACTION del formulario eliminamos este evento para que se ejecute el programa PHP que almacena los datos. Si por el contrario no incluye el evento *onclick*, el botón SUBMIT funciona como siempre, invocando al programa PHP que establecemos en la etiqueta FORM por lo que no es necesario modificarlo.
- Si el tipo de entrada no es SUBMIT entonces la reescritura se hace tal cual pero almacenando el valor del atributo NAME en el programa PHP que procesa los datos. De esta forma, estamos generando un programa PHP dinámicamente que puede almacenar los valores que la víctima introduce al saber el nombre de los controles.

3.5.4. Direcciones de correo electrónico: <A HREF = "mailto:usuario@dominio" ...>

Los enlaces que apuntan a alguna dirección de correo electrónico son también susceptibles de ser modificados por el atacante. Siguiendo esta línea, el atacante podría modificar todos estos enlaces de una página para que cuando la aplicación de correo se active tenga como destinatario su propia dirección. En nuestra implementación del SWS damos a elegir en la configuración la opción de modificar los enlaces de correo en las páginas.

4. CONTRAMEDIDAS

Como en la vida real, algunas imitaciones son simplemente idénticas a la original. Teóricamente esto puede ocurrir con la técnica *Web Spoofing*. El ataque en sí mismo permite reproducir prácticamente todo y es muy difícil de manera sistemática detectar el fraude. Algunas medidas que puede adoptar el usuario para defenderse de este ataque son: (a) desactivar el soporte de *Java*, *JavaScript* y *Active-X* en su navegador, (b) asegurarse de que la barra de direcciones está siempre visible y muestra el sitio real, (c) vigilar los cambios que sobre ésta puedan producirse y las acciones que realiza el navegador en todo momento, (d) comprobar el código fuente de las páginas y observar que no se producen reescrituras

extrañas y (e) personalizar el navegador con algún tipo de fondo especial, tipo de letra exclusivo, etc.

Esta estrategia de protección puede resultar excesiva para evitar este riesgo pero constituye las forma más segura para protegerse del ataque. El soporte de *Java*, *JavaScript*, *Active-X*, etc. permite añadir mucha más funcionalidad a las páginas pero a costa de aumentar el riesgo de este tipo de ataques. El desactivar este soporte dejaría al navegador menos capacitado para presentar ciertas páginas web pero mucho más seguro. Si el usuario precisa incondicionalmente de estas características debería intentar asegurarse de otra forma de que el sitio que visita es realmente el que dice ser.

Además de estas medidas, el usuario puede optar por instalar algún software especialmente dedicado a esto. En esta categoría podemos encontrar entre otros:

- *Anonymizer* [19]: se trata de un programa que aprovecha la técnica de reescritura de URI en beneficio del propio usuario suministrando privacidad, evitando seguimiento *on-line* y *spam*, etc.
- *Mozilla* con SRD (*Synchronized Random Dynamic*) *Boundary* [20]: se trata de un navegador que muestra distintos tipos de ventanas. Así, el borde de las ventanas varía de color en función del grado de confianza de su contenido. Además, estos bordes cambian de estilo de forma aleatoria y periódica por lo que es imposible falsearlos.

5. CONCLUSIONES

Como se ha mostrado, las nuevas tecnologías para la web traen consigo nuevos riesgos y con ellos deben aparecer por tanto nuevos métodos de seguridad que garanticen una comunicación segura. En este trabajo se ha estudiado la técnica *Web Spoofing* como método de ataque a través de Internet. Se trata de una variante del clásico ataque *man-in-the-middle* en el que un ordenador intermedio analiza y registra información sensible. En concreto, el método de ataque mediante *Web Spoofing* es casi imposible de detectar con los navegadores convencionales. Es decir, la mayoría de los navegadores, entre ellos los más populares, carecen de la seguridad necesaria para garantizarnos protección frente a estos y otros métodos. Además de todo esto, hay que tener en cuenta que las conexiones TLS/SSL pueden ser falseadas mediante métodos como éste.

En este artículo se han analizado los diferentes aspectos a considerar a la hora de aplicar esta técnica así como los detalles de la implementación realizada para mostrar su funcionamiento. Y por último, se han presentando una serie de contramedidas que el usuario puede adoptar con objeto de protegerse de este tipo de ataques. Así, el trabajo en su conjunto se plantea como una forma atractiva de mostrar al alumno algunos de los aspectos más importantes de la seguridad en Internet.

6. BIBLIOGRAFÍA

[1] R. Graham, *Hacking Lexicon*, on-line: http://www.robertgraham.com/pubs/hacking-dict.html, march 8, 2004.

[2] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter, *Uniform Resource Identifiers (URI): Generic Syntax*, on-line: http://www.ietf.org/rfc/rfc2396.txt, The Internet Engineering Task Force (IETF), january, 1999.

[3] E.W. Felten, D. Balfanz, D. Dean, and D.S. Wallach, *Web Spoofing: An Internet Con Game*, on-line: http://www.cs.princeton.edu/sip/pub/spoofing.html, 1996.

[4] D. Goodman, *JavaScript Bible*, 4th Ed., Hungry Minds, Inc., New York (USA), 2001.

[5] *Core JavaScript Guide*, on-line: http://devedge.netscape.com/library/manuals/2000/javascript/1.5/guide/, Netscape Communications Corp., september 28, 2000.
[6] J. D. Tygar and Alma Whitten, WWW Electronic Commerce and Java Trojan Horses, *Proc. 2nd USENIX Workshop on Electronic Commerce*,1996.

[7] F. De Paoli, A. L. DosSantos, and R. A. Kemmerer, Vulnerability of 'Secure' Web Browsers, *Proc. National Information Systems Security Conference*. 1997.

[8] E.Z. Ye, Y. Yuan, and S. Smith, *Web Spoofing Revisited: SSL and Beyond*, Technical Report TR2002-417, Department of Computer Science, Dartmouth College, Hanover, NH (USA), february 1, 2002.

[9] *Web Content Accessibility Guidelines 1.0. W3C Recommendation 5-May-1999*, on-line: http://www.w3.org/TR/WAI-WEBCONTENT/, World Wide Web Consortium (W3C), may 5, 1999.

[10] T. Dierks and C. Allen, *The TLS Protocol. Version 1.0*, on-line: http://www.ietf.org/rfc/rfc2246.txt, The Internet Engineering Task Force (IETF), january, 1999.

[11] S.S. Bakken, A. Aulbach, E. Schmid, J. Winstead, L.T., Wilson, R. Lerdorf, A. Zmievski, and J. Ahto, *PHP Manual*, on-line: http://www.php.net/manual/en/, PHP Documentation Group, march 1, 2004.

[12] L. Welling and L. Thomson, *PHP and MySQL Web Development*, Sams Publishing, Indianapolis, Indiana (USA), 2001.

[13] F.J. Gil, J.A. Tejedor, A. Yagüe, S. Alonso, and A. Gutiérrez, *Creación de sitios web con PHP4*, McGraw-Hill, Madrid (Spain), 2001.

[14] *The CGI Resource Index*, on-line: http://www.cgi-resources.com/, march 9, 2004.

[15] C. Aulds, *Linux Apache Web Server Administration*, Sybex, Inc., Alameda, California (USA), 2001.

[16] *The Apache Software Foundation*, on-line: http://www.apache.org/, march 9, 2004.

[17] *cURL*, on-line: http://curl.haxx.se/, march 9, 2004.

[18] D. Franklin and B. Patton, *Macromedia Flash 5*, Prentice Hall, Madrid (Spain), 2001.

[19] *Anonymizer*, on-line: http://www.anonymizer.com/, Anonymizer, Inc., 2003.

[20] E. Ye, *Mozilla with SRD (synchronized random dynamic) Boundary*, on-line: http://www.cs.dartmouth.edu/~pkilab/demos/countermeasures/, april 24, 2003.