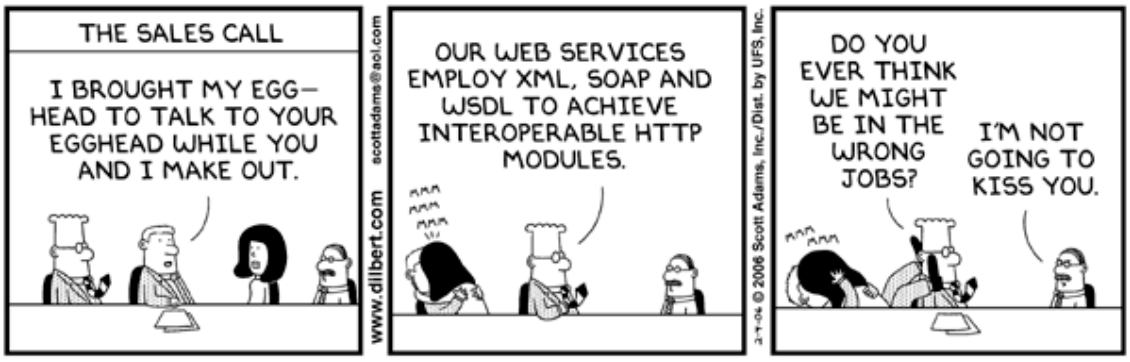
Comercio Electrónico

# Práctica 8: Seguridad en los accesos




© Scott Adams, Inc./Dist. by UFS, Inc.

**José Luis Salazar**
jsalazar@unizar.es

**Antonio Sanz**
ansanz@unizar.es

**Rafael del Hoyo**
rdelhoyo@ita.es

# Objetivo de la Práctica

Poner en práctica algunos de los conceptos vistos en la teoría relativos a la seguridad de nuestra infraestructura de comercio electrónico. Se verán de forma práctica varios aspectos de la programación segura de aplicaciones web y el uso de una app de una red social para la identificación de usuarios que usa AJAX para transferir la función de identificación.

# ¿Qué hay preparar de forma previa a la práctica?

Será necesario revisar y tener frescos todos los conocimientos adquiridos en la teoría, así como haber repasado los conceptos de PHP y SQL vistos en prácticas anteriores (que serán necesarios para la primera parte de la práctica).

# ¿Cuál es el resultado de la práctica?

Se obtiene como resultado de la práctica un servidor web capaz de realizar transacciones seguras gracias al protocolo SSL.

# ¿Qué se aprende con esta práctica?

Se aprenden los principales problemas de seguridad relativos al desarrollo de accesos a páginas web así como las soluciones a implantar para evitarlos.

# Seguridad en aplicaciones web

## *Preparación inicial*

- En primer lugar, descargar de la página web de la asignatura el fichero prueba.zip, que contiene todo el código necesario para realizar la práctica, y descomprimirlo en un directorio temporal.

- Crear una BD en el MySQL empleando el PHPMyAdmin (se recomienda crearla con el nombre “prueba”). Dentro de la mismo interfaz, se creará un usuario y se le asignarán privilegios totales únicamente sobre la BD antes creada (se recomienda crearla con el nombre “prueba” y la contraseña “password”).

- A continuación se poblará esta BD importando la estructura y datos contenidos en el fichero pruebaBD.sql (dentro de PHPMyAdmin puede realizarse dentro de la pestaña de “Ejecutar SQL”).

- Copiar los ficheros con extensión .php y .htm en la carpeta con visibilidad dentro del servidor web (en el equivalente a la carpeta /htdocs en cada caso).

- Modificar el fichero D:\xamppN\xampp\apache\bin\php.ini, y cambiar el valor de la variable “magic_quotes_gpc” de ON a OFF.

- Reiniciar el Apache para que aplique los nuevos valores de la configuración.

## *Inyección SQL*

La inyección SQL (SQL injection) consiste en aprovecharse de la facilidad de modificar la entrada a un programa para generar parámetros de entrada maliciosos que permitan realizar operaciones no autorizadas sobre la propia BD.

### *Evitar mecanismos de login*

Dentro del código de prueba se tienen dos ficheros:

- login.htm: Formulario HTML que pide un nombre de usuario y una contraseña, y los envía a un PHP que realiza las tareas de autenticación.

- Autentica.php: Programa en PHP que recoge los datos del formulario anterior, y realiza una consulta contra la BD (contra la tabla “usuarios”) para verificar el login. Si es correcto muestra un mensaje de bienvenida, y si es incorrecto muestra un mensaje de error.

Puede probarse el correcto funcionamiento del código con el usuario “prueba” y la contraseña “password”. El código PHP que se muestra es vulnerable a un ataque de inyección SQL, ya que si introducimos en el campo contraseña una cadena similar a:

```
' or '1'='1
```

La sentencia SQL quedaría entonces :

```
SELECT login FROM usuarios WHERE login ='loquesea' AND
            password ='' or '1'='1';
```

Por lo que podremos saltarnos el mecanismo de autenticación establecido.

**NOTA:** Este ataque ha sido posible ya que PHP5 tiene por defecto activadas las “magic_quotes”, que escapan de forma automática todos los valores recogidos por los métodos POST y GET. Aunque es una excelente medida de seguridad, no afecta a datos leídos de ficheros u obtenidos por otros medios.

### *Obtener información de la BD*

En este otro ejemplo tenemos los siguientes ficheros:

- select_producto.htm : Formulario HTML que permite al usuario realizar una búsqueda de productos en la BD (dentro de la tabla “productos”).

- Busca.php : Programa en PHP que busca en la tabla de productos con el parámetro del formulario anterior. Si la consulta da algún resultado lo muestra por pantalla, y si no da resultados produce un mensaje de error.

Puede probarse el correcto funcionamiento del código con la cadena de búsqueda "libros". Este código PHP es vulnerable también a un ataque de inyección SQL, ya que si introducimos una cadena de búsqueda similar a :

```
' or 1='1
```

La sentencia SQL quedaría entonces :

```
SELECT nombre, precio FROM productos WHERE nombre ='' or
1='1';
```

Podremos alterar el resultado de la consulta, pudiendo obtener obtener todos los productos de la BD. Es más, podríamos obtener datos de otra tabla si modificamos un poco más la consulta empleando el comando SQL UNION:

```
' union select * from usuarios where login like '%
```

La sentencia SQL quedaría entonces :

```
SELECT nombre, precio FROM productos WHERE nombre ='' union
       select * from usuarios where login like '%';
```

**Nota**: La función mysql_query() solo admite una consulta SQL por ejecución, por lo que desde PHP solo podemos hacer una única consulta (no es posible entonces hacer algo similar a "SELECT * from tabla; SHOW TABLES; SELECT * from tabla2";).

### *Enlaces de interés*

Descripción de la Inyección SQL
http://www.unixwiz.net/techtips/sql-injection.html

Wikipedia – SQL Injection
http://en.wikipedia.org/wiki/SQL_injection

PHP – mysql_real_escape_string
http://au2.php.net/manual/es/function.mysql-real-escape-string.php

## *Cross Side Scripting (XSS)*

El XSS (Cross Side Scripting) se basa en la capacidad de los servidores de aceptar código ejecutable dentro de una entrada de parámetros, código que luego puede ser ejecutado de forma involuntaria por otros clientes.

### *Insertar JavaScript malicioso en una página web*

En el tercer ejemplo tenemos los siguientes ficheros :

- comenta.htm: Formulario en HTML que permite comentar un producto, introduciendo el comentario en la BD (en la tabla de productos ).

- Sube_comentario: Programa en PHP que recoge el comentario y crea una consulta SQL que lo introduce en la BD.

Puede probarse el programa con el producto "producto1", introduciendo el comentario que se desee (los comentarios adicionales se van mostrando de forma sucesiva). Este programa es vulnerable a un ataque de XSS, ya que es posible introducir código JavaScript con la etiqueta HTML <SCRIPT> de la forma siguiente:

<SCRIPT>alert(\'Te pille!\')</SCRIPT>

Si se recarga la página, se podrá ver el resultado con facilidad. Podría en principio ejecutarse cualquier otro código JavaScript que se deseara (incluso desde otra página web con algo similar a: <SCRIPT SRC=http://miweb/xss.js></SCRIPT>)

### *Enlaces de interés*

FAQ sobre XSS
http://www.cgisecurity.com/articles/xss-faq.shtml

Wikipedia – Cross-Site Scripting
http://en.wikipedia.org/wiki/Cross_site_scripting

PHP – Función htmlentities()
http://es2.php.net/htmlentities

## Las redes sociales y el comercio electrónico

La irrupción de la web 2.0 trajo como mayor consecuencia la "humanización" de la web. Hasta entonces el usuario era un elemento pasivo que utilizaba la web solamente para recibir información. Con la llegada de los blogs y las redes sociales la web se convirtió en un vertedero de información proveniente de todo el mundo, pero también de todo tipo de personas.

Entre esta información proveniente de la web se encontraban las opiniones de los clientes de negocios tanto electrónicos como los clásicos. Estas opiniones se aglutinaban y creaban corrientes de opinión que podían desencadenar que un producto o una tienda pudiera tener tanto un crecimiento como un decrecimiento acelerado y repentino.

Esto no pasó inadvertido para las empresas que de repente se encontraron con otro mecanismo de interacción con su clientela y aparecieron los "community manager" empleados de las mismas para canalizar toda esa información de forma bidireccional.

Por su parte, las redes sociales se encontraron con la fuente de financiación de las empresas que querían alojar su publicidad e invitaban a las redes a albergar las llamadas de atención que salían publicadas en las páginas de sus componentes. La cosa no acabó ahí, sino que además las empresas consiguieron establecer negocios dentro de las aplicaciones de las redes sociales y nació lo que se ha dado en llamar el "f-commerce".

Además, las redes empezaron una vertiginosa carrera para ofrecer servicios añadidos que hicieran más atractivos sus contenidos para sus componentes, y en consecuencia para las empresas que querían hacer uso de ellas. Uno de esos servicios fue la delegación de la identificación.
La idea no era nueva. Ya existían webs que se dedicaban a ese servicio (quizás la más conocida era OpenId), pero en este caso las redes sociales tenían como ventaja la confianza (casi ciega) de sus miembros y una alta participación de ellos en la misma.

## Una app de identificación de Facebook

Sin duda, la red social con mayor impacto a nivel mundial es Facebook. Por eso no es de extrañar que fuera una de las primeras que dio la posibilidad de que sus usuarios pudiesen identificarse en otras webs con las herramientas propias de Facebook (aparte de la apetecible información que consigue de sus miembros con la provisión del servicio).
Actualmente esta red ofrece varias posibilidades de autenticación, dependiendo de la tecnología que se quiera emplear, la seguridad que se quiera suministrar y el grado de compatibilidad que se quiera alcanzar.
En http://developers.facebook.com/docs/concepts/login/login-architecture/ podemos las diferentes arquitecturas que ofrece facebook. Nosotros elegiremos la más sencilla. Antes de describir el proceso debemos darnos de alta en Facebook. Una vez hecho esto nos podemos conectar a:

http://developers.facebook.com/docs/howtos/login/getting-started/

y seguir los pasos que nos proponen incrustando código Javascript en la página HTML donde queremos insertar el servicio de autenticación.
En el cuerpo del código HTML deberíamos insertar este script:

```
<div id="fb-root"></div>
<script>
  // Additional JS functions here
  window.fbAsyncInit = function() {
    FB.init({
      appId      : 'YOUR_APP_ID', // App ID
      channelUrl : '//WWW.YOUR_DOMAIN.COM/channel.html', // Channel
File
      status     : true, // check login status
      cookie     : true, // enable cookies to allow the server to
access the session
      xfbml      : true  // parse XFBML
    });

    // Additional init code here

  };

  // Load the SDK Asynchronously
  (function(d){
     var js, id = 'facebook-jssdk', ref =
d.getElementsByTagName('script')[0];
     if (d.getElementById(id)) {return;}
     js = d.createElement('script'); js.id = id; js.async = true;
     js.src = "//connect.facebook.net/en_US/all.js";
     ref.parentNode.insertBefore(js, ref);
   }(document));
</script>
```

No debemos olvidar que tenemos que añadir el fichero channel.html en el que sólo será obligatoria la siguiente línea:

```
<script src="//connect.facebook.net/en_US/all.js"></script>
```

Añadiremos el siguiente código para controlar el valor de la función **FB.getLoginStatus** y nos dice si el usuario se ha autenticado o no a través de la variable response.status.

```
FB.getLoginStatus(function(response) {
  if (response.status === 'connected') {
    // connected
  } else if (response.status === 'not_authorized') {
    // not_authorized
  } else {
    // not_logged_in
  }
 });
```

Si queremos dar la posibilidad de autenticarse a aquel usuario que no lo esté debemos añadir este código cuando sea oportuno:

```
function login() {
    FB.login(function(response) {
        if (response.authResponse) {
            // connected
        } else {
            // cancelled
        }
    });
}
```

## *Trabajo a realizar*

Una de las soluciones que pueden prevenir o al menos mitigar los ataques de inyección SQL es escapar los caracteres peligrosos antes de introducirlos en la consulta. En PHP esto puede realizarse con la función mysql_real_escape_string()

- Modificar el código anterior para que no sea vulnerable a la inyección SQL empleando la función mysql_real_escape_string().

- Usando la documentación de los enlaces que se facilitan en la práctica, proponer (que no codificar) otra solución que prevenga este tipo de ataques.

Para prevenir el ataque XSS es posible escapar el uso de caracteres extraños, siendo esto posible en PHP empleando la función: htmlentities().

- Modificar el código anterior empleando la función anteriormente descrita para que no sea vulnerable a un ataque de XSS.

De manera opcional se propone un plus para esta práctica consistente en:

- Ofrecer un servicio de identificación en facebook, en el que se le anuncie al cliente si ha sido identificado o no como usuario de facebook.