VULNERABILIDADES EN PROGRAMAS Y APLICACIONES: PARTE 2

PROGRAMACIÓN SEGURA EN ENTORNOS WEB

Autor: Brais López Yáñez

Objetivo de la práctica

El objetivo de esta sesión es aprendera realizar las pruebas de penetración más comunes en el ámbito web. Concretamente, al finalizar la práctica el alumno será capaz de identificar (¡y explotar!) las siguientes vulnerabilidades:

Suplantación de identidad mediante la modificación de cookies.

- Inyección SQL.
- Crackear contraseñas hasheadas.
- Creación de una webshell empleando un sistemas de subida de archivos vulnerable.
- Cross-Site Scripting.

Para la realización de estas prácticas emplearemos los ejercicios y máquinas virtuales disponibles en la página http :// pentesterlab. com.

Ejercicio 2.1: From SQL injection to Shell

En este ejercicio se empleará inyección SQL para obtener el usuario y contraseña del administrador de un sitio web vulnerable. Una vez se ha hecho login como administrador de la aplicación, el alumno explotará una vulnerabilidad relacionada con la subida de archivos para ejecutar código de forma remota.

El guión y la máquina virtual a emplear en el ejercicio se encuentran en la web: http :// pentesterlab. com/ exercises/ from _ sqli _ to _ shell/.

Para realizar los ataques el alumno puede emplear cualquier máquina con conectividad con la máquina vulnerable. Creemos que la mejor opción es emplear la máquina anfitrión como máquina atacante. Para que la máquina anfitrión tenga acceso al servidor web de la máquina virtual configuraremos esta última en modo NAT. En las opciones avanzadas hacemos click en Port Forwarding y añadimos una nueva regla. Por ejemplo:

De esta forma, cuando las peticiones a la máquina anfitrión vayan dirigidas al puerto 8888, estas se redirigirán a la máquina virtual al puerto 80.

Notas sobre el ejercicio 2.1

En la sección “Cracking the password” se afirma que “un hash sin sal puede ser fácilmente crackeado …” ¿Qué significa en este contexto un hash “salado”?¿Cómo evita “ la sal” que se rompa una contraseña?

Realización de la práctica

Para introducirnos en este ejercicio deberíamos explicar la afirmación "un hash sin sal puede ser fácilmente crackeado...". Esta afirmación se basa en que si un hacker, puede hacer consultas SQL (inyección sql) en una base de datos, y este conoce las contraseñas generalmente usadas, como pueden ser: 1234,mundo, etc.

Y hace el proceso inverso de loguearse, genera un diccionario con estas contraseñas cifradas en MD5 y comprueba si existen usuarios en la base de datos con esa contraseña. Si existen, ya tiene acceso como un usuario más.

```
SELECT username
FROM UserTable
WHERE password = '#KH##NM@MM@M'
```

Para evitar esto, lo que se usa es un hash "salado", que es lo que va a solventar el anterior problema. El método consiste en añadir campos únicos para cada usuario a la tabla hash, como resultado de los campos únicos la encriptación va a ser diferente para cada tipo de usuario, ya que la función hash que se va a usar utilizará diferentes valores de datos. Esto va a dificultar la labor del hacker, pasando de una tarea fácil a una de alta complejidad.

Como ejemplos de hash salado, serían: añadir los campos de fecha de nacimiento, nombre, apellidos, nacionalidad, ciudad en la que vives, etc.

Con esto se logrará crear una encriptación única para cada usuario.

Una vez introducidos en el mundo de la inyección SQL, procedemos a la realización del ejercicio:

Después de examinar toda la página, empezamos a darnos cuenta que podemos acceder a la base de datos con consultas SQL, en la url de las páginas donde aparezca id=1, ahí es nuestra oportunidad para intentar una consulta.

Tras ver que nos sale un error SQL, de que no podemos acceder a la base de datos, nos damos cuenta de que estamos en lo cierto, que se pueden hacer consultas. Por lo tanto procedemos a una etapa de intentar probar consultas que tengan lógica y que nos puedan servir de algo.

Tras varios intentos fallidos, logramos obtener algo que nos va a ser de mucha utilidad:

```
UNION SELECT 1,concat(login,':',password),3,4 FROM users;
```

picture: admin:8efe310f9ab3efeae8d410a8e0166eb2

Hemos obtenido, de la base de datos, un usuario con su contraseña cifrada. Para ver qué tipo de encriptación se trata buscamos en Google, la información más relevante que nos ayude a descifrar la contraseña.

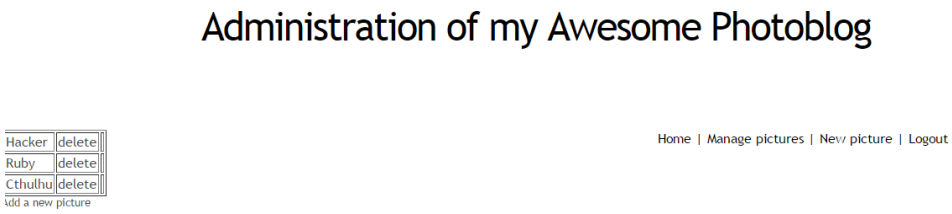
Al poco tiempo, encontramos una página que "desencripta" MD5, esta página lo que hace es comparar la información que ya existe y se concuerda con nuestra contraseña:

La página empleada:

http://md5.gromweb.com/?md5=8efe310f9ab3efeae8d410a8e0166eb2

Contraseña obtenida: P4ssw0rd

Tras la obtención de la contraseña, lo que haremos será loguearnos, para comprobar que ha sido un éxito:

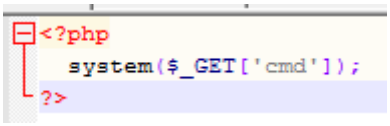


Ya en el panel de administración, vemos que hay imágenes subidas, y lo que vamos a hacer ahora es probar a subir algo, para ver si podemos ejecutar código dentro de los ficheros que subimos:

Tras hacer algún que otro intento vemos que si se puede hacer algo si subimos algún fichero. Pero no con la extensión php, ya que Apache protege contra esta extensión, pero no contra php3.

Y probamos con los siguientes archivos:

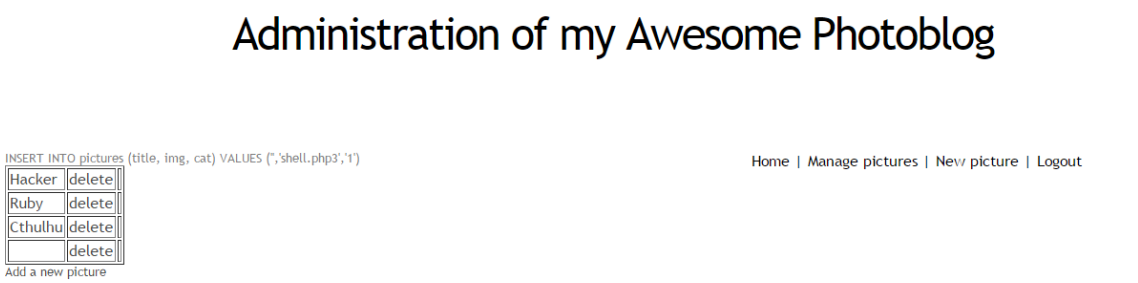
Para ejecutar los comandos:



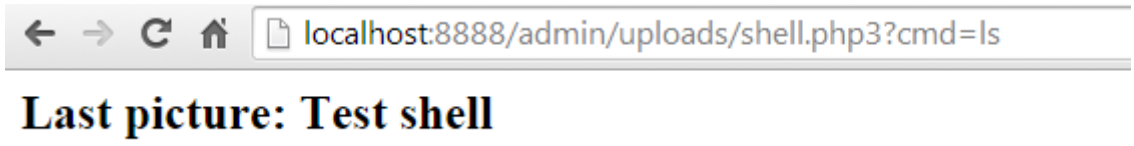
Una vez, sabemos que van los scripts, subimos el ejemplo de recomendación de pentesterlab. com., que sería en php3 para ver cuál es la ruta de nuestro archivo (y poder así ejecutar comandos):

```
<div class="content">
    <h2 class="title">Last picture: Test shell</h2>

    <div class="inner" align="center">
      <p>
        <img src="admin/uploads/shell.php3" alt="Test shell" /> </p>
    </div>
  </div>
```



Tras subir los 2 archivos, sin problemas, nos disponemos a acceder a la ruta propuesta:



Y acabamos el ejercicio verificando, que podemos introducir comandos en linux desde la url del navegador. Lo hemos conseguido gracias, a los dos archivos subidos anteriormente, que nos permitieron introducir código al navegador.

Finalizamos la práctica concluyendo que la seguridad está presente en todos los aspectos de un programador, desde las tareas más sencillas a las más complejas. Con los lenguajes de alto nivel, tenemos un mayor número de posibilidades de crear elementos pero también es más fácil que nos equivoquemos y dejemos una puerta abierta a esta serie de ataques. Realizar este tipo de pruebas puede ser muy beneficioso a la hora de testear el nivel de seguridad de nuestros códigos.