INGENIERIA INFORMATICA
Escuela Politécnica Superior
Universidad Autónoma De Madrid

Ciberseguridad

Atacando una aplicación web

Práctica 3

David Teófilo Garitagoitia Romero
Daniel Cerrato Sánchez

20/04/2023

Índice de Contenidos

Lista de Figuras

1. Introducción

El objetivo de esta práctica es conocer los distintos métodos de ataque a una aplicación para paliar y evitar dichos errores de diseño y codificación en el futuro. Es importante conocer estos puntos de seguridad, puesto que son bastante comunes a la hora de crear aplicaciones de esta índole.

Para esta práctica haremos uso de un entorno preparado proporcionado por la universidad, en la que se presta una máquina virtual que hará de host para el servicio de la aplicación web "bWAPP".

Esta web contiene gran cantidad de diseños de página web, con diversas funcionalidades a explorar y explotar. Además, ofrece un sistema de dificultad para los más curiosos, atrevidos o hambrientos de conocimiento y práctica.

Este sistema se importa directamente desde un archivo .ova a VirtualBox, se ejecuta y se deja corriendo durante la práctica. Simplemente necesitaremos obtener la IP que tiene la máquina para poder acceder a ella desde el navegador.

```
bee@bee-box:~$ ifconfig
eth1      Link encap:Ethernet  HWaddr 08:00:27:4b:b6:35
          inet addr:192.168.56.101  Bcast:192.168.56.255  Mask:255.
          inet6 addr: fe80::a00:27ff:fe4b:b635/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1188 (1.1 KB)  TX bytes:8818 (8.6 KB)
          Base address:0xd010 Memory:f0200000-f0220000
```

Ilustración 1 - IP de la máquina servidor web

Como se observa en la imagen superior, la IP del host es **192.168.56.101**, así que en nuestro navegador buscaremos la siguiente url: *http://192.168.56.101/bWAPP/login.php*

Por otra parte, usaremos una herramienta llamada "BurpSuite", la cual nos permitirá capturar y mostrar los mensajes entre el navegador y el servidor web, así como un "Man in the Middle". Esta herramienta será necesaria para llevar a cabo algunos de los ejercicios a realizar en esta práctica.

2.HTML Injection Reflected

2.1. GET

En este ejercicio aparecen dos campos a rellenar (Nombre y apellido) y un botón de acción. Si escribimos cualquier cosa en estos campos y enviamos la respuesta con el botón, vemos que lo que se ha escrito aparece escrito en la página.

Si capturamos los mensajes, podemos observar que las respuestas van escritas directamente en la cabecera del mensaje, lo que permite modificar esos valores si se captura; y que, como mensaje de respuesta, se envía un HTML donde los valores de los campos se escriben según se reciben y en texto plano (imágenes a continuación)

```
1  GET /bWAPP/htmli_get.php?firstname=hola&lastname=mundo&form=submit
   HTTP/1.1
2  Host: 192.168.56.101
3  Upgrade-Insecure-Requests: 1
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
   ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6  Referer: http://192.168.56.101/bWAPP/htmli_get.php
7  Accept-Encoding: gzip, deflate
8  Accept-Language: es-ES,es;q=0.9
9  Cookie: PHPSESSID=8ed86f3f5ff45ab775eb8c0d586f1878; security_level=0
10 Connection: close
```

Ilustración 2 - HTML Injection Reflected GET Request

```
78        </form>
79
80        <br />
81        Welcome hola mundo
82      </div>
83
84      <div id="side">
```

Ilustración 3 - HTML Injection Reflected GET Response

Esto nos permite insertar código en los campos para ejecutar algún script, por ejemplo.

Ilustración 4 - HTML Injection Reflected GET Introduciendo script

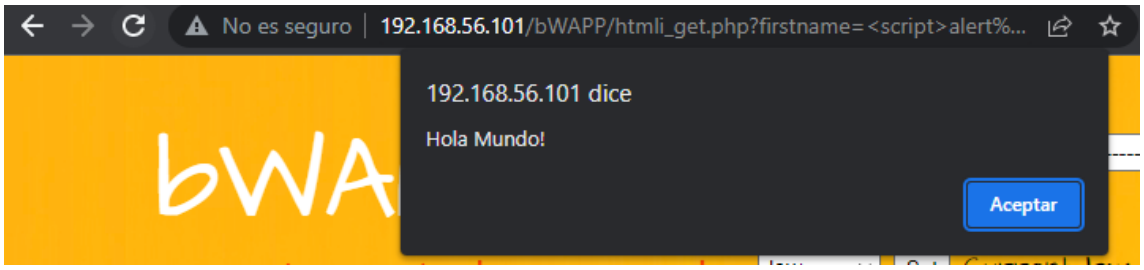

Ilustración 5 - HTML Injection Reflected GET Resultado Script

2.2. POST

En este apartado, aparece el mismo diseño de página. Volvemos a capturar los mensajes y esta vez, como se trata de un POST, la información de los campos ya no está en la cabecera del mensaje, sino en el cuerpo. La respuesta sigue siendo un HTML con las respuestas en plano. (Ver Ilustración 3)

```
POST /bWAPP/htmli_post.php HTTP/1.1
Host: 192.168.56.101
Content-Length: 41
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.56.101
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.56.101/bWAPP/htmli_post.php
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9
Cookie: PHPSESSID=8ed86f3f5ff45ab775eb8c0d586f1878; security_level=0
Connection: close

firstname=hola&lastname=mundo&form=submit
```

Ilustración 6 - HTML Injection Reflected Post Request

Por lo que probamos exactamente lo mismo que en el apartado anterior.

Ilustración 7 - HTML Injection Reflected POST Introduciendo script

Como esperábamos, el resultado fue el mismo que en el apartado anterior. (Ver Ilustración 5)

2.3. CURL

En el último apartado de este ejercicio, accedemos a la página que nos muestra la URL actual en la que está el navegador. De nuevo, capturamos el mensaje que envía el servidor y comprobamos que, en la cabecera, existe un campo llamado "Host" donde se indica la URL o la IP del host del servicio web.

Haciendo uso de la herramienta BurpSuite, tras capturar el mensaje, modificamos este campo.

```
GET /bWAPP/htmli_current_url.php HTTP/1.1
Host: PuedoCambiarElHost.es
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/111.0.5563.111 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.56.101/bWAPP/portal.php
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9
Cookie: PHPSESSID=75467e93a4d26aca4a9357caed116182; security_level=0
Connection: close
```

Ilustración 8 - HTML Injection Reflected CURL Request

Y el resultado es el siguiente:

Ilustración 9 - HTML Injection Reflected CURL Resultado

3.SQL Injection

3.1. GET/Search

En este apartado, se nos presenta un campo donde se espera que escribamos el nombre de una película y accionamos la búsqueda en base de datos a través de un botón de acción.

Suponemos que la búsqueda tendrá un formato como el siguiente: *select [atributos] from* ***pelis*** *where* ***title*** *like '%**[entrada]**%'*; por lo que probamos a introducir la típica sentencia para estos casos: *' [sentencia] #*. A través de la sentencia, si no se comprueba el formado de la entrada, podemos llegar a explorar la base de datos e incluso añadir o eliminar información.

En nuestro caso, probaremos a explorar la base de datos. Para ello, primero necesitamos saber cuántas columnas está sacando de la consulta, para poder realizar una unión con otra consulta propia. La sentencia a introducir es: *kkk' union select 1,2,3,4,5,6,7'*. Es necesario que se introduzcan siete columnas a recibir para que la consulta no falle e introducimos "kkk" al inicio para que la primera consulta no devuelva nada, básicamente poner un título de película que no tenga la base de datos.

Ilustración 10 - SQL Injection GETSearch Pruebas iniciales

Como podemos ver, de la consulta solo se muestran las columnas 2, 3, 4 y 5; las cuales usaremos para mostrar la información que queramos obtener de la base de datos.

A continuación, buscamos el nombre de la base de datos con la siguiente sentencia: *kkk' union select 1,database(),1,1,1,1'*

Ilustración 11 - SQL Injection GETSearch Database

Una vez conocemos el nombre de la base de datos, podemos buscar las tablas que contiene la base de datos. Por ejemplo, podríamos intentar conocer la tabla donde se guardan los datos de los usuarios. Para ello, introducimos una consulta como la siguiente: *kkk' union select 1,table_name,1,1,1,1,1 from information_schema.tables #*

En la imagen siguiente vemos que existe una tabla llamada "users", entre otras tablas.

movies	1	1	1	Link
users	1	1	1	Link
visitors	1	1	1	Link

Ilustración 12 - SQL Injection GETSearch Buscando tablas

Una vez sabemos esto, usamos la siguiente sentencia para obtener las columnas de esta tabla: *kkk' union select 1,column_name,1,1,1,1,1 from information_schema.columns where table_name='users' #*

Title	Release	Character	Genre	IMDb
id	1	1	1	Link
login	1	1	1	Link
password	1	1	1	Link
email	1	1	1	Link
secret	1	1	1	Link
activation_code	1	1	1	Link

Ilustración 13 - SQL Injection GETSearch Tabla Users

En la imagen anterior se muestran algunas columnas obtenidas de la última consulta. De modo que usaremos una última consulta para obtener información clave de los usuarios de la base de datos. Esta consulta es: *kkk' union select 1,login,password,secret,email,1,1 from users #*

Title	Release	Character	Genre	IMDb
A.I.M.	6885858486f31043e5839c735d99457f045affd0	bwapp-aim@mailinator.com	A.I.M. or Authentication Is Missing	Link
bee	6885858486f31043e5839c735d99457f045affd0	bwapp-bee@mailinator.com	Any bugs?	Link

Ilustración 14 - SQL Injection GETSearch Datos usuarios

3.2. SQLite

Para este apartado se nos presenta una página prácticamente igual a la anterior, así que empezamos como en el anterior: primero es necesario saber cuántas columnas se están pidiendo en la consulta para que no falle. En este caso son seis columnas.

Como ya conocemos la base de datos, buscamos la misma información que en el apartado anterior pero con la consulta modificada: *kkk' union select 1,login,password,secret,email,1 from users --*

Es necesario remarcar que en el apartado anterior había que terminar las sentencias con una almohadilla (#) puesto que es el símbolo que se usa para iniciar los comentarios en MySQL. En este caso se está usando SQLite, por lo que el símbolo de comentario es el doble guion (--).

Como añadido, buscamos la versión de SQLite usada a través de la siguiente sentencia: *kkk' union select 1,2,sqlite_version(),4,5,6 --*

Title	Release	Character	Genre	IMDb
2	3.4.2	5	4	Link

Ilustración 15 - SQL Injection SQLite Version SQLite

3.3. Stored (Blog)

En este apartado, se muestra una página en la que podemos introducir texto para una entrada de blog, que se guardará en base de datos usando el botón de acción. Como en casos anteriores, si introducimos código de script, este se ejecutará al recargar la página; pero como esta vez el texto introducido se guarda en base de datos, cada vez que se muestren las entradas de blog de esta página, se ejecutará el script.

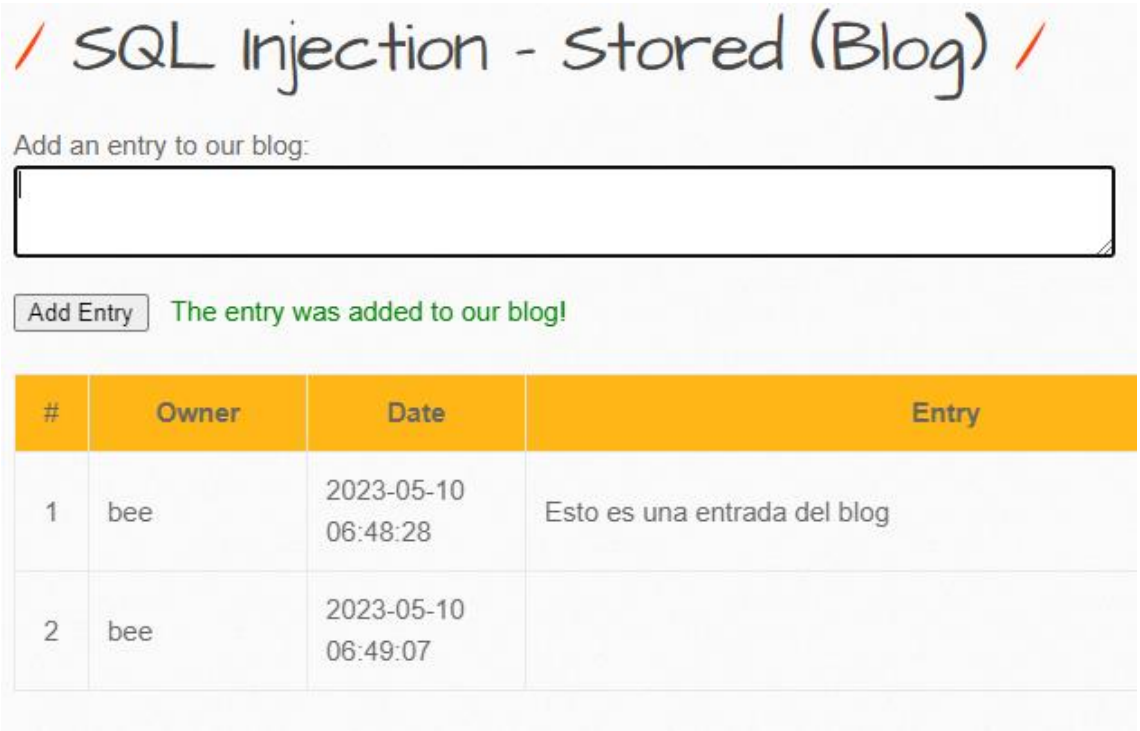

Ilustración 16 - SQL Injection Stored (Blog) Scripting

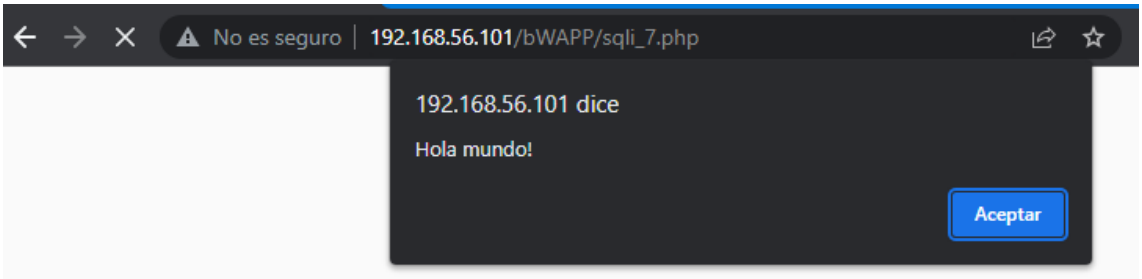

Ilustración 17 - SQL Injection Stored (Blog) Resultado

Como se ve en la Ilustración 16, la segunda entrada está vacía pero realmente contiene el código que ejecuta el script que tiene como resultado la Ilustración 17.

Pero estamos tratando la inyección SQL, así que podemos realizar consultas como las anteriores y obtener la misma información, solo que cambiando el formato de la consulta:

1', (select group_concat(login,password) from users where login="bee"))#

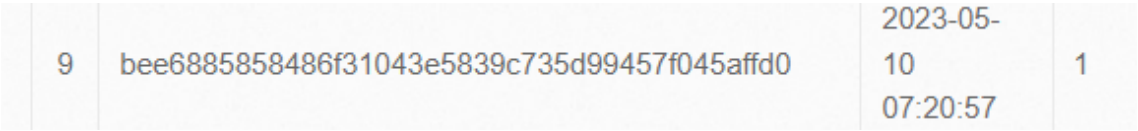

Ilustración 18 - SQL Injection Stored (Blog) Informacion Bee

En este caso, necesitamos usar *group_concat* para que nos muestre todo en una sola línea, ya que estamos "añadiendo" una nueva entrada de blog. Otra forma es usando *limit 1*.

4. Authentication

4.1. CAPTCHA Bypassing

En este nuevo apartado, nos muestran una página bastante común a la hora de hacer *login* en alguna página web. Esta contiene dos campos para usuario y contraseña, pero a continuación, se muestra una imagen CAPTCHA y el campo correspondiente para rellenar con los caracteres de la imagen.

La idea de este apartado es saltarse el CAPTCHA para poder hacer *login*. Para ello, pulsamos F12, accedemos a "Aplicación" y luego a "Cookies". Escogemos la cookie que estamos usando en esta página y la borramos.

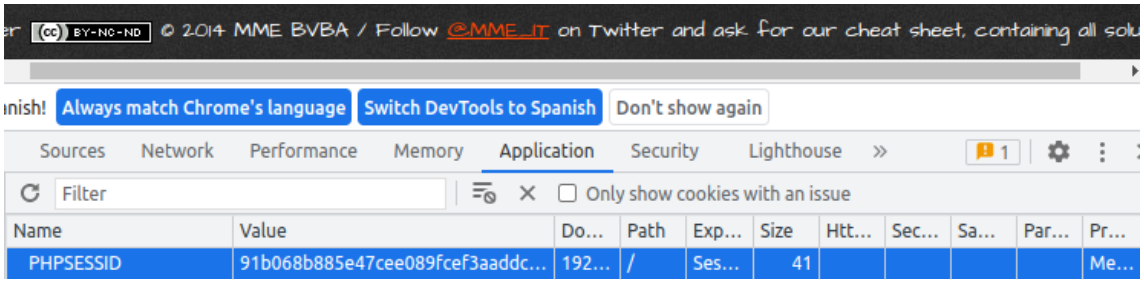

Ilustración 19 - CAPTCHA Borrar cookie

A continuación, iniciamos sesión de nuevo. Ahora, usando la herramienta BurpSuite, capturamos los mensajes enviados y volvemos a la página del CAPTCHA. La request la dejamos pasar, pero tiramos la reponse. Esta contiene la información del CAPTCHA y el botón para recargar la imagen.

Ilustración 20 - CAPTCHA Tras tirar el segundo response

Llegados a este punto, mantenemos las capturas de mensajes y enviamos el formulario relleno con datos correctos. Ahora, en la herramienta BurpSuite eliminamos el campo “captcha_user” del cuerpo del POST:

```
POST /bWAPP/ba_captcha_bypass.php HTTP/1.1
Host: 192.168.56.101
Content-Length: 48
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.56.101
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/si
gned-exchange;v=b3;q=0.7
Referer: http://192.168.56.101/bWAPP/ba_captcha_bypass.php
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9
Cookie: security_level=0; PHPSESSID=8cca1c8a3ed7bedc6ac0ab11e443eafe
Connection: close

login=bee&password=bug&captcha_user=&form=submit
```

Ilustración 21 - CAPTCHA Eliminar campo captcha_user

Por último, dejamos pasar el mensaje y hecho, conseguimos entrar.

Ilustración 22 - CAPTCHA Login conseguido

4.2. Session ID en URL

En esta página, se muestra un mensaje escrito de advertencia, pero lo interesante está en la URL del buscador. Se puede ver que en la URL se indica la ID de sesión del usuario en cuestión.

Para este apartado vamos a crear un usuario nuevo en un navegador distinto y accedemos a la página de este apartado para observar cuál es la ID de sesión. Podemos guardarla para comparar luego y comprobar que todo es correcto, pero no es necesario.

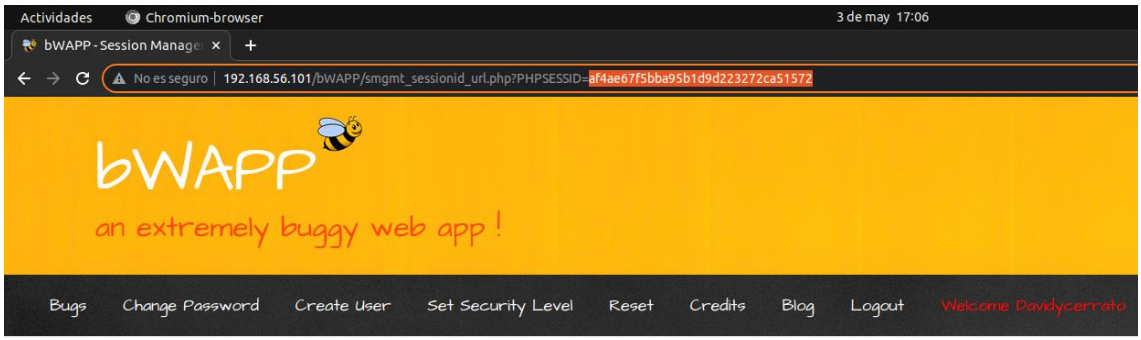

Ilustración 23 - ID en URL ID del usuario nuevo

La ID de sesión del usuario “Bee” sí que es necesario guardarla. En un escenario real, suponemos que ya tendríamos esa ID copiada.

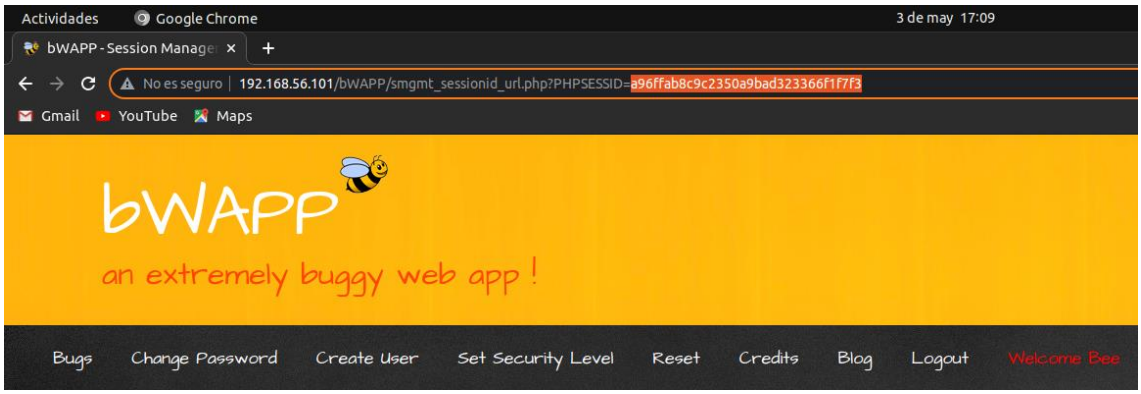

Ilustración 24 - ID en URL ID de Bee

En la sesión del usuario nuevo, vamos a acceder a la página de cambio de contraseña. Antes de enviar el formulario, capturaremos los mensajes con la herramienta BurpSuite. Una vez enviado el formulario de cambio de contraseña, en BurpSuite cambiaremos el campo "PHPSESSID" con ID de Bee. Podemos cambiar el de abajo o los dos que aparecen en la cabecera del mensaje como se ve en la imagen siguiente:

```
GET /bWAPP/smgmt_sessionid_url.php?PHPSESSID=a96ffab8c9c2350a9bad323366f1f7f3
HTTP/1.1
Host: 192.168.56.101
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/111.0.5563.111 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.56.101/bWAPP/password_change.php
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9
Cookie: security_level=0; PHPSESSID=a96ffab8c9c2350a9bad323366f1f7f3 Obligatorio
Connection: close
```

Ilustración 25 - ID en URL Cambiando ID

Una vez cambiado, enviamos y podemos observar como el servidor piensa que somos Bee, ya que muestra le mensaje "Welcome Bee" en vez de "Welcome [usuario nuevo]"

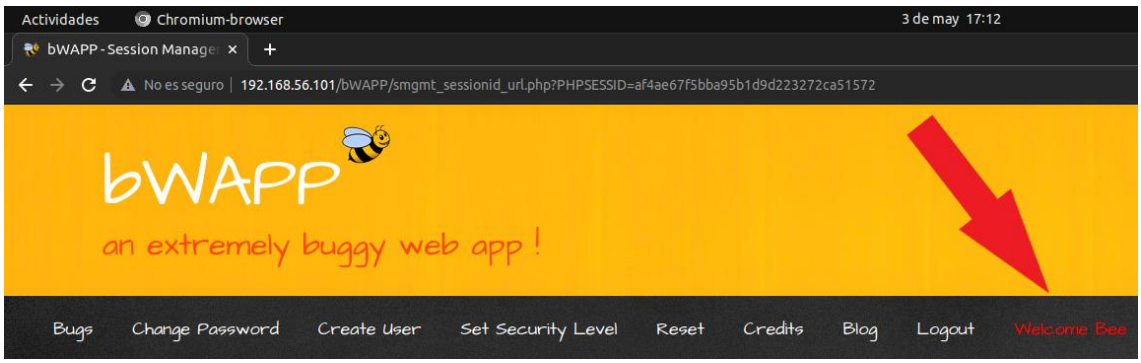

Ilustración 26 - ID en URL Resultado

4.3. Cookies (HTTPOnly)

Este apartado es muy simple pero muy instructivo. Se presenta una página en la que podemos clicar un botón que nos muestra la información de nuestra cookie en una tabla y un enlace que ejecuta un script de JavaScript que pretende hacer lo mismo, pero a través de un "alert".

Si pulsamos F12, vamos a "Aplicación", luego a "Cookies" y desplegamos nuestra cookie, podemos ver que no está activa la opción "HTTPOnly". Si esto no está activo en ningún dato, la página se muestra así:

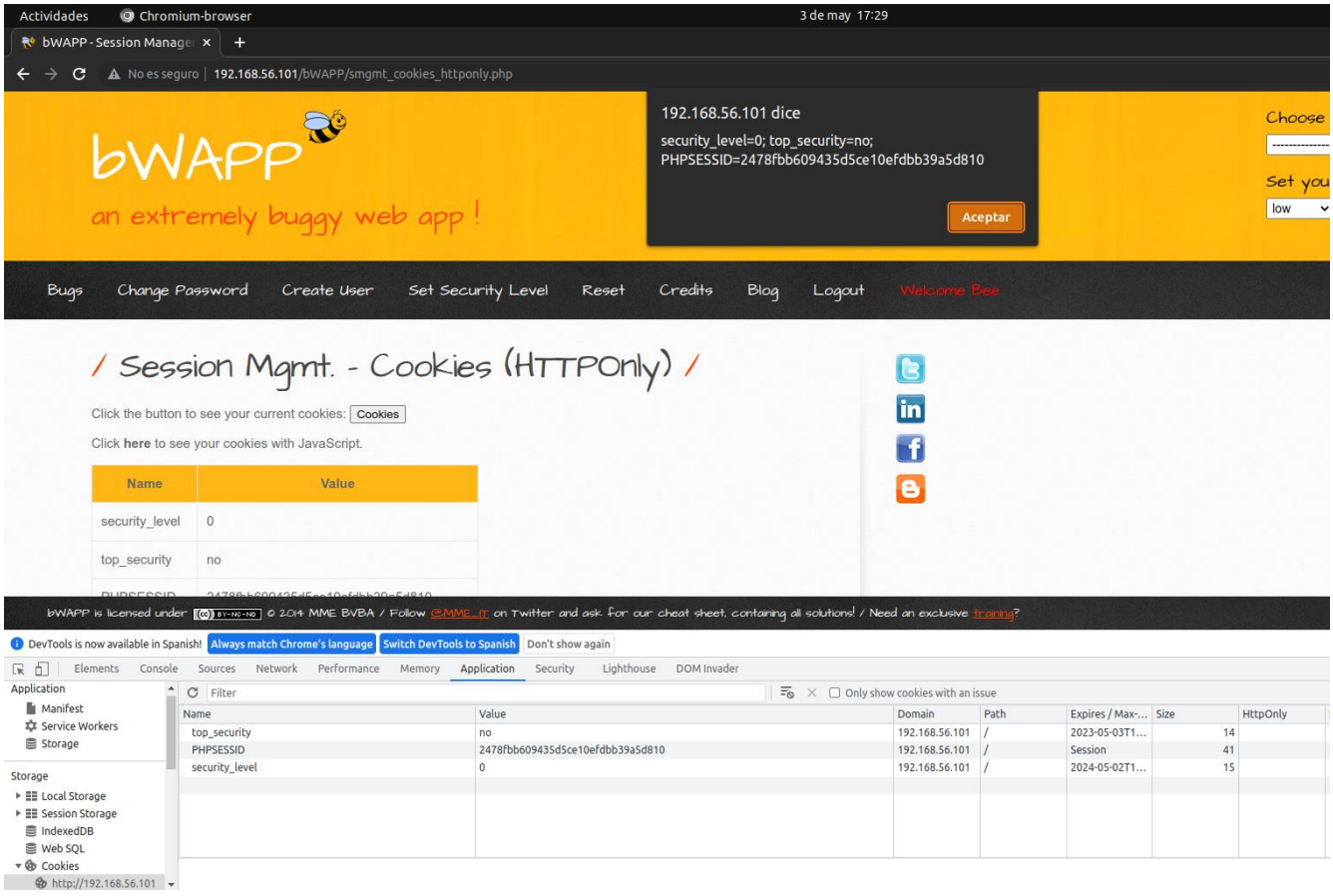

Ilustración 27 - Cookies HTTPOnly No activo

Pero si activamos la casilla de HTTPOnly de algunos o todos los datos, la página se ve así:

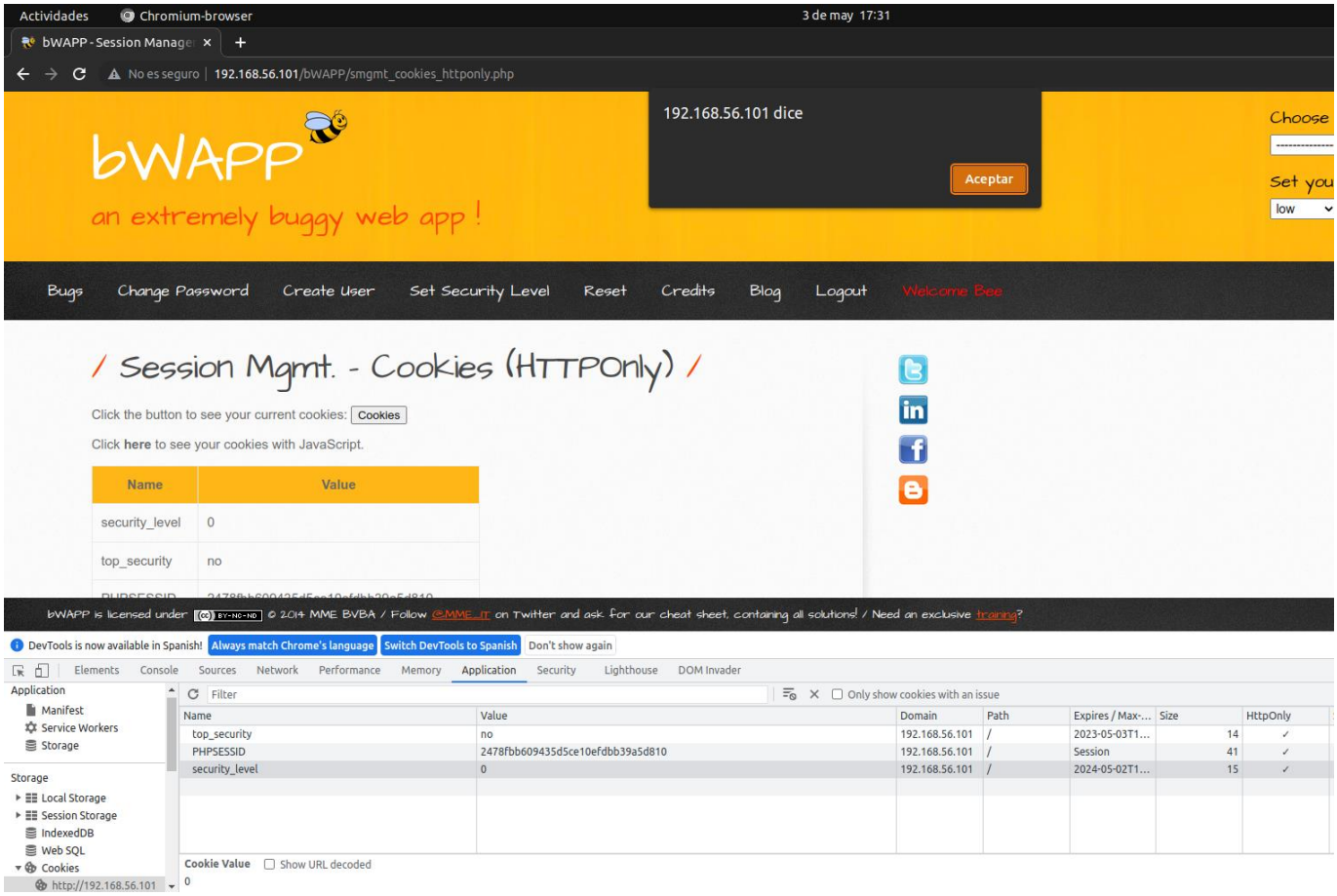

Ilustración 28 - Cookies HTTPOnly Activado

Se puede observar que la tabla sigue mostrando los datos de la cookie sin problema alguno, mientras que el mensaje de alerta generado con JavaScript no muestra nada. Esto es así porque al activar el modo HTTPOnly, los datos de las cookies solo son accesibles a través de las peticiones con protocolo HTTP.

4.4. Directory Traversal

4.4.1. Directories

En este apartado, la página muestra una lista de nombres de archivos PDF convertidos a enlace. Si accedemos a alguno de ellos, podemos ver su contenido.

Lo curioso está de nuevo en la URL del buscador. En la URL existe un campo "directory" que inicialmente tiene el valor "documents". Fácilmente se puede llegar a la conclusión de que ese valor es el nombre de un directorio del servidor y que la página nos está mostrando su contenido como si usásemos el comando "ls" en Linux.

Ilustración 29 - Directories Ruta en URL

Por lo que también es fácil pensar que podemos modificar ese valor para acceder a otros directorios. Por ejemplo, intentar ir al directorio raíz:

Ilustración 30 - Directories Directorio Raíz

Esto nos puede servir para acceder al servidor a través de una puerta trasera que hayamos conseguido crear en algún momento. Desde el directorio de documentos inicial hacia atrás, investigamos un poco los directorios y su contenido. Simplemente yendo al directorio padre de "documents" podemos observar que existe un enlace llamado "backdoor.php".

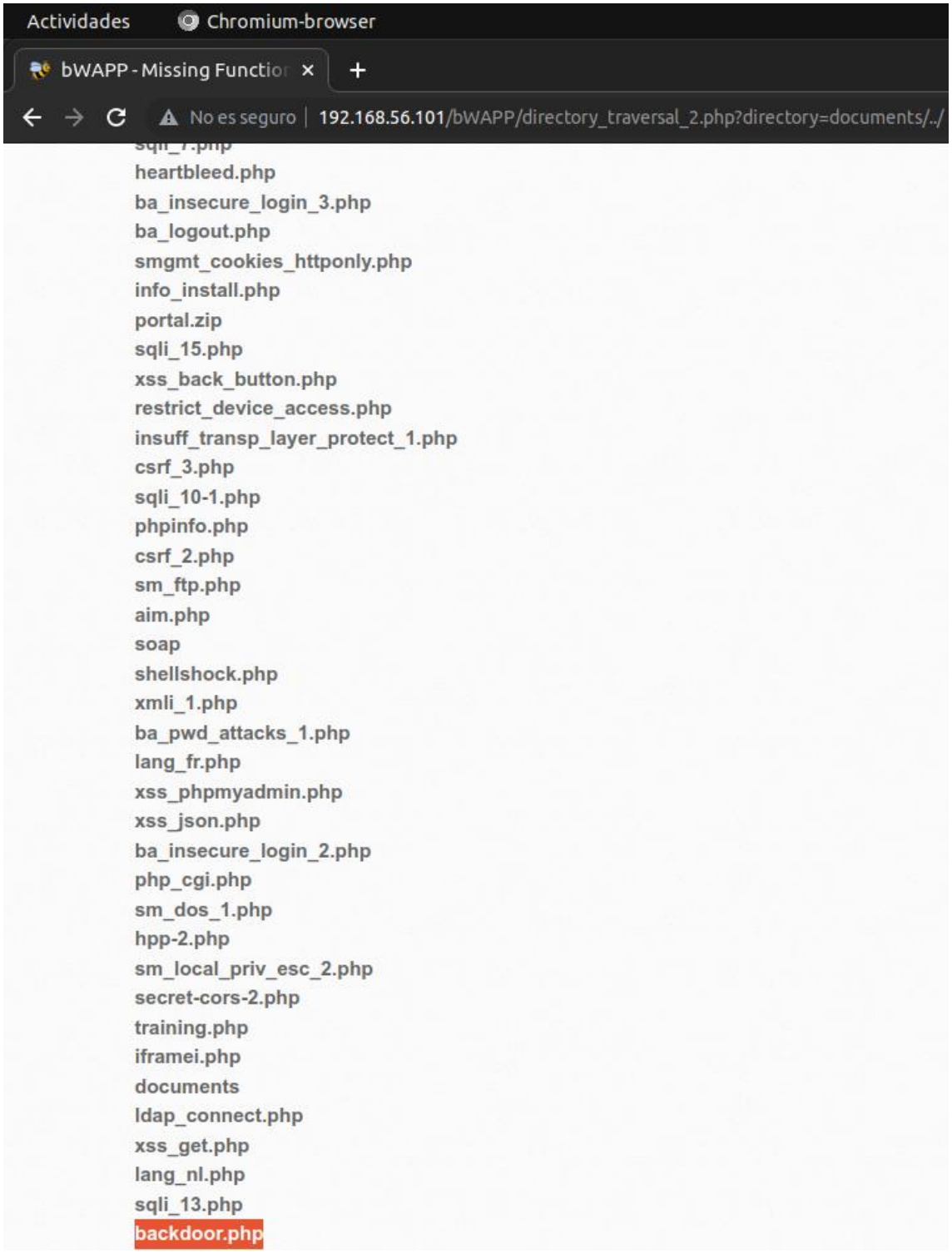

Ilustración 31 - Directories Backdoor.php en directorio padre de documents

Abriendo este enlace, llegamos a una página en la que se pueden subir archivos y colocarlos en el directorio que se indique. Esto tiene mucho peligro, ya que podemos subir cualquier tipo de archivo, que luego puede ser abierto por cualquier usuario que entre a la página inicial, por ejemplo.

Ilustración 32 - Directories Backdoor.php

En nuestro caso, vamos a subir simplemente una imagen y la colocaremos en el directorio donde se encuentran los enlaces que vimos al inicio de este apartado.

Finalmente, la página principal queda así:

Ilustración 33 - Directories Resultado

4.4.2. Files

En esta página se muestra un mensaje que invita a "trepar", como queriendo decir que subamos en directorios como hicimos anteriormente. De nuevo, existe un campo en la URL llamado "page" con valor "message.txt". En este caso, parece ser que se está mostrando el contenido del fichero cuya ruta se indica en el campo de la URL. Como si se ejecutara el comando "cat" en Linux.

Esto invita a buscar archivos que contengan información sensible, como puede ser el archivo de contraseñas de la máquina. Si el servidor tiene permiso para poder ver este archivo, nos lo mostrará en la página web simplemente indicando la ruta del archivo. Este archivo en cuestión sabemos que tiene la siguiente ruta: */etc/passwd*.

Ilustración 34 - Files Encontrando /etc/passwd

5. Cross-Site-Scripting

5.1. Reflected GET

En este apartado, se nos vuelve a mostrar un formulario simple de Nombre y Apellido con un botón de acción. Así que de nuevo podemos tratar de ejecutar un script introduciéndolo en uno de los campos del formulario.

Ilustración 35 - XSS Introduciendo script

Y de nuevo, obtenemos el resultado esperado

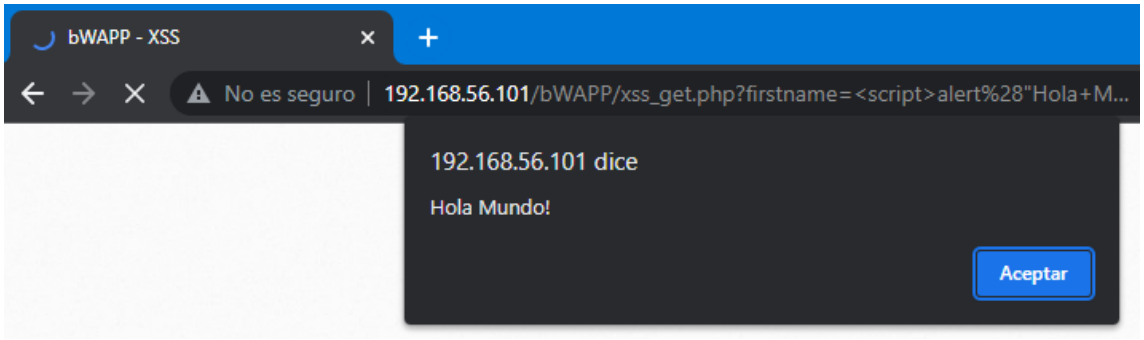

Ilustración 36 - XSS Resultado

6. Otros

6.1. Base64 Encoding

7. Conclusiones

En esta práctica hemos podido aprender gran variedad de cosas, como pueden ser las distintas vulnerabilidades y fallos de diseño e implementación de una página o servicio web; y hemos podido conocer y aprender a usar mínimamente una nueva herramienta como es BurpSuite, que actúa como “Man in the Middle”, entre sus diferentes usos.

Está práctica complementa muy bien la práctica anterior. Anteriormente vimos las vulnerabilidades que puede tener una máquina como tal, siendo atacada directamente; mientras que en esta, hemos podido atacar a la máquina (y por tanto a sus encargados) a través de un servicio que corría usando la máquina como host.

De nuevo nos encontramos con una práctica que nos abre los ojos y nos da un toque de atención para el futuro, para estar pendientes de nuestros diseños de aplicación, de toda la seguridad que se puede ofrecer con el hecho de estar informado de estas “grietas” y de lo importante que es también compartir tus conocimientos sobre el tema con tus compañeros para que puedan evitar cometer estos fallos. De nada sirve que solo una persona en un equipo sepa dónde puede haber fallos y agujeros de seguridad.

[FINAL DE DOCUMENTO]