Experiment 4: SQL Injection Attacks on Web Applications

Scenario:

The DVWA application's login and search functionalities are suspected to lack proper input validation. The company needs confirmation that attackers can extract sensitive data using SQL injection.

Tasks:

- Use SQLMap against DVWA's vulnerable pages to enumerate databases, tables, and potentially user credentials.
- Confirm that an attacker could retrieve confidential information from the backend database.

Deliverable:

Proof (screenshots/logs) of extracted database entries and a brief report on the risk to the organization.

INTRODUCTION TO DVWA

Damn Vulnerable Web Application (DVWA) is a web application designed specifically for security professionals and learners to practice and improve their skills in a safe environment. It simulates common security issues found in websites, such as SQL injection, cross-site scripting (XSS), and command injection. DVWA allows you to experiment and learn without the risk of affecting real websites. The tasks described below were performed using DVWA in a local setup to explore and understand SQL injection vulnerabilities.

DVWA Setup Instructions

Step 1: Install Docker

sudo apt install docker.io

Step 2: Start and Enable Docker

sudo systemctl start docker

sudo systemctl enable docker

Step 3: Download

sudo docker pull vulnerables/web-dvwa

Step 4: Run DVWA

sudo docker run --rm -it -p 80:80 vulnerables/web-dvwa

Step 5: Access DVWA in Browser

- URL: http://localhost
- Credentials:
  - Username: admin
  - Password: password
- Click "Reset Database" under the Setup tab.
- Set Security Level to Low under the DVWA Security tab.
- Navigate to the SQL Injection section and enter '1' in the USERID parameter to generate the URL for testing the injection vulnerability.

Extracting PHPSESSID (Required for SQLMap)

1. Log into DVWA using the default credentials.
2. Open Developer Tools (Right-click → Inspect).
3. Go to the Storage tab → Cookies → http://localhost.
4. Look for the PHPSESSID value and copy it.
5. Use this session ID in your SQLMap commands for authenticated access.

INTRODUCTION TO SQLMAP

SQLMap is a tool used to detect and exploit SQL injection vulnerabilities in web applications. It automates the process of testing for vulnerabilities and can retrieve sensitive data like usernames and passwords from a vulnerable database. In this experiment, SQLMap will be used to test for SQL injection in DVWA and extract information from its backend database.

Step 1: Enumerate Available Databases

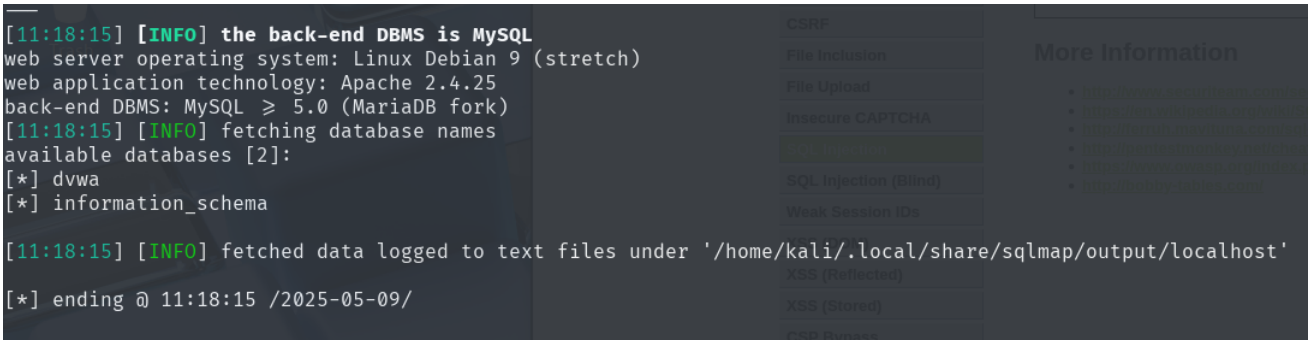
Command:

sqlmap -u " <url_dvwa_sqli>" --cookie="security=low; PHPSESSID=<your-session-id>" --batch --dbs

What It Does:

- Sends SQL injection payloads to identify backend databases.
- --dbs restricts output to only list database names.

Output:



Step 2: Enumerate Tables in the ‘dvwa’ Database

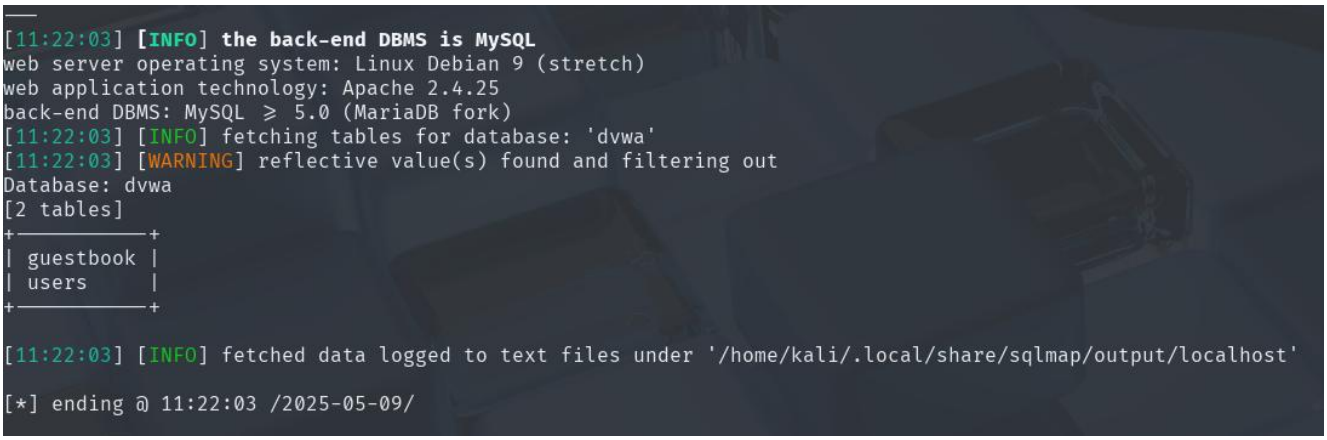
Command:

sqlmap -u " <url_dvwa_sqli>" --cookie="security=low; PHPSESSID=<your-session-id>" --batch -D dvwa --tables

What It Does:

- Targets the dvwa database.
- Lists all tables inside it using the --tables flag.

Output:



Step 3: Dump Data from the ‘users’ Table

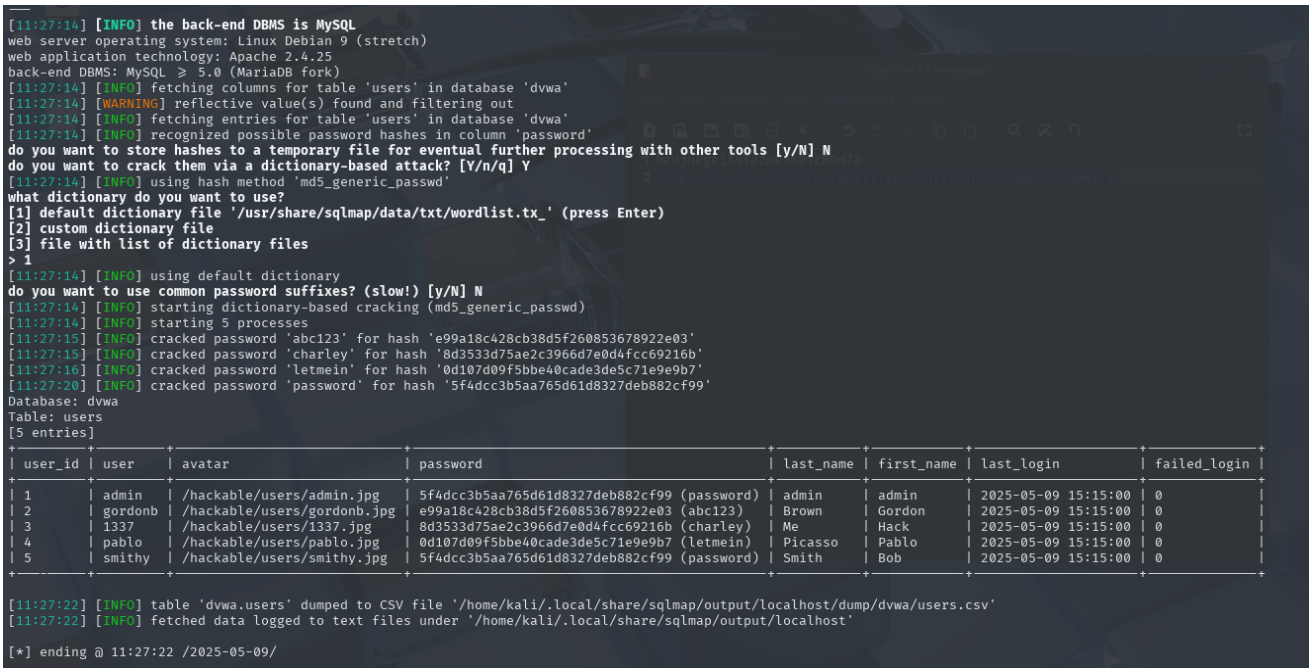
Command:

sqlmap -u " <url_dvwa_sqli>" --cookie="security=low; PHPSESSID=<your-session-id>" --batch -D dvwa -T users --dump

What It Does:

- Accesses and dumps all data from the users table in dvwa database.

Output:



Brief Risk Report:

SQL Injection Risk Summary

- **Vulnerability**: Authenticated SQL injection in user input (ID parameter)
- **Risk Level**: **High**
- **Impact**:
    - Database enumeration and full table dumps possible
    - Attackers can steal **user credentials**, escalate privileges, or tamper with data
- **Required Fix**: Implement parameterized queries or ORM-based input handling, validate and sanitize inputs, enable WAF

Conclusion

This experiment showed how web applications like DVWA can be attacked using SQL injection to get sensitive data like usernames and passwords. We used SQLMap to demonstrate how attackers can find databases, list tables, and extract important user information. This highlights how important it is for developers to handle user input carefully. SQL injection can lead to data breaches, unauthorized access, and legal issues. The best way to prevent these attacks is by using safe coding practices, like input validation, prepared statements, and regular security testing. This experiment reminds us how important it is to build security into web applications from the start and to always be proactive in defending against attacks.

****