Experiment No. 8

Aim: Performing SQL Injection Attacks Manually Using SQLmap (Authentication Bypass & Data Extraction)

Learning Objective:

- Understand the process of performing manual SQL injection attacks using SQLmap.
- Learn to bypass authentication and extract data from vulnerable databases.

Theory:

Introduction to SQL Injection

SQL injection (SQLi) is a critical security vulnerability that allows attackers to interfere with the queries an application makes to its database. This can result in unauthorized data access, authentication bypass, and in severe cases, complete database takeover.

SQLmap Overview

SQLmap is an open-source penetration testing tool designed to automate detecting and exploiting SQL injection flaws. It supports a range of injection techniques and multiple database systems and can be used to extract large amounts of data efficiently.

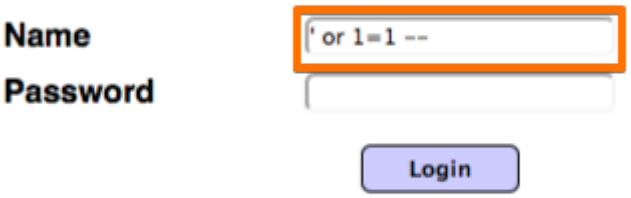
Manual SQL Injection Steps

1. Identifying SQL Injection Points

- Use manual payloads such as ' OR '1'='1-- or ' AND 1=0 UNION SELECT null-- in URL parameters or form fields to detect vulnerable inputs.
- Look for anomalies like error messages, unexpected data display, or authentication bypass after payload injection.

2. Authentication Bypass Example

- Many login forms use queries like:
  SELECT * FROM users WHERE username = '<input>' AND password = '<input>';
- If inputs are not sanitized, entering the following in the username field:



3. Using SQLmap for SQL Injection Attacks

a. Basic Usage

- Test a parameter for SQL injection:
- This detects and tests for vulnerabilities in the specified parameter.

b. Authentication Bypass (Login Page Example)

- Capture the HTTP login request (using Burp Suite or browser dev tools).
- Save the request to a file, e.g., login.req.
- Run SQLmap with:

c. Database Enumeration

- Once injection is found, enumerate databases
- Lists all databases on the target server.

d. Table and Column Discovery

- List tables in a database (replace dbname):
- List columns in a table (replace table):

e. Data Extraction

- Extract all data from a specific table:
- To target only certain columns
- This efficiently retrieves sensitive user credentials.

```
└─$ sqlmap -r req.txt -D public --tables
```

```
Database: public
[2 tables]
+-----------+
| tracking  |
| users     |
+-----------+
```

```
└─$ sqlmap -r req.txt -D public -T users --columns
```

```
Database: public
Table: users
[3 columns]
+-----------+-----------+
| Column    | Type      |
+-----------+-----------+
| email     | varchar   |
| password  | varchar   |
| username  | varchar   |
+-----------+-----------+
```

Recommended Procedure

1. **Reconnaissance**: Find forms or URL parameters possibly vulnerable to SQLi.
2. **Manual Injection Testing**: Insert simple payloads to watch for unusual application behavior.
3. **Automated Exploitation**: Use SQLmap to confirm and exploit the injection point.
4. **Authentication Bypass**: Target login points by manipulating input or using SQLmap with captured requests.
5. **Data Extraction**: Progressively enumerate and extract databases, tables, columns, and their data.

Learning Outcome:

- Explain the theory behind CSRF attacks and their potential impact.
- Demonstrate how to simulate a CSRF attack on a vulnerable application.
- Apply industry-standard practices (anti-CSRF tokens, SameSite, header validation) to mitigate CSRF risks.
- Assess and enhance the security of web applications against CSRF vulnerabilities.

Conclusion:

……

……

……

……

Name:

Class: BE-CSE

Roll No.:

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				