CYBER SECURITY INTERNSHIP

Ministry of MSME, Govt. of India

Task 8

INJECTION PRACTICAL EXPLOITATION

JASHMI KS

SQL Injection is one of the most common and critical web application vulnerabilities, where an attacker manipulates input fields to execute unauthorized SQL queries on the backend database. This vulnerability occurs when user input is not properly validated or sanitized before being processed by the database.

SQLMap is an automated penetration testing tool used to detect and exploit SQL Injection vulnerabilities. It helps security testers identify injectable parameters, extract database information, and understand the impact of insecure coding practices.

In this practical, SQLMap is used on a deliberately vulnerable web application in a controlled environment using Kali Linux in VirtualBox to demonstrate how SQL Injection attacks are performed and why proper security measures are essential to protect web applications.

SQL Injection Security Level

DVWA provides four security levels for SQL Injection to help learners see how different protections affect attacks:

1. Low Security

The app takes your input and directly puts it into the SQL query with no filtering.

$id = $_GET['id'];$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";

- **Entering ':** Breaks the query and makes the database throw an error, revealing it is vulnerable.
- **Entering 1' OR '1'='1:** Tricks the query into always being true, so all users are returned.
- **Entering 1' UNION SELECT user, password FROM users--:** Joins another query to fetch hidden data like usernames and passwords.

2. Medium Security

The app applies basic input sanitization using functions like addslashes() to escape '.

$id = addslashes($_GET['id']);$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";

How can be Attack:

A simple ' injection won't work anymore (because it becomes \').

But attackers can still bypass using numeric injection (since numbers don't need quotes).

Example:

1 OR 1=1

This still returns all records.

3. High Security

The app uses prepared statements (parameterized queries) to safely handle user input.

$stmt = $pdo->prepare("SELECT first_name, last_name FROM users WHERE user_id = ?");$stmt->execute([$id]);

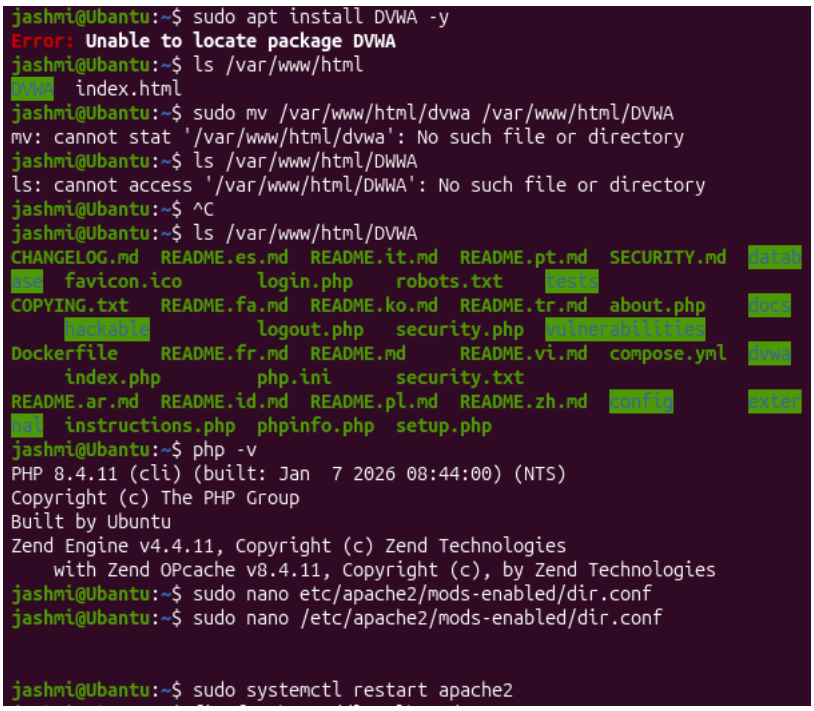
Attack:

Attempts like ', OR 1=1, or UNION SELECT no longer work.

The query treats all input as data, not SQL code.

Identify Injectable Parameters

Injectable parameters are input fields or URL values that interact directly with the database without proper validation, making them vulnerable to SQL Injection attacks.



Step 1: Identify Injectable Parameters

Injectable parameters are input fields or URL parameters that accept user input and are directly used in database queries without proper validation. These parameters can be identified by manually inserting special characters such as a single quote (') or logical conditions like OR 1=1.

If the application displays database errors or unexpected behavior, it indicates that the parameter is injectable and vulnerable to SQL Injection.

Step 2: Run SQLMap

SQLMap is an automated SQL Injection exploitation tool used to detect and exploit vulnerable parameters. Once an injectable parameter is identified, SQLMap is executed with the target URL to verify the vulnerability and begin exploitation.

SQLMap automatically tests the parameter and confirms whether SQL Injection is possible.

Step 3: Extract Database Names

After confirming the vulnerability, SQLMap is used to enumerate the available databases on the target server. This step reveals all databases that the application user has access to.

Extracting database names helps attackers understand the backend structure and identify sensitive databases.

Step 4: Extract Tables

Once a target database is selected, SQLMap is used to list all tables present within that database. Tables contain structured data such as user credentials, application data, and configuration information.

This step helps identify tables that store sensitive information.

Step 5: Extract User Data

After identifying important tables, SQLMap is used to extract column names and dump data stored in those tables. This commonly includes usernames, passwords, email addresses, and other confidential information.

This step demonstrates how attackers can steal sensitive data using SQL Injection.

Step 6: Analyze Impact

The impact of SQL Injection exploitation includes:

- Unauthorized access to sensitive data
- Leakage of user credentials
- Modification or deletion of database records
- Bypass of authentication mechanisms
- Complete compromise of the application database

This vulnerability poses a serious threat to application security.

Step 7: Document Attack Flow

The complete attack flow is documented step-by-step, including:

- Identification of vulnerable input
- Confirmation of SQL Injection
- Database enumeration
- Table extraction
- Data dumping

Screenshots are captured at each stage to provide proof of exploitation.

Step 8: Suggest Fixes

To prevent SQL Injection attacks, the following security measures are recommended:

- Use prepared statements and parameterized queries
- Implement proper input validation and sanitization
- Avoid displaying database error messages
- Use least-privilege database accounts
- Enable web application firewalls (WAF)

The SQL Injection vulnerability was successfully identified and exploited using SQLMap. Sensitive database information was extracted, proving that improper input handling can lead to severe security breaches.