SQL Injection

This project focuses on understanding and practically demonstrating a critical web security vulnerability known as **SQL Injection (SQLi)**. By simulating an attack environment, the project showcases how attackers can exploit poorly secured input fields to gain unauthorized access to a database, extract sensitive information, and compromise system integrity.

Goals and Objectives

- To simulate a SQL Injection attack in a controlled lab environment.
- To educate developers and cybersecurity students on how such vulnerabilities occur and how they can be prevented.
- To perform hands-on exploitation using industry-standard tools like SQLmap and Kali Linux.

Technologies and Tools Used

Component	Tool/Environment
Operating System	Kali Linux
Exploitation Tool	SQLmap
Test Web Server	DVWA (Damn Vulnerable Web App) or custom-built
Backend Database	MySQL
Browsers/Terminals	Firefox, Linux Terminal

What is SQL Injection?

SQL Injection is a code injection technique that exploits vulnerabilities in an application's database layer. It occurs when untrusted input is embedded directly in SQL queries without proper sanitization or escaping.

For example:

SELECT * FROM users WHERE username = 'admin' AND password = '1234';

If the input is not sanitized, a malicious user could enter:

' OR '1'='1

Turning the query into:

SELECT * FROM users WHERE username = '' OR '1'='1';

This returns all users, effectively bypassing authentication.

Project Workflow

1. **Vulnerable Web App Setup**
    - Deployed a test application with insecure login forms (e.g., DVWA or PHP-based login system).
    - Configured MySQL as the backend with dummy user data.
2. **Launching SQLmap**
    - Ran SQLmap against the login form to automate SQL Injection.
    - Discovered injectable parameters and extracted:
        - Table names
        - Usernames and hashed passwords
        - Database version and structure
3. **Result Analysis**
    - Demonstrated how attackers can enumerate databases, read confidential data, and even modify or delete tables.
4. **Mitigation Explained**
    - Use of prepared statements (parameterized queries)
    - Input validation and sanitization
    - Web Application Firewalls (WAFs)

Key Takeaways

- SQL Injection remains one of the **OWASP Top 10** vulnerabilities.
- Even basic knowledge of SQL and access to free tools can lead to significant breaches if protections are not in place.
- Secure coding practices are essential to prevent injection attacks.

Mitigation Techniques

Best Practice	Description
Prepared Statements	Use placeholders rather than dynamic SQL
Input Validation	Reject suspicious or malformed inputs
Escaping User Input	Escape special characters that can break queries
Least Privilege	Restrict DB user roles to avoid destructive actions
Error Handling	Avoid showing DB errors to the user

Ethical Usage Disclaimer

This demonstration was conducted in a safe lab environment for educational purposes only. Never perform such attacks on real systems without explicit permission, as doing so is illegal and unethical.