Cyber Security Task 1

🛡️ Web Application Security Testing

Intern Name: Dhrumi Sonani

Report Title: VAPT Report on DVWA

Internship provider: Future Intern

Domain: Cyber Security

Table of Contents

Objective

The objective of this project is to identify and assess common web application vulnerabilities in DVWA, such as SQL Injection, XSS (cross-site scripting), and CSRF attacks and Brute Force attacks, using tools like Burp Suite, Nikto. The goal is to understand how attackers exploit these flaws and to recommend effective remediation measures.

Test Environment Setup

A controlled test environment was created using **Kali Linux** and the **Damn Vulnerable Web Application (DVWA)** to simulate a realistic web application penetration testing scenario. The environment includes intentionally vulnerable software and industry-standard security testing tools.

Access URL : http://localhost/dvwa

Security level: Set to low via DVWA Security panel

Methodology & Tools

- **Firefox** – Used for testing and interacting with DVWA
- **Burp Suite** – Intercepted and analyzed HTTP requests
- **DVWA** – Target application for vulnerability testing

Task Overview

- Logged into the DVWA application hosted on the local server

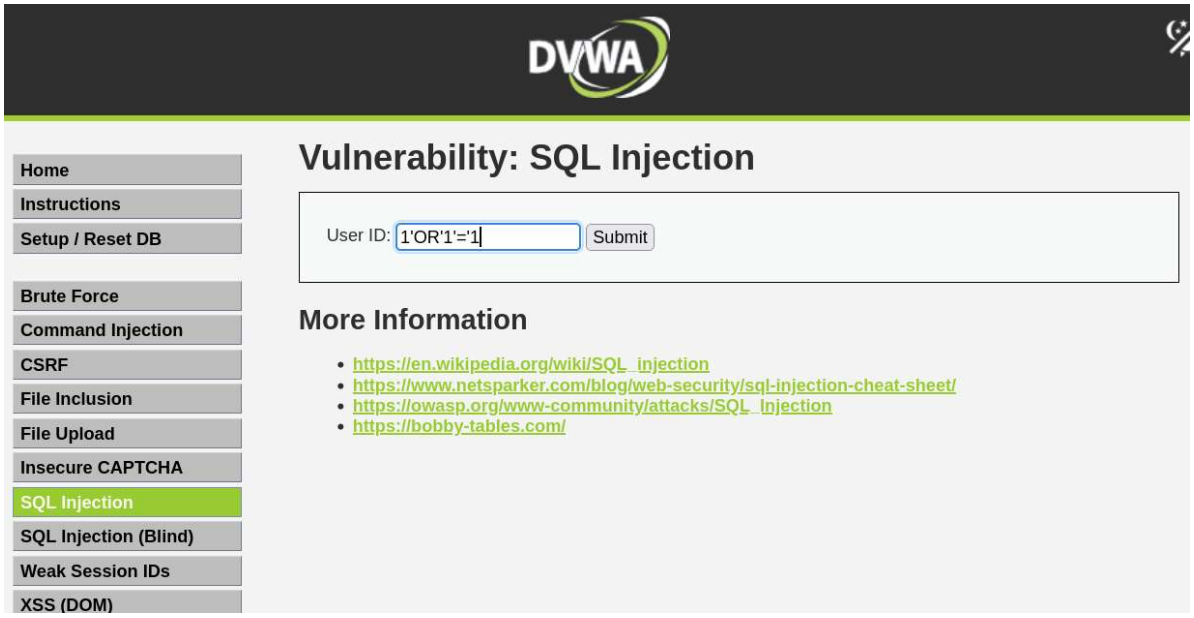
- Switched the DVWA **security level between Low and Medium** to observe vulnerability behavior under different configurations
- Explored and tested key vulnerabilities including **SQL Injection**, **XSS**, and **CSRF**

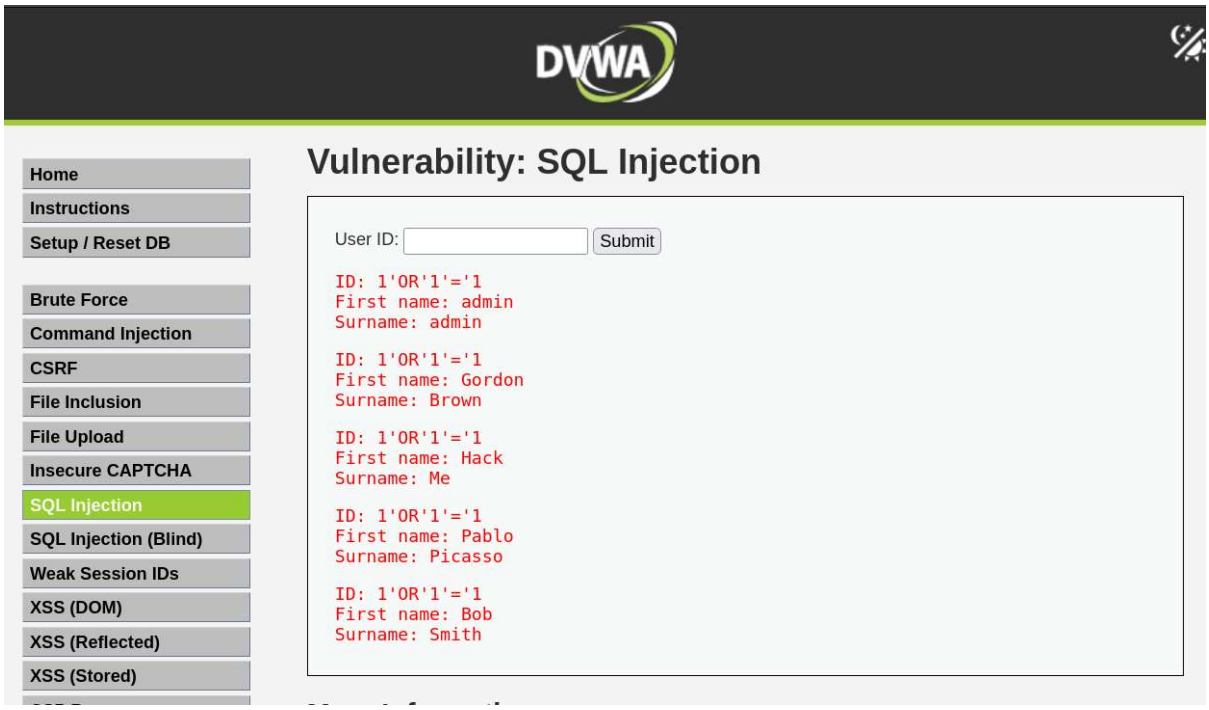
Test for Vulnerabilities

1. SQL Injection Testing (DVWA > SQL Injection Testing)

SQL Injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It can allow attackers to:

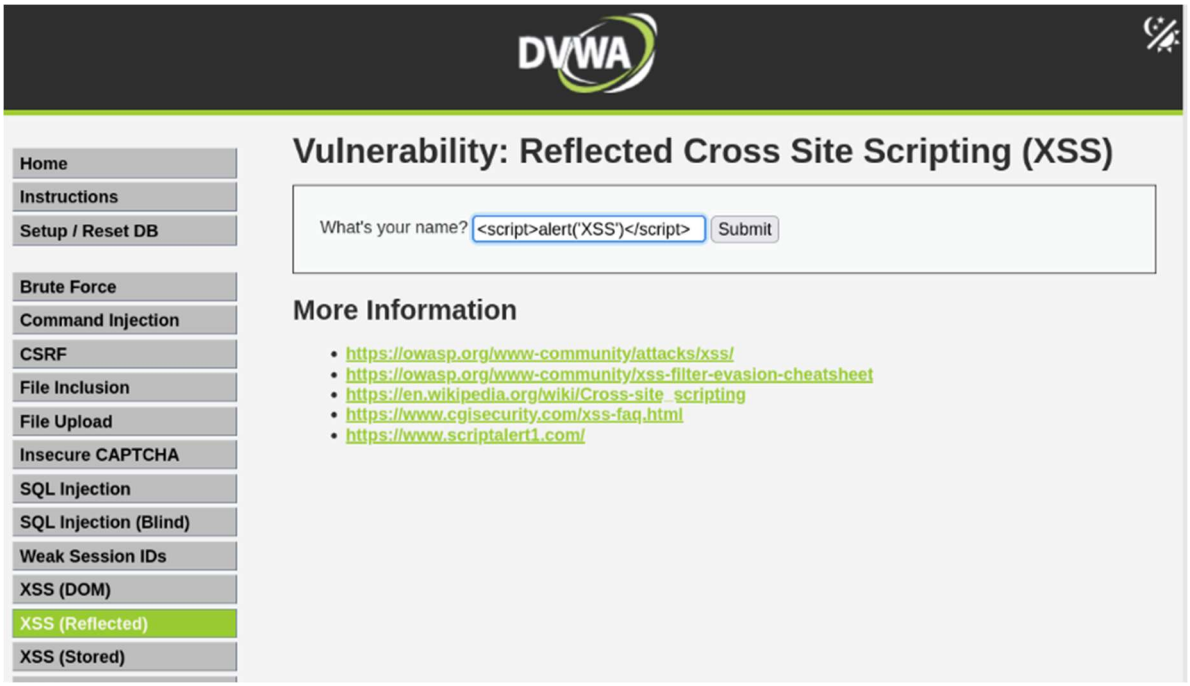
- Bypass authentication
- Retrieve, modify or delete data
- Execute administrative operations

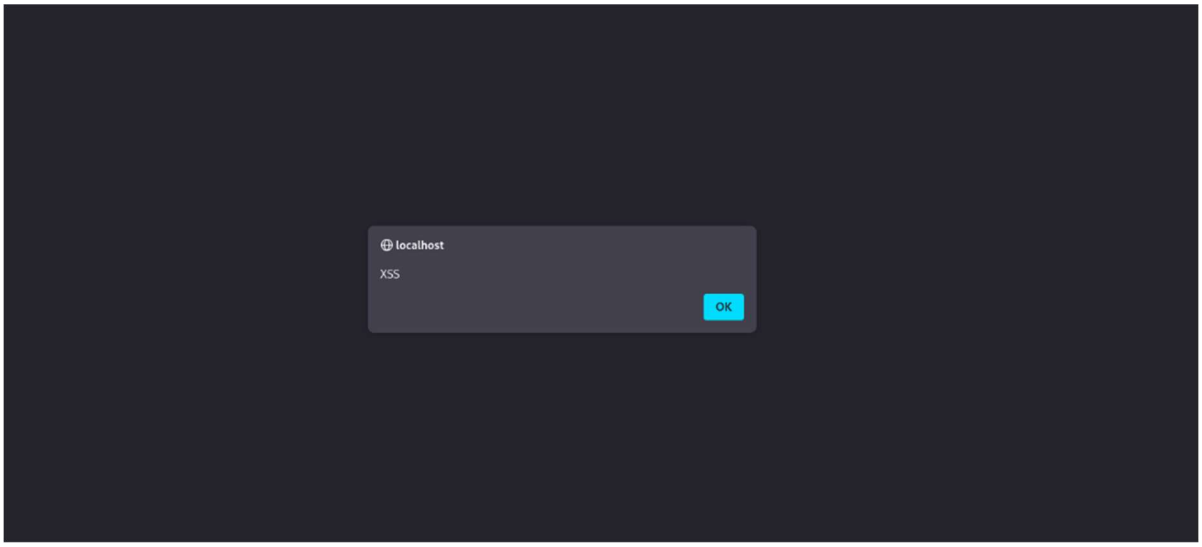




2. Vulnerability: Reflected Cross-Site Scripting (XSS)

Reflected XSS is a client-side vulnerability where user input is immediately back by the server without proper validation or encoding. This allows attackers to inject malicious JavaScript, which executes in the victim's browser.



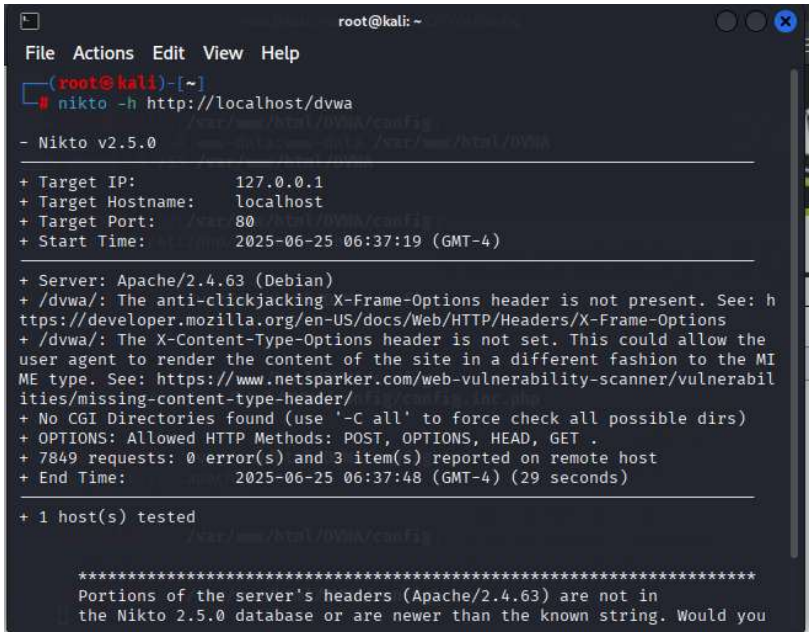


3. Vulnerability Scanning using Nikto on DVWA

Vulnerabilities are detectable by scanners like Nikto.

Nikto will start scanning and output results like:

- HTTP headers
- Server banner
- Directory structure
- Security issues



4. Vulnerability Assessment: Cross-Site Request Forgery (CSRF)

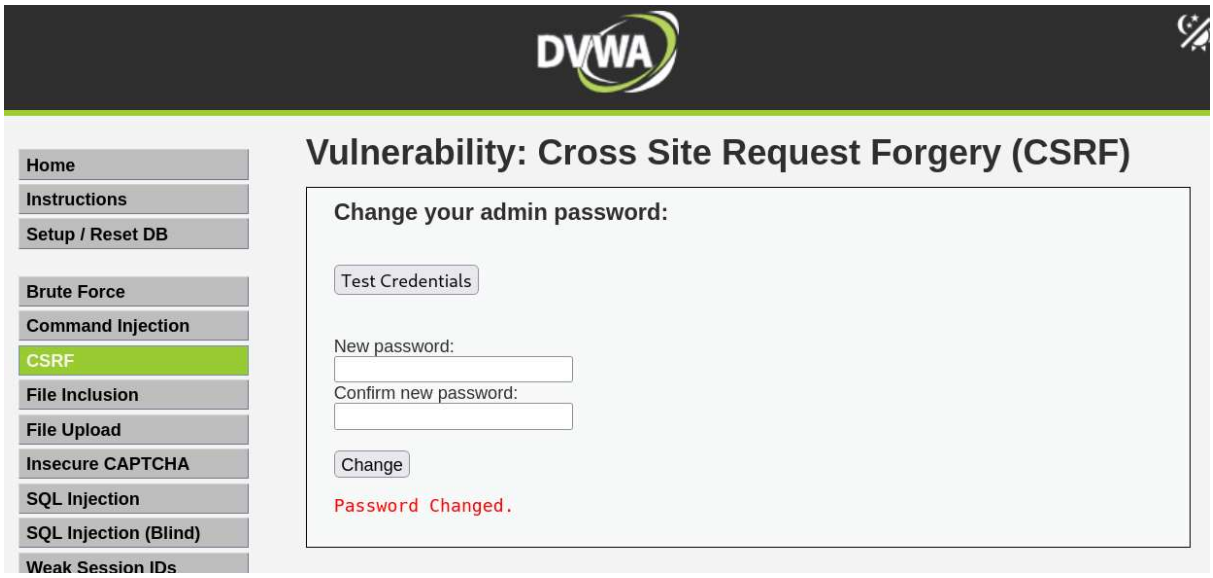
CSRF occurs when a malicious website or script causes a user's browser to perform unwanted actions on a web application where they are authenticated. Since DVWA lacks CSRF tokens and origin validation, it accepts requests without verifying if they came from a trusted source.

Impact:

An attacker can:

- Change the victim's password
- Perform sensitive actions on behalf of the victim
- Potentially gain full control of user accounts

DVWA showing password changed successfully.

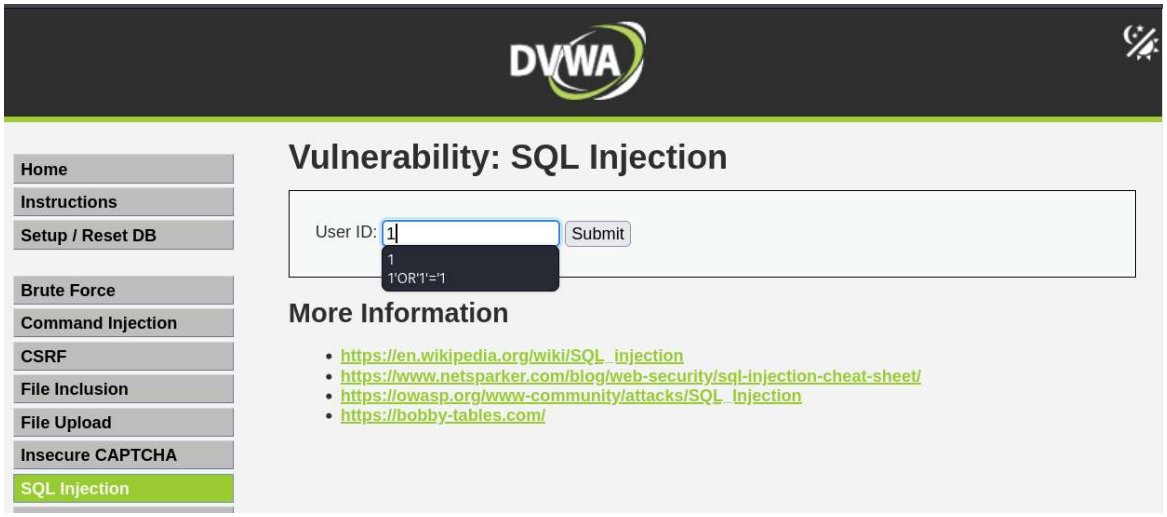


5. Vulnerability: Insecure Direct Object Reference (IDOR)

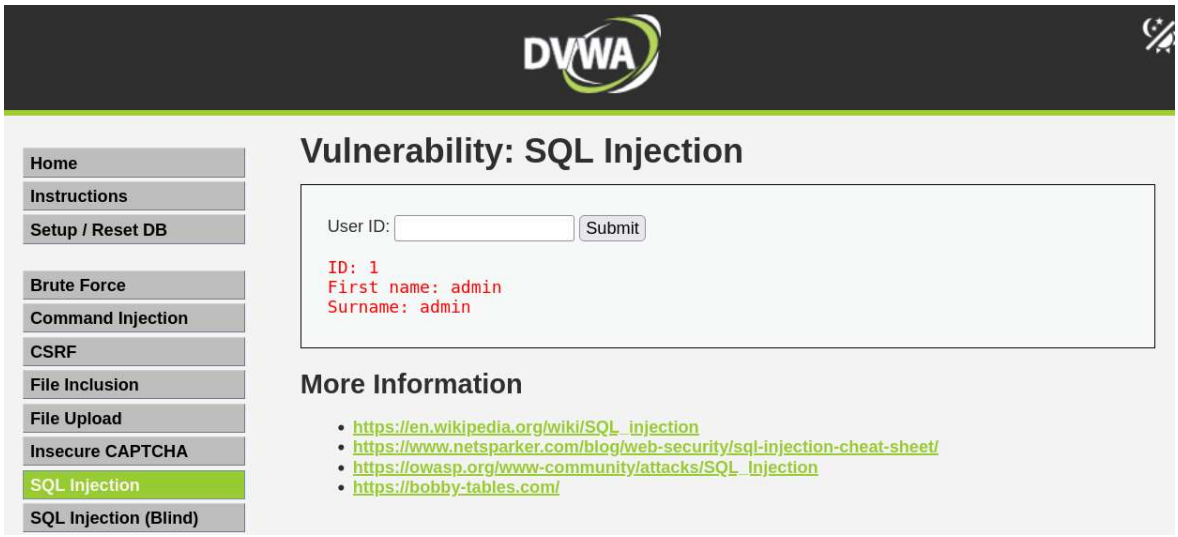
IDOR occurs when an application exposes object identifiers (e.g., user IDs) in the URL or request, and fails to properly enforce access control. Attackers can manipulate these references to access unauthorized data.

Impact:

- Unauthorized access to sensitive data
- Potential for data leakage or privilege escalation



Showing data for user 1



Showing data for user 2

📊 Findings Summary

- The DVWA application was tested under low-security settings to identify common web vulnerabilities. Four major issues were observed: SQL Injection, Reflected Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and Insecure Direct Object Reference (IDOR).
- The **SQL Injection** vulnerability was found in the login form, where unsanitized input allowed malicious SQL queries to be executed. This posed a high risk of unauthorized access or data manipulation.
- The **Reflected XSS** vulnerability was observed in the XSS module, where user input was reflected back in the response without proper encoding. This allowed JavaScript execution in the browser, resulting in a medium-level security risk.
- The **CSRF** vulnerability affected the password change functionality. The absence of CSRF tokens allowed unauthorized requests to be submitted on behalf of authenticated users, leading to a high-severity issue.
- A simulated **IDOR** vulnerability demonstrated that by modifying object references (e.g., user ID in the URL), it was possible to access other users' data without proper authorization checks. This was categorized as a high-risk issue.

🛠️ Recommendations

- Use input validation and output encoding to prevent SQLi and XSS.
- Implement CSRF tokens and verify request origins.
- Apply access control checks to protect against unauthorized access (IDOR).
- Use secure identifiers (e.g., UUIDs) instead of direct object references.
- Keep all systems and applications updated with security patches.