CYBER SECURITY INTERNSHIP REPORT

TASK 1: WEB APPLICATION SECURITY TESTING

ASSIGNMENT

BY

ATHANASIUS J.K GADOSEY

Introduction

Task Overview

Task 1: Web Application Security Testing

Objective of Task

This report summarizes a web application security assessment conducted on the Damn Vulnerable Web Application (**DVWA**), an intentionally insecure web app designed for learning and practicing security techniques.

The purpose of this task was to simulate a real-world vulnerability assessment using ethical hacking techniques and to identify common web application security flaws as defined by the **OWASP TOP 10.**

Methodology

The following approach was used during the assessment:

Test Environment:

- **DVWA:** A learning platform designed with vulnerable web app scenarios for practice.

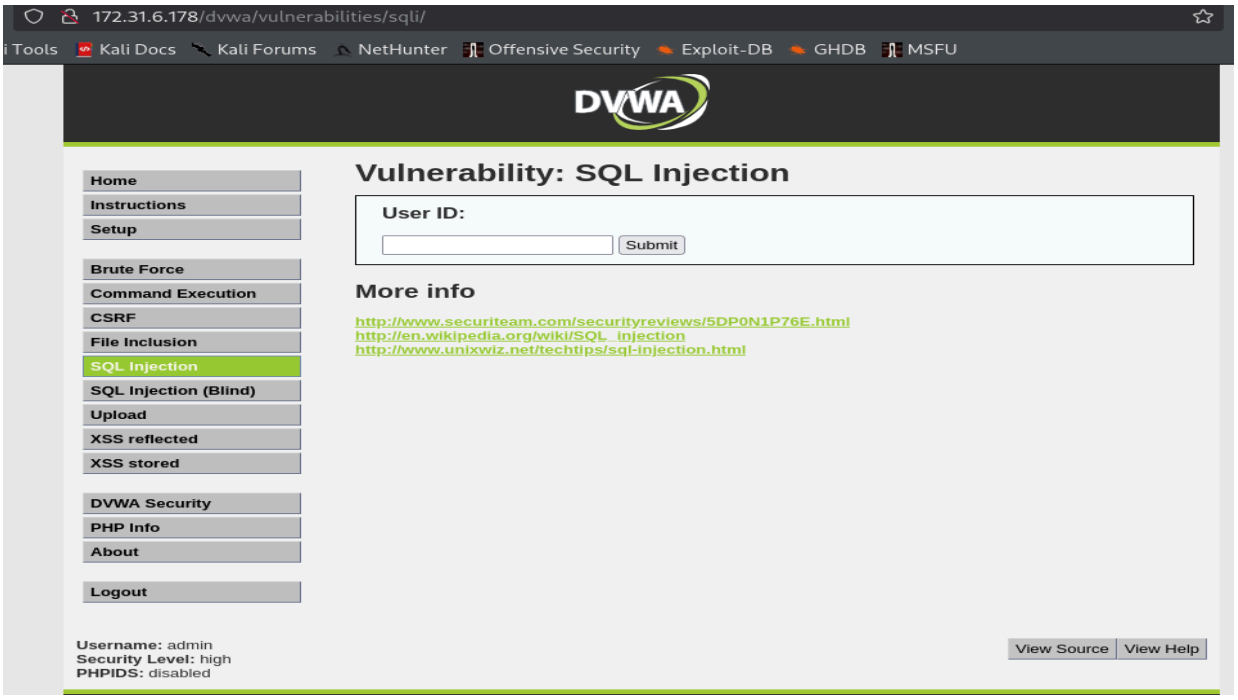
Tools I Used:

- **OWASP ZAP (Zed Attack Proxy):** A security tool that helps find weaknesses in websites by scanning and analyzing web traffic.
- **Browser (e.g., Firefox or Chrome):** Used to interact with **DVWA** while routing traffic through ZAP.
- **Nikto:** Used to identify potential security vulnerabilities and misconfigurations.
- **Burp Suite:** Used to intercept and scan traffic

Findings In The Report:

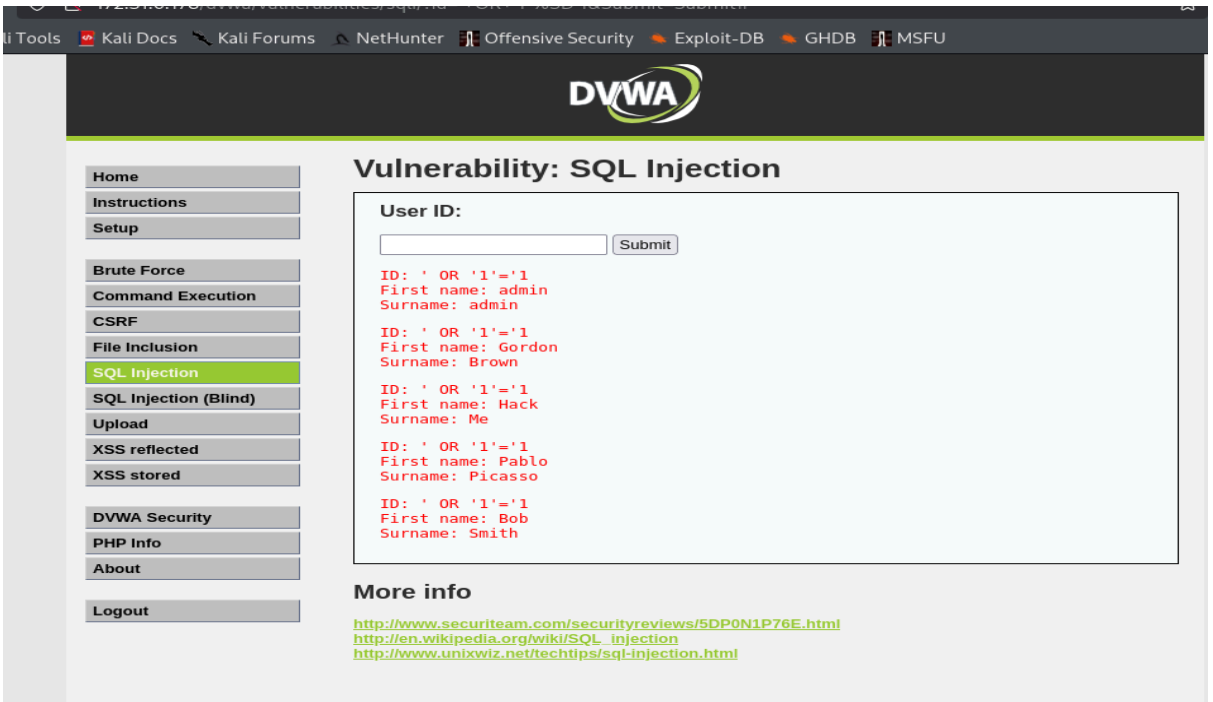
Vulnerability 1: SQL Injection in "**id**" parameter

- **Location**: SQL Injection module – ID Field
- **Description:** User input is directly concatenated into an SQL query, allowing attackers to manipulate database queries.
- **Impact:** High – could lead to data leakage or full DB access.
- **Steps to Reproduce:**
    - Go to http://localhost/dvwa/vulnerabilities/sqli/



This shows the page of the SQL Injection Vulnerability along with the field to execute the command

- Input ' OR '1'='1 in the ID field, click Submit, and the Application returns user data from the database.



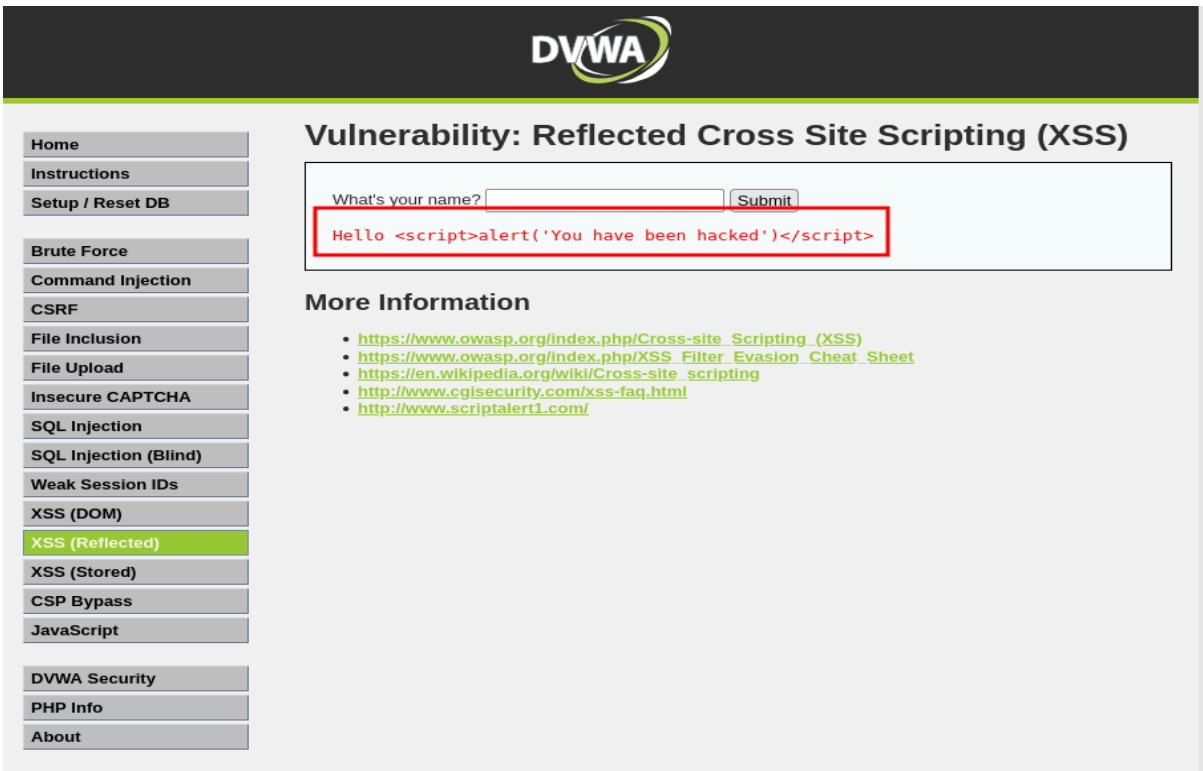
This shows a list of usernames after the command was successfully executed

- **OWASP Category:** A03: Injection
- **Mitigation:** Use prepared statements or parameterized queries.

Vulnerability 2: Reflected Cross-Site Scripting (XSS)

- **Location: XSS (Reflected)** module – input field
- **Description:** User input is rendered directly on the page without sanitization
- **Impact: Medium–** could allow hijacking or defacement.
- **Step to Reproduce:**
  - Input **<script>alert('XSS')</script>** into the form.

This shows the input field where the script command was entered

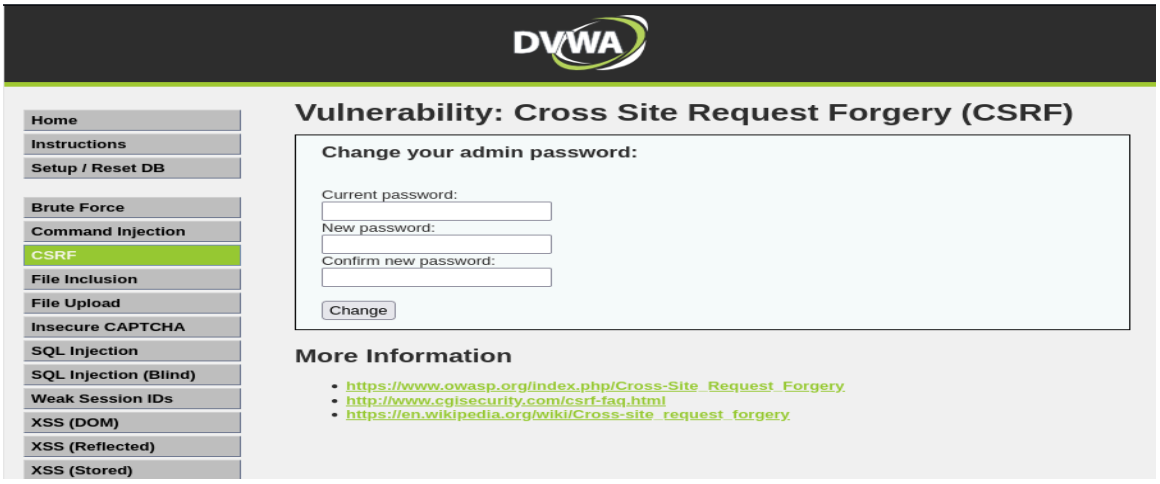


This shows the results output after the script was executed

- **OWASP Category:** A07: Cross-Site Scripting (XSS)
- **Mitigation:** Apply output encoding and input validation.

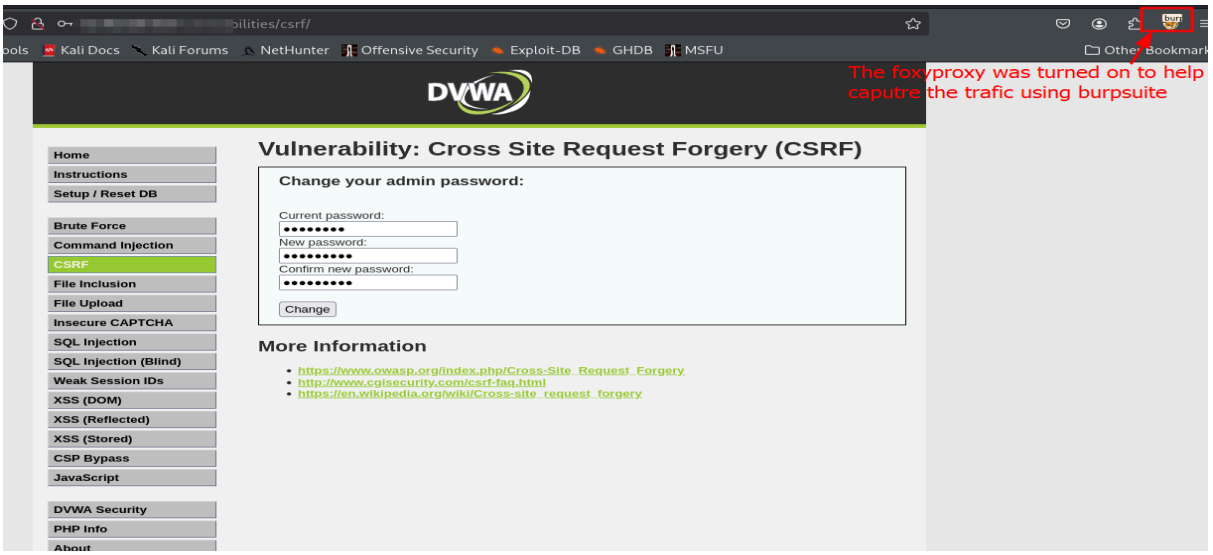
Vulnerability 3: Cross-Site Request Forgery (CSRF)

- **Location: DVWA → CSRF** module (password change page).
- **Description:** The password change functionality does not include CSRF protection. Any attacker who knows the request format can forge it and force the action without user consent.
- **Impact**: High – attackers can take control of user accounts or perform unauthorized actions
- **Steps to Reproduce:**
    - Open the Web Page of the DVWA web application and locate the CSRF vulnerability.



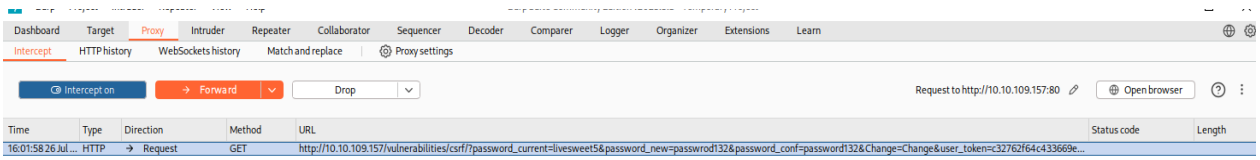
This is the initial page of the CSRF where the password would be changed.

- Turn on your Firefox proxy (I used **FoxyProxy**)

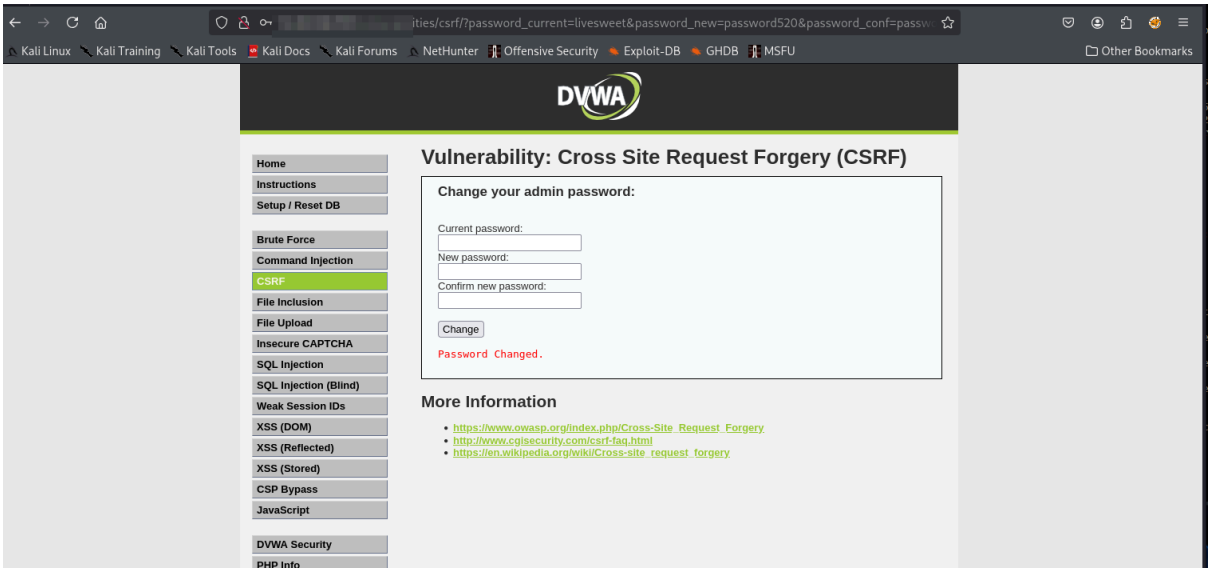


CSRF vulnerability page

- Capture the password change request using Burp Suite



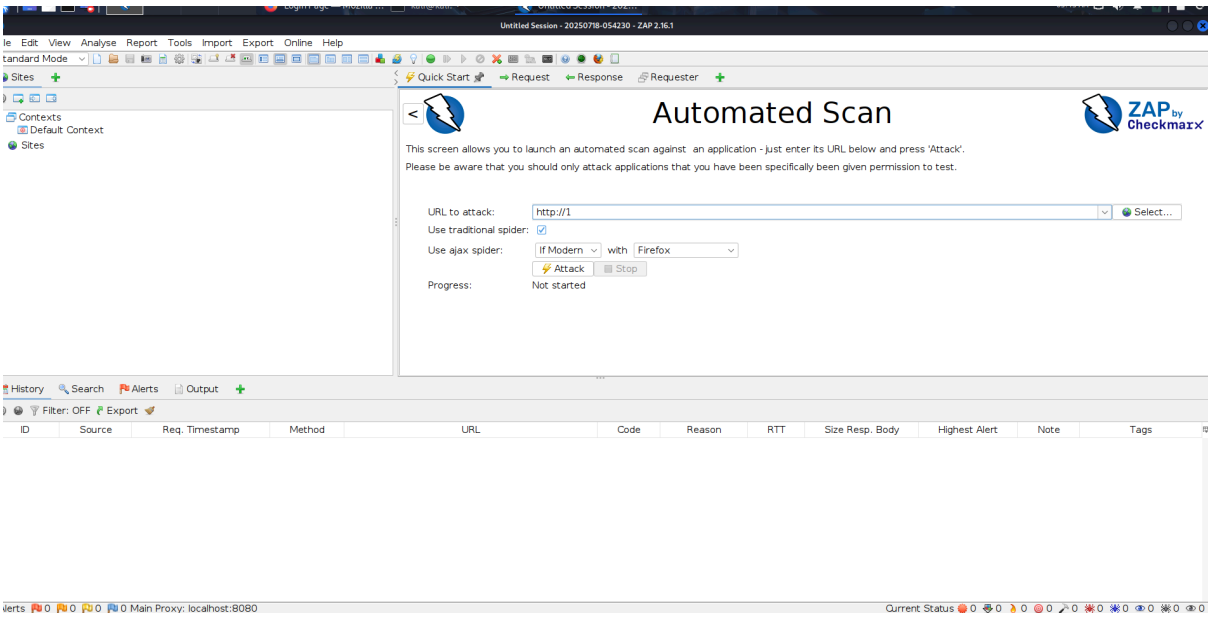
- Send it to Repeater and modify the parameters (e.g., password_new=password520).



After replaying the request, password changes are made without any token validation.

- **OWASP Category:** A01:2021 - Broken Access Control.
- **Mitigation:** Implement anti-CSRF tokens and validate them server-side.

- After starting the DVWA, Foxy captured the traffic, and ZAP was used to scan the app automatically.



Overview of the ZAP Page

OWASP Table

OWASP Top 10 (2021)	Tested	Vulnerabilities Found
A01: Broken Access Control	✅	CSRF (password change), Potential IDOR
A02: Cryptographic Failures	❌	Not tested
A03: Injection	✅	SQL Injection, Reflected XSS
A04: Insecure Design	❌	Not tested
A05: Security Misconfiguration	✅	Command Injection
A06: Vulnerable & Outdated Comp	❌	Not tested
A07: Identification & Auth Fail.	❌	Brute Force Login
A08: Software/Data Integrity Failures	❌	Not tested
A09: Security Logging & monitoring Failures	❌	Not tested
A10: Server-Side Request Forgery (SSRF)	❌	Note tested

Final Report Summary

This assessment evaluated the security posture of the **Damn Vulnerable Web Application (DVWA)** using ethical hacking techniques aligned with the OWASP Top 10 (2021) framework. The objective was to identify common vulnerabilities, demonstrate exploitation, and recommend mitigations.

Key Vulnerabilities Identified

- SQL Injection (A03: Injection) - Unsanitized input allowed database manipulation and data leakage.
- Reflected Cross-Site Scripting (XSS) (A03/A05) - User input was rendered without encoding, enabling script injection.
- Cross-Site Request Forgery (CSRF) (A01: Broken Access Control) - Critical actions could be performed without user consent.

Tools Used

- **Burp Suite** – Interception, brute force, and CSRF testing.
- **OWASP ZAP** – Automated scanning.
- **Kali Linux environment** – Optional testing platform.

Risk Overview

- Several vulnerabilities are **high severity**, including SQL Injection, Command Injection, and CSRF.
- These flaws could lead to data compromise, server takeover, or unauthorised account access.

Recommendations

- Implement input validation, parameterized queries, CSRF tokens, rate limiting, and secure session management to enhance security.
- Align web application security with OWASP best practices.