Web Application Security Testing

CyberSecurityTask1 | Futureinterns

Name: Safik Rahman
Task: Conduct security testing on a sample web application to identify vulnerabilities like SQL
injection, XSS, and authentication flaws.
Skills Gained: Web application security, ethical hacking, penetration testing.

Objective

To identify and document security vulnerabilities in a sample web application using penetration testing techniques and OWASP Top 10 as a reference. The findings aim to demonstrate common web app vulnerabilities and recommend mitigation strategies.

Tools Used

Tool	Purpose
OWASP ZAP	Automated vulnerability scanning
Burp Suite	Manual testing (interception, payload)
Firefox	User interaction & testing
DVWA	Vulnerable web app for simulation

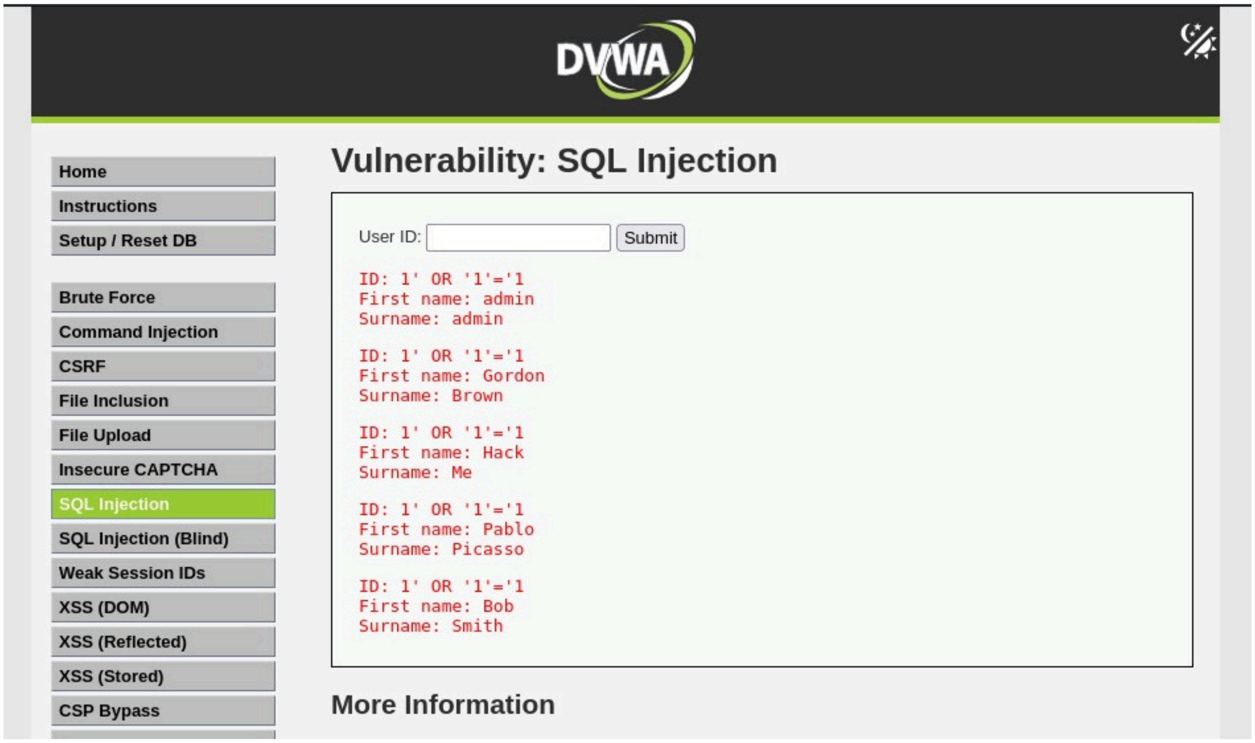
Vulnerabilities Found (Manual + ZAP)

1.SQL Injection (Manual Test)

Tested URL: http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit

Payload Used: `'OR '1'='1`

Result:Successfully bypassedSQL logic andfetched multiple users
Impact:High–Unauthorized access to sensitive data (users, credentials)
OWASP Category: A01 – Broken Access Control / A03 – Injection



Vulnerability: SQL Injection

```
ID: 1' OR '1'='1
First name: admin
Surname: admin

ID: 1' OR '1'='1
First name: Gordon
Surname: Brown

ID: 1' OR '1'='1
First name: Hack
Surname: Me

ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'='1
First name: Bob
Surname: Smith
```

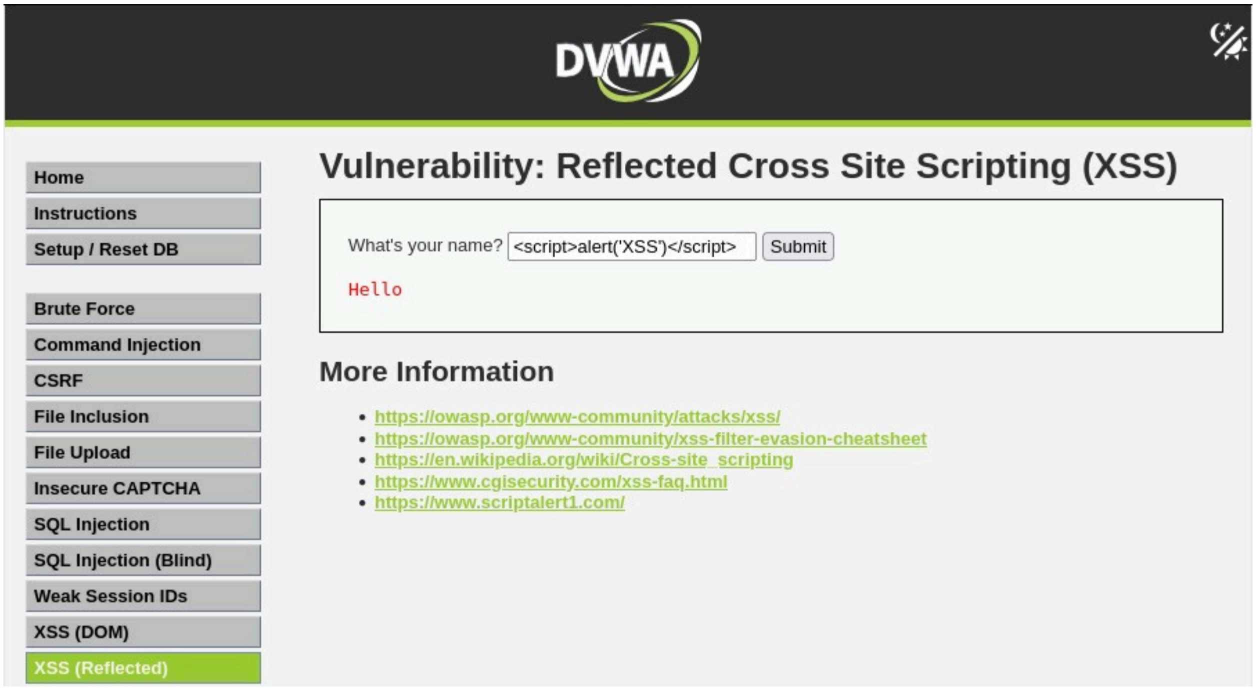
More Information

2.Reflected Cross-Site Scripting (XSS)

Tested URL: `http://localhost/DVWA/vulnerabilities/xss_r/`

Payload Used: `<script>alert('XSS')</script>`

Result: JavaScriptexecutedon page
Impact: Medium–Can be used for session hijacking or phishing
OWASP Category: A07 – XSS



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <script>alert('XSS')</script>

Hello

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/

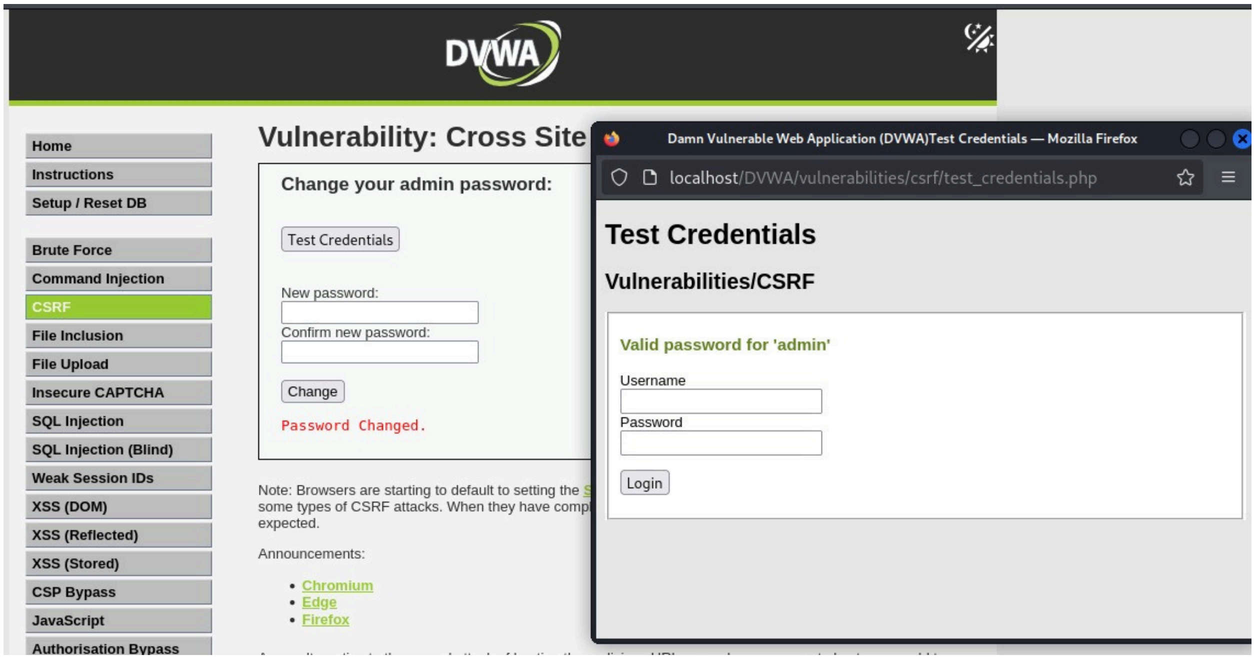
3.Automated Alerts by OWASP ZAP

Vulnerability	Risk	Description
Absence of Anti-CSRF Tokens	High	Forms vulnerable to CSRF attacks
Missing HTTP Security Headers	Medium	Headers like CSP, X-Frame-Options missing
Cookie Security Misconfigurations	Medium	Missing HttpOnly and SameSite flags
Directory Browsing	Low	Access to file directories enabled
Server Version Exposure	Info	Server: header leaks version info
Content Security Policy Not Set	Medium	Allows inline scripts, increasing XSS Risk

- **OWASP Categories Covered:**
  - ○ A05 – Security Misconfiguration
  - ○ A06 – Vulnerable Components
  - ○ A08 – Software & Data Integrity Failures

4. Cross-Site Request Forgery (CSRF)

Tested URL: `http://localhost/DVWA/vulnerabilities/csrf/`
Action Performed: Admin password was changed without authentication using a crafted request
Result: CSRFprotection missing, server processed password change via forged request
Impact: High–Attacker can change sensitive settings or hijack sessions on behalf of an authenticated user
OWASP Category: A01 – Broken Access Control / A05 – Security Misconfiguration



Vulnerability: Cross Site

Change your admin password:

New password:

Confirm new password:

`Password Changed.`

Note: Browsers are starting to default to setting the S
some types of CSRF attacks. When they have comp
expected.

Announcements:

- Chromium
- Edge
- Firefox

Damn Vulnerable Web Application (DVWA)Test Credentials — Mozilla Firefox

localhost/DVWA/vulnerabilities/csrf/test_credentials.php

Test Credentials

Vulnerabilities/CSRF

Valid password for 'admin'

Username

Password

Conclusion:

In this assessment, we successfully identified key vulnerabilities in the DVWA application using tools like OWASP ZAP, Burp Suite, and manual testing. Exploits included SQL Injection, Reflected XSS, and CSRF, demonstrating how attackers can bypass input validation, execute malicious scripts, and change user data without authorization. The scan also revealed missing security headers and cookie misconfigurations. These findings reflect real-world risks and highlight the need for secure coding practices, proper input handling, and session protection in web applications.

Report Prepared By:- Safik Rahman