Web Application Security Assessment Report

Intern Name: Prathyusha Kale  
Intern ID: APSPL\_UID9364  
Task: 3 – Web Application Security  
Organization: ApexPlanet Software Pvt. Ltd.  
Date: 30-10-2025

Introduction

This report presents the results of a web application security assessment conducted using DVWA (Damn Vulnerable Web Application) as part of my internship training. The objective of this task was to explore and understand common web vulnerabilities, learn how they can be exploited, and apply appropriate prevention techniques.

Testing Environment

* Tool Used: DVWA (Practice Web Application)
* Operating System: Kali Linux
* Browser: Mozilla Firefox
* Testing Period: October 2025

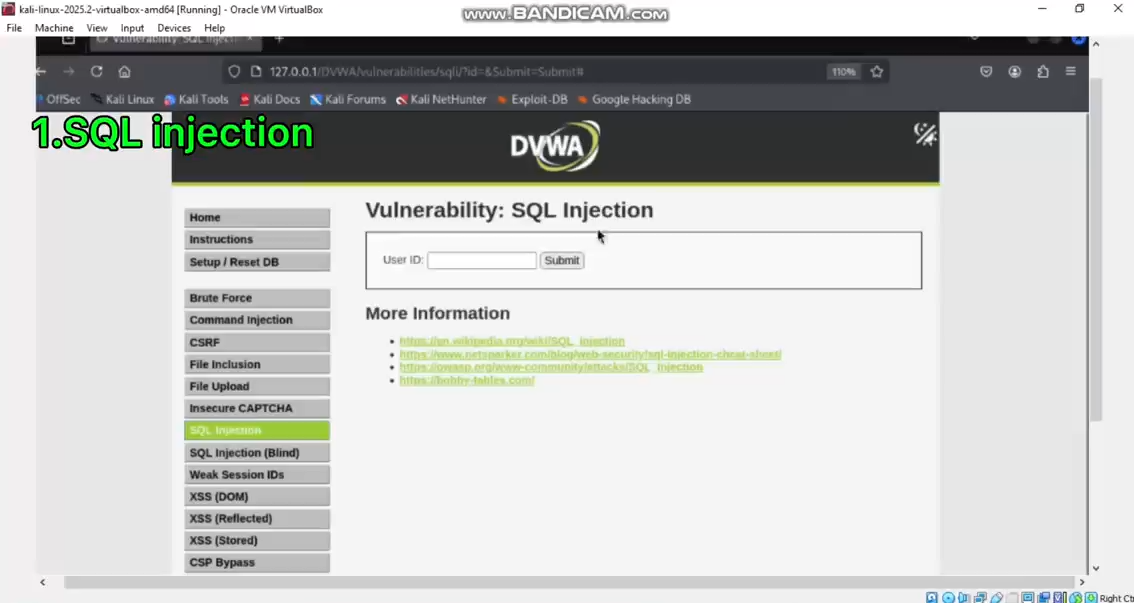
Vulnerabilities Tested

1. SQL Injection

Description:  
SQL Injection (SQLi) occurs when an attacker inserts malicious SQL statements into input fields to manipulate a database query, allowing unauthorized access to sensitive data.

Testing Steps:

1. Opened the SQL Injection page in DVWA
2. Entered the test payload: 1' OR '1'='1
3. Observed that all user records were displayed

[Screenshot 1: SQL Injection Test]

Result:  
At low security level, the application was vulnerable. The attack bypassed authentication and revealed all user data.

Mitigation Measures:

* Use prepared statements and parameterized queries
* Validate and sanitize all user inputs
* Avoid displaying detailed database error messages

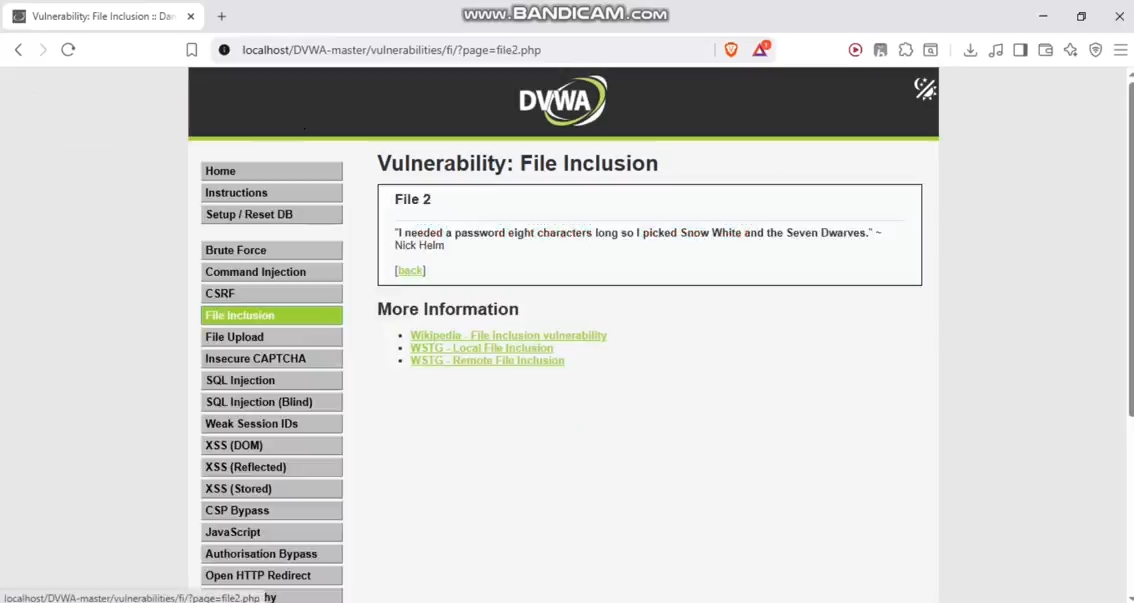
2. Cross-Site Scripting (XSS)

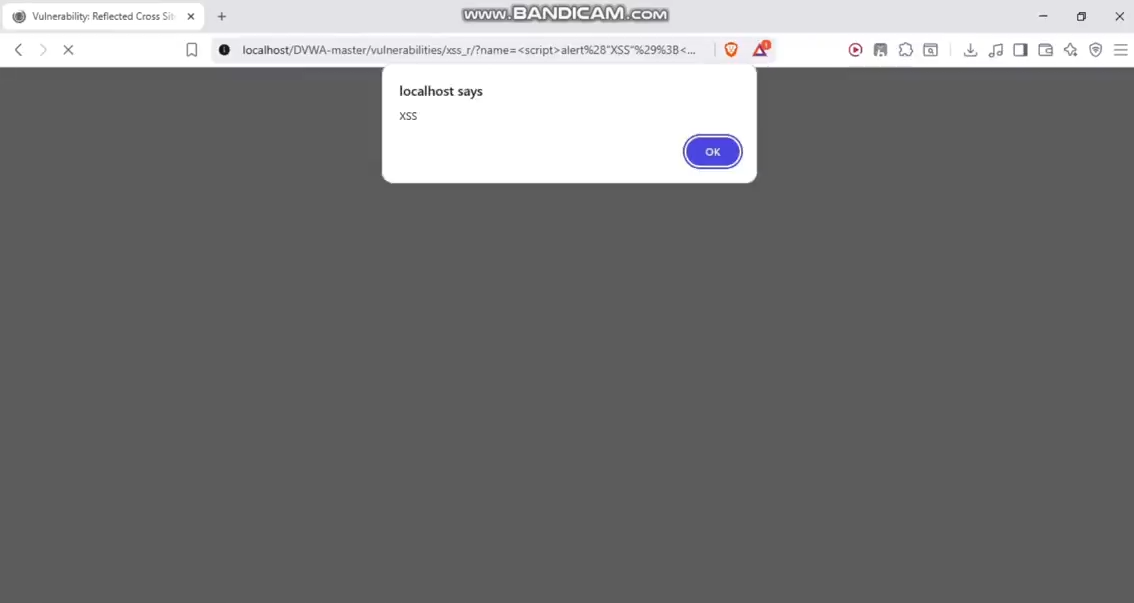
Description:  
Cross-Site Scripting (XSS) allows attackers to inject and execute malicious scripts in a victim’s browser, potentially stealing cookies or session data.

Testing Steps:

1. Tested Reflected XSS using payload: <script>alert("XSS")</script>
2. Tested Stored XSS in the Guestbook section

[Screenshot 2: Reflected XSS Alert]

  
[Screenshot 3: Stored XSS in Guestbook]



Result:  
Both Reflected and Stored XSS vulnerabilities executed successfully at low security level.

Mitigation Measures:

* Filter or escape special characters in user input
* Apply proper output encoding
* Implement a Content Security Policy (CSP)

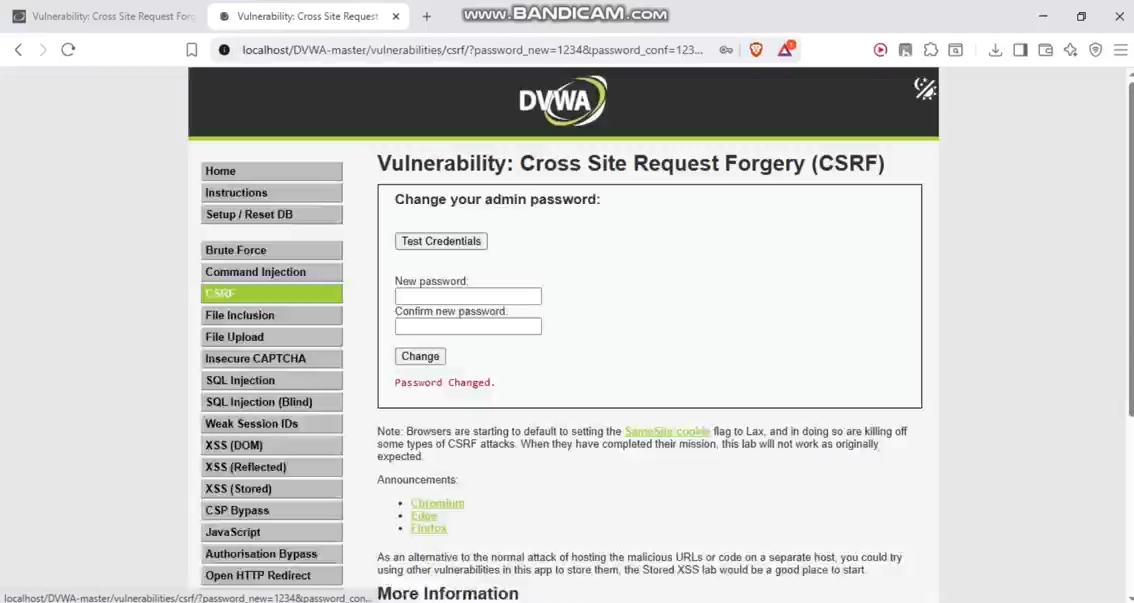
3. Cross-Site Request Forgery (CSRF)

Description:  
CSRF exploits the trust a website places in a user’s browser, tricking authenticated users into performing unwanted actions.

Testing Steps:

1. Created a malicious HTML form to change a user’s password
2. Sent the link to the target user
3. Observed the password change without user consent

[Screenshot 4: CSRF Password Change]



Result:  
At low security level, the password change request was processed without validation, confirming vulnerability.

Mitigation Measures:

* Implement anti-CSRF tokens
* Validate the origin and referrer headers
* Require password re-entry for sensitive actions

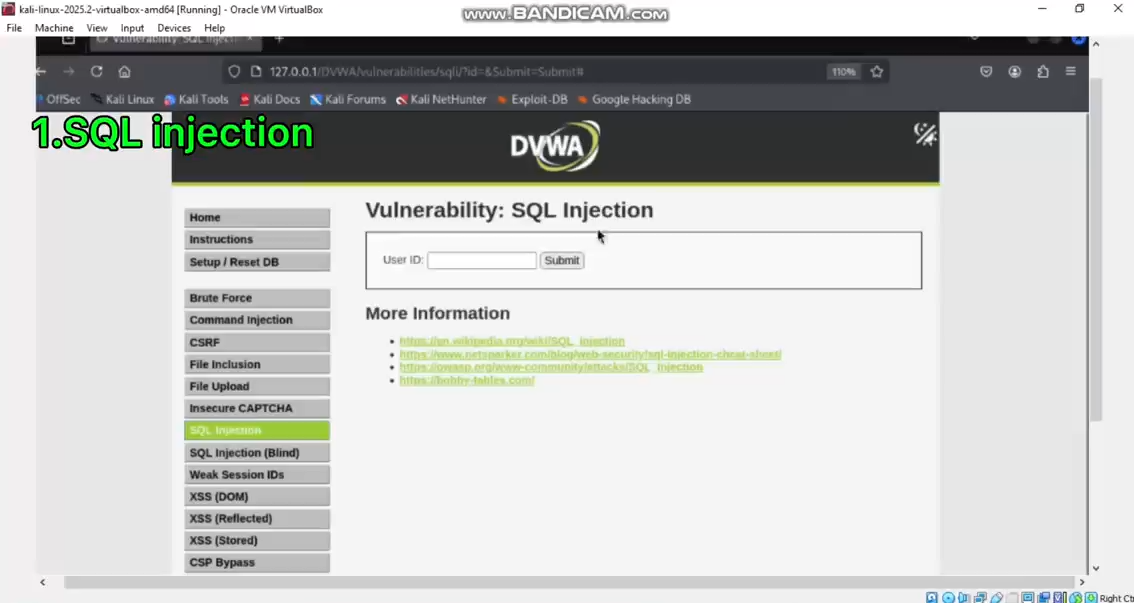
4. File Inclusion

Description:  
File Inclusion vulnerabilities allow attackers to include unauthorized files from the local or remote server, exposing sensitive information.

Testing Steps:

1. Attempted Local File Inclusion using payload: ../../../../../../etc/passwd
2. Successfully accessed restricted system files

[Screenshot 5: File Inclusion Attack]



Result:  
Sensitive system files were accessible at low security level.

Mitigation Measures:

* Restrict file access to predefined allowlists
* Avoid using user input directly in file paths
* Implement strict input validation and sanitization

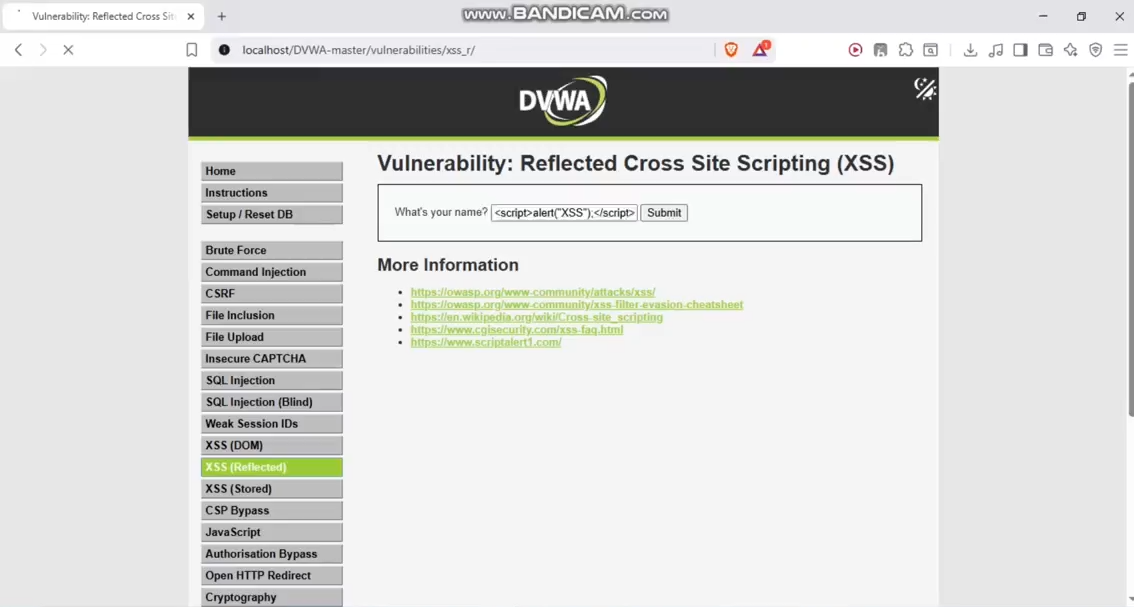
5. Content Security Policy (CSP) Bypass

Description:  
CSP is a browser-level defense mechanism that restricts resources (scripts, styles, etc.) that can load on a web page, reducing the risk of XSS attacks.

Testing Steps:

1. Reviewed existing CSP headers
2. Attempted various CSP bypass techniques

[Screenshot 6: CSP Bypass]



Result:  
Gained an understanding of how CSP enhances browser security and how improper configurations can be exploited.

Summary of Findings

| Vulnerability | Security Level Tested | Result |
| --- | --- | --- |
| SQL Injection | Low | Vulnerable |
| XSS (Reflected) | Low | Vulnerable |
| XSS (Stored) | Low | Vulnerable |
| CSRF | Low | Vulnerable |
| File Inclusion | Low | Vulnerable |

Key Learnings

1. Input Validation is Critical – Always treat user input as untrusted.
2. Defense in Depth – Multiple layers of security significantly reduce risk.
3. Secure Coding Practices – Adhering to secure development standards prevents most attacks.
4. Systematic Testing – Following a structured methodology ensures comprehensive security coverage.

Recommendations

* Validate and sanitize all user inputs.
* Use trusted security libraries and frameworks.
* Keep all software and dependencies up to date.
* Implement robust error handling and logging.
* Conduct regular security testing and code reviews.

Conclusion

This hands-on assessment provided valuable insights into the most common web application vulnerabilities and their mitigations. By experimenting with DVWA across different security levels, I gained practical knowledge of how secure coding practices and layered defenses can effectively prevent attacks. This experience will guide me in writing more secure, resilient web applications in the future.