Task 1 – Web Application Security Report

Name: B Sandhya
Internship: Future Interns – Cyber Security
Date: 25 November 2025

1. Introduction

This report explains the basic security tests I performed on the demo.testfire.net website for Task 1 of the internship. I checked for three common issues: XSS, SQL Injection, and weak login passwords. I also used Burp Suite to capture the traffic and understand how the website responds.

All the screenshots related to these tests are uploaded in the Screenshots folder of my GitHub repository.

2. Scope of Work

For this task, I focused on:

- Testing if the website accepts script input (XSS)
- Checking if login can be bypassed using SQL Injection
- Testing weak/default login credentials
- Intercepting browser requests using Burp Suite

3. Tools Used

- Burp Suite Community Edition
- Google Chrome
- Target site: demo.testfire.net

4. Tests Performed

4.1 Cross-Site Scripting (XSS)

Payload used:

```
<script>alert('XSS')</script>
```

Summary:
I entered the above script in the search bar. The website showed a small popup alert box, which confirms that the site is vulnerable to XSS.

Evidence:

- XSS Vulnerability-1.png
- XSS Vulnerability-2.png

4.2 SQL Injection

Payload used:

```
' OR 1=1 --
```

Summary:
I typed this payload in the login page. The website did not block it properly, which shows that the SQL queries are not fully protected.

Evidence:

- SQL Injection-1.png
- SQL Injection-2.png

4.3 Weak Authentication

Credentials tested:

```
Username: jsmith
Password: Demo1234
```

Summary:
The website allowed weak login attempts without showing any strong password rules or protection. This means the authentication system is weak.

Evidence:

- Weak login-1.png
- Weak login-2.png

4.4 Burp Suite – HTTP History

I used Burp Suite to intercept the requests while testing the site. The HTTP History tab clearly shows the requests made by the browser.

Evidence:

- History Page of Burp Suite.png

5. Recommendations

Based on the tests, here are my suggestions:

- Validate and sanitize user input to prevent XSS
- Use parameterized queries to stop SQL Injection
- Enforce strong password rules
- Block repeated login attempts and add account lockouts
- Improve backend validation to avoid risky behavior

6. Conclusion

The website has multiple security issues including XSS, SQL Injection, and weak authentication. These problems can allow attackers to misuse the site. Fixing them will make the site more secure. All findings are supported with screenshots, which are available in my GitHub repository.