Web Application Security Testing Report

Cyber Security Internship – Future Interns
Track Code: CS
Task: 1 – Web Application Security Testing
Name: Samitha Muthyala
Date: 16.12.2025

---

1. Objective

The objective of this task is to understand the basics of web application security by identifying common vulnerabilities present in web applications. The task focuses on learning ethical hacking concepts such as SQL Injection, Cross-Site Scripting (XSS), and authentication weaknesses using a deliberately vulnerable application. This exercise helps in understanding how attackers exploit vulnerabilities and how developers can mitigate them.

---

2. Scope of Testing

- Testing was performed only on a **deliberately vulnerable web application**.
- No real-world or live websites were tested.
- The testing was conducted in a controlled local environment for educational purposes.

---

3. Target Application

- **Application Name:** OWASP Juice Shop
- **Application Type:** Vulnerable Web Application
- **Deployment:** Localhost
- **URL:** http://localhost:3000

OWASP Juice Shop is an intentionally insecure web application designed for learning and practicing web security concepts.

---

4. Tools and Environment Used

Tool	Description
Kali Linux	Operating system used for security testing
Burp Suite	Used to intercept and analyze HTTP requests
Web Browser (Firefox)	Used to interact with the web application
VirtualBox	Used to run Kali Linux safely

---

5. Methodology

The testing was carried out using a basic penetration testing approach:

1. Application exploration
2. Input testing and manipulation
3. Identifying vulnerabilities
4. Analyzing impact
5. Suggesting mitigation techniques

---

6. Vulnerabilities Identified

6.1 SQL Injection

Description:
SQL Injection is a vulnerability that allows an attacker to interfere with the database queries of an application. This can lead to unauthorized access without valid credentials.

Test Performed:
A malicious SQL payload was entered into the login form.

Payload Used:

' OR 1=1--

Observation:
The application allowed login without verifying valid credentials.

Impact:

- Unauthorized access to user accounts

- Possible data leakage
- Compromise of database integrity

Mitigation:

- Use prepared statements and parameterized queries
- Validate and sanitize user inputs
- Avoid displaying database errors to users

6.2 Cross-Site Scripting (XSS)

Description:
Cross-Site Scripting allows attackers to inject malicious JavaScript code into a web application, which executes in the victim's browser.

Test Performed:
JavaScript code was injected through the search input field.

Payload Used:

<script>alert("XSS")</script>

Observation:
A pop-up alert was displayed, confirming the execution of injected JavaScript.

Impact:

- Session hijacking
- Phishing attacks
- User data theft

Mitigation:

- Encode user inputs before displaying them
- Implement Content Security Policy (CSP)
- Avoid inline JavaScript execution

6.3 Weak Authentication Mechanism

Description:
The application lacks strong authentication controls such as account lockout or CAPTCHA.

Test Performed:
Multiple incorrect login attempts were made without any restriction.

Observation:
The application allowed unlimited login attempts.

Impact:

- Brute-force attacks
- Unauthorized account access

Mitigation:

- Implement account lockout after failed attempts
- Use CAPTCHA
- Enforce strong password policies

---

7. Screenshots

Screenshots were captured as evidence for:

- SQL Injection login bypass
- XSS alert popup
- Login page showing weak authentication

(All screenshots are attached in the GitHub repository.)

---

8. Learning Outcomes

- Understood the basics of web application vulnerabilities
- Learned how SQL Injection and XSS attacks work
- Gained hands-on experience with Burp Suite
- Learned the importance of secure coding practices

---

9. Conclusion

This task provided a practical introduction to web application security testing. Multiple vulnerabilities were identified in the target application, demonstrating how insecure coding practices can be exploited. Implementing proper security controls and following secure development guidelines can significantly reduce security risks.

End of Report