Web Application Security Testing Report

Task: Cyber Security – Task 1
Application Tested: OWASP Juice Shop
Testing Type: Web Application Security Testing

1. Introduction

Web application security is an important part of cybersecurity, as many applications handle sensitive user data. This task focuses on performing basic security testing on a sample web application to understand how common vulnerabilities are identified in real-world scenarios.

For this task, OWASP Juice Shop was used, which is an intentionally vulnerable application designed for learning and practice.

2. Objective

The objective of this task is to perform security testing on a web application and identify common vulnerabilities such as:

- SQL Injection
- Cross-Site Scripting (XSS)
- Security misconfigurations

The task also aims to gain hands-on experience using automated security testing tools.

3. Tools Used

The following tools were used during this task:

- OWASP Juice Shop – Vulnerable web application
- OWASP ZAP – Automated web application security scanner
- Docker – Used to run the application locally
- Web Browser – Used for manual testing

4. Application Setup

OWASP Juice Shop was deployed locally using Docker.
The application was accessed through the following URL:

http://localhost:3000

The application is intentionally vulnerable and is used only for educational and testing purposes.

5. Security Testing Performed

5.1 SQL Injection Testing

Description:
SQL Injection testing was performed on the login functionality by injecting SQL payloads into the input fields.

Observation:
The application detected the malicious input and blocked the request, showing an error message.

Impact:
If such input validation was not present, attackers could bypass authentication and gain unauthorized access.

Mitigation:
Use parameterized queries and proper input validation to prevent SQL Injection attacks.

5.2 Cross-Site Scripting (XSS) Testing

Description:
A reflected XSS payload was tested using the search functionality of the application.

Observation:
The injected script was not executed, indicating that the application sanitized the input.

Impact:
If XSS vulnerabilities are present, attackers could execute malicious scripts in users' browsers.

Mitigation:
Proper input sanitization, output encoding, and implementation of Content Security Policy (CSP).

5.3 Automated Security Scan (OWASP ZAP)

An automated security scan was conducted using OWASP ZAP on the locally running application.

Findings:
The scan identified the following issues:

- Content Security Policy (CSP) Header Not Set
- Information Disclosure
- Cross-Domain Misconfiguration
- Timestamp Disclosure

These findings mainly indicate missing security headers and potential information exposure.

6. Screenshots and Evidence

Screenshots were captured during each phase of testing, including:

- Application running in the browser
- SQL Injection testing
- XSS testing

- OWASP ZAP automated scan
- OWASP ZAP alert details

These screenshots provide evidence of the testing process and results.

7. Conclusion

This task provided practical exposure to web application security testing using both manual techniques and automated tools. It helped in understanding how common vulnerabilities are tested and how security mechanisms work to prevent attacks. The task also highlighted the importance of secure coding practices and proper security configurations in web applications.