Practical 7: Web Application Vulnerability Testing

Aim

To perform web application vulnerability testing on a deliberately vulnerable web application and identify security flaws using a web proxy tool.

Tools & Environment

- Operating System: Kali Linux
- Web Proxy Tool: Burp Suite Community Edition
- Vulnerable Application: OWASP Juice Shop

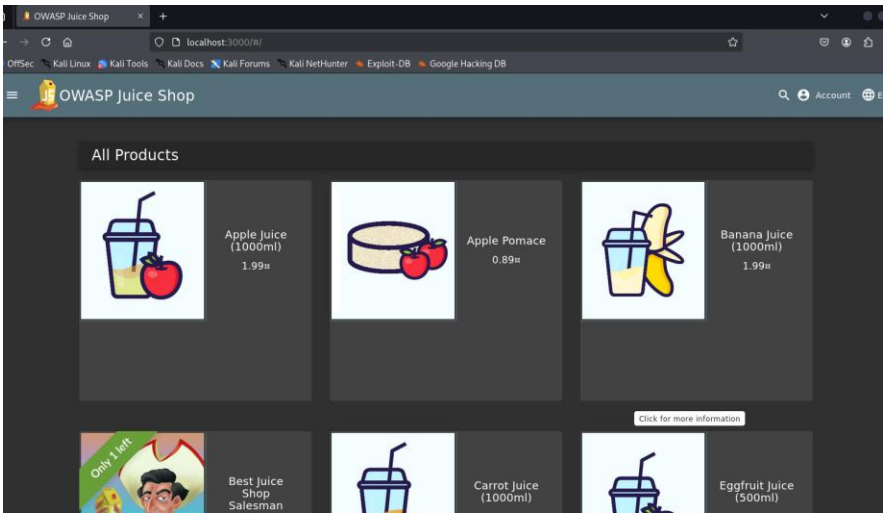
Application Setup

The OWASP Juice Shop application was deployed using a container-based approach. The application was accessed locally through a web browser and verified to be running successfully.

Command used:

podman run -d -p 3000:3000 docker.io/bkimminich/juice-shop

Application URL:http://localhost:3000

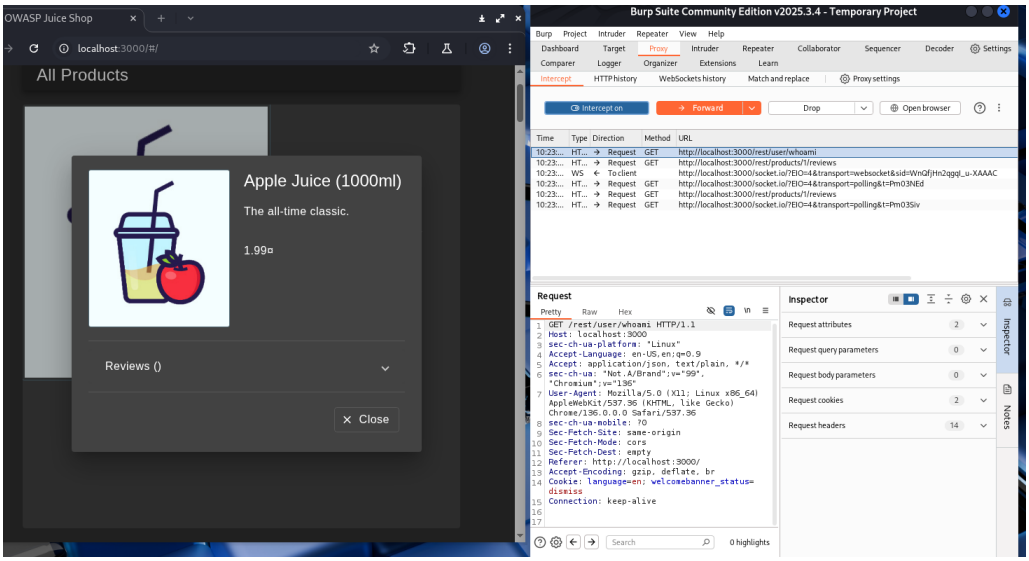


Proxy Configuration

The browser was configured to route all HTTP traffic through the web proxy tool. This allowed interception and modification of requests between the browser and the web application.

Request Interception

After configuring the proxy, HTTP requests from the browser were successfully intercepted using Burp Suite.



SQL Injection Testing

Test Description

The login functionality was tested by modifying input parameters in the intercepted HTTP request.

Payload Used

' OR '1'='1 --

Observation

The application responded abnormally, allowing authentication without valid credentials. This indicates improper input validation.

Cross-Site Scripting (XSS) Testing

Test Description

The search input field was tested by injecting a script payload to check for client-side script execution.

Payload Used

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

Observation

The injected script executed successfully in the browser, confirming the presence of a Cross-Site Scripting vulnerability.

Response Analysis

The application responses differed based on user input. Malicious input resulted in unexpected behavior, confirming the presence of vulnerabilities.

Identified Vulnerabilities

Vulnerability	Risk Level
SQL Injection	High
Cross-Site Scripting (XSS)	Medium

Impact Analysis

- Unauthorized access to application functionality
- Potential exposure of sensitive information
- Execution of malicious scripts in user browsers

Mitigation Measures

- Implement input validation and sanitization
- Use parameterized queries

- Encode output data
- Apply secure coding practices
- Conduct regular security assessments

Conclusion

This practical demonstrated the identification of common web application vulnerabilities using a web proxy tool. The results highlight the importance of secure input handling and defensive coding practices to protect web applications.