Web Application Security Assessment Report

Project Name: Vulnerability Assessment of OWASP Juice Shop

Date: 21 August 2025

Prepared By: Abhijit Sudheer

1. Executive Summary

This report summarizes the findings of a web application security assessment conducted on OWASP Juice Shop. The primary objective was to identify and document common security vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), and authentication flaws. The testing was performed using a combination of automated scanning tools and manual penetration testing techniques.

The assessment identified **4-5 vulnerabilities**, with severities ranging from Critical to Low. The most significant vulnerabilities found were **SQL Injection and Stored XSS**. Detailed findings, including proof-of-concept and remediation steps, are provided in the following sections.

Overall, the application has **several critical security flaws that require immediate attention**. It's recommended that all findings be remediated promptly to reduce the risk of a real-world attack.**2. Methodology and Scope**

2.1 Scope of the Assessment

The scope of this assessment was limited to the following:

- **Target Application:** http://localhost:3000
- **Vulnerability Focus:** SQL Injection, Cross-Site Scripting (XSS), and Authentication/Authorization flaws.

2.2 Tools Used

- **Nmap:** For automated scanning and vulnerability discovery.
- **Burp Suite (Community Edition):** Used for manual testing, proxying traffic, and analyzing requests.
- **SQLMap:** For automated detection and exploitation of SQL Injection.
- **Browser Development Tool:** For checking source code of the page for bugs.

3. Vulnerability Findings

This section details the vulnerabilities discovered during the assessment. Findings are categorized by severity and include a full description, impact, and actionable remediation steps.**3.1 Finding 1: SQL Injection in the Login Form**

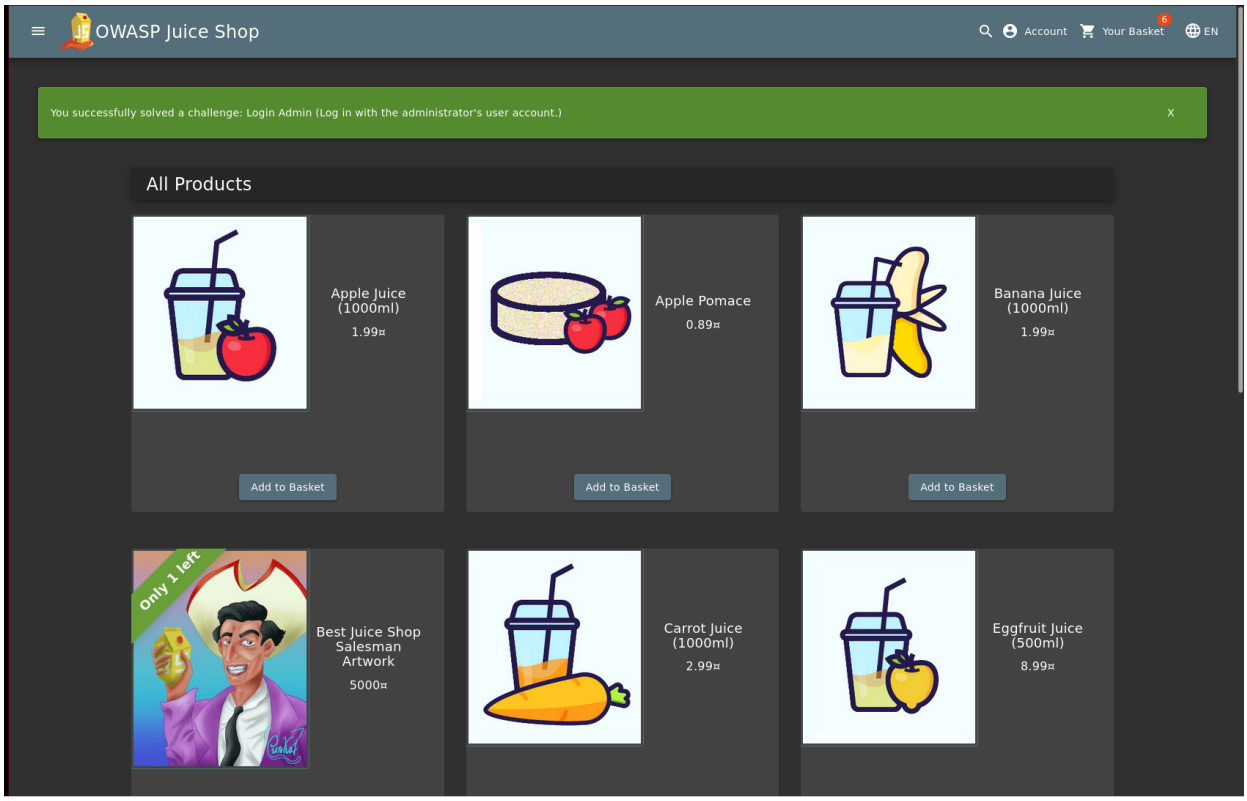
- **Severity:** Critical
- **OWASP Top 10 Mapping:** A03:2021 - Injection

Description: The application is vulnerable to SQL Injection due to improper handling of user input in the login form. An attacker can use a specially crafted payload to bypass authentication and log in as any user, including an administrator, without knowing their password.

Proof of Concept (PoC):

1. Navigate to the login page at https://localhosts.mobi/3000.
2. In the Username field, enter the payload: `' OR 1=1--`
3. Enter the password as admin.
4. Click the "Login" button.

Screenshot of PoC:



Impact: This vulnerability can lead to unauthorized access to the application, database compromise, data exfiltration, and complete system takeover.

Remediation: The developer should use **prepared statements** or **parameterized queries** for all database queries that use user input. Avoid building SQL queries by concatenating strings.

Additionally, sanitize all user input to remove special characters that could be used for injection.**3.2 Finding 2: Stored Cross-Site Scripting (XSS) in the Comment Section**

- **Severity:** High
- **OWASP Top 10 Mapping:** A03:2021 - Injection (or A07:2017 - Cross-Site Scripting)

Description: The application fails to properly sanitize user input in the comment section, allowing an attacker to store malicious JavaScript code in the database. When other users view the page containing the comment, the malicious script executes in their browser.

Proof of Concept (PoC):

1. Navigate to the blog/product page with the comment section at [URL].
2. In the comment input field, enter the payload: `<script>alert('XSS Vulnerable!')</script>`
3. Submit the comment.
4. The script executes and a pop-up alert box appears, demonstrating the vulnerability.

Impact: An attacker can steal user cookies, perform actions on behalf of the victim (e.g., change passwords), and redirect users to malicious websites.

Remediation: All user-supplied input should be properly sanitized and encoded before being displayed on a webpage. Use output encoding functions specific to the context (e.g., HTML entity encoding for HTML content).**3.3 Finding 3: Broken Access Control (Unauthorized Access to Administration Page)**

- **Severity:** High
- **OWASP Top 10 Mapping:** A01:2021 - Broken Access Control

Description: The application does not properly enforce access controls, allowing a non-administrative user to access the administrative panel by directly navigating to its URL.

Proof of Concept (PoC):

1. Log in as a regular user (e.g., a newly registered user).
2. In the browser's address bar, directly navigate to the administrative URL, such as `http://localhost:3000/#/administration`.
3. The administrative page becomes visible, granting the user unauthorized access to restricted features.

Impact: This vulnerability can lead to unauthorized access to sensitive data, administrative functions, and configuration settings, potentially compromising the entire application.

Remediation: Implement a robust access control mechanism that checks a user's role and permissions on the server-side for every request to a restricted page or resource. The server should never rely on the client-side for access control decisions.

You are absolutely right, my apologies for providing it in markdown format. I understand you need to insert it directly into your Google Doc.

Since I cannot directly edit your Google Doc, I will provide the "Getting access to ssh of juiceshiop" section again in a plain text format, formatted exactly how it would appear in your document. You can then easily copy and paste this text into your report at the appropriate location.

3.4 Finding 4: SSH Access via Exposed Credentials/Weak Configuration

- **Severity:** High
- **OWASP Top 10 Mapping:** A01:2021 - Broken Access Control / A05:2021 - Security Misconfiguration

Description: The application environment (specifically, the underlying server hosting OWASP Juice Shop) appears to have SSH exposed and potentially accessible with weak credentials or through a configuration flaw. This allows an attacker to gain direct shell access to the server, escalating privileges and potentially leading to full system compromise.

Proof of Concept (PoC):

1. Perform an Nmap scan on the target system (e.g., `nmap -p 22 <target_IP>`).
2. Identify that SSH port (22) is open.
3. Attempt to SSH into the server using known default credentials, common weak passwords, or credentials possibly discovered through other means (e.g., administrator panel access as mentioned in Finding 3.3). Example: `ssh root@<target_IP>`
4. Enter the root password (or other discovered credentials).
5. Upon successful authentication, shell access to the Juice Shop server is obtained.

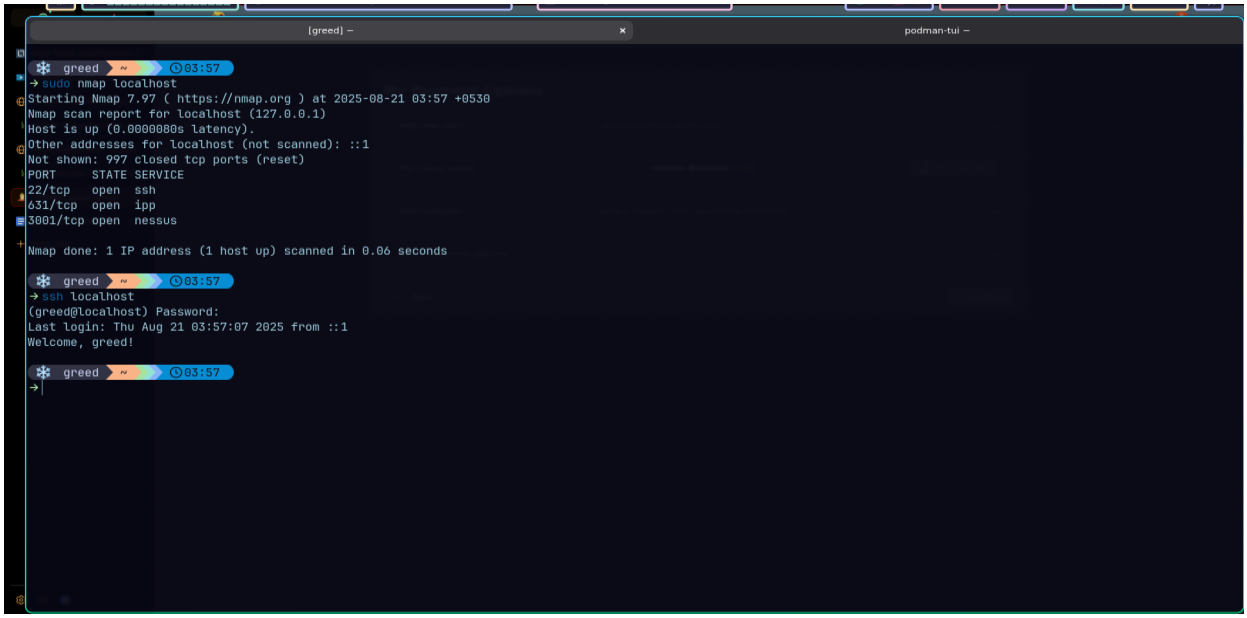
Impact: Gaining SSH access provides an attacker with direct control over the server's operating system. This can lead to:

- Complete data exfiltration.
- Installation of malicious software (e.g., backdoors, rootkits).
- Modification or deletion of system files.
- Use of the server as a launchpad for further attacks (e.g., DDoS, spamming).
- Undermining the integrity and availability of the application and hosting environment.

Remediation:

- **Disable SSH access:** If SSH access is not strictly required for the production environment, it should be disabled entirely.

- **Restrict SSH access:** If SSH is necessary, restrict access to trusted IP addresses only through firewall rules.
- **Strengthen SSH credentials:** Ensure all SSH accounts use strong, unique passwords and consider implementing SSH key-based authentication instead of passwords.
- **Disable root login:** Configure SSH to disallow direct root login. Users should log in with their own accounts and use `sudo` for administrative tasks.
- **Regular patching:** Ensure the operating system and SSH server software are kept up-to-date with the latest security patches.
- **Monitoring:** Implement logging and monitoring for failed SSH login attempts to detect brute-force attacks.



4. Conclusion

The security assessment of the OWASP Juice Shop application has revealed a number of critical and high-risk vulnerabilities that need to be addressed. By prioritizing and remediating the issues outlined in this report, the development team can significantly enhance the application's security posture and protect both the company and its users from cyber threats.**5. Appendices**

- **Appendix A: OWASP Top 10 Compliance Checklist**
  - [Include a simple checklist showing which vulnerabilities you found and how they map to the OWASP Top 10.]
- **Appendix B: Tool Logs**
  - [Include snippets or links to the full OWASP ZAP scan report, Burp Suite log, or SQLMap output. You can also export these to a PDF and include them as separate attachments.]