Black Box Penetration Testing Report

For Intern Intelligence

V1.0

March 10th, 2025

By: Aman Gupta

Report Details

Title	Black Box Penetration Testing Report
Version	V1.0
Author	Aman Gupta
Tester(s)	Aman Gupta
Classification	Confidential

Version Control

Version	Date	Author	Description
V1.0	10/03/2025	Aman Gupta	Final Draft

Table of Contents:

List of Tables

List of Figures

1. Executive Summary

This report details the security assessment conducted on OWASP Juice Shop; a deliberately vulnerable web application designed for security testing. The assessment was carried out to identify and exploit vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), brute force attacks, credential access, price manipulation, administrative file access, and improper error handling. The goal was to assess the application's security posture and provide recommendations for mitigating risks.

1.1 Scope of Work

This security assessment covers the remote penetration testing of an accessible OWASP Juice Shop ,Try hack me , Port Swigger Web Application. The assessment was carried out from a black box perspective, with the only supplied information was the name of the web application. No other information was assumed at the start of the assessment.

Penetration Testing	Start Date/Time	End Date/Time

Pen test 1	18/02/2025	22/02/2025

1.2 Project Objectives

This security assessment is carried out to gauge the security posture of OWASP's Juice shop. The result of the assessment is then analyzed for vulnerabilities. Given the limited time that is given to perform the assessment, only immediately exploitable services have been tested. The vulnerabilities are assigned a risk rating based on threat, vulnerability and impact.

1.3 Assumption

While writing the report, it was assumed that the given Web application is considered open to public.

1.4 Timeline

The timeline of the test is as below:

Table 1 Penetration Testing Time Line

1.5 Summary of Findings

Value	Number of Risks
Low	0
Medium	3
High	4
Critical	2

Table 2 Total Risk Rating

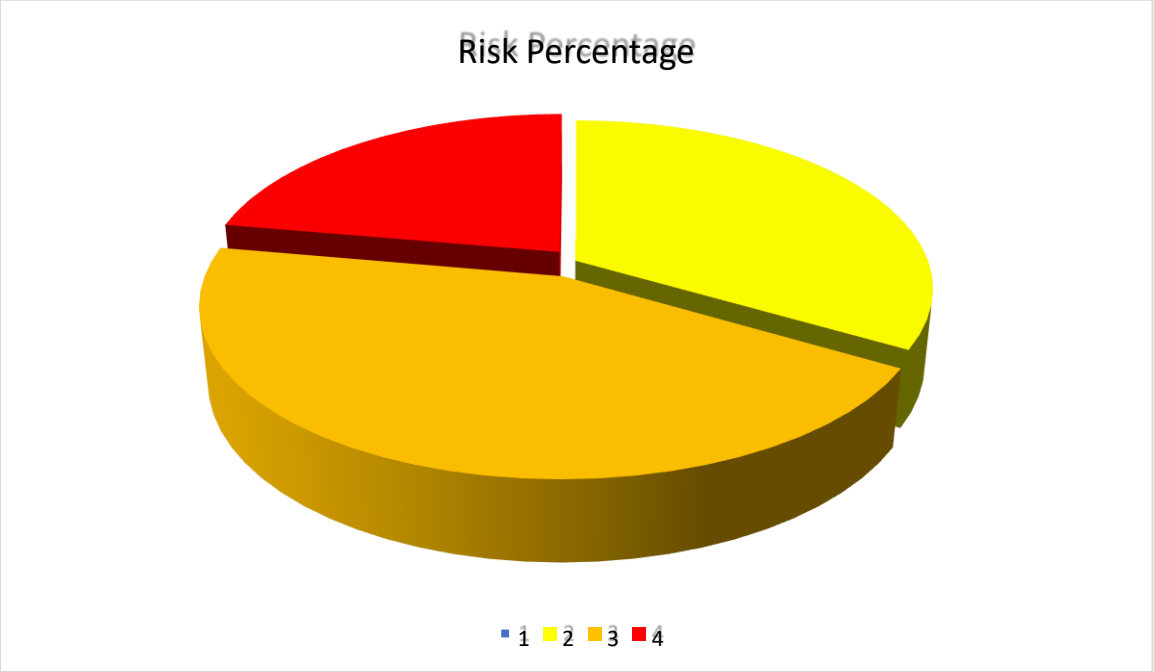


Figure 1 Total Risks

SQL Injection: Implement prepared statements and parameterized queries to prevent SQL injection attacks.

IDOR: Enforce proper access controls and implement authorization checks before granting access to sensitive resources.

Admin Panel Exposure: Restrict access to the admin panel using IP whitelisting, VPN, or strong authentication mechanisms.

Stored XSS: Sanitize and validate user input properly, and implement Content Security Policy (CSP) to mitigate XSS risks.

Outdated jQuery: Upgrade jQuery to the latest stable version and regularly patch dependencies.

Credentials in Plaintext: Use HTTPS (TLS encryption) for secure transmission of credentials and sensitive data.

Open Port 3000: Identify the service running on port 3000, disable unnecessary services, and restrict access using firewall rules.

2. Methodology

2.1 Planning

During planning the information was gathered through public sources to learn about target:

- Technical Infrastructure
- Common Vulnerabilities and Exposures

Then, the running services and its versions were determined and detected.

2.2 Exploitation

Utilizing the information gathered in Planning, started to find the vulnerability for every service that we discovered after that trying to exploit it.

2.3 Reporting

Based on the results from the first two steps, start analysing the results. The risk rating is based on this calculation:

Risk= Threat * Vulnerability * Impact

Threat		Low				Medium				High				Critical			
Vulnerability		L	M	H	C	L	M	H	C	L	M	H	C	L	M	H	C
Impact	Low	1	2	3	4	1	4	6	8	3	6	9	12	4	8	12	16
	Medium	2	4	6	8	4	8	12	16	6	12	18	24	8	16	24	32
	High	3	6	9	12	6	12	18	24	9	18	27*	36	12	24	36	48
	Critical	4	8	12	16	8	16	24	32	12	24	36	48	16	32	48	64

Table 3 Risk Analysis

L	Low	1-16
M	Medium	17-32
H	High	33-47
C	Critical	48-64

Table 4 Rating Calculation

After calculating the risk rating, we start writing the report on each risk and how to mitigate it.

**Based on my analysis risks that falls under this category will be considered as High.*

3. Detail Findings

3.1 Detailed Web App Information

1. Open Port 3000 (Unidentified Service)

Figure 3 Nmap – Open Ports

Impact:

Medium

Risk Rating:

High

Threat Level:

High

Analysis:

- Port 3000 is open and detected as "ppp?" (potentially unknown service).
- If Juice Shop is running, it might expose admin functions without authentication.
- A vulnerable API may allow SQL injection, IDOR, or privilege escalation attacks.

CVE:

CVE-2017-16089: Unauthenticated API access in certain Node.js applications.

CVE-2022-21661: Prototype pollution vulnerability in Express.js-based apps.

Recommendations:

Restrict Port Access**:** Use firewall rules (e.g., iptables or UFW) to limit exposure.

Implement Authentication**:** Ensure proper access control on any web services**.**

Disable Unused Services**:** If the port is not required**,** close it to reduce attack surface.

Update and Patch: If running Node.js/Express, ensure all dependencies are patched**.**

2. SQL Injection via login bypass

```
"email":"admin' or 1=1--",
"password":"admin' or 1=1--"
```

Figure 4 Burp Suite SQL Payload

Impact:

High - Can allow unauthorized access to admin accounts.

Risk Rating:

Critical

Threat Level:

Critical

Analysis:

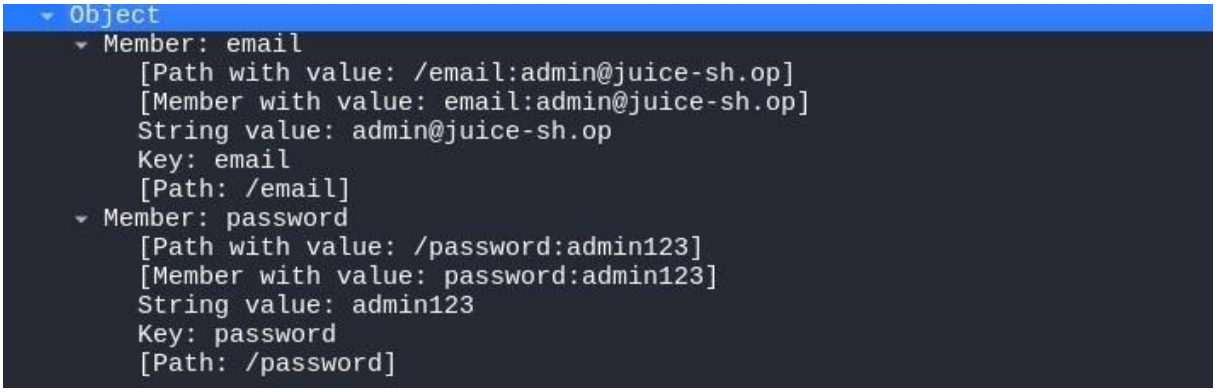
- SQL injection occurs when unvalidated user input is directly executed in SQL queries.
- This can lead to authentication bypass, data exfiltration, and even full system compromise.

CVE: CVE-2022-40888

Recommendation:

- Use prepared statements (parameterized queries).
- Implement input validation and allow-list filtering.
- Use Web Application Firewalls (WAF) to block malicious SQL patterns.

3. Credentials Sent in Plaintext Figure 5 WireShark – Packet Sniffing



4. Brute force Impact:

High

Risk Rating:

High

Threat Level:

Critical

Analysis:

- Login request sends email:admin@juice-sh.op and password:admin123 in plaintext.
- No HTTPS/TLS encryption, making credentials vulnerable to sniffing.
- Likely a misconfiguration or lack of proper security enforcement.

CVE: CVE-2017-8917

Recommendations:

- Enforce HTTPS: Implement TLS (SSL) to encrypt login credentials.
- Use Secure Authentication: Implement OAuth, JWT tokens, or hash passwords before transmission.
- Security Headers: Add Strict-Transport-Security (HSTS) & X-Content-Type-Options: no sniff.
- Monitor Traffic: Use tools like Wireshark or Burp Suite to detect insecure transmissions.

5. Insecure Direct Object Reference (IDOR)

Vulnerability: Access to user data without authentication.

Impact:

High – Can lead to unauthorized access to user accounts.

Risk Rating:

High

Threat Level:

High

Analysis:

- IDOR occurs when an attacker can manipulate object identifiers (e.g., user IDs) to gain unauthorized access to resources. **CVE:** CVE-2021-35449 **Recommendation:**
- Implement access control checks at the API level.
- Use session-based authentication with proper authorization mechanisms.
- Log and monitor API access to detect suspicious activity.

6. Unprotected Admin Panel Exposure

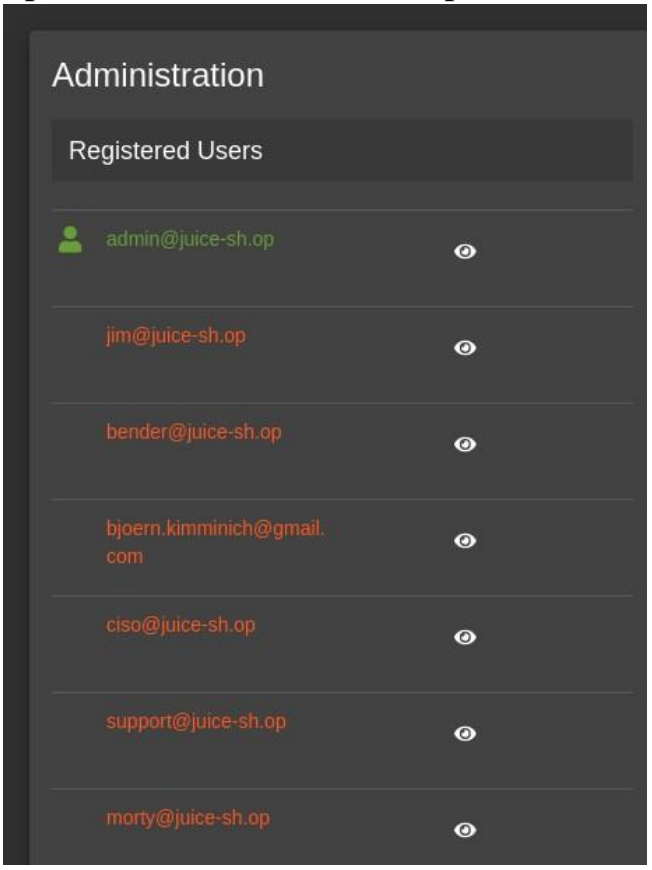


Figure 6 Target Web App Admin Panel

Impact:

High

Risk Rating:

High

Threat Level:

High

Analysis:

- An exposed admin panel allows attackers to enumerate user emails and potentially escalate privileges.

CVE: Not specific, but similar to CVE-2023-28369.

Recommendation:

- Restrict admin panel access with IP-based whitelisting.
- Require strong authentication (e.g., MFA) for admin access.
- Implement role-based access control (RBAC).

7. Cross-Site Scripting (XSS) Vulnerability:

- xss attack on search bar
- <script> tag has been filtered out but and other tag not
- <img src=x onerror=alert('hacked')>
- <iframe src="javascript:alert('XSS')"> </iframe>
- <form action="https://timely-dragon-0022e1.netlify.app/" method="POST">

<input type="submit" value="Click Me">

</form> executed

on the website.

Impact:

Medium

Risk Rating:

Medium

Threat Level:

Medium

Analysis:

- XSS occurs when an attacker injects malicious JavaScript, which executes in a victim's browser.

CVE: CVE-2023-38361

Recommendation:

- Implement output encoding (e.g., use htmlspecialchars() in PHP).
- Enforce Content Security Policy (CSP) to restrict script execution.
- Use HTTP-only cookies to prevent session hijacking.

Summary of Findings

Vulnerability	Impact	Risk Rating	Threat Level
SQL Injection (Search Bar)	Critical	Critical	High
Brute Force Attack	High	High	High
Credential Access via IDOR	High	High	High
Price Manipulation	High	High	High
Admin File Access & Modification	High	Medium	Medium
Cross-Site Scripting (XSS)	Medium	Medium	Medium
Improper Error Handling	Medium	Medium	Medium

4. References

https://nvd.nist.gov/vuln