VAPT Testing Report

For Intern Intelligence

Version: 1.0
Date: 3rd FEB 2025
Author: SHAIKH LIYAKAT

Report Details:

Title	Black Box Penetration Testing Report
Version	V1.0
Author	Shaikh Liyakat
Tester(s)	Shaikh Liyakat

Classification	Confidential

Version Control:

Version	Date	Author	Description
V1.0	22/02/2025	Shaikh Liyakat	Final Draft

Table of Contents:

List of Illustrations

List of Tables

List of Figures

Methodology

The review was conducted manually by:

1. **Reviewing Key PHP Files:** Examining the source code for database queries, user input handling, file inclusion, and session management.
2. **Identifying Vulnerabilities:** Matching code practices against the OWASP Top 10 vulnerabilities.
3. **Code Comparison:** Documenting both the vulnerable (old) code and the secure (fixed) code along with explanations.
4. **Risk Rating:** Assigning severity based on the potential impact and likelihood of exploitation.

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.

2.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.

3.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.

4.Assumption:

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

Reporting

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

Risk= Threat * Vulnerability * Impact

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

OWASP Top 10 2017		change	OWASP Top 10 2021 proposal	
A1	Injections	**as is**	A1	Injections
A2	Broken Authentication	**as is**	A2	Broken Authentication
A3	Sensitive Data Exposure	**down 1**	A3	Cross-Site Scripting (XSS)
A4	XML eXternal Entities (XXE)	**down 1 + A8**	A4	Sensitive Data Exposure
A5	Broken Access Control	**down 1**	A5	Insecure Deserialization **(merged with XXE)**
A6	Security Misconfiguration	**down 4**	A6	Broken Access Control
A7	Cross-Site Scripting (XSS)	**up 4**	A7	Insufficient Logging & Monitoring
A8	Insecure Deserialization	**up 3 + A4**	A8	**NEW:** Server Side Request Forgery (SSRF)
A9	Known Vulnerabilities	**as is**	A9	Known Vulnerabilities
A10	Insufficient Logging & Monitoring	**up 3**	A10	Security Misconfiguration

1. SQL Injection via login bypass:

email = ' OR 1=1--

password = ' OR 1=1--

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.

2.Brute force:

Common wordlist Refrence:
https://github.com/danielmiessler/SecLists/blob/master/Passwords/Default-Credentials/default-passwords.txt

Impact:

High

Risk Rating:

High

Threat Level:

Critical

Analysis:

- Login request sends
{"email": " admin@juicesh.op","password":"admin123"}

- 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.

3. 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.

4.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:

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

CVE: CVE-2023-38361

Recommendation:

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

5. References

https://nvd.nist.gov/vuln