



# Penetration Test

**XYZ Company**

**Report of Findings**

PENETRATION TESTING REPORT TEMPLATE

## Contents

Document Control	2
CONFIDENTIALITY	2
DISCLAIMERS	2
Proprietary Information	3
Executive Summary	3
Assessment Summary	4
Strategic Recommendation	4
Technical Summary	5
Finding Severity Ratings	5
Scope	6
Scope Exclusions	6
Post Assessment Clean-up	7
Findings Overview	7
Technical Detail	7
Vulnerability 1:	7
Vulnerability 2:	9
Vulnerability 3:	11
Vulnerability 4:	13
Vulnerability 5:	15
Vulnerability 6:	17
Summary of Findings	19
Conclusion	20
Appendices	21

## Document Control

### CONFIDENTIALITY

Geez Security (GTST) shall not be held responsible for any special, incidental, indirect, or consequential damages resulting from the use of this information. The contents of this document are the confidential and proprietary property of **Solomon Tesfaye**. Unauthorized distribution of this document or its contents is strictly prohibited.

Geez Security grants the designated client point of contact permission to review and share this document as deemed appropriate, following **Geez Security's** data handling policies. This document must be labeled "**CONFIDENTIAL**" and shared only with individuals who have a legitimate "**need to know.**"

Address questions regarding the proper and legitimate use of this document to:

Geez Security

Attention: Contracts Manager

### DISCLAIMERS

The information in this document is provided "**as is**" with no guarantees or warranties. Vulnerability assessments reflect a **snapshot in time**, meaning the environment may have changed since testing was conducted. Additionally, new vulnerabilities may have emerged after this report was generated.

As a result, this report should be treated as a **guideline**—not an absolute or complete assessment of the risks to your systems, networks, or applications

### Proprietary Information

The content of this document is considered proprietary information and should not be disclosed outside of the recipient organization's network. Geez Security gives permission to copy this report for the purposes of disseminating information within your organization or any regulatory agency

Contact information		
name	title	Contact information
Geez Security		
Solomon Tesfaye Ensermu	Junior Ethical pentester	Email:solomontesfaye654@gmail.com office:(+251) 942347974

### Executive Summary

**Geez Security** assessed the external security posture of the web application through an **external network penetration test** conducted from **June 29th, 2025** to **July 3rd, 2025**.

## PENETRATION TESTING REPORT TEMPLATE

By simulating real-world attack techniques, **critical vulnerabilities** were identified, which could allow an attacker to gain **full internal network access** to the **[Company Name]** office. Immediate remediation is strongly advised to mitigate these risks.

Six vulnerabilities were discovered and exploited, ranging from **Critical (SQL Injection)** to **Medium (Security Question Bypass)** severity based on CVSS v3 scoring. These findings demonstrate the importance of secure coding practices and the risks posed by improper input validation and poor authentication controls.

### Assessment Summary

Based on the security assessment for **Owasp Juice shop** web applications the current status of the identified vulnerabilities set the risk at a "CRITICAL" level, which if not addressed in time (strongly recommended before going in a live production environment), these vulnerabilities could be a trigger for a Cyber security breach. These vulnerabilities can be easily fixed by following the best practices and recommendations given throughout the report. The following table represents the penetration testing in-scope items and breaks down the issues, which were identified and classified by severity of risk. (note that this summary table does not include the informational items):

Vulnerability	CVSS v3 Score	Severity
SQL Injection – Admin Login Bypass	9.8	Critical
Broken Authentication – Brute Force Attack	8.8	High
Insecure Direct Object Reference (IDOR)	7.5	High
Persistent XSS – Last Login IP	7.4	High
Directory Listing – Sensitive File Exposure	6.5	Medium
Security Question Bypass – Jim’s Account	6.1	Medium

Phase	Description	Critical	High	Medium	Total
1	Web/API Penetration Testing	1	3	2	6
	Total	1	3	2	6

### Strategic Recommendation

Immediate focus should be placed on fixing the **Critical and High severity vulnerabilities**, especially SQL Injection and Authentication issues. The organization should also:

- Implement secure coding guidelines (OWASP Top Ten).
- Conduct regular vulnerability assessments.

## PENETRATION TESTING REPORT TEMPLATE

- Use input validation libraries and frameworks.
- Implement multi-factor authentication and better session management

## Technical Summary

This test covered:

- Authentication weaknesses
- Input validation flaws
- Access control failures
- Improper data protection
- Information disclosure vulnerabilities

Tools Used:

- Burp Suite (Intruder, Repeater)
- Nmap
- Web Browsers
- TryHackMe CTF environment

## Finding Severity Ratings

The table below gives a key to the risk naming and colors used throughout this report to provide a clear and concise risk scoring system. It should be noted that quantifying the overall business risk posed by any of the issues found in any test is outside our scope. This means that some risks may be reported as high from a technical perspective but may, as a result of other controls unknown to us, be considered acceptable by the business.

Severity	CVSS V3 Score Range	Definition
<b>Critical</b>	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
<b>High</b>	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
<b>Moderate</b>	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan

## PENETRATION TESTING REPORT TEMPLATE

Severity	CVSS V3 Score Range	Definition
		of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

## Scope

Assessment	Details
External Penetration Test	10.16.0.0/8, 10.16.1.0/8

- Target: OWASP Juice Shop (TryHackMe Instance)
- Testing Date: 07/01/2025
- Test Type: Black Box Web Application Penetration Test

## Scope Exclusions

- No live production systems were tested.
- Physical and social engineering attacks were not in scope.

## PENETRATION TESTING REPORT TEMPLATE

## Post Assessment Clean-up

- Test accounts and sessions were terminated.
- No persistent payloads or backdoors remain.

## Findings Overview

All the issues identified during the assessment are listed below with a brief description and risk rating for each issue. The risk ratings used in this report are defined in Risk Ratings Section

Ref id	Findings	Status	Risk/Severity
ref-1-1	SQL Injection – Admin Login Bypass	Open	CRITICAL
ref-2-1	Broken Authentication – Brute Force Attack	Open	HIGH
ref-2-2	Insecure Direct Object Reference (IDOR)	Open	HIGH
ref-2-3	Persistent XSS – Last Login IP	Open	HIGH
ref-3-1	Directory Listing – Sensitive File Exposure	Open	MEDIUM
ref-3-2	Security Question Bypass – Jim’s Account	Open	MEDIUM

## Technical Detail

## Vulnerability 1:

Ref id	ref-1-1
Vulnerabilities name	SQL Injection – Admin Login Bypass
Risk/Severity:	CRITICAL



PENETRATION TESTING REPORT TEMPLATE

Vulnerability Explanation	<p>Affects: Login Authentication System of OWASP Juice Shop</p> <p>Parameter(s): Email and Password input fields on the login page</p> <p>Attack Vectors: Web login form</p> <p>References: OWASP Top 10 - A1:2017 Injection <a href="#">OWASP SQL Injection Cheat Sheet</a></p>
Vulnerability Description	<p>During testing, it was discovered that the login form on the OWASP Juice Shop application was vulnerable to SQL Injection. By using the classic SQL injection payload ' OR 1=1--, the attacker successfully bypassed authentication and gained unauthorized access to the administrator account. This exposes the system to serious risks, including data leakage, unauthorized changes, and complete compromise of user accounts.</p>
proof	<pre> 1 POST /rest/user/login HTTP/1.1 2 Host: oswap 3 User-Agent: Mozilla/5.0 (X11; Linux   Firefox/128.0 4 Accept: application/json, text/plair 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/json 8 Content-Length: 39 9 Origin: http://oswap 10 Connection: keep-alive 11 Referer: http://oswap/ 12 Cookie: io=HISaFKkL50u5aIToAAAZ; lar 13 Priority: u=0 14 15 {   "email": "' OR 1=1 --",   "password": "xx" } </pre> <p><i>Response:</i> HTTP 200 OK (Login Successful)  </p>
Recommendation	<p><b>Who:</b> IT Team / Development Team</p> <p><b>Action:</b></p> <ul style="list-style-type: none"> <li>- Implement input validation and sanitization on all user input fields.</li> <li>• Use parameterized queries (prepared statements) to prevent SQL injection.</li> <li>• Employ a web application firewall (WAF) to detect and block SQL injection attempts.</li> <li>• Conduct regular code reviews and security testing on input-handling functions.</li> </ul>

PENETRATION TESTING REPORT TEMPLATE

	<ul style="list-style-type: none"> <li>Implement multi-factor authentication for administrative accounts.</li> </ul> <p><b>Action 1:</b> Restrict login attempts to prevent brute-force and automated SQL Injection Attempts</p> <p><b>Action 2:</b> Update password policy (this is more for the weak password issue but can also help here for defense in depth):</p> <ul style="list-style-type: none"> <li>Minimum of 14 characters</li> <li>No dictionary words or proper names</li> <li>Unique passwords for each system</li> </ul> <p><b>Item 3:</b> Synchronize error messages for invalid logins to prevent user enumeration and reduce SQLi information disclosure.</p> <p><b>Additionally:</b></p> <ul style="list-style-type: none"> <li>Train developers on secure coding practices.</li> <li>Use static and dynamic analysis tools to detect input validation flaws.</li> <li>Periodically test web applications for injection vulnerabilities</li> </ul>
--	---

Vulnerability 2:

Ref id	ref-2-1
Vulnerabilities name	Broken Authentication – Brute Force Attack
Risk/Severity:	<b>HIGH</b>
Vulnerability Explanation	<p>Affects: Login Authentication System</p> <p>Parameter(s): Password field on the login form</p> <p>Attack Vectors: Burp Suite Intruder</p> <p>References: OWASP Top 10 - A2:2017 Broken Authentication <a href="#">OWASP Brute Force Protection Cheat Sheet</a></p>
Vulnerability Description	The OWASP Juice Shop login endpoint does not restrict the number of failed login attempts. This allowed Geez Security to perform a brute-force

## PENETRATION TESTING REPORT TEMPLATE

attack using the Burp Suite Intruder tool and a password wordlist. The attack successfully discovered the administrator password without being blocked or throttled by the application. This indicates a critical weakness in the authentication mechanisms.

proof

For the payload, we will be using the best1050.txt from Seclists. (Which can be installed via: apt-get install seclists)

You can load the list from: /usr/share/wordlists/SecLists/Passwords/Common Credentials/best1050.txt

Once the file is loaded into Burp, start the attack. You will want to filter for the request by status.

A failed request will receive a 401

Unauthorized Whereas a successful request will return a 200 OK.

Once completed, login to the account with the password.

We going to intruder.

```
13 Priority: u=0
14
15 {"email": "admin@juice-sh.op", "password": "5test5"}
```

The screenshot displays the Burp Suite Intruder tool interface. At the top, there is a 'Start attack' button. Below it, the 'Payloads' tab is active, showing a list of payloads. A red arrow points to the 'Load items from file' button. The list of payloads includes 'admin000' through 'admin008'. Below the payloads list, a table shows the results of the attack. The table has three columns: 'Request', 'Payload', and 'Status code'. The status codes for requests 121 through 127 are 401, while request 124 has a status code of 200, indicating a successful login. A red arrow points to the '200' status code.

Request	Payload	Status code
121	admin120	401
122	admin121	401
123	admin122	401
124	admin123	200
125	admin124	401
126	admin125	401
127	admin126	401
128	admin127	401

We find password. Success!!

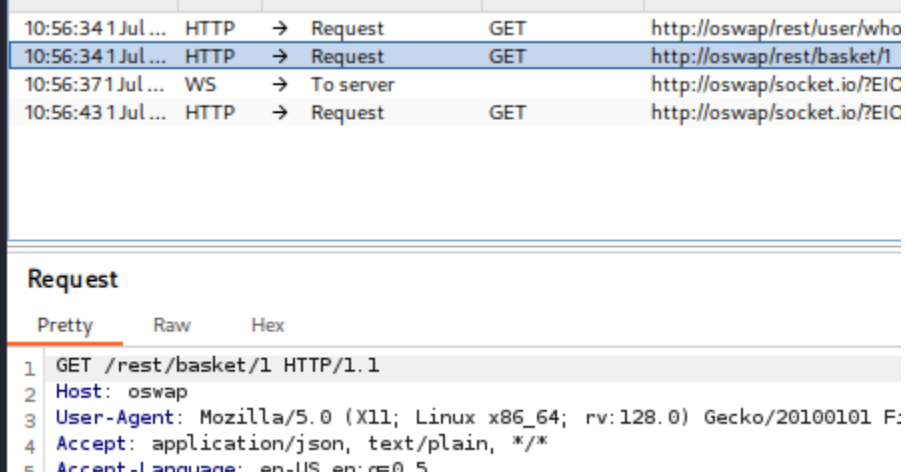
PENETRATION TESTING REPORT TEMPLATE

Recommendation	<p><b>Who: IT Team</b></p> <p><b>Action:</b> Implement account lockout or exponential backoff after a small number of failed login attempts (e.g., 5 failed attempts).</p> <ul style="list-style-type: none"> <li>▪ Deploy CAPTCHA after failed logins to prevent automated attacks.</li> <li>▪ Monitor for unusual login activity.</li> <li>▪ Review and enhance password policies.  </li> </ul> <p><b>Action 1:</b>OWA permitted unlimited login attempts. Geez Security recommends restricting logon attempts against their service.</p> <p>Action 2: The Juice Shop login system permitted a successful login via a password spraying attack, signifying a weak password policy. Geez Security recommends the following password policy per the Center for Internet Security (CIS):</p> <ul style="list-style-type: none"> <li>▪ 14 characters or longer</li> <li>▪ Use different passwords for each account accessed</li> <li>▪ Do not use words and proper names in passwords, regardless of language</li> </ul> <p>Item 3 The Juice Shop application permitted user enumeration (response status clearly showed account existence). Geez Security recommends synchronizing valid and invalid login messages.</p> <p>Additionally, Train employees on how to create strong passwords</p> <ul style="list-style-type: none"> <li>▪ Check employee credentials against known breached password databases (e.g., HaveIBeenPwned API).</li> <li>▪ Discourage employees from using work email addresses as usernames for non-corporate services.</li> </ul>
----------------	--

Vulnerability 3:

Ref id	ref-2-2
Vulnerabilities name	Insecure Direct Object Reference (IDOR)
Risk/Severity:	<b>HIGH</b>

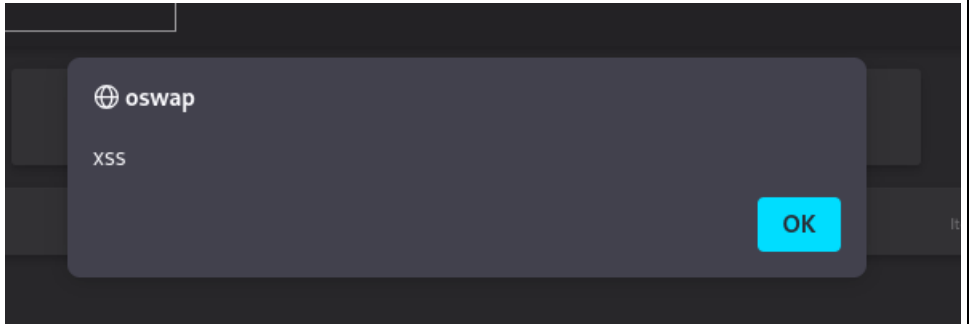
# PENETRATION TESTING REPORT TEMPLATE

Vulnerability Explanation	<p>Affects: Broken Access Control — allows unauthorized users to access objects (like user baskets) by manipulating object references in requests.</p> <p>Parameter(s): /rest/basket/&lt;id&gt; (e.g., /rest/basket/1)</p> <p>Attack Vectors: Horizontal Privilege Escalation — an attacker changes the ID in the request to view or manipulate data belonging to another user.</p> <p>References: THM Juice Shop - Task 6, Question #2: View another user's shopping basket</p>
Vulnerability Description	<p>The application fails to verify user authorization when accessing sensitive object references, such as shopping basket IDs. This allows an attacker to tamper with the ID parameter and access other users' data.</p>
proof	<p>we are going to change the number 1 after /basket/ to 2</p>  <p>The screenshot shows a network traffic capture with four entries. The second entry is highlighted, showing a GET request to http://oswap/rest/basket/1. Below the capture, a detailed view of the request is shown, including the URL, host, user-agent, and accept headers.</p>
Recommendation	<p><b>Who: IT Team</b></p> <p><b>Action:</b></p> <p><b>Action 1:</b> Ensure object access is always validated server-side against the current user's session or role.</p> <p><b>Action 2:</b> Implement access control checks before processing any object reference.</p> <p><b>Action 3:</b> Avoid exposing direct object IDs when possible; use indirect references or tokens.</p> <p><b>Additional Security Measures:</b></p>

PENETRATION TESTING REPORT TEMPLATE

	<ul style="list-style-type: none"> <li>○ Implement logging for unusual access patterns to catch abuse</li> <li>○ Conduct regular access control reviews in code and API design</li> <li>○ Educate developers on OWASP top 10, especially broken access control</li> </ul>
--	---

Vulnerability 4:

Ref id	ref-2-3
Vulnerabilities name	Persistent XSS – Last Login IP
Risk/Severity:	<b>HIGH</b>
Vulnerability Explanation	<p><b>Affects:</b> Server-side input handling of HTTP headers (specifically True-Client-IP)</p> <p>Improper sanitization allows stored JavaScript execution on later page views.</p> <p><b>Parameter(s):</b> True-Client-IP (used in request headers during logout)</p> <p><b>Attack Vectors:</b></p> <p>Injected payload &lt;iframe src="javascript:alert('xss')"&gt; via header is stored and reflected on the admin "Last Login IP" page.</p> <p>This executes JavaScript on page load, making it a <b>persistent/stored XSS</b>.</p> <p><b>References:</b> THM Juice Shop – Task 7, Question #2: Perform a persistent XSS</p>
Vulnerability Description	The application logs the IP address using the unsanitized value of the True-Client-IP HTTP header. This allows an attacker to inject JavaScript, which is executed when the admin later views the "Last Login IP" page.
proof	

PENETRATION TESTING REPORT TEMPLATE

	Screenshot of alert box triggered by <code>&lt;iframe src="javascript:alert('xss') "&gt;</code> on the "Last Login IP" page after logout/login cycle using Burp Suite.
Recommendation	<p><b>Who: IT Team</b></p> <p><b>Action:</b></p> <p>The login mechanism permitted injection of malicious payloads via HTTP headers without sanitization. Geez Security recommends sanitizing all inputs, including HTTP headers, and disabling the reflection of user-controllable values in administrative pages</p> <p>Action 2: The application permitted a persistent XSS attack via the <code>True-Client-IP</code> header. Geez Security recommends adopting proper input validation and encoding strategies, including:</p> <ul style="list-style-type: none"> <li>• Reject or sanitize any IP-related headers containing HTML or JavaScript characters</li> <li>• Use context-aware output encoding (e.g., HTML encode before rendering to DOM)</li> <li>• Avoid reflecting user input in admin dashboards unless necessary</li> </ul> <p><b>Item 3:</b> The admin dashboard permitted user enumeration through exposed login metadata (IP visibility). Geez Security recommends limiting visibility of sensitive data and contextual info unless essential..</p> <p><b>Additionally, Geez Security recommends that Juice Shop:</b></p> <ul style="list-style-type: none"> <li>• Train developers on secure coding practices and common XSS prevention techniques</li> <li>• Implement HTTP response headers like <code>Content-Security-Policy (CSP)</code></li> <li>• Regularly test all input fields and headers with both automated tools and manual review</li> <li>• Log and monitor unusual or suspicious inputs (e.g., non-IP string values in IP headers)</li> </ul>

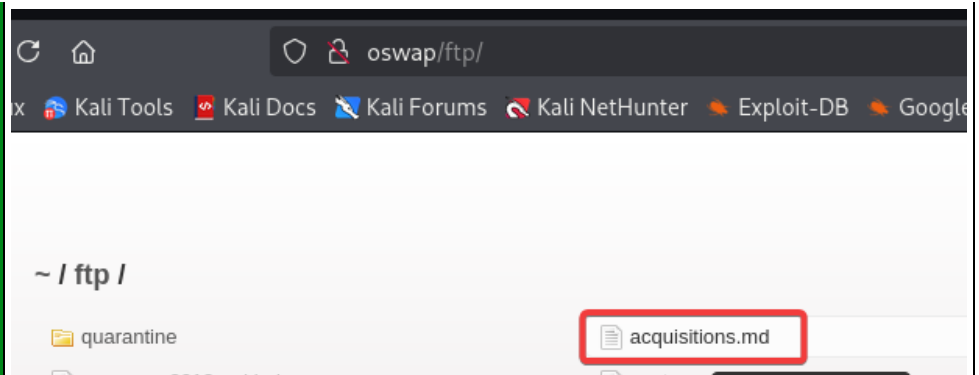
## PENETRATION TESTING REPORT TEMPLATE

## Vulnerability 5:

Ref id	ref-3-1
Vulnerabilities name	Directory Listing – Sensitive File Exposure
Risk/Severity:	<b>Medium</b>
Vulnerability Explanation	<p><b>Affects:</b> Web Application — specifically, the <code>/ftp/</code> endpoint exposed on the Juice Shop web server</p> <p><b>Parameter(s):</b></p> <ul style="list-style-type: none"><li>- URL path: <code>/ftp/</code></li><li>- File name manipulation: <code>package.json.bak%2500.md</code></li></ul> <p><b>Attack Vectors:</b></p> <ul style="list-style-type: none"><li>- Direct access to known or guessable paths (<code>/ftp/legal.md</code>, <code>/ftp/acquisitions.md</code>)</li><li>- Null byte injection (<code>%2500.md</code>) to bypass file extension restrictions</li></ul> <p><b>References:</b></p> <ul style="list-style-type: none"><li>• [Juice Shop Write-up, Task 5 - Question 1 &amp; 3]</li><li>• OWASP Cheat Sheet: <a href="#">Directory Listing</a></li></ul>
Vulnerability Description	The <code>/ftp/</code> directory allows unauthenticated users to view and download sensitive internal files. While <code>.md</code> and <code>.pdf</code> are allowed, attackers used <b>Poison Null Byte (%00)</b> injection (e.g., <code>package.json.bak%2500.md</code> ) to bypass server file-type restrictions and download <code>.bak</code> files.
proof	Accessing <code>http://&lt;target&gt;/ftp/acquisitions.md</code> and displaying the file

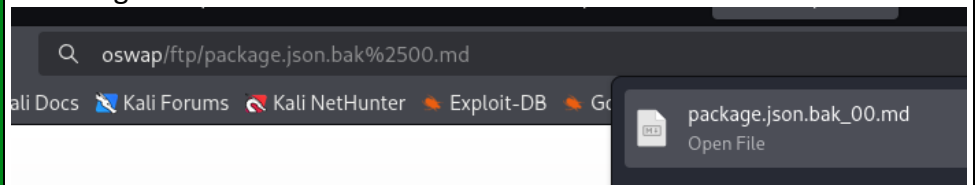


PENETRATION TESTING REPORT TEMPLATE



You can click and open it!

Using a browser or curl to request `package.json.bak%2500.md` and receiving the `.bak` file contents:



Recommendation

**Who: IT Team**

**Action:**

Directory Listing and unrestricted file access were permitted. Geez Security recommends disabling directory listing and enforcing strict file access controls.

Action 2 The application permitted bypassing download restrictions via Null Byte injection. Geez Security recommends input validation and MIME type checks to prevent such bypasses

Item 3: The system permitted sensitive file enumeration. Geez Security recommends implementing proper authorization checks and sanitizing file path inputs to prevent access to unintended files.

**Additionally, Geez Security recommends that the organization:**

- Train developers on secure file handling practices
- Regularly audit and restrict access to backup and internal files
- Apply server-side filtering and proper error handling to avoid revealing

PENETRATION TESTING REPORT TEMPLATE

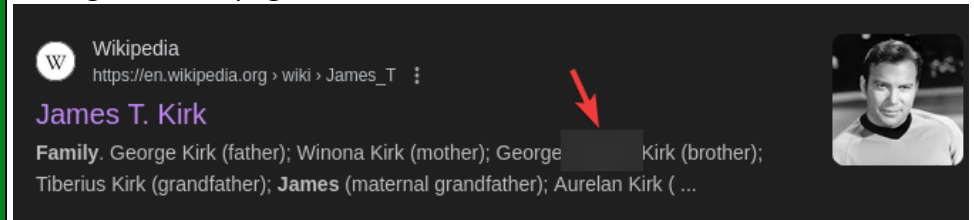
	<p>internal structure</p> <ul style="list-style-type: none"> <li>▪ Monitor access logs for suspicious download patterns</li> </ul>
--	--

Vulnerability 6:

Ref id	ref-3-2
Vulnerabilities name	Security Question Bypass – Jim’s Account
Risk/Severity:	<b>MEDIUM</b>
Vulnerability Explanation	<p><b>Affects:</b> Password recovery mechanism for user accounts (specifically Jim’s account).</p> <p><b>Parameter(s):</b> email and securityAnswer in the “Forgot Password” form.</p> <p><b>Attack Vectors:</b> An attacker can:</p> <ul style="list-style-type: none"> <li>• Input Jim’s email (jim@juice-sh.op) into the "Forgot Password" page.</li> <li>• Use publicly accessible information (Star Trek reference) to deduce that the answer is Samuel (middle name of Jim’s brother in the show).</li> <li>• Reset the password and gain unauthorized access.</li> </ul> <p><b>References:</b></p> <ul style="list-style-type: none"> <li>• OWASP Broken Authentication</li> <li>• TryHackMe Juice Shop Room — Task 4, Question 2</li> <li>• Star Trek lore (used for social engineering)</li> </ul>
Vulnerability Description	<p>The account recovery process relied on a weak security question. The attacker exploited knowledge that Jim referenced Star Trek, leading to the discovery that his brother’s middle name was “Samuel,” which was used as the answer to the password reset security question. This granted unauthorized access to Jim's account without needing his original password.</p>

proof

When inputted into the email field in the Forgot Password page, Jim's security question is set to "Your eldest siblings middle name?". In Task 2, we found that Jim might have something to do with Star Trek. Googling "Jim Star Trek" gives us a wiki page for Jame T. Kirk from Star Trek. Looking through the wiki page we find that he has a brother.



Inputting that into the Forgot Password page allows you to successfully change his password. You can change it to anything you want!  
security question : his eldest siblings middle name!

The image shows a 'Forgot Password' form. The email field is filled with 'jim@juice-sh.op'. The security question field is highlighted with a green border and contains six dots. The new password field contains five dots, with a note 'Password must be 5-20 characters long.' and a character count '5/20'. The repeat new password field also contains five dots with a '5/20' character count. A 'Show password advice' toggle is at the bottom, and a 'Change' button is at the very bottom.

Recommendation

**Who: IT Team**

**Action:**

**Action 1:** The account recovery system allowed bypass via weak personal knowledge questions. Geez Security recommends using multi-factor authentication (MFA) or password reset links sent via email instead of security questions

**Action 2:** The password reset feature permitted successful account takeover using publicly available trivia, which is a form of social engineering. Geez Security recommends the following password policy, per the Center for Internet Security (CIS):

- 14 characters or longer

PENETRATION TESTING REPORT TEMPLATE

- Use different passwords for each account accessed
- Do not use words and proper names in passwords, regardless of language

**Item 3:** The reset form indirectly permitted user enumeration (confirmation that Jim's email exists). Geez Security recommends synchronizing valid and invalid account response messages to prevent enumeration.

**Additionally, Geez Security recommends that the application:**

- Train employees on how to create a proper password.
- Check employee credentials against known breached passwords.
- Discourage employees from using work emails and usernames as login credentials to other services unless absolutely necessary.

## Summary of Findings

The identified vulnerabilities highlight critical issues in input validation, access control, and authentication mechanisms. Immediate remediation is strongly advised.

## Conclusion

The OWASP Juice Shop assessment revealed several realistic and high-impact vulnerabilities that are often found in live production systems when proper security measures are lacking. It is essential that the development team carefully review each finding, apply the recommended fixes, and perform follow-up testing.

The penetration testing engagement, conducted by Geez Security, identified multiple vulnerabilities within the target application. The critical and high-severity issues discovered pose serious risks to the organization's data and system integrity. If left unaddressed, these vulnerabilities could allow attackers to gain unauthorized access, exfiltrate sensitive information, or potentially take full control of the system.

These vulnerabilities are accessible through commonly known attack techniques and would likely be exploited by determined threat actors. Remediating the issues detailed in this report will greatly reduce the organization's risk exposure and strengthen its overall security posture.

Immediate attention should be given to resolving the critical and high-risk vulnerabilities, particularly before moving the system into a production environment. Additionally, implementing ongoing security monitoring, conducting periodic testing, and following industry best practices will help maintain a strong security stance.

Geez Security remains available to provide further consultation and conduct re-testing after remediation efforts to ensure all identified vulnerabilities have been properly addressed and that no new security gaps have emerged.

## Appendices

### Appendix A: Tools and Techniques Used

1. Network Scanning: Nmap, Masscan
2. Vulnerability Scanning: Nessus, OpenVAS
3. Exploitation Framework: Metasploit, Cobalt Strike
4. Web Application Testing: Burp Suite, OWASP ZAP
5. Password Cracking: Hydra, John the Ripper

### Appendix B: Glossary of Terms

- Reconnaissance: The process of gathering information about the target before launching an attack.
- SQL Injection (SQLi): A type of attack where an attacker inserts malicious SQL code into a web form to access or manipulate the database.
- Cross-Site Scripting (XSS): An attack where an attacker injects malicious scripts into content from otherwise trusted websites.
- Privilege Escalation: Gaining elevated access to resources that are normally protected.

### Appendix C: Vulnerability Severity Levels

PENETRATION TESTING REPORT TEMPLATE

- Critical: A vulnerability that could lead to full system compromise and immediate action is required.
- High: A serious vulnerability that could allow unauthorized access or significant disruption.
- Medium: A vulnerability that presents some risk but may require certain conditions or skills to exploit.
- Low: A minor vulnerability that is unlikely to be exploited but should still be remediated as part of best practices.