



# SECURITY ASSESSMENT

## Owasp Juice Shop Report

Done by :

- Ahmed Sameh
- Amir Serry
- Adham Adel
- Reem Lotfy

Date of Testing: 10/10/2024

Date of Report Delivery: 24/10/2024



# Table of Contents

## Contents

<b>SECURITY ENGAGEMENT SUMMARY.....</b>	<b>2</b>
ENGAGEMENT OVERVIEW.....	2
SCOPE.....	2
Executive Risk Analysis.....	3
1. High-Risk Vulnerabilities:.....	3
2. Medium-Risk Vulnerabilities:.....	3
3. Low-Risk Vulnerabilities:.....	3
Risk Impact Overview.....	3
Executive Recommendation.....	4
1. Immediate Actions (Within 30 Days):.....	4
2. Short-Term Actions (Within 60 Days):.....	4
3. Long-Term Actions (Within 90 Days and Beyond):.....	4
<b>SIGNIFICANT VULNERABILITY DETAIL.....</b>	<b>5</b>
SQLI inLogin.....	5
ADMIN PASSWORD GUESTING.....	5
Support Chat giving promotional code.....	6
Post Customer feedback with another user.....	7
idoor in admin page.....	8
unauthorize access in admin page.....	9
idoor in Privacy policy.....	10
Testing JSON Web Token Lead To 0-Click ATO.....	10
Reflected XSS In Search Functionality.....	14
Repetitive Registration.....	14
View another users shopping basket.....	15
Make a free order (place an order that make you rich).....	16
<b>METHODOLOGY.....</b>	<b>17</b>
ASSESSMENT TOOLSET SELECTION.....	17
ASSESSMENT METHODOLOGY DETAIL.....	18
Conclusion.....	19



# Security Engagement Summary

## Engagement Overview

The Penetration testing team evaluated the security posture of the Juice Shop Web Application through a Penetration Test which showed the different flaws in configuration and implementation of the Juice Shop Web service. A penetration test emulates an external threat actor which is trying to compromise different *External* Systems through the exploitation of multiple vulnerable configurations in the provided service. In this current Web Application Penetration Test the objective was to analyze the external security posture of the web application Juice Shop and discover possible vulnerabilities on the Juice Shop to gain Administrative access on the application and extract sensitive client information and transactions.

## Scope

### Allowed Scope

The allowed scope for this engagement was the following: OWASP Juice Shop: <http://localhost:3000/>

The testing team was not provided accounts for testing

## Methodology Used

Starting on the Saturday 10 of October 2024 the Penetration testing team engaged on a penetration test of the Juice Shop Service. All of the testing was performed with the following methodology:

- Discovery
- Scanning
- Exploitation
- Reporting

Along this report the team has provided screenshots and important files used during the assessment.

## Executive Risk Analysis

The OWASP Juice Shop project, intentionally designed to demonstrate web application vulnerabilities, highlights real-world security threats. A comprehensive penetration test was conducted to identify risks associated with its architecture. The following is a summary of key risk findings:

---

### 1. High-Risk Vulnerabilities:

- **SQL Injection:**  
Allows unauthorized access to the backend database, compromising sensitive data such as user accounts and transactions.
- **Broken Authentication:**  
Weak session management enables account takeover by exploiting insecure tokens and session IDs.
- **Cross-Site Scripting (XSS):**  
Injected malicious scripts allow attackers to hijack sessions or perform phishing attacks on end-users.
- **Insecure Direct Object References (IDOR):**  
Poor access controls allow unauthorized users to access other users' personal information.

---

### 2. Medium-Risk Vulnerabilities:

- **Cross-Site Request Forgery (CSRF):**  
Attackers can trick users into performing unintended actions on their accounts.
- **Unvalidated Redirects and Forwards:**  
Users are redirected to malicious websites, increasing phishing risks.
- **Security Misconfigurations:**  
Unnecessary information in error messages and exposed API endpoints increase the attack surface.

---

### 3. Low-Risk Vulnerabilities:

- **Outdated Software Components:**  
Use of vulnerable versions of libraries and frameworks could be exploited in specific scenarios.
  - **Information Disclosure:**  
Leaking sensitive data such as stack traces or application paths provides attackers with insights for further attacks.
-



## Risk Impact Overview

- **Business Impact:** High — Unauthorized access and data leaks could lead to reputational damage, financial penalties, and loss of customer trust.
- **Likelihood of Exploitation:** High — Many vulnerabilities are easily detectable using automated tools.
- **Compliance Risks:** Potential non-compliance with data privacy regulations (e.g., GDPR, CCPA) due to weak authentication and data security practices.

## Executive Recommendation

Based on the identified risks, the following recommendations are proposed to improve the security posture of the Juice Shop project:

### 1. Immediate Actions (Within 30 Days):

- **Patch and Update Software Components:**  
Ensure all dependencies and libraries are updated to their latest stable versions to mitigate known vulnerabilities.
- **Remediate Critical Vulnerabilities:**  
Address SQL injection, XSS, and authentication flaws with high priority. Implement prepared statements and input validation for SQL queries and sanitize all user inputs to prevent script injection.
- **Implement Strong Authentication and Session Management:**  
Use multi-factor authentication (MFA) and secure session handling mechanisms to prevent session hijacking and account takeovers.

### 2. Short-Term Actions (Within 60 Days):

- **Enhance Access Control Mechanisms:**  
Apply the principle of least privilege (POLP) to limit access to critical resources. Use role-based access controls (RBAC) to prevent unauthorized access.
- **Add Security Headers:**  
Implement HTTP security headers (e.g., CSP, X-Frame-Options) to mitigate XSS and clickjacking risks.
- **Conduct Employee Awareness Training:**  
Educate staff and developers on secure coding practices and the risks of common web vulnerabilities.

### 3. Long-Term Actions (Within 90 Days and Beyond):

- **Implement Automated Security Testing:**  
Integrate security testing tools into the CI/CD pipeline to detect vulnerabilities early during development.
- **Adopt a Bug Bounty Program:**  
Encourage responsible disclosure of vulnerabilities by incentivizing external researchers to report issues.
- **Perform Regular Security Audits and Penetration Testing:**  
Schedule regular assessments to ensure ongoing security improvements and compliance with industry standards.

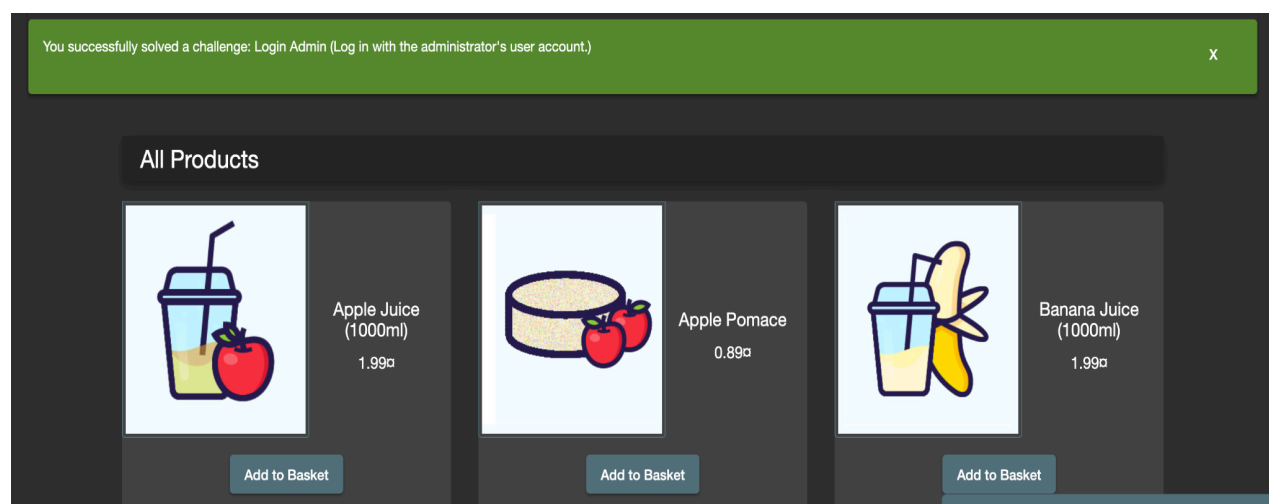
## Significant Vulnerability Detail

### SQLI inLogin

#### Level High

#### Vulnerability detail

- you can login using sql " ' OR '1'='1' -- -"



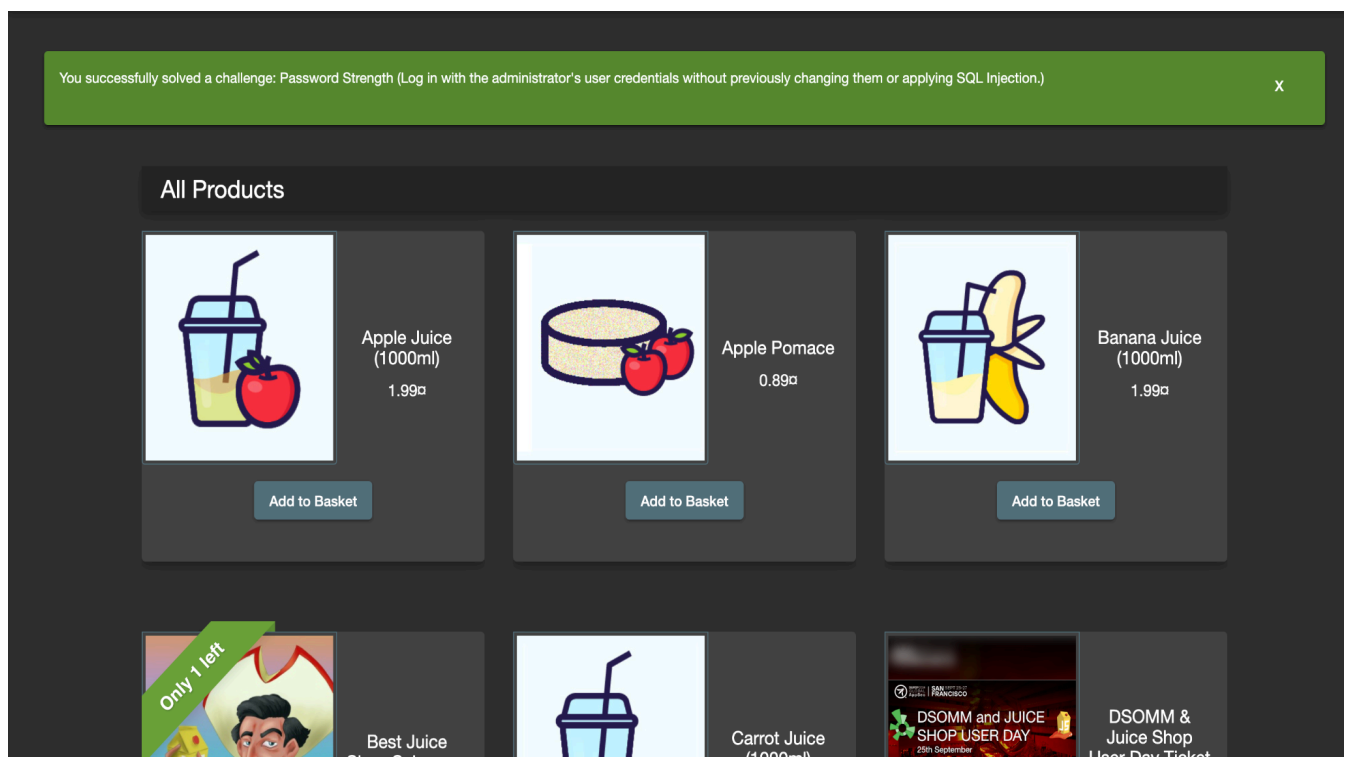
- Probability is high.
- All the website will be affected
- Developer need to validate user input

## Admin Password Guesting

### Level High

#### Vulnerability detail

- you can guest admin password



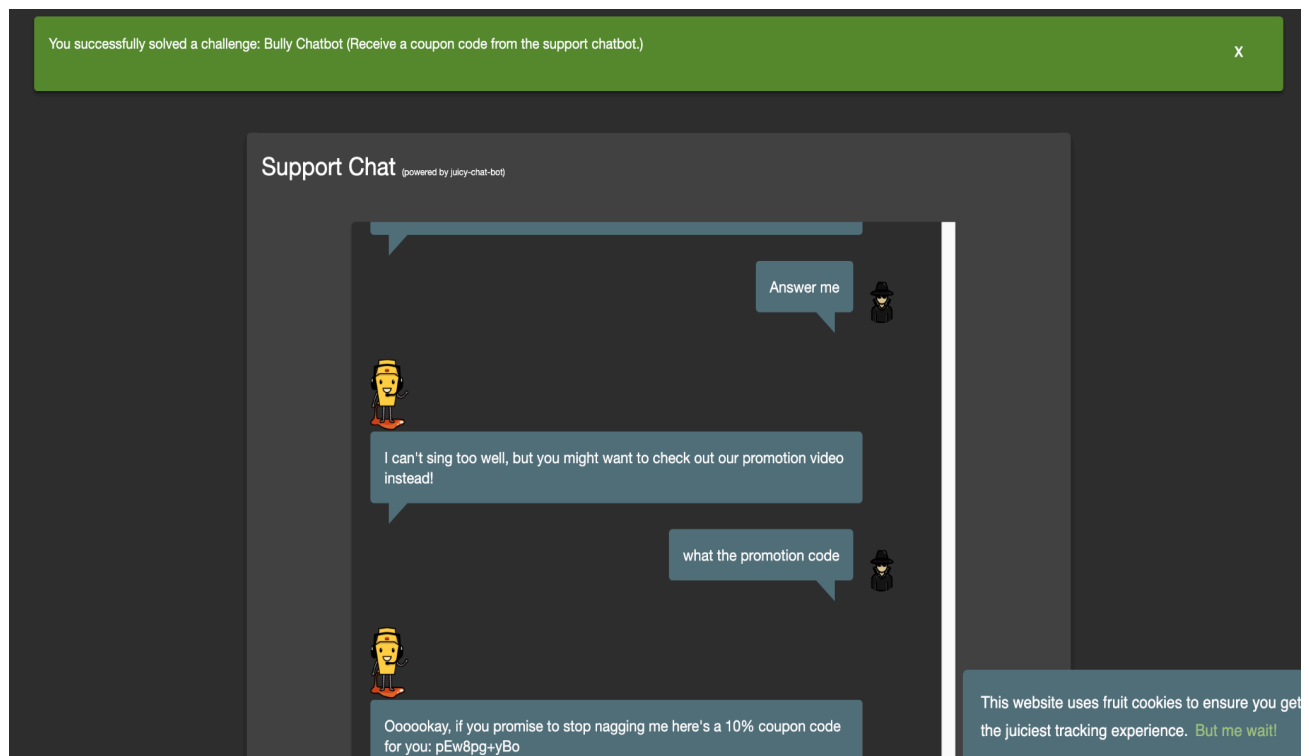
- probability is high.
- All the website will be affected
- inforce password strength

## Support Chat giving promotional code

### Level low

#### Vulnerability detail

- you can talk to support chatbot to give you promotional code



- probability is medium.
- promotional code part only is affected
- use better chatbot

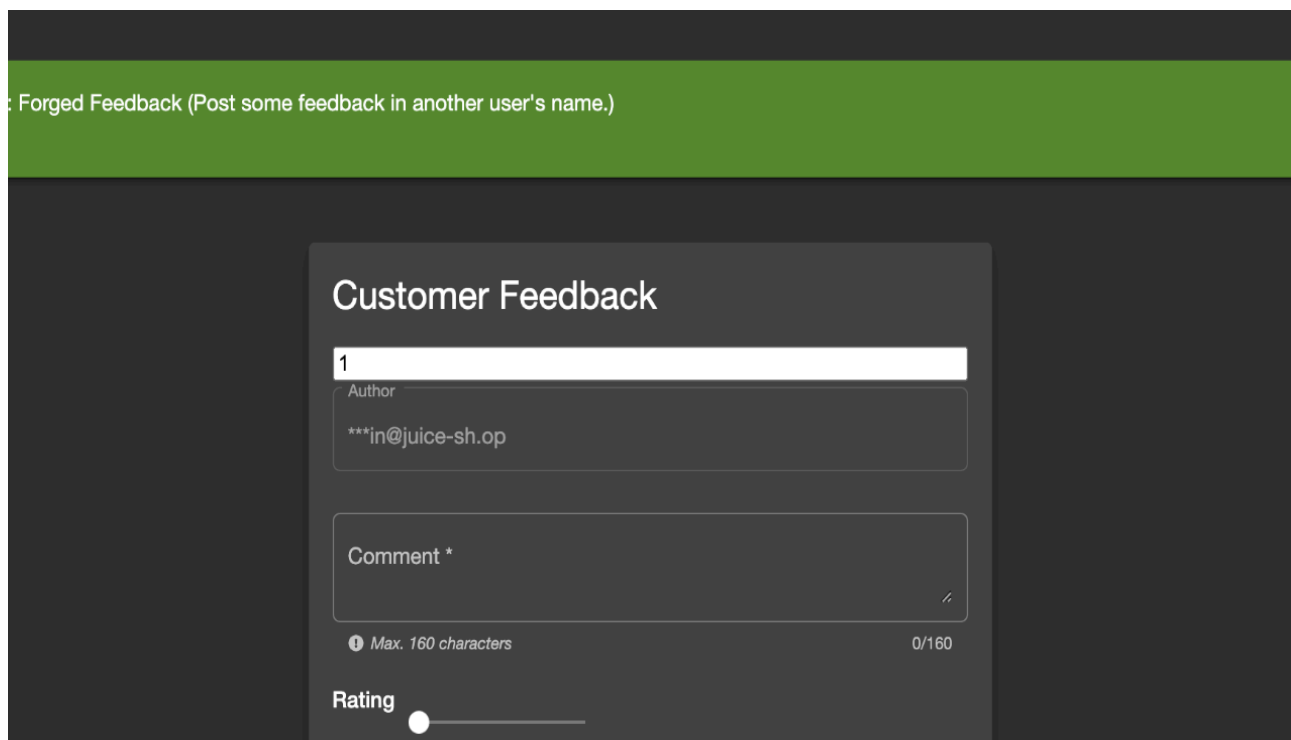


## Post Customer feedback with another user

### Level Medium

#### Vulnerability detail

- you can post customer feedback with another user by unhide the textfield of the



The screenshot shows a web application interface with a green header bar containing the text "Forged Feedback (Post some feedback in another user's name.)". Below this is a dark grey form titled "Customer Feedback". The form contains a text input field with the value "1", an "Author" field with the value "\*\*\*in@juice-sh.op", a "Comment \*" field, a character count "Max. 160 characters" and "0/160", and a "Rating" field with a slider.

id and change it

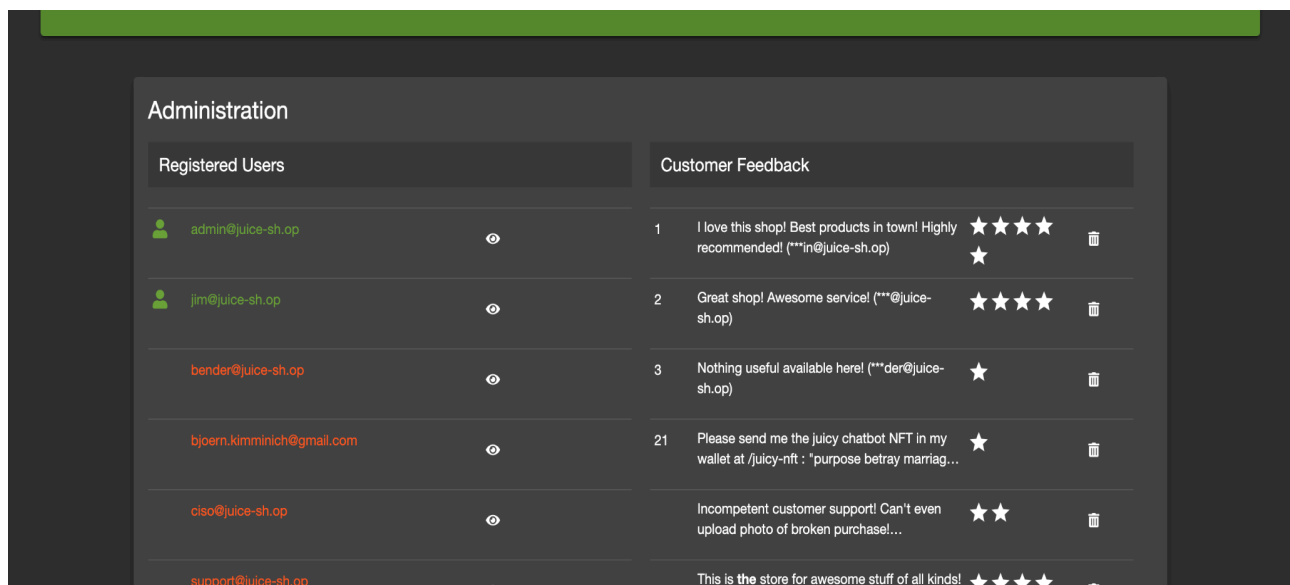
- probability is low.
- customer feedback only is affected
- take the id from session not hidden field

## idoor in admin page

### Level Medium

#### Vulnerability detail

- going to admin by url



- Probability is medium.
- feedback for admin will be affected
- Developer need authorize access

## Unauthorized access in admin page

### Level Medium

#### Vulnerability detail

Administration			
Registered Users		Customer Feedback	
		<a href="#">Click for more information</a>	
amy@juice-sh.op	👁	2	Great shop! Awesome service! (**@juice-sh.op) ★★★★★ 🗑
bjoern@juice-sh.op	👁	3	Nothing useful available here! (**der@juice-sh.op) ★ 🗑
bjoern@owasp.org	👁	21	Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriag... ★ 🗑
accountant@juice-sh.op	👁		Incompetent customer support! Can't even upload photo of broken purchase!... ★★ 🗑
uvogin@juice-sh.op	👁		This is the store for awesome stuff of all kinds! (anonymous) ★★★★★ 🗑
demo	👁		Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous) ★★★★★ 🗑

- going to admin by url and remove five star feedback
- Probability is medium.
- feedback for admin will be affected
- Developer need authorize access

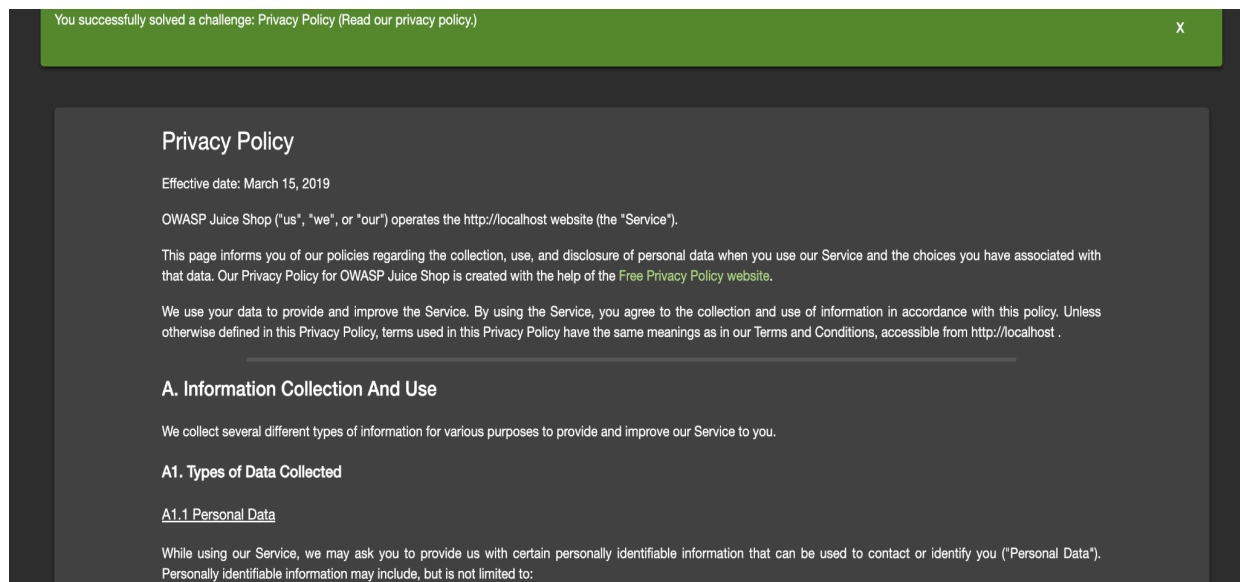


## idoor in Privacy policy

### Level Low

#### Vulnerability detail

- going to privacy policy by url



- Probability is low.
- privacy will be affected
- Developer need authorize access







### 3. By Changing The ALG To **None** and Removing Signature We Will be able to Alter The Payload Data

eyJ0eXAiOiJKV1QiLCJhbGciOiJIb251In0.ewo  
gICJzdGF0dXMiOiAic3VjY2VzcyIsCiAgImRhdG  
EiOiB7CiAgICAiaWQiOiAyMywKICAgICJ1c2Vyb  
mFtZSI6ICIIAogICAgImVtYWlsIjogImFkaGFt  
YWRLbDMzQGdtYWlsLmNvbSIsCiAgICAicGFzc3d  
vcnQiOiAiOThlMjE2YmQxMjYxMGVlZTMzOWI1NT  
JhMDNiMWU2ODEiLAogICAgInJvbGUiOiAiYWRTa  
W4iLAogICAgImRlbHV4ZVRva2VuIjogIiIsCiAg  
ICAibGFzdExvZ2ZuSXAiOiAidW5kZWZpbmVkIiw  
KICAgICJwcm9maWx1SW1hZ2U0IiAiL2Fzc2V0cy  
9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsd  
C5zdmciLAogICAgInRvdHBTZWNYZXQ0IiAiIiwK  
ICAgICJpc0FjdGJ2ZSI6IHRydWUsCiAgICAiY3J  
lYXRlZEF0IjogIjIwMjQ0MDk0MDk0MDk0NDg0ND  
guMDIxICswMDowMCIIsCiAgICAidXBkYXRlZEF0I  
jogIjIwMjQ0MDk0MDk0MDk0NDk0NTAuODM0ICsw  
MDowMCIIsCiAgICAiZGVzXHRlZEF0IjogbnVsbAo  
qIH0sCiAgImIhdCI6IDE3MjY0OTM0NTAzMkFkQ.

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "None"
}
```

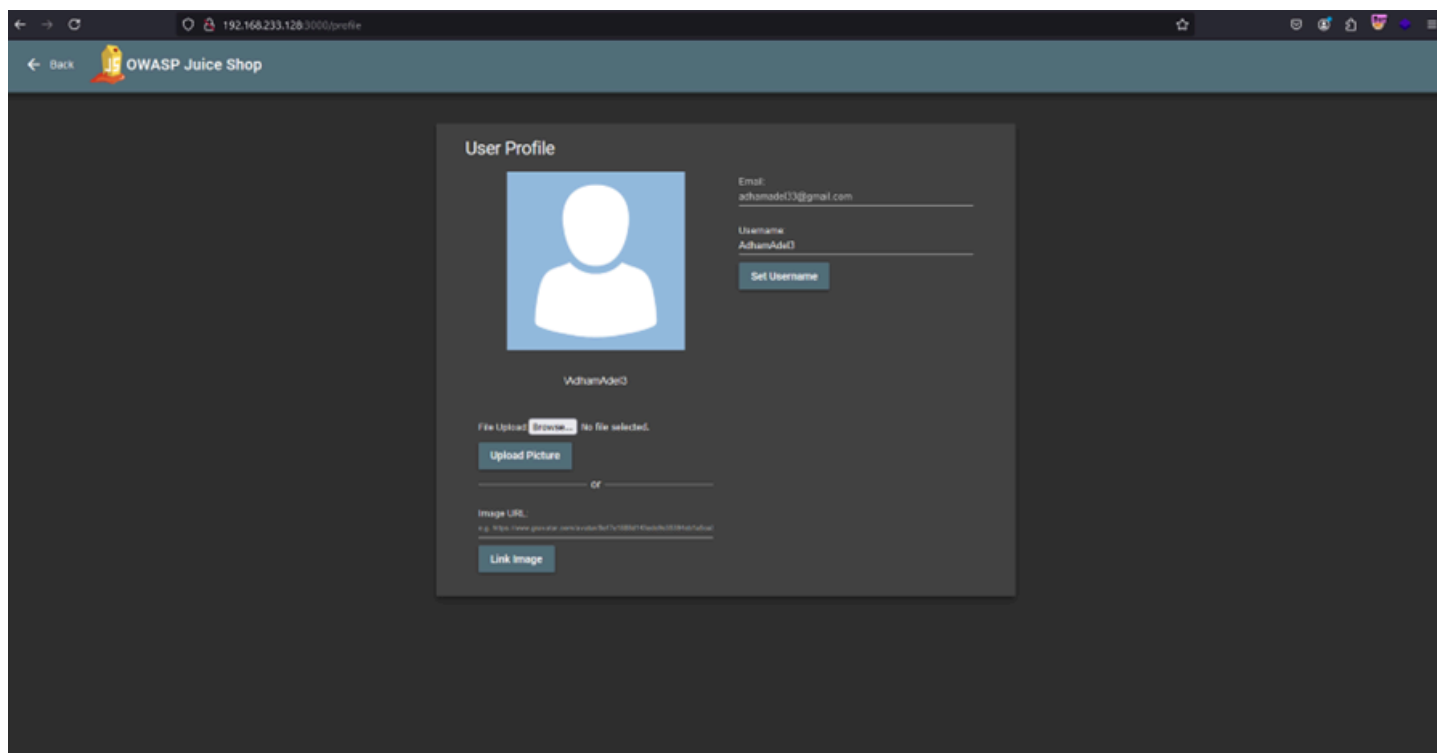
**PAYLOAD:** DATA

```
{
  "status": "success",
  "data": {
    "id": 23,
    "username": "",
    "email": "adhamadel33@gmail.com",
    "password": "98e216bd12910ebe339b552a03b1e681",
    "role": "admin",
    "deluxeToken": "",
    "lastLoginIp": "undefined",
    "profileImage":
"/assets/public/images/uploads/default.svg",
    "totpSecret": "",
    "isActive": true,
    "createdAt": "2024-09-09 14:48:48.021 +00:00",
    "updatedAt": "2024-09-09 14:49:50.838 +00:00",
    "deletedAt": null
  },
  "iat": 1725893473
}
```

New Token Will be

'eyJ0eXAiOiJKV1QiLCJhbGciOiJOc25lbn0.ewogIjZjdGF0dXMiOiAic3VyY2VzcylsCiAgImRhdGEiOiB7CiAgICAiaWQlOiAyMywKICAgLjC1c2VybmFtZSI6IChLaoglCAglmVtYWlsIjogImFkaGFtYWRIbDMzQGdtYWlsLmNvbSIsCiAgLCAicGFzc3dvcmQiOiAiOThIMjE2YmQxMjkxMGViZTMzOWI1NTJhMDNiMWU2ODEiLAoglCAglnJvbGUiOiAiYWRTaW4iLAoglCAglmRibHV4ZVRva2VuljogIlIsCiAgLCAibGFzdExvZ2luSXAiOiAidW5kZWZpbmVkIiwKICAgLjJwcm9maWxlSW1hZ2UiOiAiL2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMuZGVmYXVs dC5zdmciLAoglCAglnRvdHBTZWNYZXQiOiAiliwKICAgLjpc0FjdGl2ZSI6IHRydWUsCiAgLCAiY3JlYXRIZEF0ljogIjllwMjQtMDktMDkgMTQ6NDg6NDguMDIxIiwucSwMDowMCIsCiAgLCAidXBkYXRIZEF0ljogIjllwMjQtMDktMDkgMTQ6NDk6NTAuODM0IiwucSwMDowMCIsCiAgLCAiZGVsZXRIZEF0ljogbnVsbAoglIH0sCiAgImIldCI6IDEzMjU0OTM0NmKfQ.'

#### 4. So We Take The Account NOW



#### Who Would be Impacted

Any Application User

#### Potential Remediation

1. Disable or Reject the Use Of the None Algorithm
  - a. ``jwt.verify(token, secret, { algorithms: ['RS256'] })``
2. Validate the Algorithm During Verification
  - a. ``jwt.decode(token, secret, algorithms=['RS256'])``
3. Upgrade JWT Library



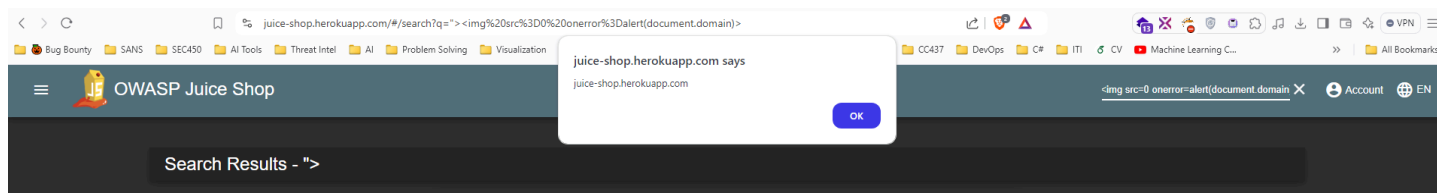


## Reflected XSS In Search Functionality

MEDIUM

### Vulnerability detail

- We Are able to Inject A malicious JS Code in search functionality that may lead to Steal User Session
- Probability is High but need user interaction
- Potential Remediation By Used Proper Validation on user input





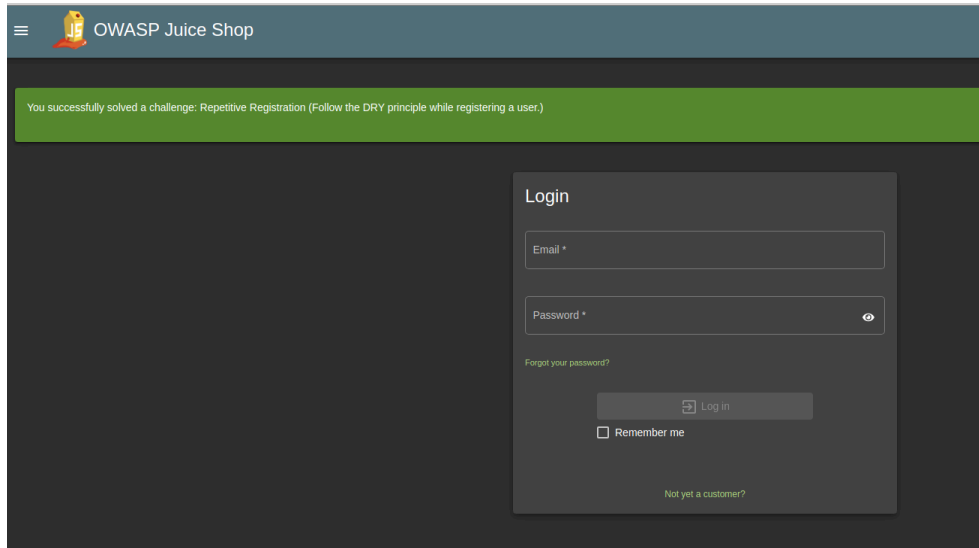
## Repetitive Registration

High

### Vulnerability details

- Register a new user without checking the confirm password and the password strengths isn't applied.
- Probability is high
- It can impact the user's profiles and the password will be easy to guessed

1096	http://192.168.69.3:3000	GET	/rest/admin/application-configuration	304	306
Request				Response	
Pretty Raw Hex				Pretty Raw Hex Render	
<pre>1 POST /api/Users/ HTTP/1.1 2 Host: 192.168.69.3:3000 3 Content-Length: 225 4 Accept-Language: en-US,en;q=0.9 5 Accept: application/json, text/plain, */* 6 Content-Type: application/json 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)   Chrome/129.0.6668.71 Safari/537.36 8 Origin: http://192.168.69.3:3000 9 Referer: http://192.168.69.3:3000/ 10 Accept-Encoding: gzip, deflate, br 11 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=   OSnE7aZLzjNqewXkQKpVMgAJl fpruw6t2TKpGoYR4b813l92rmWDJPv6yBx 12 Connection: keep-alive 13 14 {"email":"a@yahoo.com","password":"a2345","passwordRepeat":"a","securityQuestion":{"id":2,"question":   "Mother's maiden name?","createdAt":"2024-10-21T23:19:55.458Z","updatedAt":"2024-10-21T23:19:55.458Z"},   "securityAnswer":"a"}</pre>				<pre>1 HTTP/1.1 201 Created 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 X-Recruiting: /#/jobs 7 Location: /api/Users/24 8 Content-Type: application/json; charset=utf-8 9 Content-Length: 302 10 ETag: W/"12e-OnM9IRjXXXyGlykoWGuNbXHB4xo" 11 Vary: Accept-Encoding 12 Date: Tue, 22 Oct 2024 00:46:22 GMT 13 Connection: keep-alive 14 Keep-Alive: timeout=5 15 16 {   "status": "success",   "data": {     "username": "",     "role": "customer",     "deluxeToken": "",     "lastLoginIp": "0.0.0.0",     "profileImage": "/assets/public/images/uploads/default.svg",     "isActive": true,     "id": 24,     "email": "a@yahoo.com",     "updatedAt": "2024-10-22T00:46:22.120Z",     "createdAt": "2024-10-22T00:46:22.120Z",     "deletedAt": null   } }</pre>	

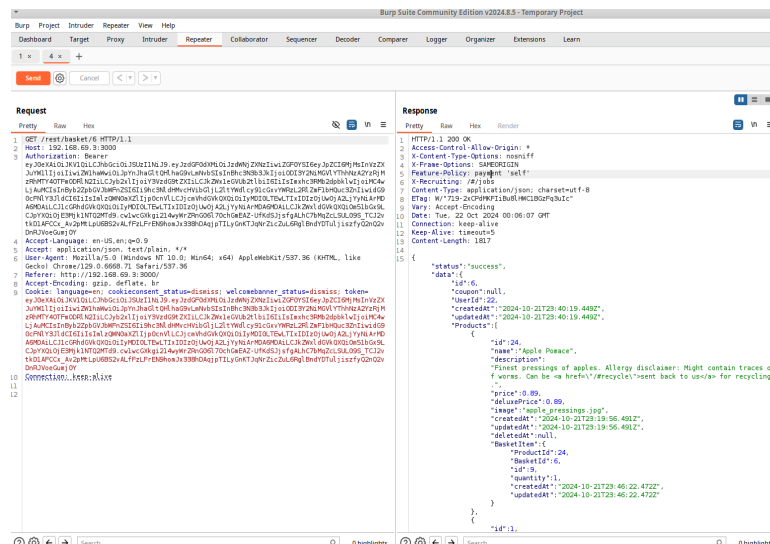


## View another users shopping basket

low

### Vulnerability details

- Easy access another user basket without any authentication by edit on the get request (change the basket id)





## Methodology

The methodology employed in testing the OWASP Juice Shop follows industry-standard frameworks like **OWASP Testing Guide**, **NIST SP 800-115** (Technical Guide to Information Security Testing and Assessment), and **PTES** (Penetration Testing Execution Standard). The purpose of this penetration test is to simulate real-world attacks on the web application to discover vulnerabilities, assess their risk levels, and provide remediation advice.

## Assessment Toolset Selection

Choosing the right set of tools is crucial to ensure a comprehensive security evaluation. Below is the toolset used for the assessment of the OWASP Juice Shop:

---

### 2.1. Reconnaissance and Information Gathering

- **Nmap:**  
Used to perform network scanning, port scanning, and service identification. Helps in identifying the open ports, services, and version details.
- **Whois/Dig/Netcat:**  
Utilized to gather DNS information and other critical metadata about the hosting server, identifying server versions, configurations, and subdomains.

---

### 2.2. Vulnerability Scanning

- **OWASP ZAP (Zed Attack Proxy):**  
An essential tool for automated vulnerability scanning, including detection of SQL injection, Cross-Site Scripting (XSS), security misconfigurations, and other OWASP Top 10 vulnerabilities.
- **Burp Suite Professional:**  
Used for both automated and manual security testing. It's crucial for identifying issues like IDOR (Insecure Direct Object References), CSRF (Cross-Site Request Forgery), and business logic flaws. Burp is used for active scanning, interception, and fuzzing.

---

### 2.3. Exploitation Tools

- **SQLMap:**  
Automated tool used to detect and exploit SQL Injection vulnerabilities. It helps in testing database security and querying sensitive data from exposed databases.



- **Metasploit Framework:**

An exploitation framework used to simulate attacks against discovered vulnerabilities. It helps in verifying if a vulnerability is exploitable and the extent of the risk.

---

## 2.4. Post-Exploitation & Privilege Escalation

- **John the Ripper/Hashcat:**

Used for cracking weak passwords or hashes, exploiting poor credential management practices after gaining initial access.

- **Hydra:**

Brute force tool used to test for weak authentication practices, especially in login forms, administrative portals, or remote services like FTP and SSH.

## Assessment Methodology Detail

The OWASP Juice Shop was tested using a comprehensive methodology to identify and verify potential security weaknesses. The testing phases are detailed below:

---

### 3.1. Phase 1: Reconnaissance and Information Gathering

Objective: Collect as much information as possible about the Juice Shop without interacting with it directly (passive information gathering). This phase helps in mapping the attack surface and identifying weak points before active testing.

Steps:

- Perform DNS enumeration and whois lookups to identify the hosting infrastructure.
  - Map network topology using Nmap, identifying open ports and services.
  - Utilize OWASP Amass to discover subdomains and related infrastructure that might be exposed.
- 

### 3.2. Phase 2: Vulnerability Discovery

Objective: Identify potential vulnerabilities in the web application by running automated vulnerability scans followed by manual verification.

Steps:

- Run OWASP ZAP and Burp Suite for an initial scan, identifying security flaws like SQL injection, XSS, CSRF, and IDOR.
  - Review the HTTP request/response cycles for anomalies, insecure cookies, and sensitive data exposure.
  - Test for security misconfigurations (e.g., default credentials, weak SSL/TLS configurations) using Nikto.
-



### 3.3. Phase 3: Manual Exploitation and Verification

Objective: Exploit identified vulnerabilities to validate the risk and impact of each vulnerability, ensuring false positives are eliminated and high-impact flaws are further analyzed.

Steps:

- Use SQLMap to test for SQL injection vulnerabilities, extracting sensitive data such as user credentials and transactional information.
  - Test identified XSS vulnerabilities
  - Test broken authentication mechanisms and privilege escalation paths using manual techniques and Metasploit.
- 

### 3.4. Phase 4: Post-Exploitation and Data Extraction

Objective: Once a vulnerability is successfully exploited, test for further access escalation, data exfiltration, and backdoor installation scenarios to simulate a real-world attacker.

Steps:

- Use Hydra or manual brute-force attempts against exposed login portals to test weak authentication mechanisms.
  - Crack captured credentials using tools like Hashcat and John the Ripper.
  - Test lateral movement techniques or privilege escalation to gain access to higher-privileged accounts or system resources.
- 

### 3.5. Phase 5: Reporting and Remediation Recommendation

Objective: Document all vulnerabilities, steps taken to discover and exploit them, and provide detailed recommendations for fixing each issue.

Steps:

- Record the vulnerabilities, categorized based on their severity (High, Medium, Low).
  - Provide code or configuration recommendations for remediation.
  - Suggest preventive measures like secure coding practices, continuous security monitoring, and automated vulnerability testing integration.
- 

## Conclusion

This methodology ensures a thorough security assessment of OWASP Juice Shop by using a combination of automated tools and manual testing techniques. The step-by-step approach of reconnaissance, vulnerability discovery, exploitation, and post-exploitation provides a clear picture of the application's security posture and offers actionable insights to mitigate risks.