Vulnerability Assessment and Penetration Test Report for OWASP Juice Shop

Date: October 22nd,
Group code: CAI1_ISS5_M4d,
Mentor: Hesham Saleh

Muhmad Wagih

Table of Contents

Contact Information

Name	Title	Contact Information
DEPI Trainees		
Muhmad Wagih Alngar	**Penetrartion Testing Trainee**	**mohamedwageh925@gmail.com**
Eslam Daoud	**Penetrartion Testing Trainee**	**EDaoudd@gmail.com**
Mahmoud Wael Abdelwahab	**Penetrartion Testing Trainee**	**nkhf748@gmail.com**
Abdulrahman Nasser Khalaf	**Penetration Testing Trainee**	**20200297@stud.fci-cu.edu.eg**
Rady mohamed osama	**Penetration Testing Trainee**	**radyosama200@gmail.com**

Assessment Overview

From October 15th, 2024 to October 22th, 2024, OWASP engaged DEPI to evaluate the security posture of its infrastructure compared to current industry best practices that included a webapp penetration test. All testing performed is based on the NIST SP 800-115 Technical Guide to Information Security Testing and Assessment, OWASP Testing Guide (v4), and customized testing frameworks. Phases of penetration testing activities include the following:

Planning – Customer goals are gathered and rules of engagement obtained.

- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.

Assessment Components

Web app Penetration Test

A web app penetration test emulates the role of an attacker on a web application. An engineer will scan the website to identify potential vulnerabilities and perform common and advanced web attacks, such as: Injections, broken access control and other Cross-site scripting (XSS) attacks, Improper Input Validation, and more. The engineer will seek to gain access to hosts through lateral movement, compromise domain user and admin accounts, and exfiltrate sensitive data.

Scope

Assessment	Details
Web-Server Penetration Testing	OWASP JUICE SHOP

Scope Exclusions

Per client request, DEPI did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Phishing

All other attacks not specified above were permitted by OWASP.

Executive Summary

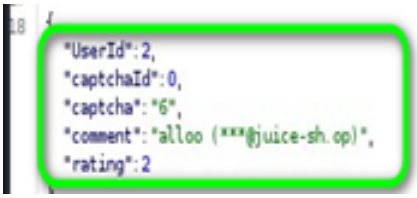
DEPI evaluated OWASP's Juice Shop posture through penetration testing from October 17th, 2024 to October 24th, 2024. The following sections provide a high-level overview of vulnerabilities discovered, successful and unsuccessful attempts, and strengths and weaknesses.

Technical Finding

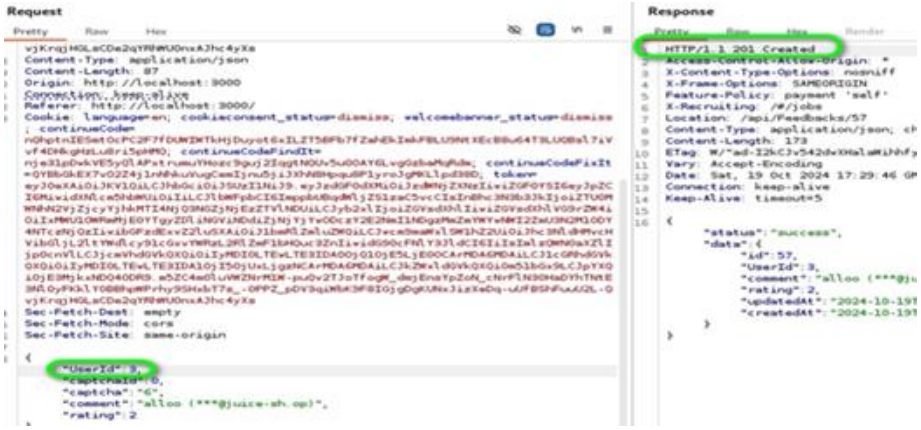
1.Broken Anti-Automation

1.CAPTCHA Bypass

- **Description:** This vulnerability allows an attacker to reuse a previously solved CAPTCHA token for multiple submissions, bypassing the CAPTCHA validation mechanism. The application fails to enforce unique CAPTCHA challenges for each submission, enabling automated attacks like spamming or brute force attempts without requiring the CAPTCHA to be solved repeatedly.
- **PoC:**
  - Initial Feedback Submission
  - Interception and Analysis the request → Notice that the CAPTCHA validation in the request relies solely on the captchaId and the provided captcha answer. Determine that the CAPTCHA system might not track or validate previous submissions effectively, possibly allowing the reuse of a single CAPTCHA solution multiple times.



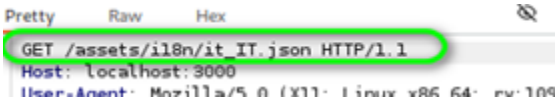
  - Automating CAPTCHA Reuse → Replaying Requests and modify the intercepted POST request to reuse the same captchaId and captcha answer. Change other fields like comment to simulate different feedback entries.
  - Rapidly replay the modified request multiple times (more than 10 times) within 20 seconds to challenge the anti-automation controls.



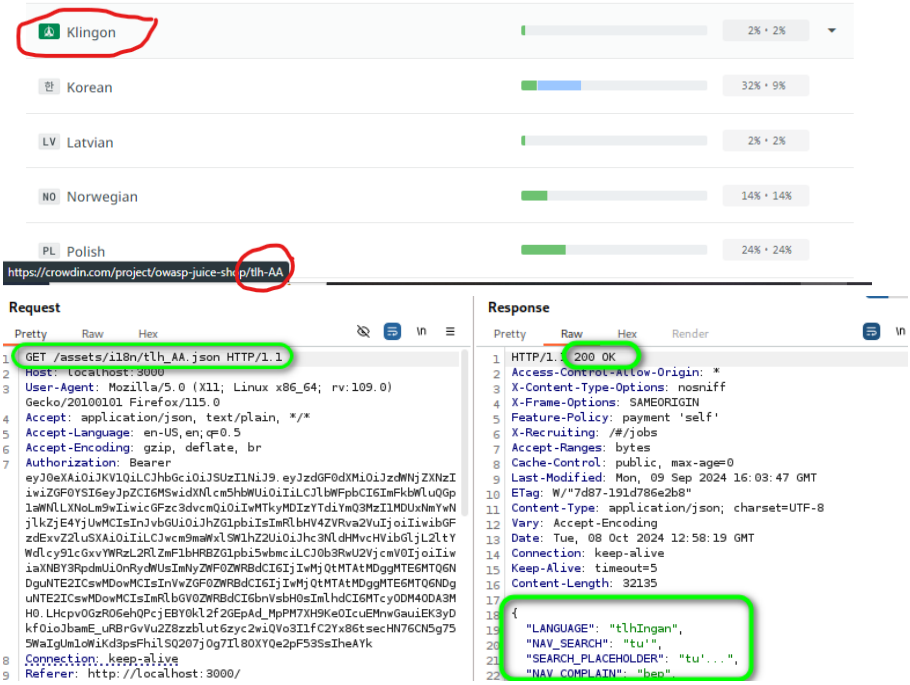
- **Impact:**
  - **Business Impact**: Attackers can automate submissions, leading to spamming or DoS-style attacks, affecting the integrity of user-generated content and application performance.
  - Medium **Risk Vulnerabilities.**

2.Extra Language

- **Description**: A vulnerability exists in the language selection feature of OWASP Juice Shop. By tampering with the language code parameter, unsupported or hidden languages can be accessed, such as the fictional "Klingon" (tlh_AA).
- **PoC**:
  - Analyzing the Language Change Request: the application retrieves language files based on two components:
    - ✓ Language Code
    - ✓ Country Code



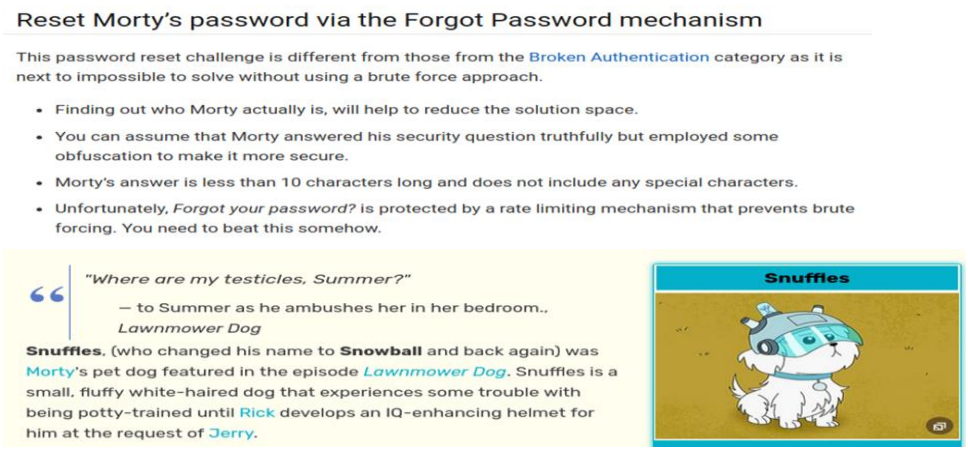
  - Exploring Available Languages: I used the official page of the project to search section related to languages development, as suggested in the hint.
  - With a hypothesis that there might be easter eggs or hidden languages, research was directed towards unusual or fictitious languages



- **Impact:**
  - **Business Impact**: These bypasses intended functionality, potentially exposing developer easter eggs or unintended data**.**
  - **High Risk Vulnerabilities.**

3.Reset Morty’s password

- **Description:** This vulnerability involves resetting Morty’s password by exploiting a weak security question system and bypassing anti-brute force mechanisms. The security question ("What is the name of your favorite pet?") can be answered using variations of known answers combined with leet speak.
- **PoC:**
  - Preparation and Initial Analysis



  - Based on provided hints, the potential answers are "Snuffles" or "SnowBall". The task is complicated by the requirement to use an obfuscated password that must be fewer than 10 characters, without special characters, and a necessity to bypass the anti-brute-force mechanism by manipulating the X-Forwarded-For header.
  - We will make a wordlist then we will brute-force : Start the tool to iterate over the generated wordlist. Each password attempt includes a modified X-Forwarded-For header to prevent detection by the brute force mechanism.
  - the password is <5N0wb41L>

- **Impact:**
  - **Business Impact:** Allows unauthorized password resets, leading to account compromise.
  - **High Risk Vulnerabilities.**

Remediation plan for the Broken Anti-Automation

1. **CAPTCHA Bypass**: Strengthen CAPTCHA mechanisms to prevent bot interactions.
2. **Extra Language:** Sanitize and validate all user inputs to block automation abuse.
3. **Reset Morty's Password:** Use strict authentication and anti-bot measures for password resets.

General Remediation: Implement robust anti-automation mechanisms, including CAPTCHA, rate limiting, input validation, and monitoring. These controls prevent automated abuse, ensuring secure user interactions.

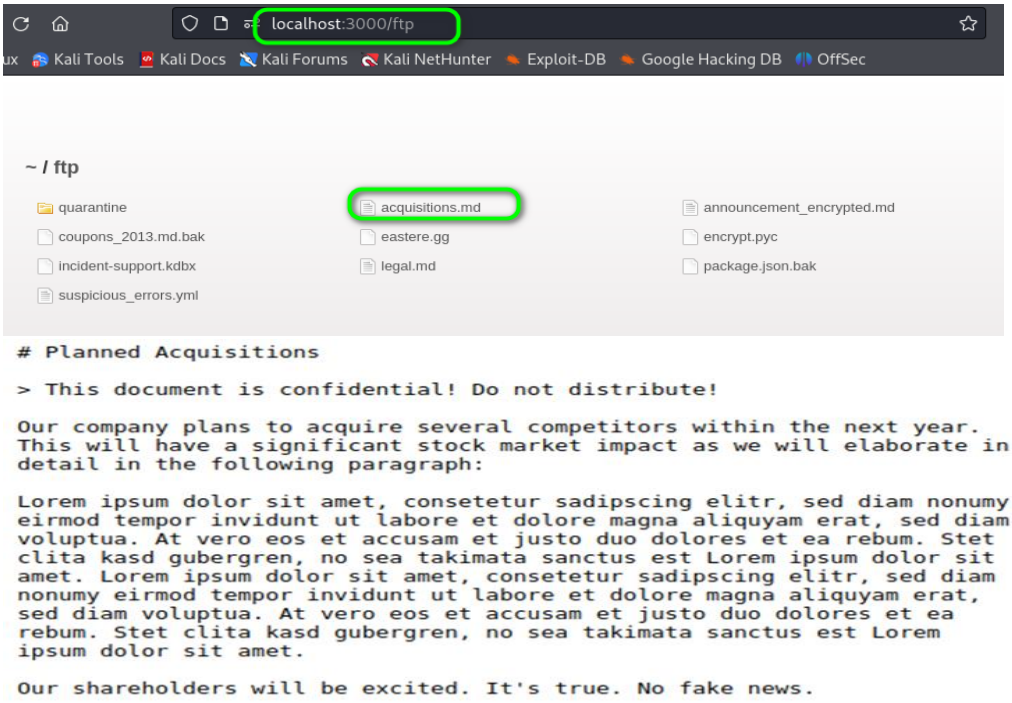
Methodology used to identify Anti-Automation

2. **Brute-force Testing:** Repeated login attempts without CAPTCHA verification.
3. **Bypassing CAPTCHA:** Exploiting weaknesses in the CAPTCHA system.
4. **Rate Limiting Tests:** Checking if the application allows excessive requests (e.g., login attempts) without throttling.
5. **Automated Input Submissions:** Testing forms or feedback mechanisms to see if automated tools can submit data without human interaction.

2.Sensitive Data Exposure

1.Confidential Document

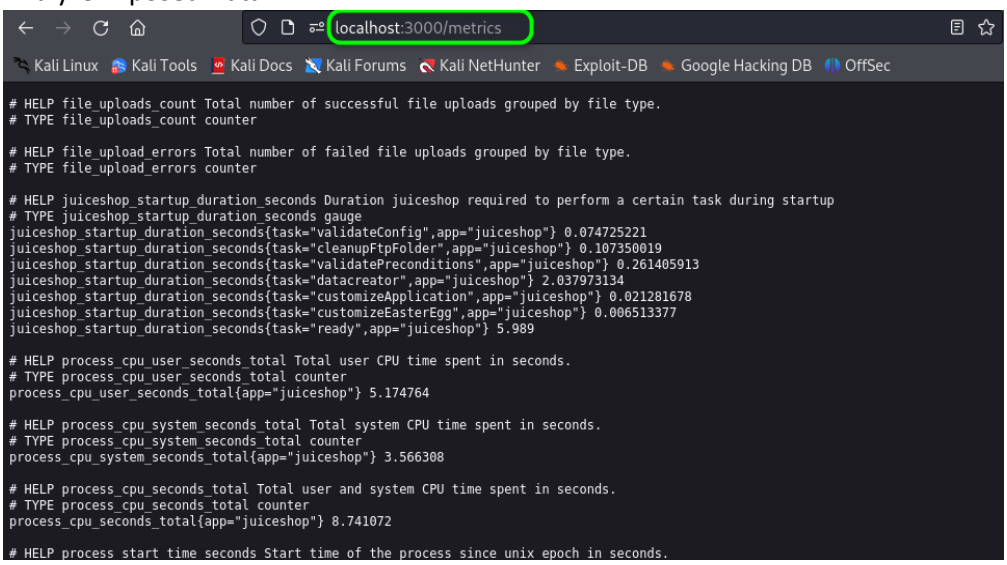
- **Description:** Sensitive data was exposed by storing a confidential document (acquisition.md) on an insecure FTP server, which could be accessed without authentication or authorization.
- **PoC:**
  - Access the FTP Server
  - Identify the Target File
  - Download and Review the File
  - Analyze the Content



- **Impact:**
  - **Business Impact:** Unauthorized access to strategic business plans, leading to potential financial or reputational damage.
  - **Low Risk Vulnerabilities.**

2.Exposed Metrics

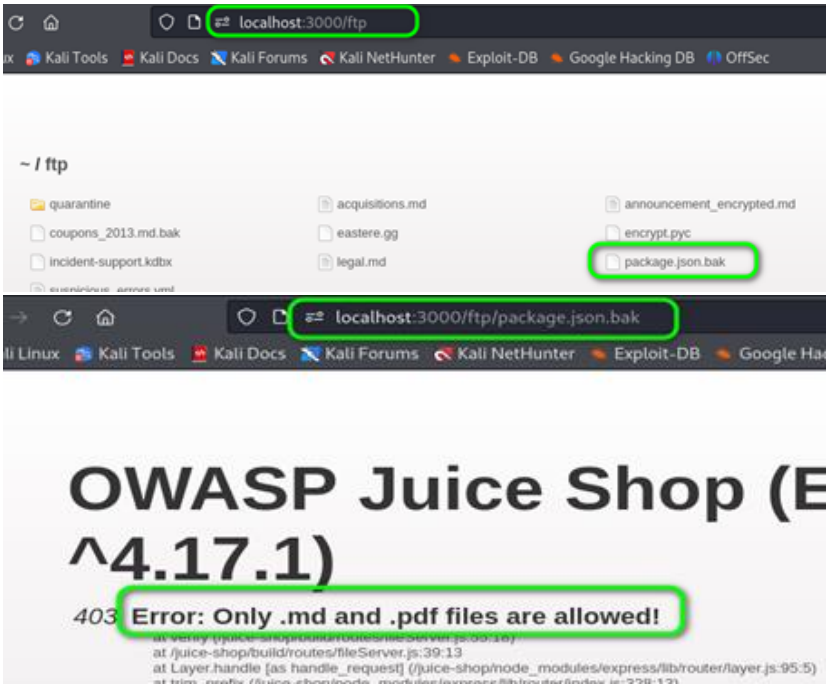
- **Description:** An exposed Prometheus metrics endpoint (/metrics) allows unauthorized access to sensitive operational data, including system health, CPU usage, and service runtime metrics.
- **PoC:**
  - Review Application Documentation
  - Access the Endpoint
  - Analyze Exposed Data



- **Impact:**
  - **Business Impact**: Exposes critical system information that could aid attackers in understanding system internals and launching targeted attacks.
  - **Low Risk Vulnerabilities.**

3.Forgotten Developer Backup

- **Description**: A developer's backup file containing sensitive data was left exposed in the public directory, allowing unauthorized access to confidential information.
- **PoC**:
  - Access the FTP Server
  - Identify the Target File

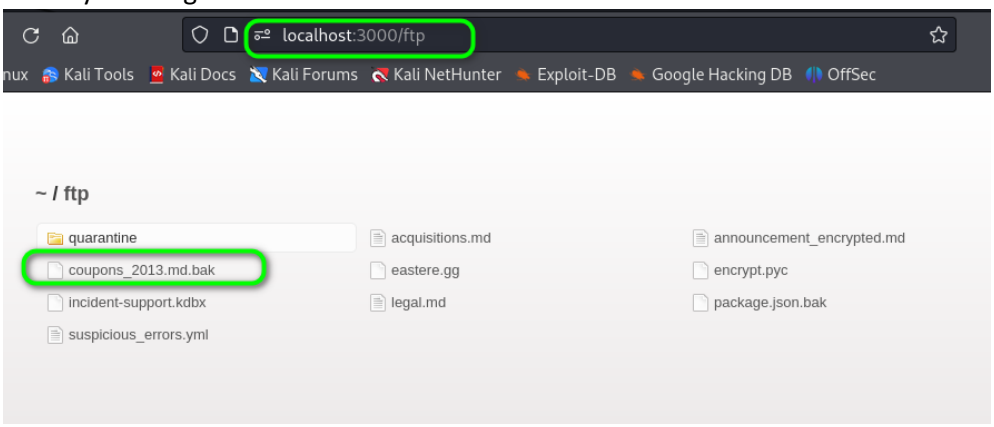


  - **Exploit NULL Byte Injection**

- **Impact**:
  - **Business Impact**: Unauthorized users can access sensitive source code, configuration files, or database dumps, leading to potential data leaks or application compromise.
  - Medium **Risk Vulnerabilities.**

4.Forgotten Sales Backup

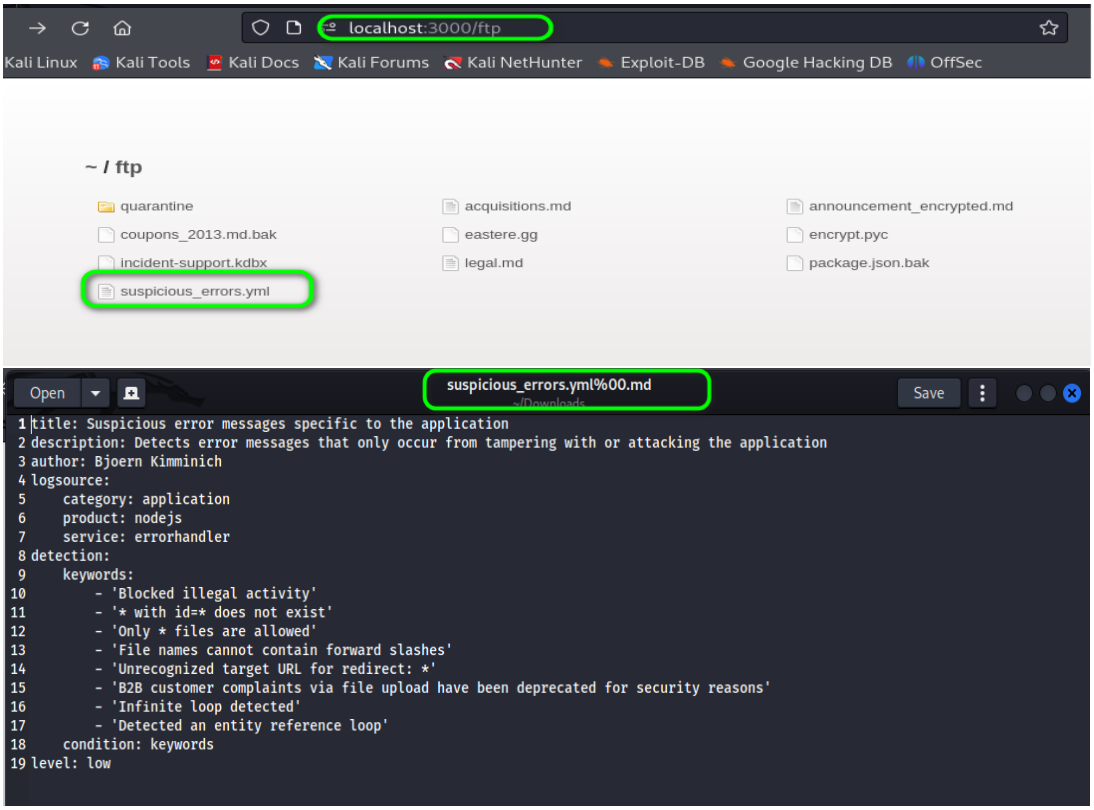
- **Description:** The Forgotten Sales Backup vulnerability occurs when a backup file (coupons_2013.md.bak) is left accessible on a public server, exposing sensitive sales data. This flaw highlights the risks of improperly secured backup files.
- **PoC:**
  - Access the FTP Server
  - Identify the Target File



- **Impact:**
  - **Business Impact:** Exposes confidential sales data, which can lead to financial loss and loss of customer trust.
  - Medium **Risk Vulnerabilities.**

5.Misplaced Signature File

- **Description**: The Misplaced Signature File vulnerability in OWASP Juice Shop occurs when a developer accidentally uploads a sensitive signature file to an insecure location within the web application. This can lead to unauthorized access to the contents of the file, potentially exposing critical data or application logic. This type of vulnerability highlights the risks associated with improper file handling and deployment practices in a web application.
- **PoC**:
  - Access the FTP Server
  - Identify the Target File



- **Impact**:
  - **Business Impact:** Unauthorized access to signature files can lead to data leakage and potential exploitation.
  - Medium **Risk Vulnerabilities.**

remediation plan for the Sensitive Data Exposure

- **Confidential Document:** Securely store confidential documents and enforce access permissions.
- **Exposed Metrics:** Restrict access to sensitive metrics and implement monitoring.
- **Forgotten Developer/Sales Backup:** Regularly audit backups and secure access to sensitive data.
- **Misplaced Signature File:** Secure sensitive files and review storage practices.

General Remediation: Enforce data encryption, access controls, and regular audits to protect sensitive information from unauthorized exposure and comply with privacy regulations.

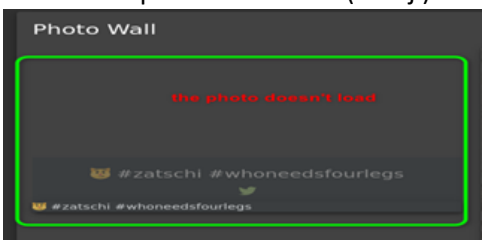
The methodology used to find Sensitive Data Exposure

1. **Testing Unencrypted Data Transmission:** Checking if sensitive data like passwords, tokens, or personal information is sent over HTTP instead of HTTPS.
2. **Exposed Endpoints:** Searching for API or web endpoints that reveal sensitive data without proper authorization.
3. **Weak Data Storage**: Verifying if sensitive information is stored insecurely (e.g., plaintext passwords or logs).
4. **Error Messages & Debug Info:** Reviewing error messages for sensitive data leakage.

3.Improper Input Validation

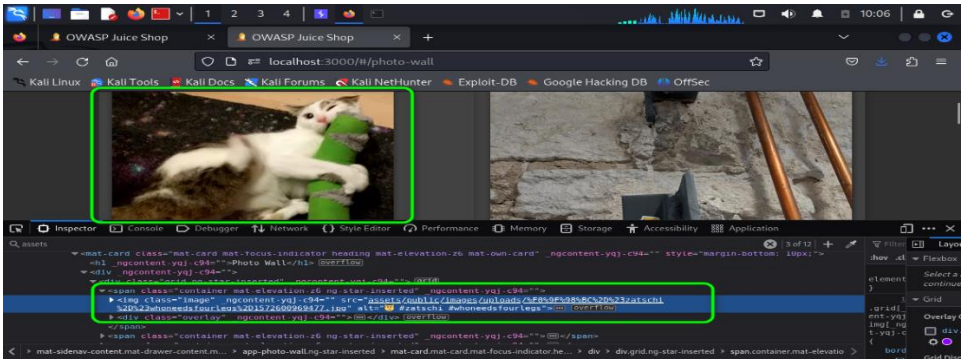
1.Missing Encoding

- **Description:** The Missing Encoding vulnerability arises when user input is not properly encoded before being rendered in the application, potentially leading to Cross-Site Scripting (XSS) attacks.
- **PoC:**
  - Identify the Issue: Noticed an image that was not loading on the photo wall of the application. Inspecting the element showed that the src attribute of the image tag contained special characters (emoji) which were not URL encoded.



  - Understand the Problem: Realized that the lack of proper URL encoding for the special characters in the image source URL caused the browser to fail in fetching the image.

  - Use URL Encoding Tool
  - Replace the URL <assets%2Fpublic%2Fimages%2Fuploads%2F%F0%9F%98%BC-%23zatschi-%23whoneedsfourlegs-1572600969477.jpg>



- **Impact:**
  - **Business Impact:** This vulnerability can lead to unauthorized actions being performed on behalf of users, including session hijacking and data theft.
  - **Low Risk Vulnerabilities.**

2.Repetitive Registration

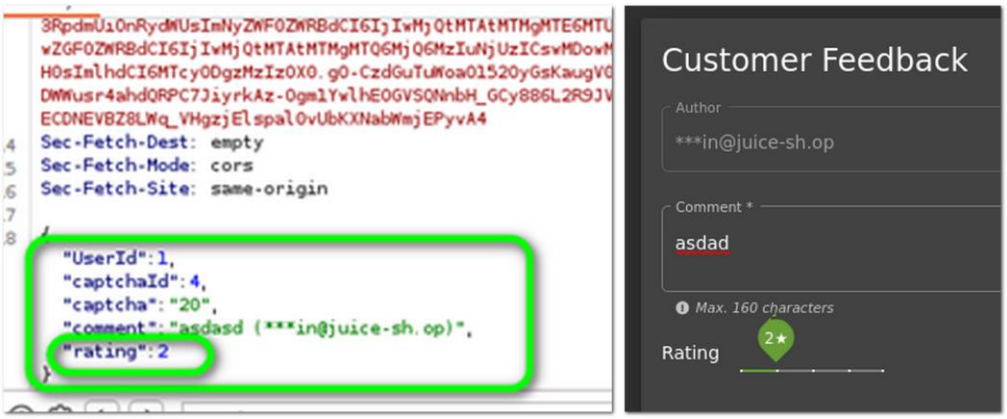
- **Description:** The Repetitive Registration vulnerability occurs when the application allows users to register with mismatched passwords due to improper server-side validation. While client-side checks may prevent submission initially, these can be easily bypassed, leading to account creation without proper password confirmation.
- **PoC:**
  - Fill Out the Registration Form

  - Modify Form Request:

- **Impact:**
  - **Business Impact:** This vulnerability can lead to unauthorized account creation, compromising user accounts and the overall integrity of the registration process.
  - Low **Risk Vulnerabilities.**

3.Zero Stars

- **Description:** The Zero Stars vulnerability involves exploiting improper input validation to submit a zero-star rating for a product, which should not be allowed by the application. This can be achieved by manipulating network requests or altering client-side controls.
- **PoC:**
  - Intercepting and Modifying Network Requests

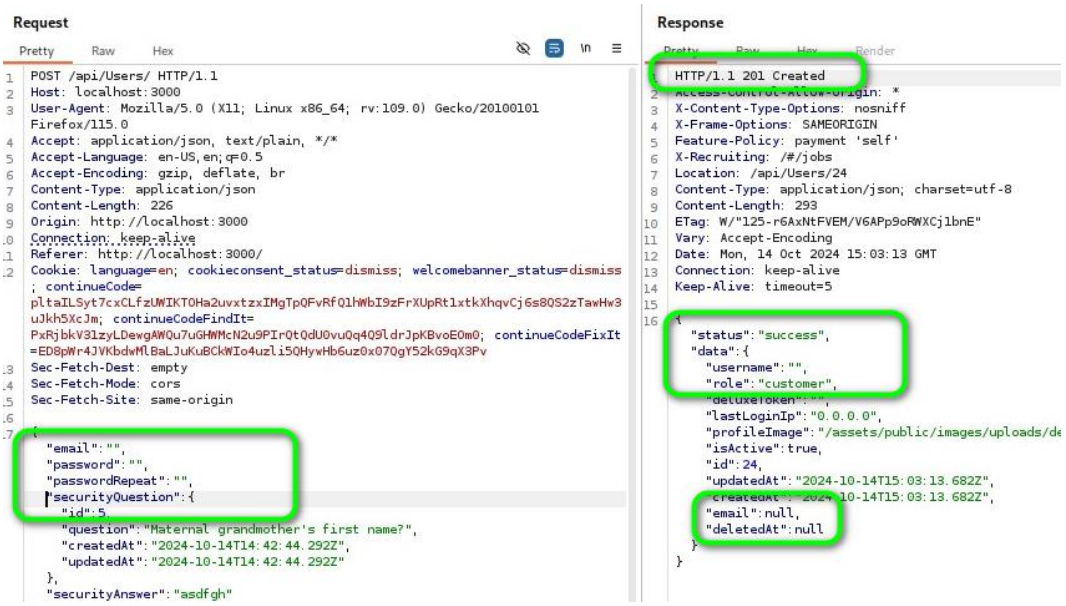


  - Manipulating the request

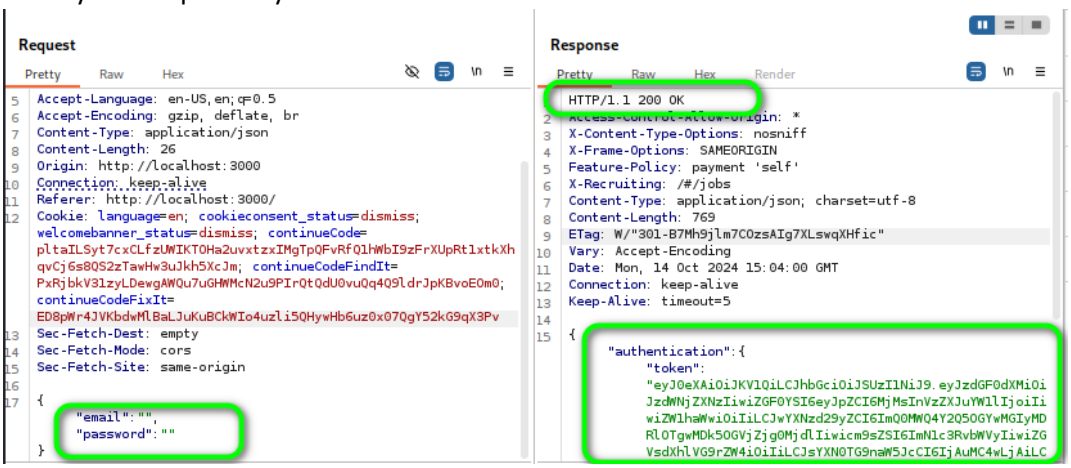
- **Impact:**
  - **Business Impact**: By enabling zero-star ratings, users can negatively impact the reputation of products, leading to potential financial losses for the business.
  - **Low Risk Vulnerabilities.**

4.Empty User Registration

- **Description:** The Empty User Registration vulnerability allows users to create an account without providing essential information, such as an email and password, due to inadequate input validation on the server side. This can be exploited by manipulating the registration request.
- **PoC:**
  - Intercept the Registration Request:



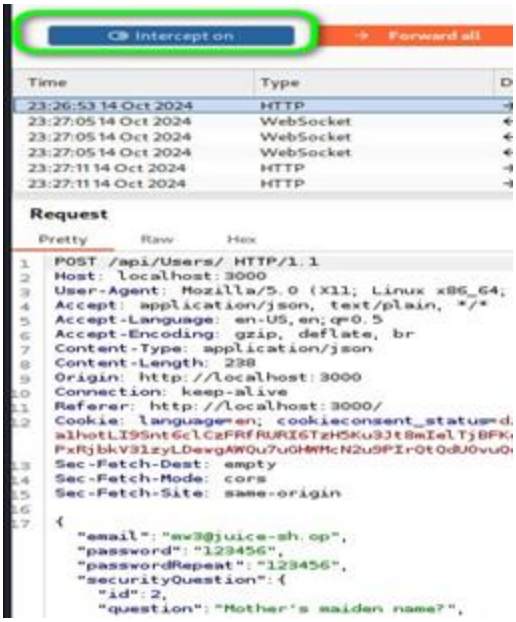
  - Modify the Request Payload:



- **Impact:**
  - **Business Impact**: This vulnerability can lead to unauthorized account creation, making it possible for malicious actors to generate accounts without valid credentials, compromising the integrity of user data.
  - **Low Risk Vulnerabilities.**

5.Admin Registration

- **Description**: The Admin Registration vulnerability occurs when an attacker can exploit improper input validation during the registration process. By modifying the role in the HTTP request from "customer" to "admin," unauthorized users can register themselves with administrative privileges.
- **PoC:**
  - Interception and Modification of Request

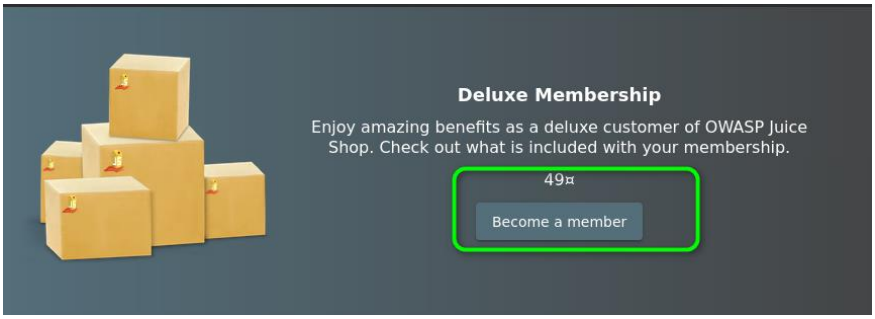


  - Modifying User Role:

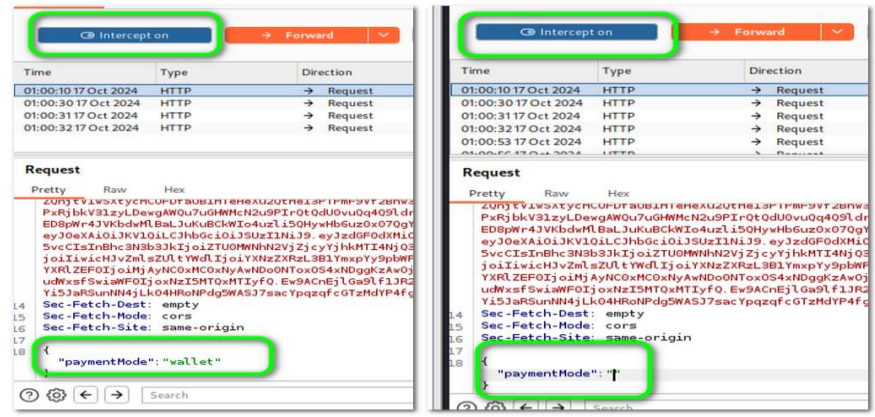
- **Impact:**
  - **Business Impact**: This leads to unauthorized elevation of privileges, allowing attackers full control over the application and its sensitive data.
  - Medium **Risk Vulnerabilities.**

6.Deluxe Fraud

- **Description:** The Deluxe Fraud vulnerability allows users to manipulate payment parameters during a Deluxe Membership purchase, bypassing the payment process entirely through improper validation.
- **PoC:**
  - Navigating to Deluxe Membership Page:



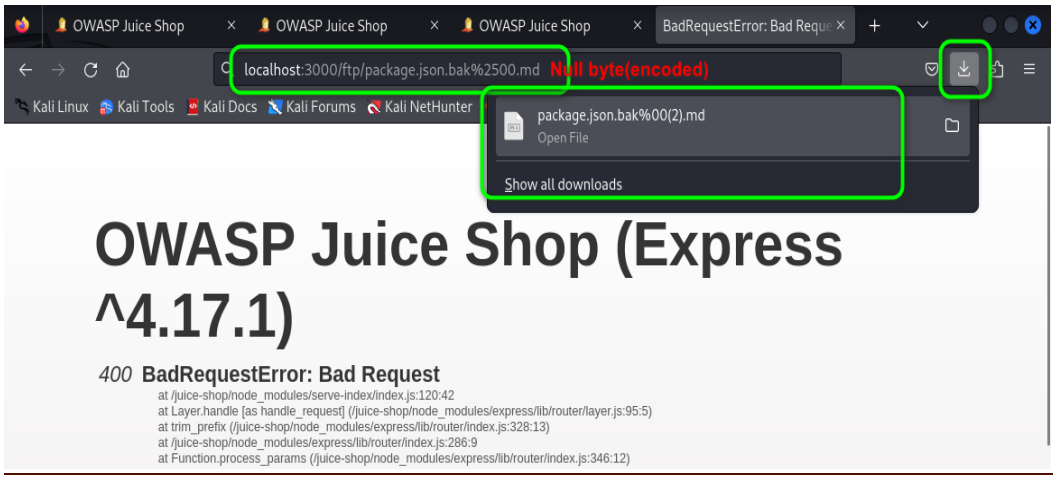
  - Intercepting API Requests and Manipulating Payment Process:



- **Impact:**
  - **Business Impact:** This vulnerability can result in financial losses for the application, as users can gain premium access without completing a legitimate transaction.
  - Medium **Risk Vulnerabilities.**

7.Poison Null Byte

- **Description:** The Poison Null Byte vulnerability occurs when an application improperly handles null bytes (%00) in file paths, allowing attackers to bypass file type restrictions by truncating strings. This can lead to unauthorized access to sensitive files.
- **PoC:**



- **Impact:**
  - **Business Impact:** This vulnerability can expose restricted files or sensitive data, leading to potential data breaches or application compromise.
  - Medium **Risk Vulnerabilities.**

Remediation plan for the Improper Input Validation

1. **Admin Registration**: Validate and sanitize input data for admin registration to prevent unauthorized access.
2. **Deluxe Fraud**: Ensure robust validation of all input fields to prevent fraudulent activities.
3. **Empty User Registration**: Implement checks to prevent empty submissions during user registration.
4. **Missing Encoding**: Implement proper encoding for all outputs to prevent injection attacks.
5. **Poison Null Byte**: Sanitize inputs to prevent null byte injection.
6. **Repetitive Registration**: Implement checks to prevent duplicate user registrations.
7. **Zero Stars**: Validate user ratings to prevent invalid submissions.

General Remediation: Implement comprehensive input validation and sanitization for all user inputs across the application to prevent exploitation and ensure data integrity.

Methodology used to find Improper Input Validation

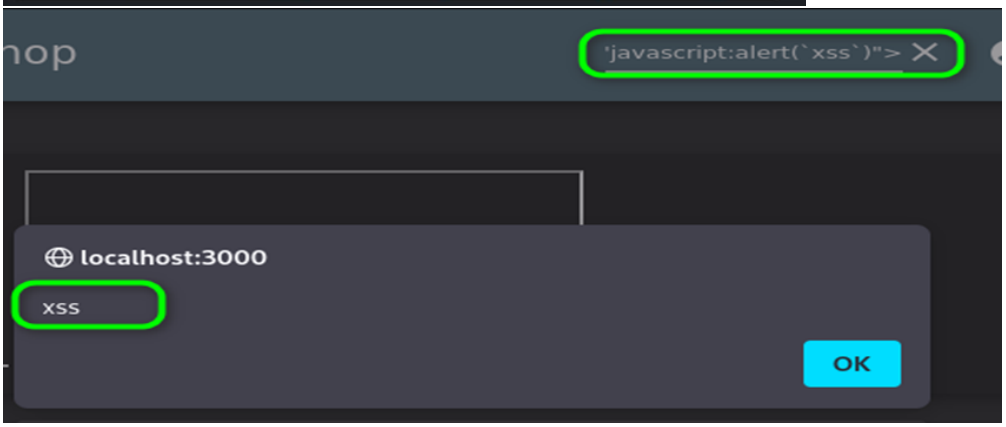
1. **Fuzzing**: Testing input fields with random or malicious data (e.g., special characters, long strings) to see if the system properly validates it.
2. **Boundary Testing**: Submitting data at or beyond allowed input limits (e.g., input size or type restrictions).
3. **Injection Attacks**: Attempting SQL, command, or code injection by exploiting improper input sanitization.
4. **Client-Side Validation Bypass**: Disabling JavaScript to bypass client-side checks and sending unauthorized data directly to the server.

3.Cross-Site Scripting (XSS)

1.DOM XSS

- **Description:** The DOM-based Cross-Site Scripting (XSS) vulnerability occurs when an attacker injects malicious scripts into a web application by manipulating the client-side JavaScript code, exploiting improper handling of user inputs.
- **PoC**
  - Identify the Injection Point and Craft the Payload:

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



- **Impact:**
  - **Business Impact:** This can lead to the execution of unauthorized scripts in the victim's browser, resulting in session hijacking, data theft, or unauthorized actions within the application.
  - Low **Risk Vulnerabilities.**

2.Bonus payload

- **Description:** The Bonus Payload vulnerability refers to an additional hidden payload that can be injected into the application by exploiting flaws in its input handling, allowing attackers to trigger unintended behavior or bypass security measures.
- **PoC:**

```
<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=tru
e&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_
teaser=true"></iframe>
```

- **Impact:**
  - **Business Impact**: This can result in unauthorized code execution or data manipulation, posing a threat to the security and functionality of the application.
  - low **Risk Vulnerabilities.**

Remediation plan for the XSS

1. **Bonus Payload:** Implement output encoding for dynamic content to mitigate injection.
2. **DOM XSS:** Validate and sanitize DOM manipulations to prevent unauthorized scripts.

General Remediation: Implement comprehensive input validation, output encoding, and security headers to protect against various XSS attacks across all application components.

Methodology used to find XSS

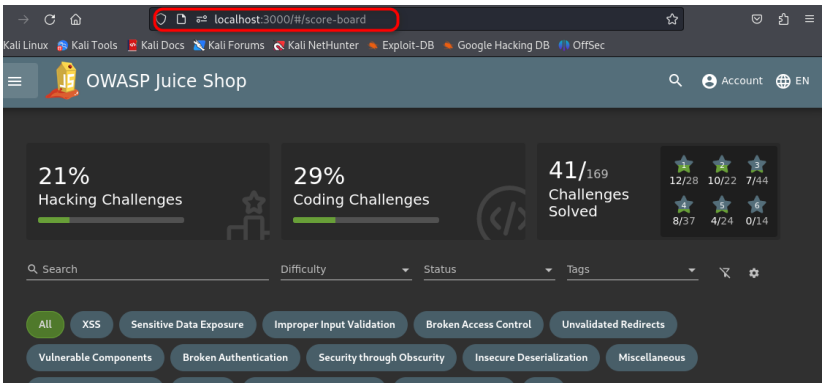
1.Analyzing JavaScript: Review client-side scripts that manipulate the DOM.

2.Crafting Payloads: Insert XSS payloads into input fields or URLs to test for execution in the browser.

4.Miscellaneous

1.Score Board

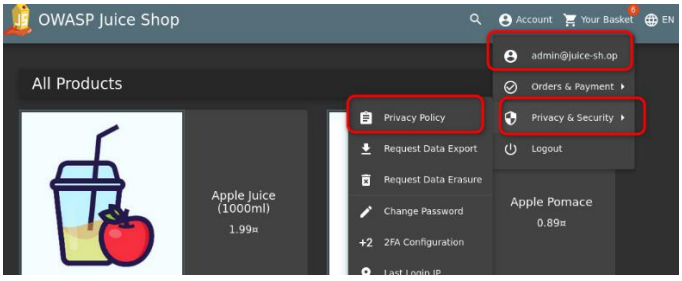
- **Description:** The Score Board vulnerability in Juice Shop involves unauthorized access to the Score Board, where challenge progress is displayed. Attackers can access or manipulate this feature by bypassing authentication or exploiting weak access controls.
- **PoC**



- **Impact:**
  - **Business Impact:** It undermines the challenge system, allowing users to falsely claim achievements or view hidden challenges without proper authorization.
  - Medium **Risk Vulnerabilities.**

2.Privacy Policy

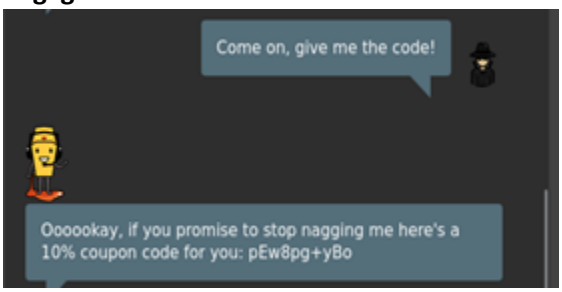
- **Description:** The Privacy Policy vulnerability occurs when sensitive information is accessible through the Privacy Policy page or related endpoints due to improper handling of data exposure and access control.
- **PoC**



- **Impact:**
  - **Business Impact:** This can lead to unauthorized access to sensitive information, including user data, potentially causing privacy breaches and legal compliance issues.
  - Medium **Risk Vulnerabilities.**

3.Bully Chatbot

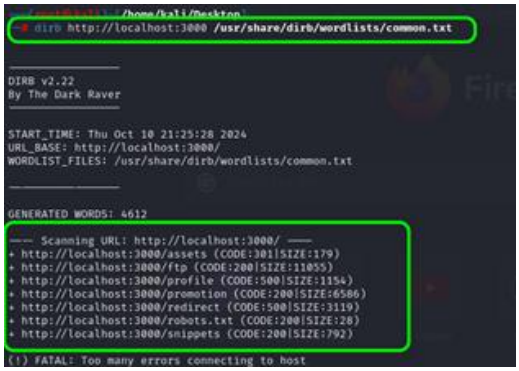
- **Description:** The Bully Chatbot vulnerability arises when users can manipulate or exploit the chatbot's responses through crafted input, causing the bot to display inappropriate or harmful content, often due to insufficient input validation.
- **PoC**
  - **Engage the Chatbot**



- **Impact:**
  - **Business Impact:** This can lead to abusive or harmful interactions with the chatbot, damaging the user experience and exposing the system to reputational harm.
  - Medium **Risk Vulnerabilities.**

4.Security Policy

- **Description:** The Security Policy vulnerability arises when the security policies of an application are improperly implemented or exposed, potentially allowing attackers to identify and exploit weak points in the system's defense mechanisms.
- **PoC**



- **Impact:**
  - **Business Impact:** This can lead to attackers gaining insights into the system's defenses, increasing the likelihood of successful attacks by exploiting known vulnerabilities or bypassing security controls.
  - Low **Risk Vulnerabilities.**

Remediation plan for the Miscellaneous

- **Bully Chatbot**: Implement content filtering and monitoring to prevent abusive interactions.
- **Privacy Policy**: Ensure the privacy policy is up-to-date and accessible.
- **Score Board**: Validate scores to prevent manipulation and ensure fair play.
- **Security Policy**: Review and enforce security policies consistently across the application.

General Remediation: Enforce strong security practices, including input validation, user monitoring, and regular updates of policies to enhance overall application security.

Methodology used to find Miscellaneous

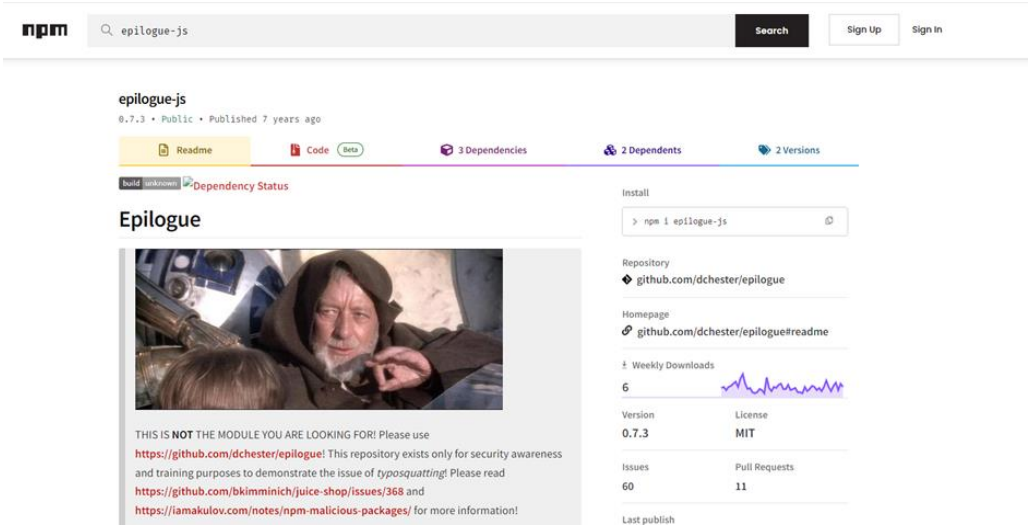
1. **Exploratory Testing:** Manually navigate through various application features to identify unexpected behaviors.
2. **Input Manipulation:** Alter request parameters, headers, or payloads to uncover vulnerabilities.
3. **Error and Response Analysis**: Review error messages and application responses for sensitive data exposure or improper handling.
4. **Functionality Testing:** Assess fewer common features (e.g., chatbots, scoreboards) for weaknesses.

5.Vulnerable components

1.Legacy Typosquatting

- **Description**: This challenge requires identifying a typosquatted library used by Juice Shop, a form of cyber threat where attackers exploit common typographical errors to push malicious or deceptive packages.
- **PoC:**
  - Review Hint and Background Information
  - Inspecting Developer's Backup

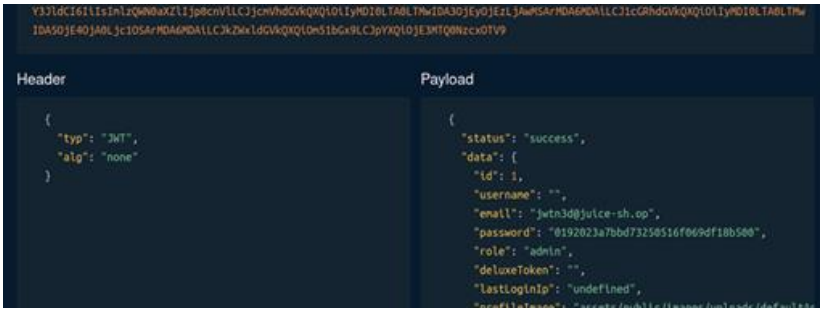
  - Investigate Packages on npm
  - Identifying the Typo squatted Package



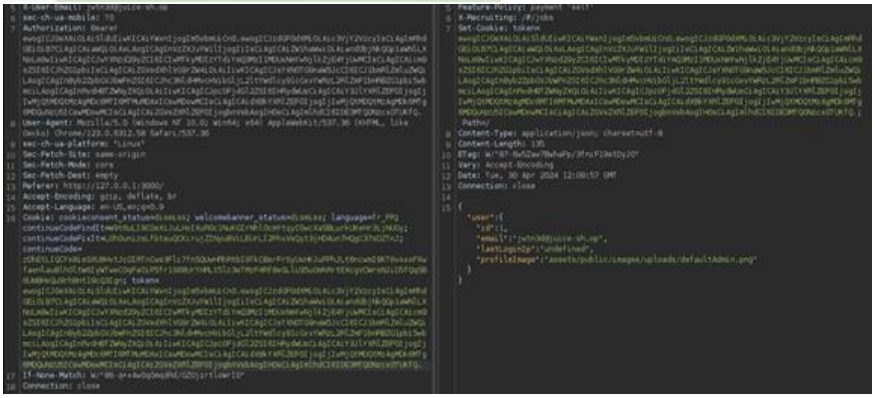
- **Impact:**
  - **Data Breach**: If the malicious package gains access to user data or system configurations, it could lead to the exposure of confidential information.
  - Code Integrity Compromise: The injected malicious package can alter the behavior of the application, resulting in unexpected crashes, vulnerabilities, or malicious activities.
  - Medium **Risk Vulnerabilities.**

2.Unsigned JWT

- **Description:** This challenge focuses on exploiting a vulnerability related to the improper verification of JSON Web Tokens (JWTs) in the application. The goal is to modify an existing JWT to bypass authentication checks without a valid signature.
- **PoC:**
  - Understanding JWT



  - Capture and Decode a JWT



- **Impact:**
  - **Business Impact:** Authentication Bypass: By manipulating the JWT header to specify "alg": "none", attackers can bypass the signature verification step, effectively allowing unauthorized access to protected resources or administrative privileges without possessing a valid token.
  - **High Risk Vulnerabilities.**

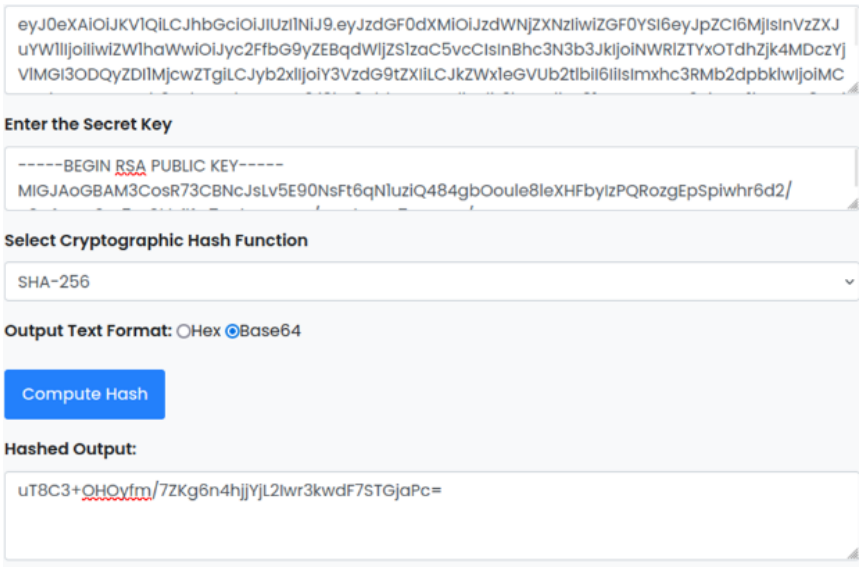
3.Forged Signed JWT

- **Description:** The objective of this challenge is to create a forged JWT that appears to be correctly signed by RSA but actually uses an algorithm confusion attack to bypass security measures.
- **PoC:**
  - Retrieve Existing JWT and Analyze JWT Structure then Obtain Public RSA Key

Like Forge an essentially unsigned JWT token this challenge requires you to make a valid JWT for a user that does not exist. What makes this challenge even harder is the requirement to have the JWT look like it was properly signed.

- The three generic hints from Forge an essentially unsigned JWT token also help with this challenge.
- Instead of enforcing no encryption to be applied, try to apply a more sophisticated exploit against the JWT libraries used in the Juice Shop.
- Getting your hands on the public RSA key the application employs for its JWTs is mandatory for this challenge.
- Finding the corresponding private key should actually be impossible, but that obviously doesn't make this challenge unsolvable.
- Make sure your JWT is URL safe!

  - Sign JWT with HMAC Using RSA Public Key

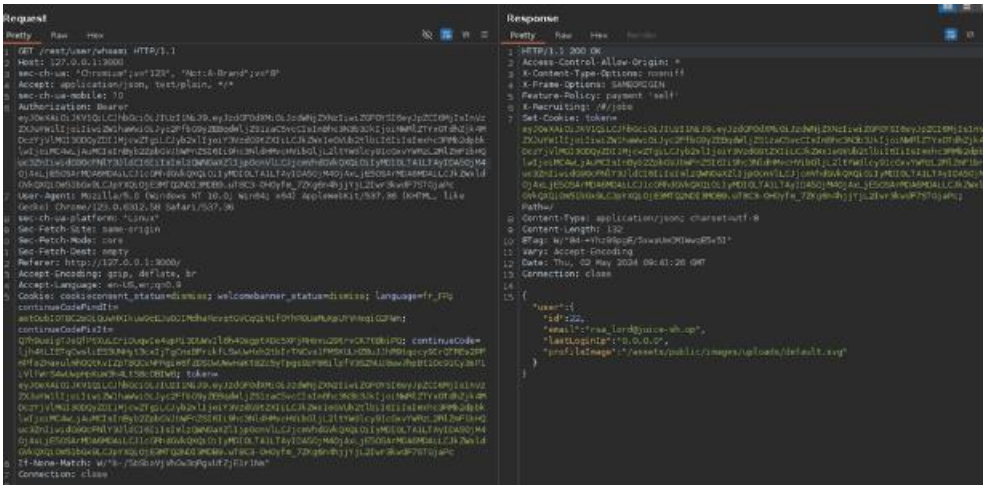


- Finally we replace older signature by the new one :

Encoded

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
JzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZ
CI6MjIsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJy
c2FfbG9yZEBqdWljZS1zaC5vcCIsInBhc3N3b3J
kIjoiNWRlZTYxOTdhZjk4MDczYjVlMGI3ODQyZD
I1MjcwZTgiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZ
Wx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoi
MC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3N
ldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZm
F1bHQuc3ZnIiwidG90cEFNlY3JldCI6IiIsImlzQ
WN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI0
LTA1LTAyIDA5OjM4OjAxLjE5OSArMDA6MDAiLCJ
1cGRhdGVkQXQiOiIyMDI0LTA1LTAyIDA5OjM4Oj
AxLjE5OSArMDA6MDAiLCJkZWxldGVkQXQiOm51b
Gx9LCJpYXQiOjE3MTQ2NDI3MDB9.uT8C3-
OHOyfm_7ZKg6n4hjjYjL2Iwr3kwdF7STGjaPc
```

- Send Forged JWT to Server



- ➢ **Impact:**
  - **Privilege Escalation:** The attacker can forge JWT tokens to impersonate users with higher privileges, such as administrators, allowing full control over the system.
  - **Authentication Bypass:** By exploiting the algorithm confusion vulnerability, the attacker bypasses authentication mechanisms, compromising the entire user authentication process.
  - **Data Exposure:** Unauthorized access to sensitive user data, including personal and financial information, can result from the attack.
  - **High Risk Vulnerabilities.**

Remediation plan for the Vulnerable Components

1. **Forged Signed JWT**: Use strong cryptographic algorithms and verify JWT signatures to prevent forgery.
2. **Legacy Typosquatting**: Transition away from legacy systems to reduce exposure to Typosquatting.
3. **Unsigned JWT**: Ensure all JWTs are signed and validated.

General Remediation: Regularly update and patch components, validate all inputs, and implement strong security practices to protect against vulnerabilities in application components.

Methodology used to find Vulnerable Components

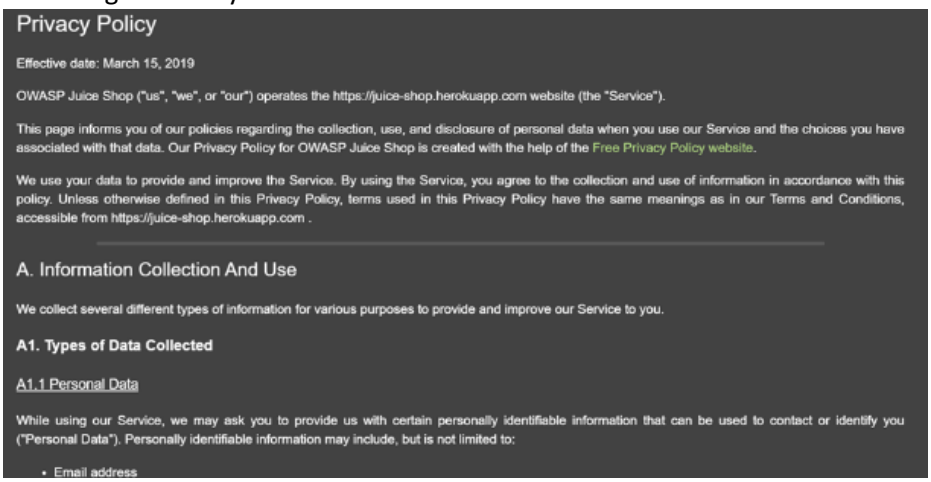
1. **Dependency Scanning**: Utilizing tools like npm audit to identify outdated or vulnerable libraries in the application.
2. **Version Checking**: Comparing current versions of components against known vulnerabilities in databases (e.g., CVE).
3. **Reviewing Documentation**: Analyzing third-party components for any security advisories or vulnerabilities reported.
4. **Manual Testing**: Exploring functionality to determine if vulnerable components can be exploited in the application.

6.Security through Obscurity

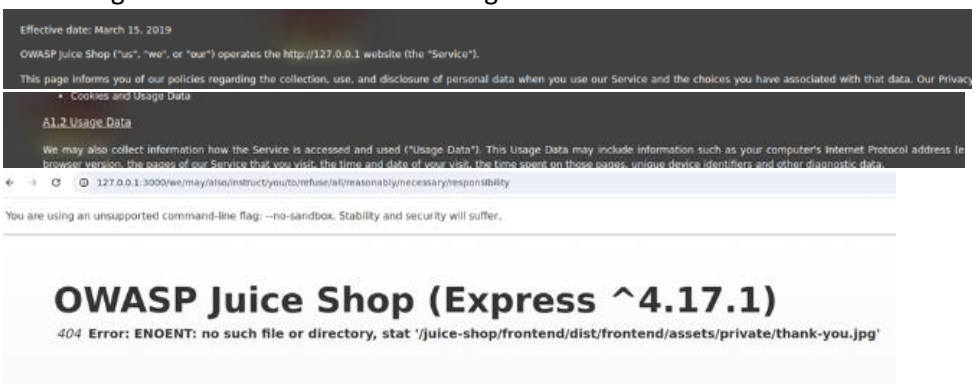
1.Privacy Policy Inspection

- **Description**: The "Privacy Policy Inspection" challenge requires to demonstrate their familiarity with the privacy policy by uncovering hidden elements or messages within the policy document.

- **PoC:**
  - Accessing the Policy



  - Examining the Document and Discovering Hidden Elements

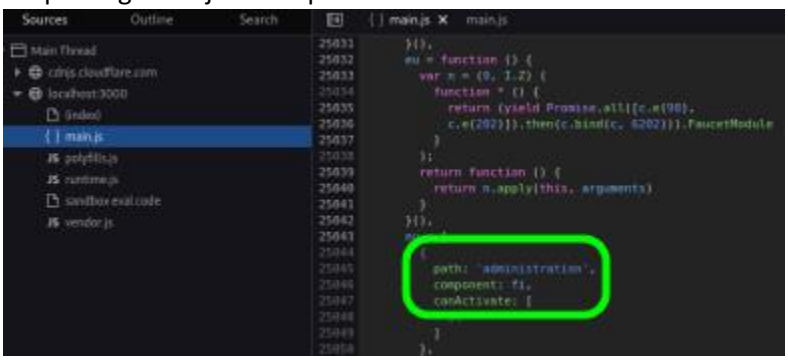


- **Impact:**
  - **Sensitive Data Disclosure:** If real-world applications implement similar techniques, it could expose unintended content or sensitive information to users or attackers who can bypass standard access controls by manipulating URLs.
  - **Unauthorized Access**: Finding hidden elements or URLs may lead to exposure of private documents, system files, or sensitive company information.
  - Medium **Risk Vulnerabilities.**
- **Remediation:**
  - **Revise Privacy Policy:** Ensure it is clear, compliant with regulations, and includes mechanisms for obtaining user consent for data collection, with regular reviews to maintain its relevance.

7.Broken Access Control

1.Admin Section

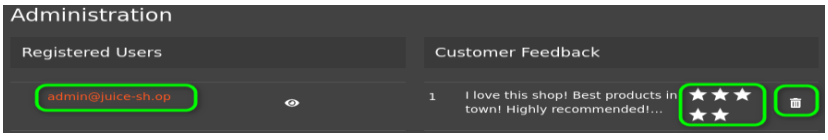
- **Description**: Admin Section (Access the administration section of the store.) The URL is http://localhost:3000/#/administration . This vulnerability allows unauthorized users to access the administration section of the store, potentially giving them control over sensitive functionalities and data.
- **PoC:**
  - Sensitive Data Exposure in Main.js
  - Inspecting main.js found path administration



- **Impact:**
  - **Unauthorized Access:** Attackers can gain access to sensitive administrative functionalities.
  - **Data Breach:** Sensitive data such as user information, product details, and configuration settings can be viewed or modified**.**
  - Medium **Risk Vulnerabilities.**

2.Five-Star Feedback

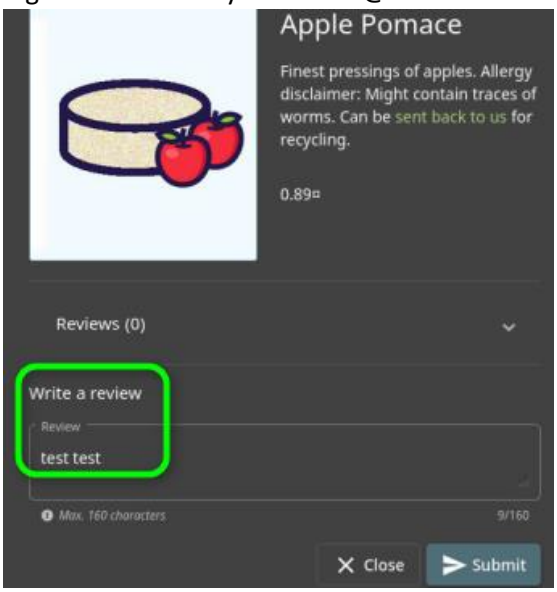
- **Description:** Five-Star Feedback (Get rid of all 5-star customer feedback.) The "Five-Star Feedback" vulnerability allows an authenticated admin user to delete all 5-star customer feedback through the Admin section**.**
- **PoC:**
  - first login as admin and use Admin section to access the admin pages and scroll in customer feedback and found five-star feedback and delete it.



- **Impact:**
  - **Reputation Damage:** Positive feedback is crucial for maintaining a good reputation. Deleting 5- star feedback can negatively impact the organization's image.
  - **Loss of Trust:** Customers may lose trust if they notice that their positive feedback has been removed, leading to potential loss of business.
  - Medium **Risk Vulnerabilities.**

3.Forged Review

- **Description:** The "Forged Review" vulnerability allows an authenticated user to post a product review as another user or edit any user's existing review.
- **PoC:**
  - login as test user by email test@test.com like this

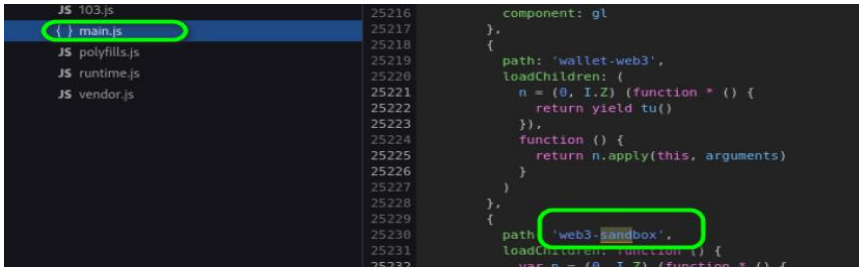


  - Then login as test and go to review the any product then go the burp suite tool and send request to repeater </rest/products/6/reviews> change the author attribute called author change the email to <hacker@hacker.com> so can any user change the reviews

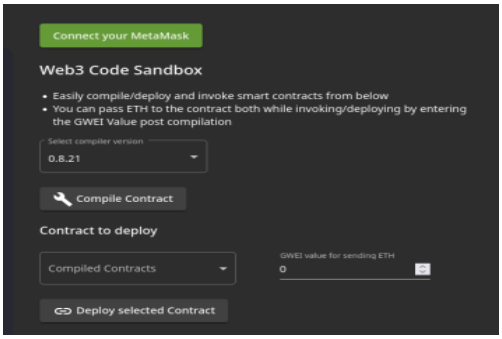
- **Impact**:
  - **Reputation Damage**: Malicious users can post negative or false reviews under other users' names, damaging the organization's reputation.
  - **Data Integrity**: The integrity of the review data is compromised, making it difficult to track genuine customer feedback.
  - Medium **Risk Vulnerabilities.**

4.Web3 Sandbox

- **Description:** Web3 Sandbox (Find an accidentally deployed code sandbox for writing smart contracts on the fly.) This environment allows users to create, test, and deploy smart contracts without proper security measures, potentially leading to unauthorized access or exploitation.
- **PoC:**
  - go to debugger tool in main.js file and search in sandbox to get path



  - after get path go to URL and found sandbox

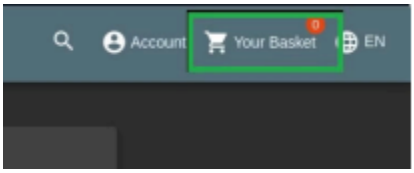


- **Impact:**
  - **Unauthorized Access:** Attackers can deploy and execute smart contracts that perform unauthorized actions on the blockchain.
  - **Data Leakage:** Sensitive data stored in the sandbox environment could be accessed or exfiltrated.
  - Medium **Risk Vulnerabilities.**

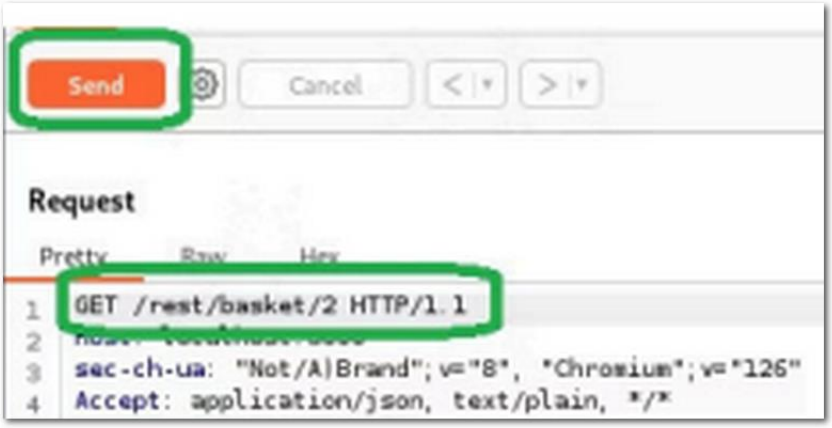
5.View Basket

- **Description**: Unauthorized access to the View Basket functionality was discovered.
- **PoC:**

- Click on "Your Basket"



- Go to burp http history and select this request then send to repeater and found that the request contain the UserId "1" change it to "2" another UserId and send the request



- **Impact:**
- **Business Impact**: Allows attackers to alter basket items and quantities, potentially leading to financial loss.
- Medium **Risk Vulnerabilities.**

Remediation plan for the Broken Access Control

1. **Admin Section**: Implement strict Role-Based Access Control (RBAC) to restrict admin privileges.
2. **Five-Star Feedback, Forged Review**: Validate user permissions for feedback submission.
3. **View Basket**: Enforce server-side validation for user actions.
4. **Web3 Sandbox**: Secure blockchain interactions with robust validation.

General Remediation: Implement **Role-Based Access Control (RBAC)** to enforce permissions, ensuring users only access what their roles allow. Use **server-side authorization** for all actions, validate user inputs, and restrict outbound network traffic to prevent SSRF. Apply **anti-CSRF tokens** to protect against unauthorized actions, and audit logs regularly for suspicious activities. This ensures robust access control and reduces the risk of unauthorized access across the system.

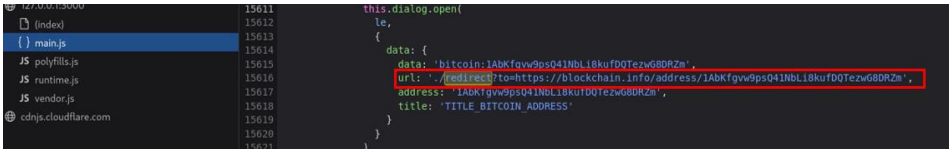
Methodology used to find Access Control

1. **Role Testing**: Attempting actions with different user roles to check for unauthorized access.
2. **Parameter Manipulation**: Altering request parameters to bypass restrictions.
3. **Functionality Exploration**: Investigating hidden or less obvious features for access control flaws.
4. **Session Management Review**: Analyzing session handling and cookie security to identify potential session hijacking risks.

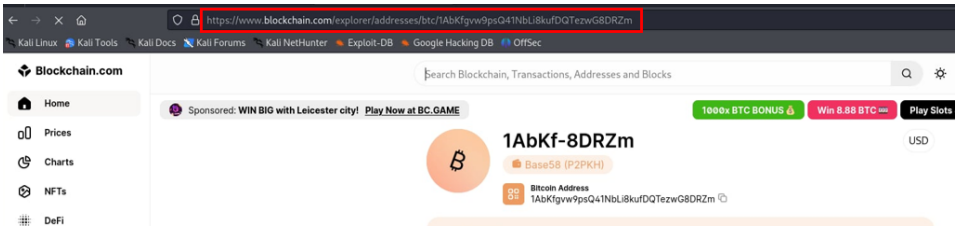
8.Unvalidated Redirects

1.Outdated Allowlist

- **Description**: The system uses an outdated allowlist for redirect URLs, allowing attackers to craft malicious URLs that can trick users into visiting external malicious sites.
- **PoC:**
  - First I searched in the JavaScript files for anything related to redirection and I found that redirection URL in main.js file



  - And when I added it to the Juice Shop URL it redirected me to a BlockChain page

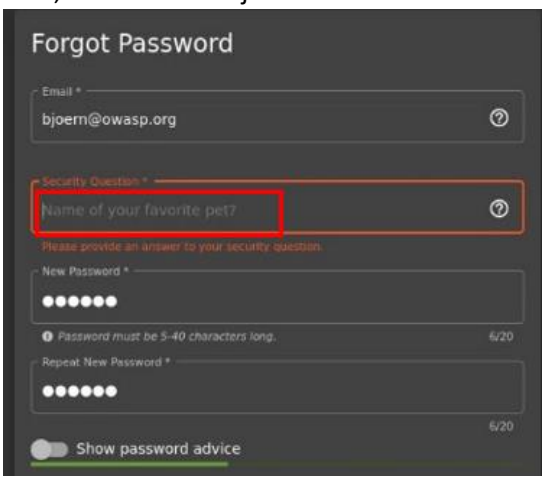


- **Impact:**
  - **Business Impact:** This vulnerability could lead to phishing attacks or the redirection of users to malicious websites, where attackers could steal sensitive information or deliver malware.
  - Medium **Risk Vulnerabilities.**
- **Remediation:**
  - **Regular Audits:** Frequently review the allowlist to remove outdated entries.
  - **Update Procedures:** Establish clear guidelines for managing the allowlist.
  - **Automated Monitoring:** Use tools to track changes and alert for outdated items.

9.Broken Authentication

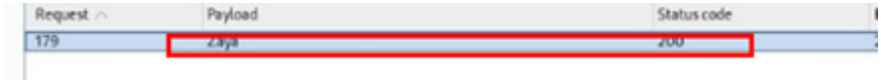
1.Bjoern's Favorite Pet

- **Description**: A user can discover Bjoern's favorite pet by exploiting a weak authentication mechanism, allowing unauthorized access to sensitive account information through weak security questions or predictable logic in the password retrieval process.
- **PoC:**
  - First, I searched in Bjoern's emails for the email that have a favorite pet question:



  - Then I intercepted the request of sending the answer using BurpSuite and send it to the intruder

  - Then I made a word list of about 120 Pets names and upload it to the intruder and start brute forcing and When I filtered the response based on it's status code I found only one request has 200 and having pet name = <Zaya>

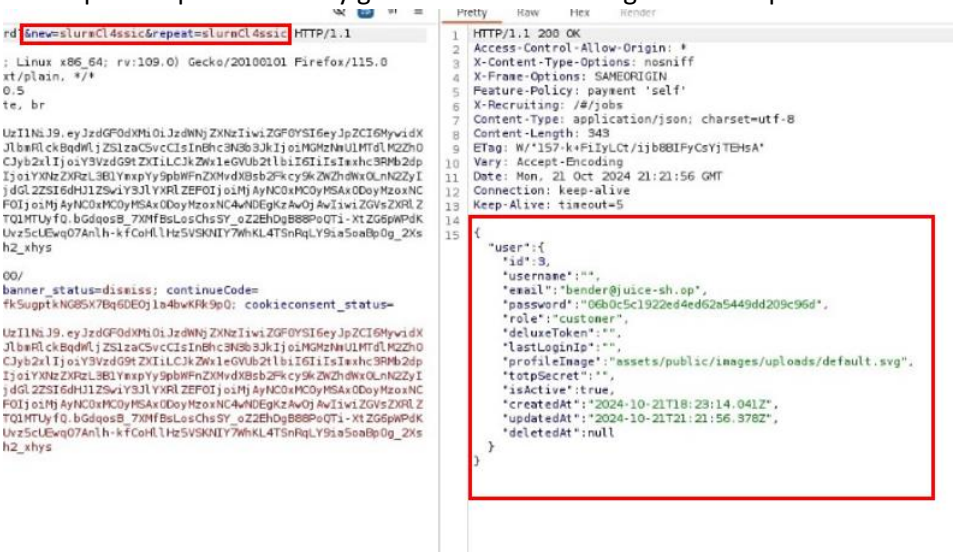


- **Impact**:
  - **Business Impact:** Attackers can potentially reset Bjoern's account by answering or guessing security questions or bypassing protections, leading to unauthorized access to sensitive information or account takeover.
  - Medium **Risk Vulnerabilities.**

2.Change Bender's Password

- **Description:** This vulnerability allows an attacker to change Bender's password, exploiting weak session handling or inadequate verification checks during the password change process.
- **PoC:**
  - First, I entered change password page when I logged in as Bender
  - then I intercepted the request of changing password using BurpSuite and tried to send a request with fake data and it told me that the old password is incorrect
  - So, I tried to remove the old password argument from the request and It seems that he didn't mind

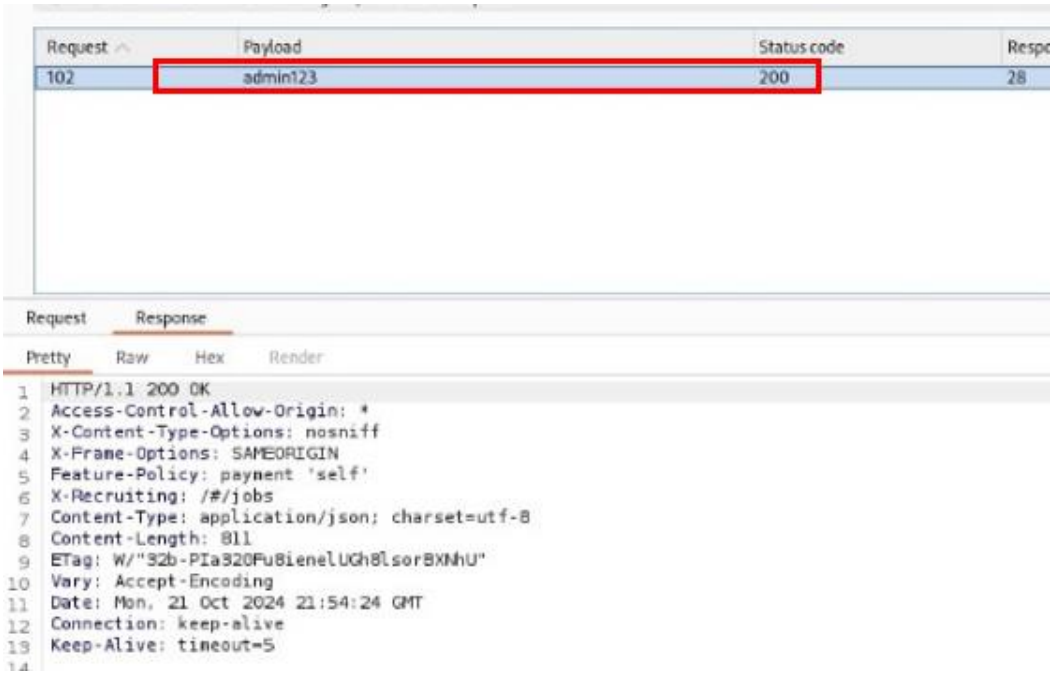
- Then I put the password they gave me in the challenge as a new password



  - And pingo !!!
- **Impact:**
  - **Business Impact:** The attacker could gain unauthorized access to Bender's account by manipulating the password change feature, leading to potential data theft or misuse of account privileges.
  - **High Risk Vulnerabilities.**

3.Password Strength

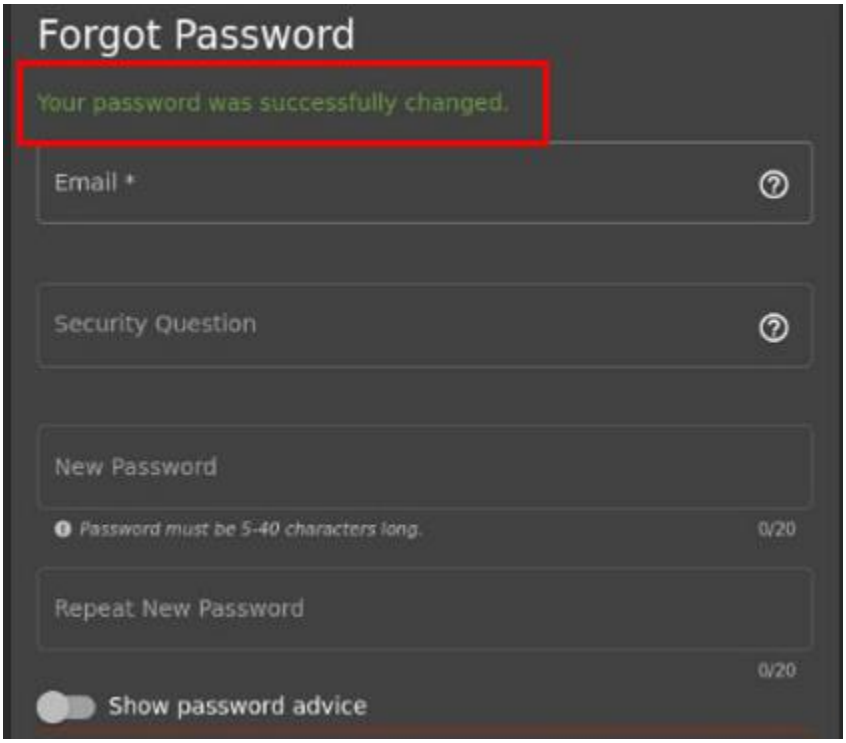
- **Description**: The system allows weak or easily guessable passwords, which can be exploited by attackers to compromise accounts through brute-force or dictionary attacks.
- **PoC:**
  - First a searched for a strong passwords word list and I found this word list on GitHub
  - Then I intercept the request of login using Burp Suite and send it to intruder and used the previous wordlist and started brute forcing filtering the responses based on status code and there was only one response with 200 having password value = admin123



- **Impact:**
  - **Business Impact:** Weak password policies increase the likelihood of unauthorized access to user accounts, leading to data breaches or further exploitation of compromised accounts.
  - **Low Risk Vulnerabilities.**

4.Reset Bjoern's Password

- **Description:** The password reset functionality is vulnerable, allowing an attacker to reset Bjoern's password without proper authorization or verification, exposing the account to potential misuse.
- **PoC:**
  - After Knowing the answer of reset password question of Bjoern from “Bjoern's Favorite Pet” Challenge I entered it and reset his password



- **Impact:**
  - **Business Impact:** Exploiting this vulnerability allows attackers to reset the password of Bjoern's account, leading to unauthorized account access and potentially compromising confidential information.
  - **High Risk Vulnerabilities.**

5.Reset Jim's Password

- **Description:** The password reset mechanism for Jim's account is flawed, allowing an attacker to bypass normal security checks and reset the password without proper authorization.
- **PoC:**
  - First, I entered reset password page and figured out that reset password question for Jim is what is his eldest siblings middle name

  - Then I searched for a word list of common names and I found this list on GitHub Then I intercepted the request of reset password and sent it to intruder And started the attack filtering the responses based on its status code and I found only one response with 200 having an answer value = Samuel

- **Impact:**
  - **Business Impact**: Attackers could gain control of Jim's account by resetting the password, leading to unauthorized access and potential misuse of Jim's privileges and data.
  - Medium **Risk Vulnerabilities.**

Remediation plan for the Broken Authentication

1. **Bjoern's Favorite Pet**: Implement proper session handling and avoid weak authentication schemes.
2. **Change Bender's Password**: Ensure secure password change processes with verification.
3. **Password Strength**: Require complex passwords and enforce strength policies.
4. **Reset Passwords (Bjoern, Jim)**: Implement secure password reset mechanisms with verification.

General Remediation: Enforce strong password policies, secure session management, two-factor authentication, and robust reset processes to prevent authentication flaws.

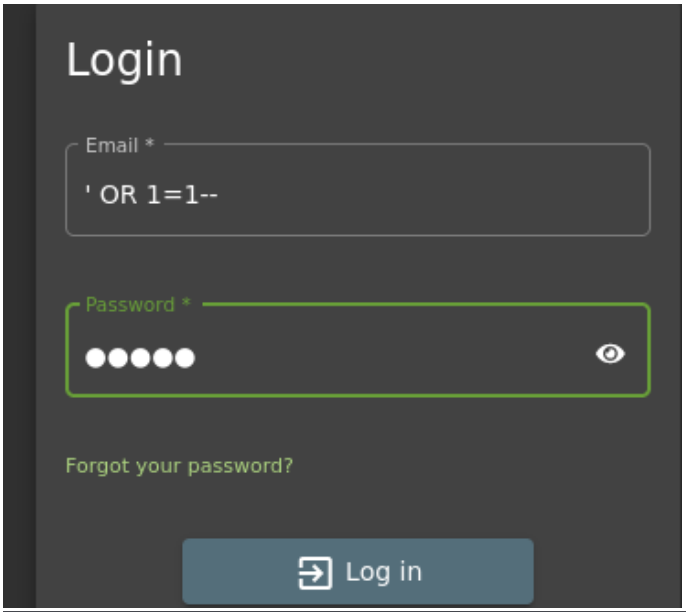
Methodology used to find Broken Authentication

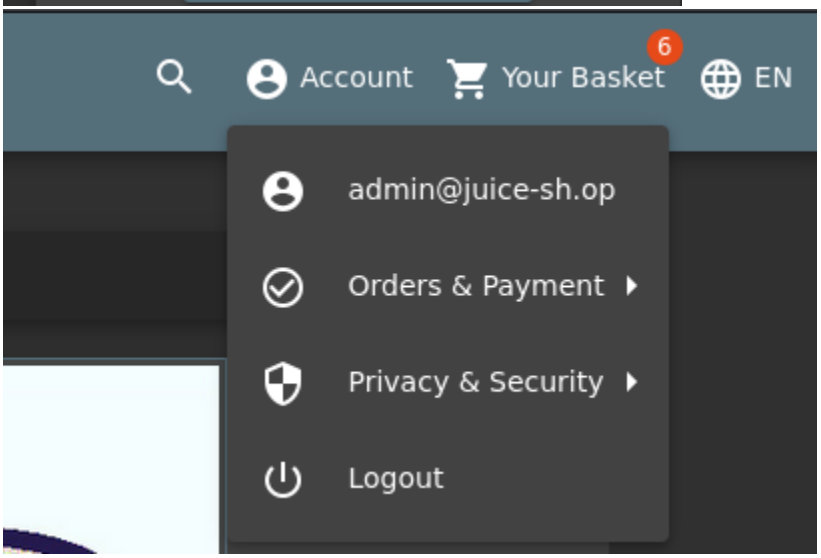
1. **Account Enumeration**: Testing for identifiable responses based on valid or invalid usernames and emails.
2. **Password Strength Testing**: Evaluating the effectiveness of password policies and strength requirements.
3. **Session Management Analysis**: Investigating how session tokens are generated, managed, and invalidated.
4. **Brute Force Testing**: Attempting to gain access through automated brute-force attacks on login forms.

10.Injection

1.Login admin

- **Description:** The Login Admin vulnerability allows an attacker to log in to the administrator's account without valid credentials by exploiting a misconfiguration in the authentication process, bypassing intended access controls.
- **PoC:**
  - Set this payload:

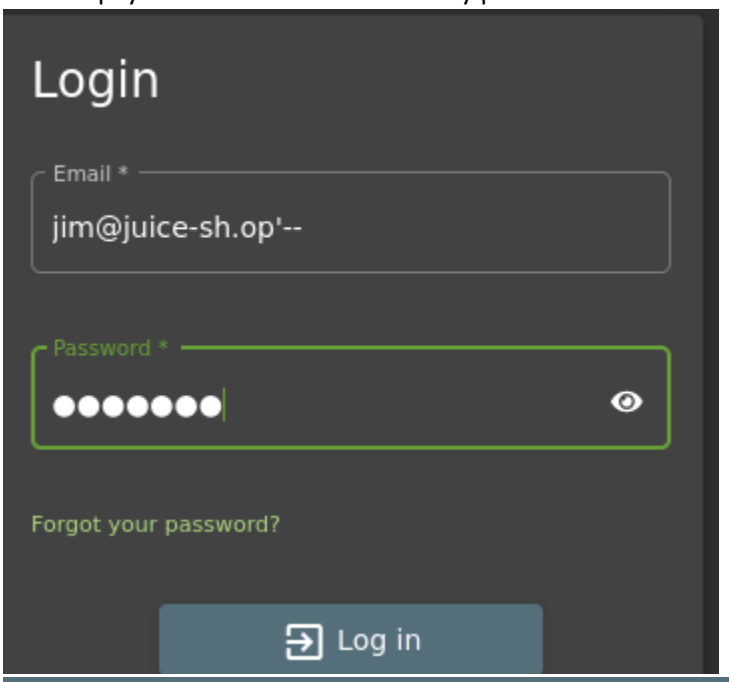


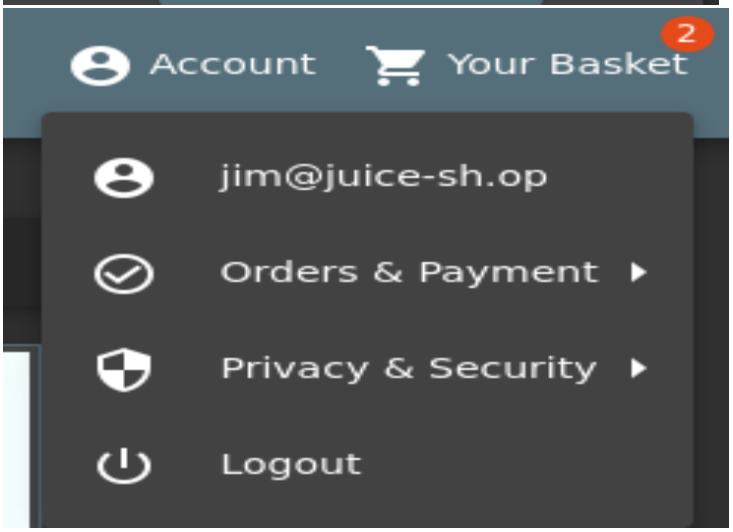


- **Impact:**
  - **Business Impact:** Unauthorized access to the admin panel allows an attacker to take full control of the application, leading to potential data leaks, system misconfiguration, or other malicious actions.
  - Low **Risk Vulnerabilities.**

2.Login Jim

- **Description:** The Login Jim vulnerability exploits a weakness in the login mechanism, allowing an attacker to access Jim's account without proper credentials, bypassing authentication measures.
- **PoC:**
  - Set this payload after the email and any password

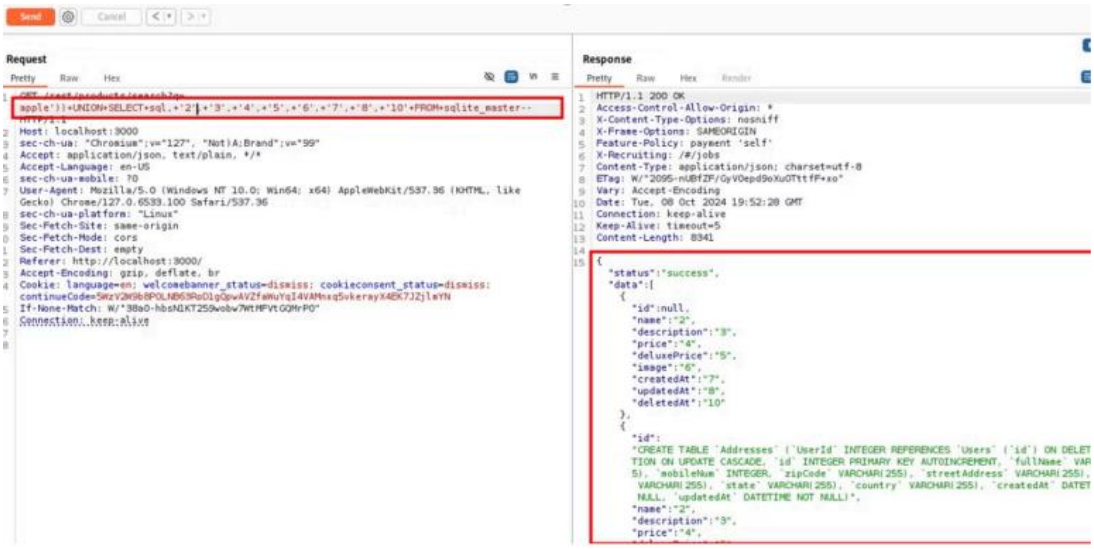




- **Impact:**
  - **Business Impact:** Gaining unauthorized access to Jim's account could allow attackers to view personal data, perform privileged actions, or exploit additional vulnerabilities.
  - Low **Risk Vulnerabilities.**

3.Database Schema

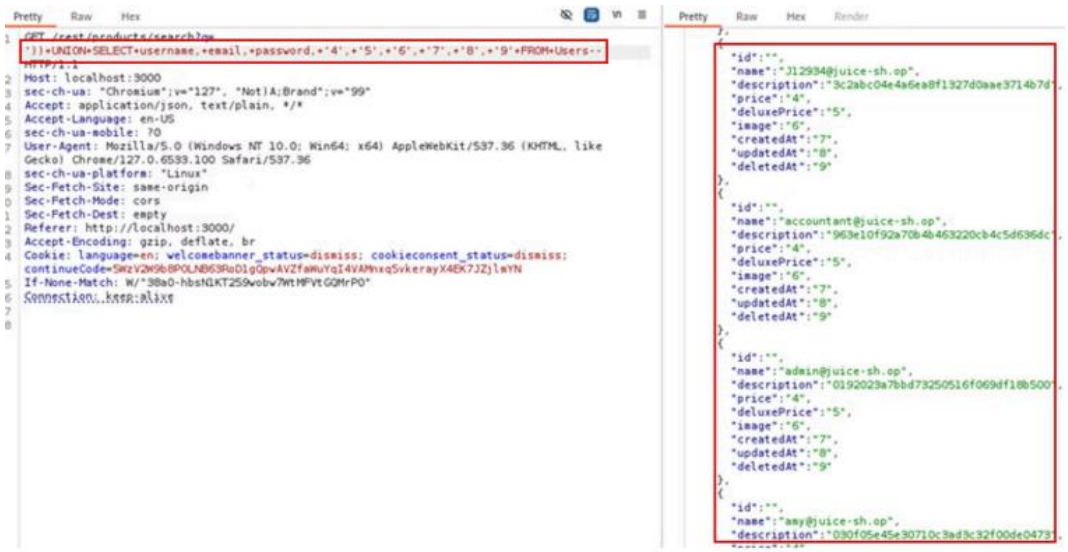
- **Description**: A SQL injection vulnerability in the product search functionality allows attackers to enumerate the database schema.
- **PoC:**
  - Finding Vulnerable Endpoint
  - Use the payload:<< test')) UNION SELECT 1, 2, 3, 4, 5, 6, 7, 8, SQL FROM sqlite_schema-- >> to align with the expected number of columns in the original query and extract schema information from the SQLite_schema table.



- **Impact**:
  - **Business Impact:** Enumerates the database schema.
  - Medium **Risk Vulnerabilities.**

4.User Credentials

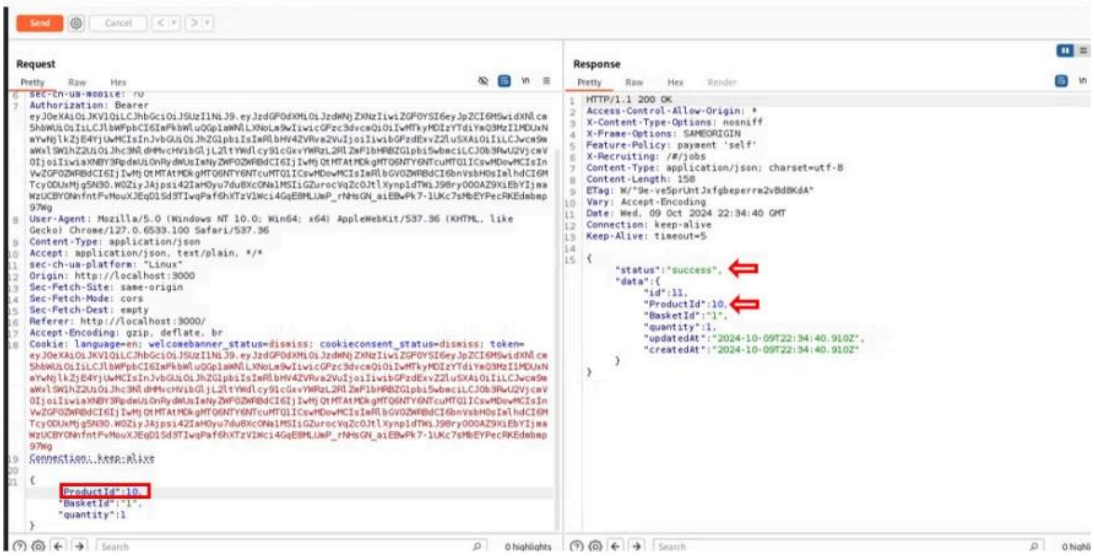
- **Description:** This vulnerability enables attackers to retrieve sensitive user data, including usernames and hashed passwords.
- **PoC**:
  - SQL Injection Vulnerability in Product Search (User Credentials Extraction)
  - the final payload:<< test')) UNION SELECT username, password, role, deletedAt, isActive, createdAt, id, email, profileImage FROM USERS-- >>.
  -



- **Impact:**
  - **Business Impact:** Extracts user credentials.
  - Medium **Risk Vulnerabilities.**

5.Christmas Special

- **Description:** A hidden feature accessible via a specific URL grants unauthorized access to restricted offers.
- **PoC:**
  -



- **Impact:**
  - **Business Impact:** Unauthorized access to restricted offers can lead to financial exploitation or unintended user benefits.
  - Medium **Risk Vulnerabilities.**

Remediation plan for the Injection

- **Christmas Special**: Sanitize user inputs to prevent injection attacks in holiday-themed functionalities.
- **Database Schema**: Securely handle database queries to prevent SQL injection.
- **Login Admin/Jim**: Implement parameterized queries to prevent injection during login attempts.
- **User Credentials**: Securely handle and validate credential submissions to prevent unauthorized access.

General Remediation: Implement input validation, parameterized queries, and proper sanitization techniques across all entry points to mitigate injection vulnerabilities and protect the application from exploitation.

Methodology used to find Injection

1. **Input Field Testing**: Injecting various payloads into input fields to identify how the application processes and sanitizes data.
2. **Error Message Analysis**: Reviewing error responses for indications of SQL or command injection vulnerabilities.
3. **Database Interaction**: Manipulating requests to test how the application interacts with the database.
4. **Automated Scanning**: Utilizing tools designed to detect injection flaws across the application.