SECURITY ASSESSMENT

Juice Shop Vulnerabilities Report

Submitted to: Dojuicer
Security Analysts: Ahmed Salah Elsayed
Moataz Mahmoud Elsayed
Seifeldeen Usama Khaled
Mohamed Mahmoud Hassan

Date of Testing: 2024/10/1
Date of Report Delivery: 2024/10/24

Table of Contents

Contents

Security Engagement Summary

Engagement Overview

- The assessment was requested by the Development Team at Dojuicer to identify security risks in a legacy web application. to help them understand what security risk the web-application is posing to the organization.
- The goal is to detect vulnerabilities, assess their potential impact, and provide insights for remediation to safeguard the organization's data integrity.
- The Information Security department at Dojuicer will conduct this assessment which is done monthly.

Scope

- The assessment is limited to the Juice Shop web application running on the provided virtual machine. The machine itself, including its operating system and network components, is out of scope as advised by the Development Team as they are irrelevant to the web application itself.

Executive Risk Analysis

Risk Level: High

The overall risk is high due to the presence of critical vulnerabilities such as SQL injection in the login functionality, SQL injection in the search bar, and reflected XSS in the search bar. These vulnerabilities allow attackers to access sensitive data, bypass authentication, and execute arbitrary code, which poses a significant threat to both user privacy and the integrity of the application. This could result in severe reputational damage and financial loss for the business.

Executive Recommendation

Remediation efforts are crucial to protect the application and its users. The highest priority should be addressing the SQL injection vulnerabilities by implementing prepared statements and parameterized queries to prevent unauthorized data access and manipulation. XSS vulnerabilities should be mitigated through proper input validation, output encoding, and the implementation of a Content Security Policy (CSP) to restrict executable scripts. Regular code reviews and security testing should be conducted to ensure the continuous identification and remediation of potential risks.

Significant Vulnerability Summary

High Risk Vulnerabilities

- SQL injection in the login functionality
- SQL injection in the search bar
- Reflected XSS in the search bar
- Insecure Object Reference in the viewing user carts
- Cross Site Request Forgery

Medium Risk Vulnerabilities

- Insecure Direct Object Reference (IDOR) in Feedback Submission

Low Risk Vulnerabilities

- None identified at this time

Significant Vulnerability Detail

SQL injection in the login functionality

<<RISK LEVEL HIGH >>

Vulnerability detail

- The risk of the vulnerability is High.
- The vulnerability was identified by adding a single quote to the admin email in the login page, this forced a 500 internal server error and also the query that the web application runs and what is causing the error in the query.
- Here is a screenshot of how the vulnerability was discovered:

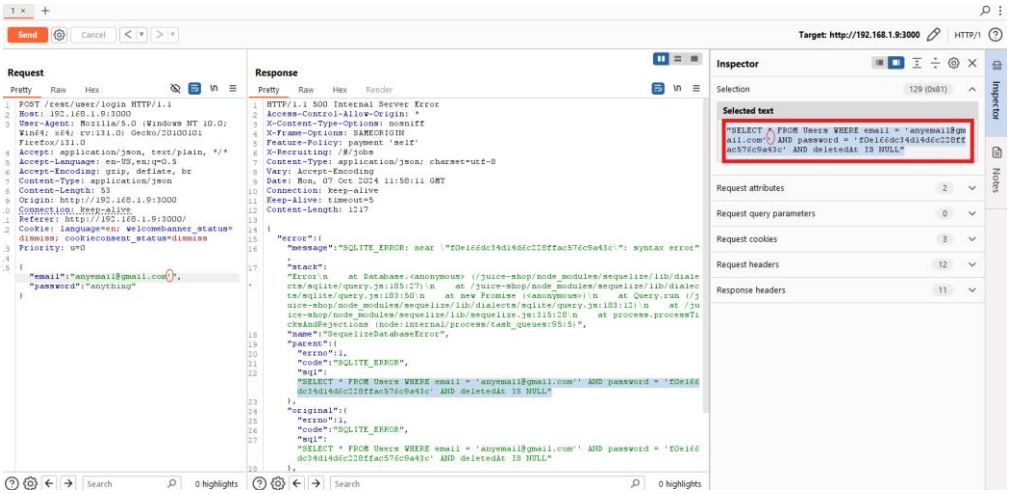


FIGURE (1)

- Here is a screenshot of how the vulnerability was exploited, this figure shows that the payload:

  "admin@juice-sh.op'--" was able to ignore the password check part of the query and therefore no password check is done:

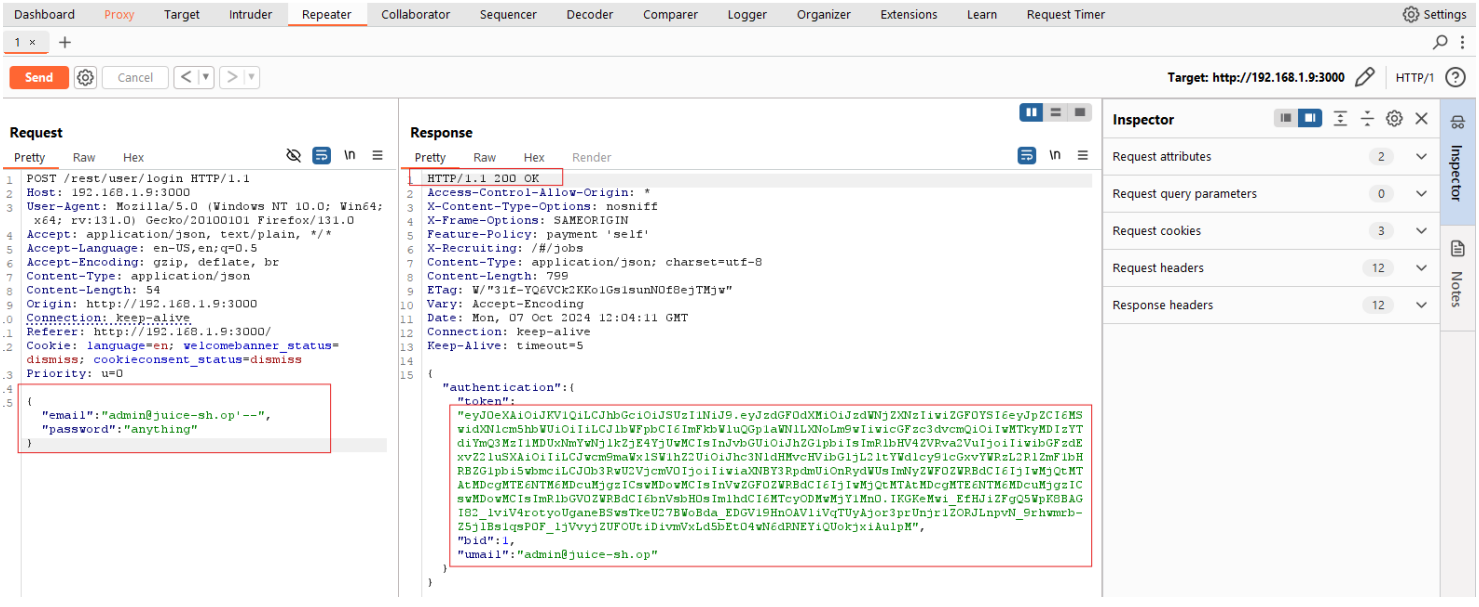


FIGURE (2)

- We were also able to claim sensitive informatio through the token we get from the response shown in figure 2 , we could try different decoding techniques, base64 was a successful try and we got the following result:

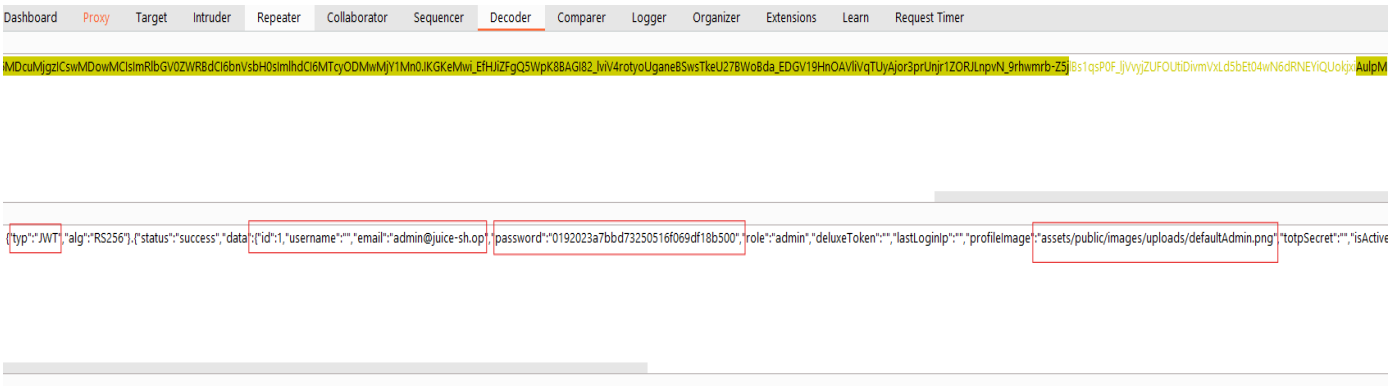


FIGURE (3)

We got the password hashed in md5, we cracked it using the rainbow table on the internet on crackstaion website:

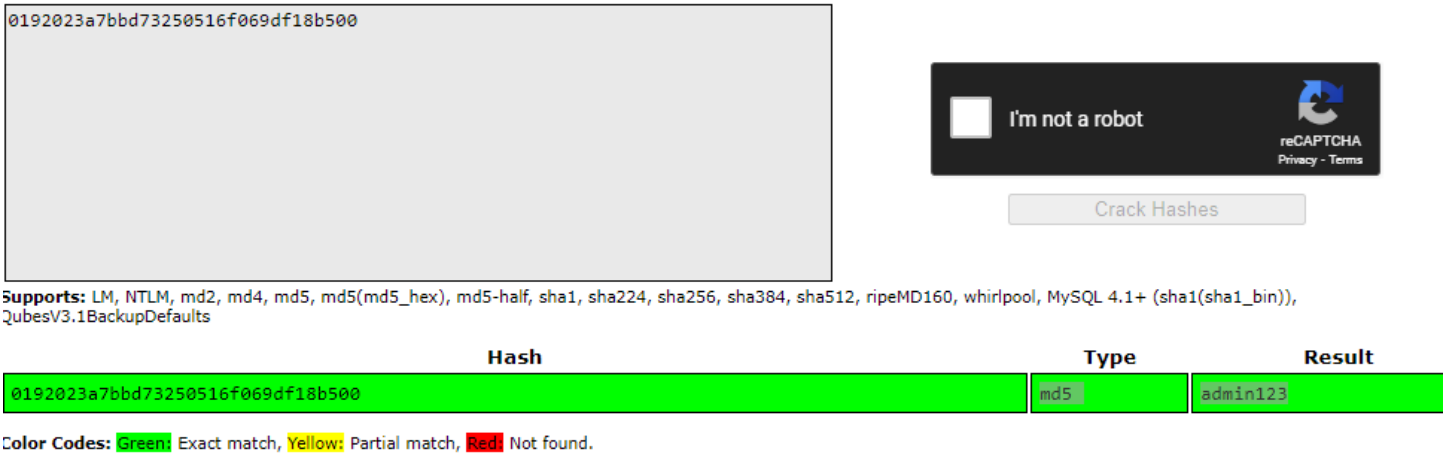


FIGURE (5)

- The attack has a high probability to be done on the web application
- All users that their emails are known (we can see that all users that wrote review before on products their email are viewed which is a bad security) including the admin could be impacted from this attack, which will decrease the reputation of the application and the revenue as a result which could lead to losing the whole business.
- Remediations:
    - Implement prepared statements and parameterized queries to prevent SQL Injection.
    - Employ proper input validation and sanitization for all user inputs.
    - Regularly perform code reviews and security testing to identify similar vulnerabilities.

SQLi in the product search bar

RISK LEVEL HIGH

Vulnerability detail

- The risk of the vulnerability is High .
- The vulnerability was identified using burp we intercepted the search request and manipulating the query parameter ‘q’:

FIGURE (6)

Now we can use a union select statement to retrieve information from the database, but we need to know how many columns does the first select query is calling and to do this we can use this payload: union select 1,2,…etc till we get a success response:

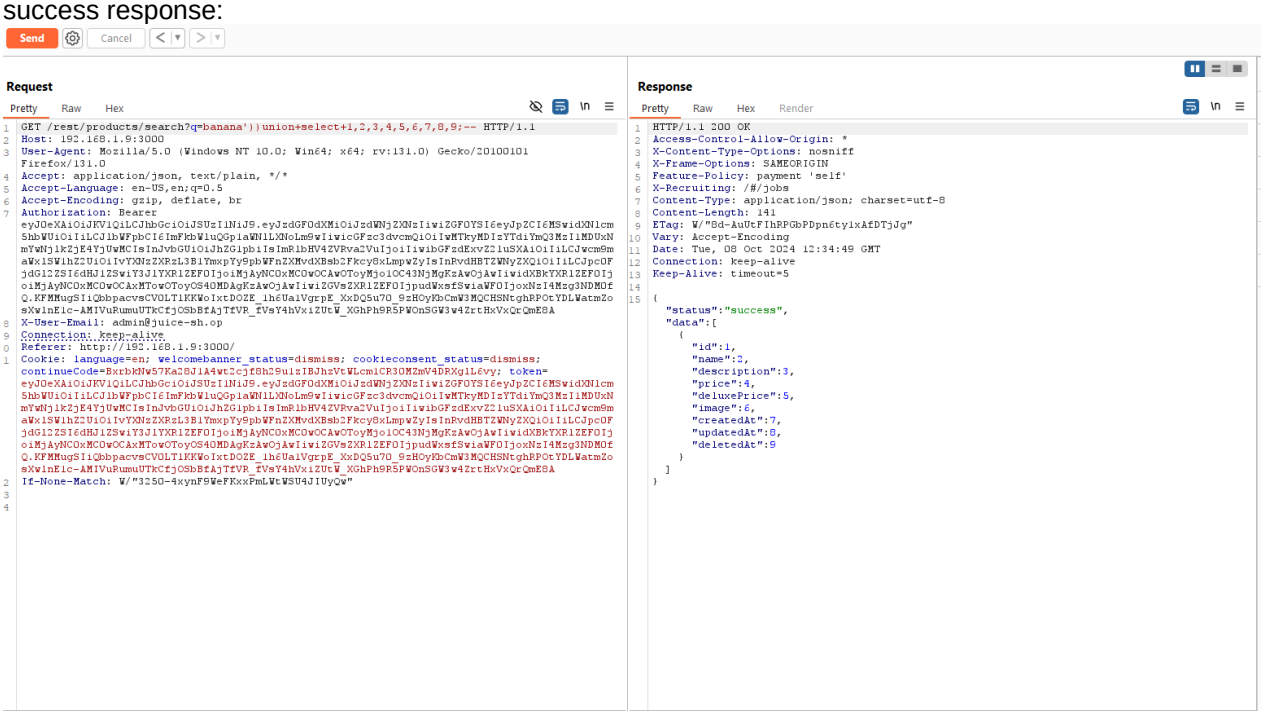


FIGURE (7)

We can see they are 9 columns, now we can inject a query to know the names of the tables in this sqllite database:

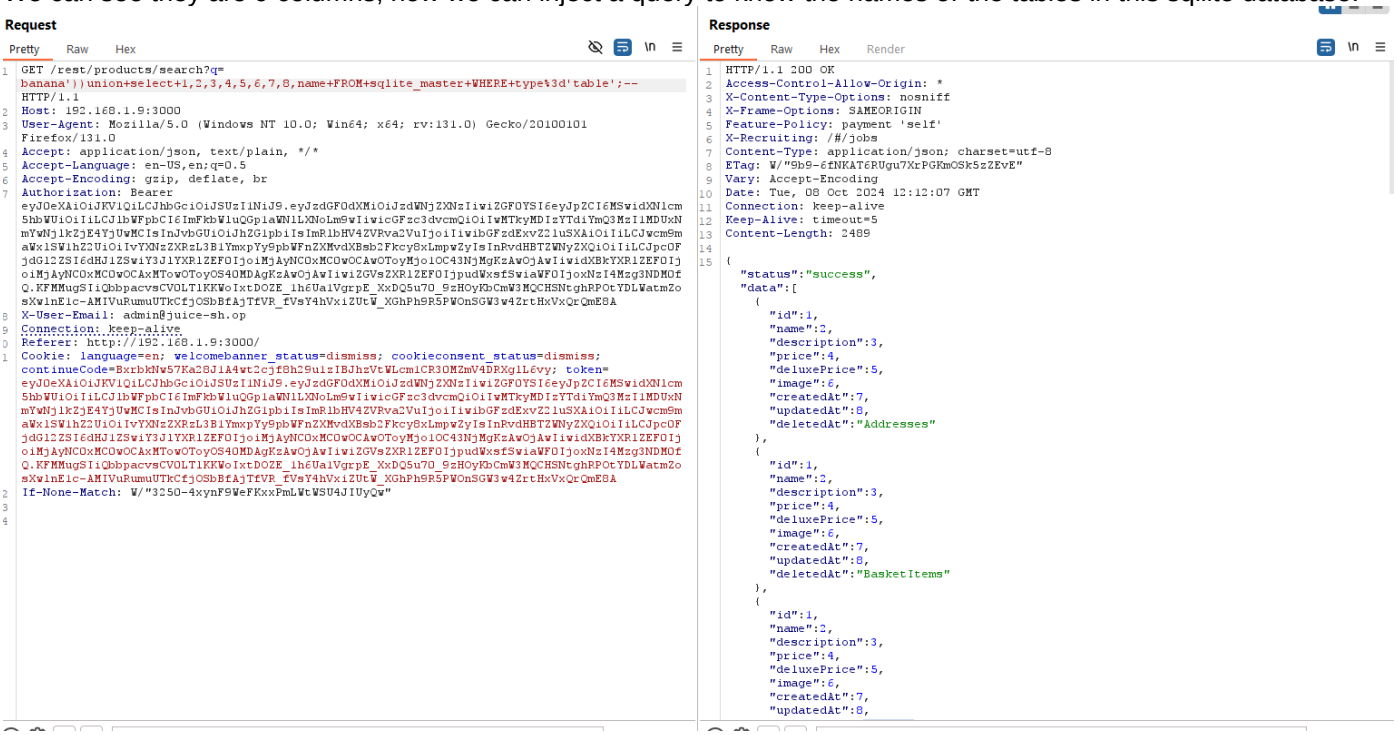


FIGURE (8)

We now got the table names: Addresses, BasketItems, Baskets, Captchas, Cards, Challenges, Complaints, Deliveries, Feedbacks, ImageCaptchas, Memories, PrivacyRequests, Products, Quantities, Recycles, SecurityAnswers, SecurityQuestions, Users, Wallets, sqlite_sequence.
Users table is interesting choice to start with to know its columns using payload:

banana'))UNION+SELECT+1,2,3,4,5,6,7,8,name+FROM+pragma_table_info('users');--

we got the following columns:

createdAt, deletedAt, deluxeToken, email, id, isActive, lastLoginIp, password, profileImage, role, totpSecret, updatedAt, username

Querying data about all users using the payload:
banana'))UNION+SELECT+username,email,password,totpSecret,deluxeToken,6,7,8,9+FROM+Users;--

- User's privacy and information are exposed to the attacker, including wallet table which is too sensitive information, which leads to destroying the business once again.
- Remediations:
    - Implement prepared statements and parameterized queries to prevent SQL Injection.
    - Employ proper input validation and sanitization for all user inputs.
    - Regularly perform code reviews and security testing to identify similar vulnerabilities.

Reflected XSS in the search bar

RISK LEVEL HIGH

Vulnerability detail

-
- Reflected XSS found in the search bar query parameter ‘q’.
- By trying to search for anything in the search bar we can see that what we search for is reflected on the web page with no validation on the input , here is a screenshot to demonstrate:

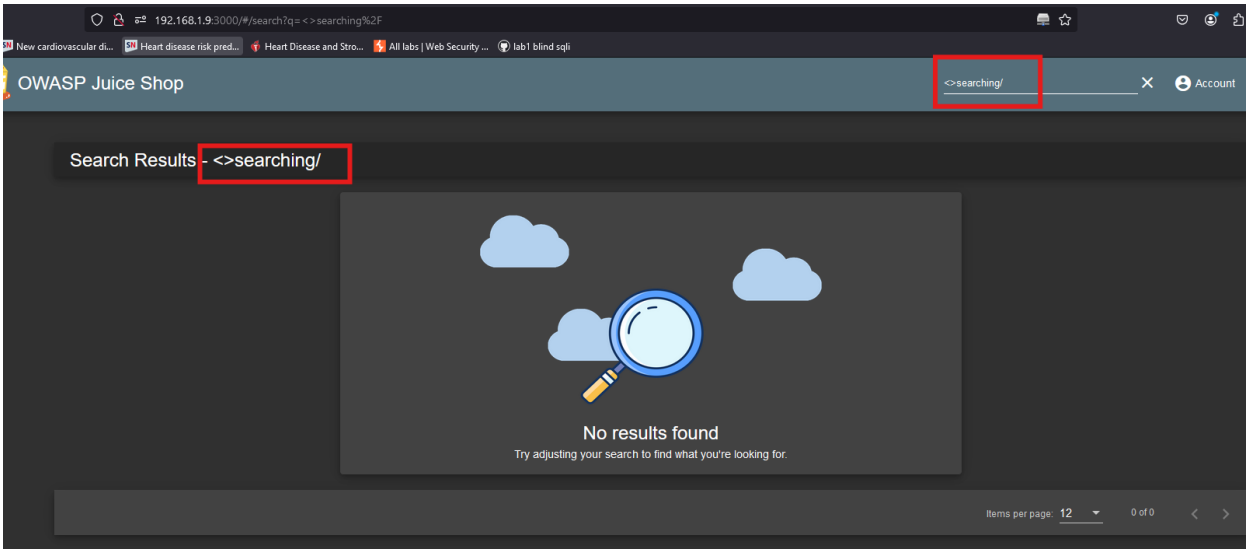
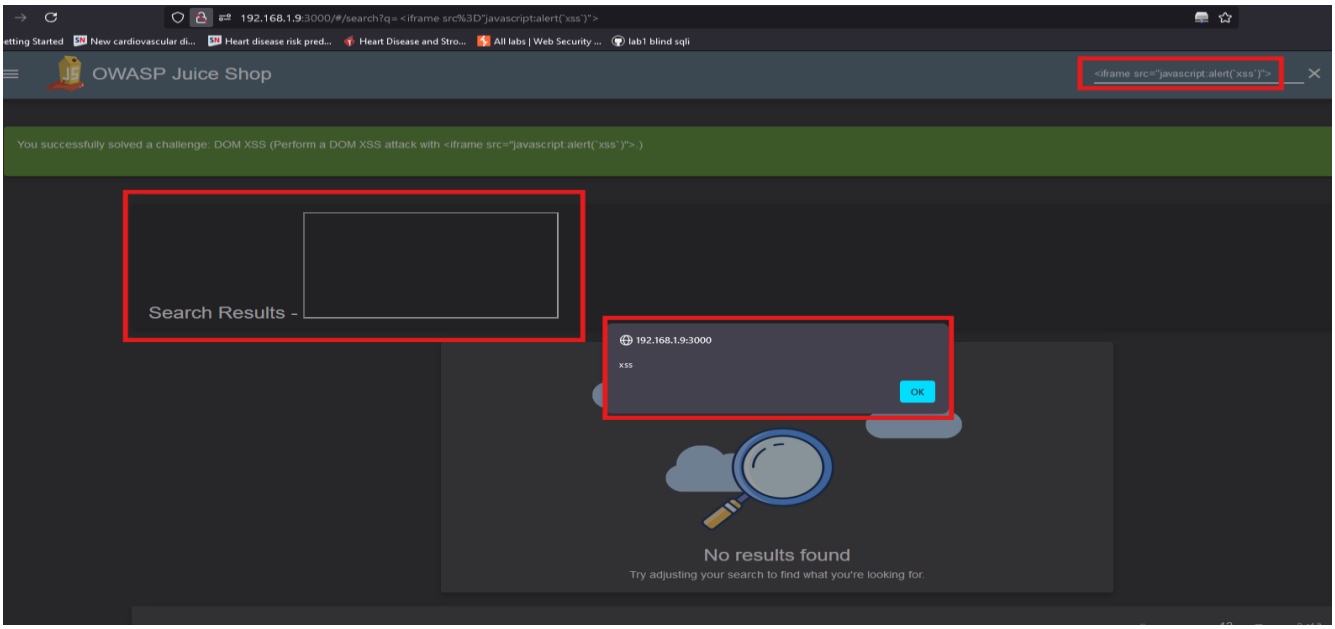


FIGURE (6)

- We tried to inject it with a payload: <script>alert(‘XSS FOUND!’)</script> but this didn’t work out, the browser didn’t execute due to content security policy activated, so we had another approach to run the script.

  We used the payload : <iframe src="javascript:alert(`xss`)" and here is the result:



We can see that the javascript code is executed, we could use also <img> tag with a non-valid source with onerror: <img src”” onerror=’alert(‘xss’)>

- Impact of Reflected XSS on the Web Application:
    - Session Hijacking: An attacker could steal session cookies of users, potentially allowing them to hijack active sessions and impersonate the victim
    - Malicious Redirects: An attacker could craft a malicious URL with embedded XSS payloads and trick users into clicking it. Once the victim clicks the link, they could be redirected to phishing or malicious websites.
- Potential remediation: Input Validation, Sanitization, Output Encoding (Encode any user-supplied data before reflecting it back on the web page), Use a trusted library or framework to sanitize all user inputs, removing any dangerous tags or content

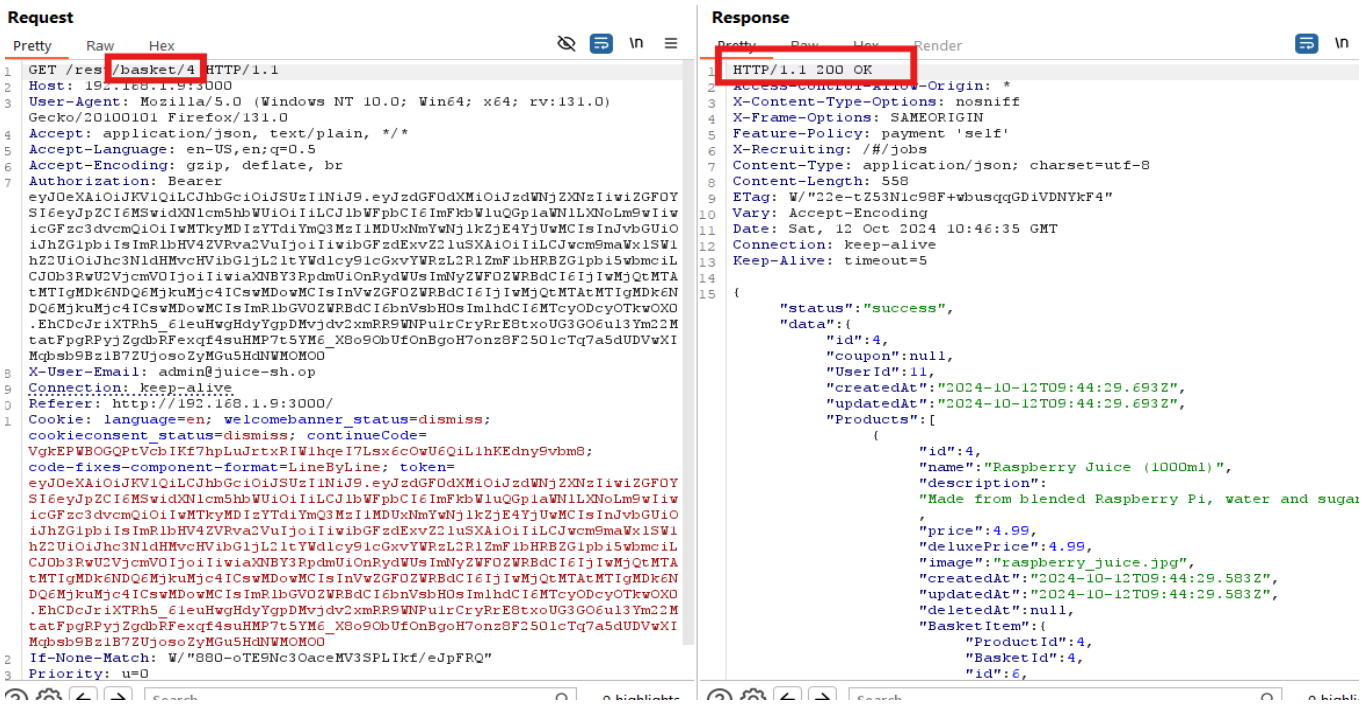
---

Insecure Object Reference in the viewing user carts

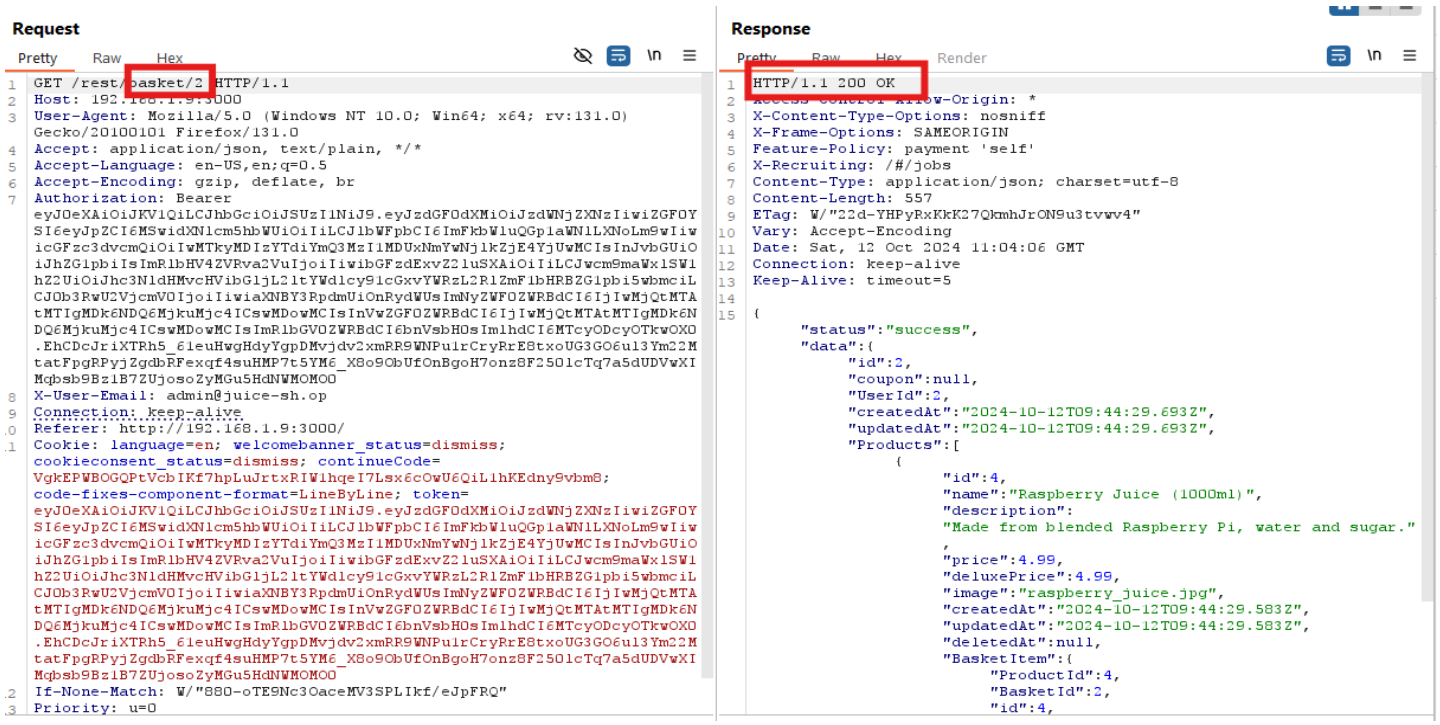
RISK LEVEL HIGH

Vulnerability detail

- The risk of the vulnerability is high.
- When we request to view the cart of the current authenticated user we can see that there is a path parameter that has the current user id:



Now let's change the user id in the path parameter:



- Users privacy is highly impacted which can lead to collecting users cart data to be sold for other purposes:

Gaining access to another user's cart can expose private data, including their selected items, prices, or even personal details if the cart data contains any sensitive information.

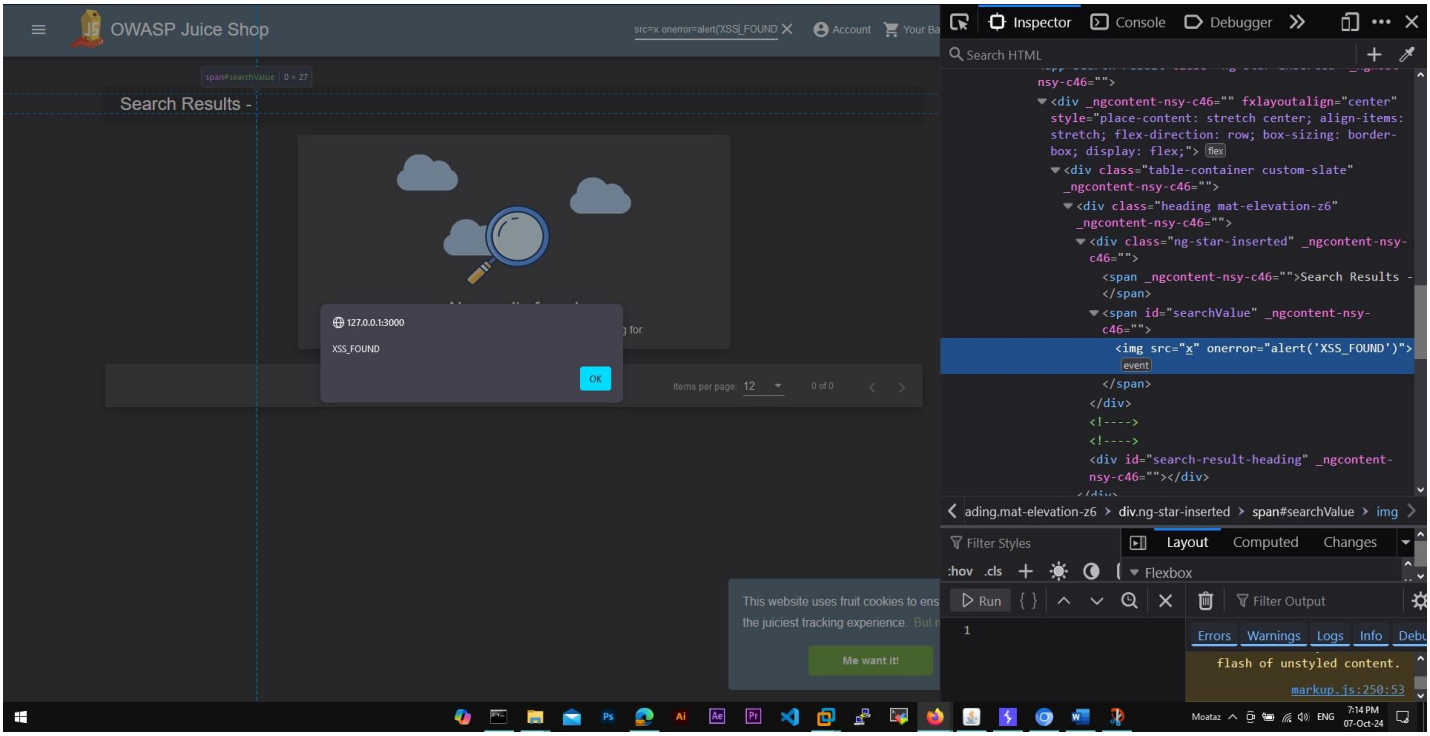
- Remediation:
    - Server-side validation: Ensure that the server checks the current user's identity and validates their permissions before returning or processing any sensitive data.
    - Access control checks: Implement strict access controls to ensure that users can only access their own data, such as their own cart, and not data belonging to others.
    - Use of secure object references: Instead of exposing direct user IDs or other sensitive identifiers in the API, use secure tokens or session-based identifiers to manage and validate requests.

Cross-Site Scripting (XSS) in Juice Shop Search Bar

RISK LEVEL HIGH

Vulnerability Detail:

- **Identification and Validation**: The XSS vulnerability was identified by entering a malicious payload (<img src=x onerror=alert('XSS_FOUND')>) into the search bar, resulting in the execution of the injected script upon submission.
- **Evidence of Validation**:



- **Probability of Exploit**: High, as XSS can be easily exploited by any user interacting with the search bar, leading to arbitrary JavaScript execution.
- **Impact on Users/Business**: If exploited, an attacker could execute arbitrary scripts in the context of users' browsers, potentially stealing cookies, session tokens, or executing actions on behalf of users without their consent.
- **Potential Remediation**: Implement proper input validation and output encoding for all user-supplied content. Use Content Security Policy (CSP) headers to restrict sources of executable scripts and sanitize user input before reflecting it back to the web page.

Broken Access Control (IDOR)

RISK LEVEL MEDIUM

Vulnerability detail

- The risk of the vulnerability is medium.
- When intercepting a submit feedback request we find the following fields is used to add this feedback:

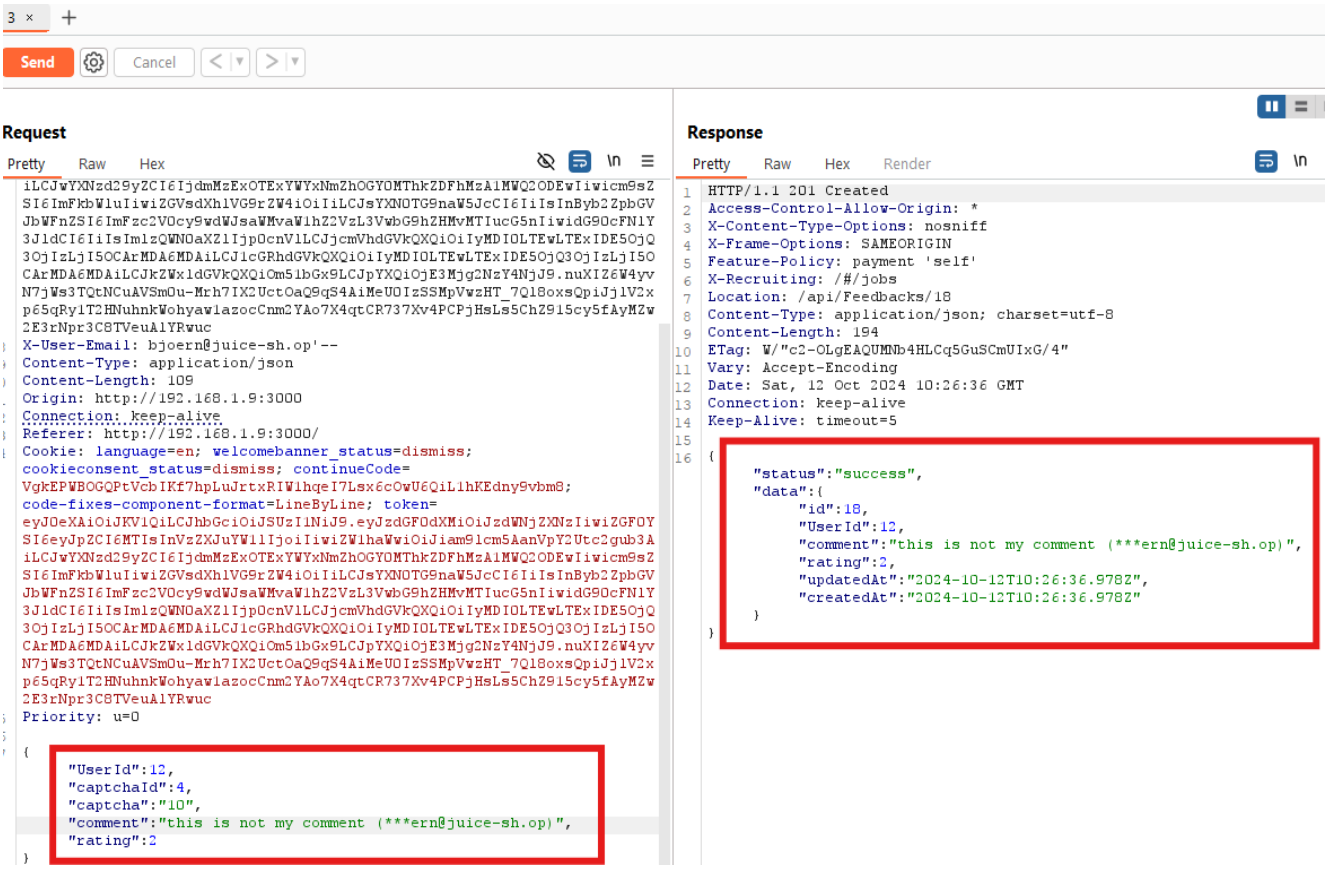


FIGURE (8)

Now we can try changing the user id which is 12 for our current authenticated user:

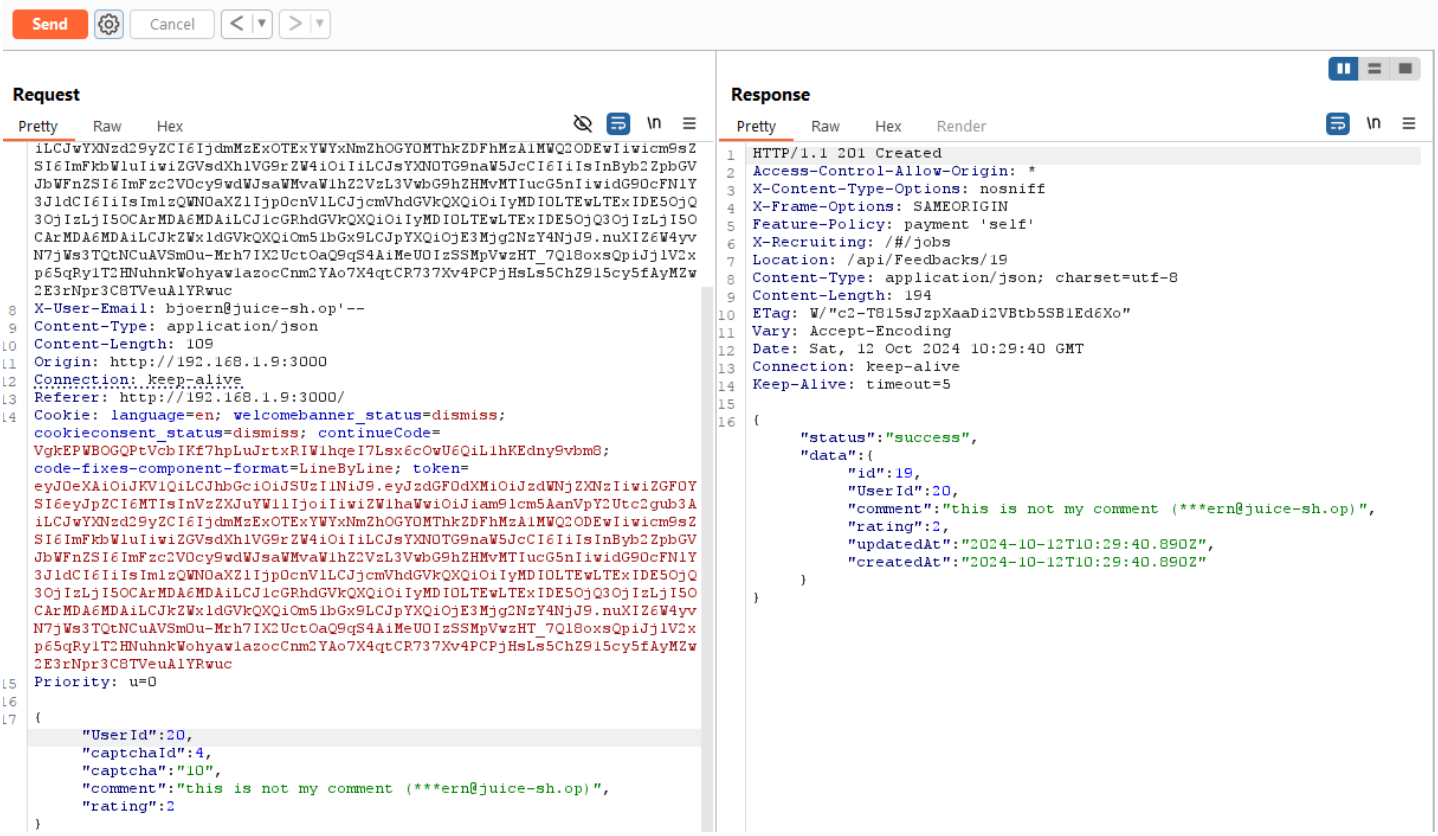


FIGURE (9)

Successfully we used the user id 20 to submit feedback!

- Potential Impact:

  Unauthorized actions: An attacker could impersonate other users to post feedback.

  Loss of trust: This type of vulnerability can cause reputational damage to the web application, especially if users realize their data or actions are being tampered with by others.

- Remediation:
  - Server-side validation: Ensure that the server checks the current user's identity and validates their permissions before returning or processing any sensitive data.
  - Access control checks: Implement strict access controls to ensure that users can only access their own data, such as their own cart, and not data belonging to others.
  - Use of secure object references: Instead of exposing direct user IDs or other sensitive identifiers in the API, use secure tokens or session-based identifiers to manage and validate requests.

Methodology

The assessment involved using a combination of automated tools and manual validation techniques to identify and exploit vulnerabilities in the Juice Shop web application. Tools such as Burp Suite were used to intercept and manipulate HTTP requests, revealing SQL injection and XSS vulnerabilities. Manual validation confirmed the presence and severity of these issues, as evidenced by screenshots and tool outputs included throughout the report.

Assessment Toolset Selection

- **Burp Suite**: Used to intercept and manipulate HTTP requests to identify vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Insecure Direct Object Reference (IDOR).
- **Nmap**: Used for network scanning to ensure no open ports or unnecessary services were exposed as part of the web application.

This concluded the vulnerability assessment methodology portion of this report.