DEPI Graduation Project

OWASP Juice Shop

submitted by

Abdelrhman Mohammed Abdelrhman

Ibrahim Lotfy Ibrahim

Amr Hamada Emam

Abdullah Ayman

Submitted to

Eng. Omar Tarek Zayed

OCTOBER 24, 2024

SPRINTS

Table of Contents

Executive Summary

Sec4Fun is a trusted cybersecurity partner, dedicated to safeguarding organizations from the ever-evolving threat landscape. As experts in penetration testing, we provide a proactive approach to security by simulatingreal-world cyberattacks to uncover vulnerabilities before they can be exploited.

Our penetration testing services are designed to thoroughly assess your organization's security posture across various domains, including network infrastructure, web applications, mobile applications, and cloud environments. By leveraging industry-recognized methodologies such as OWASP, NIST, and OSSTMM, we ensure a meticulous evaluation of your systems, identifying weaknesses that could potentially lead to unauthorizedaccess, data breaches, or other security incidents.

Our team of highly skilled and certified penetration testers brings a wealth ofexperience and knowledge to each engagement. We understand that every organization is unique, which is why our approach is tailored to meet your specific security needs and compliance requirements. Whether you are seeking to validate your security controls, comply with regulatory standards,or gain peace of mind, our services are designed to provide you with actionable insights and recommendations.

At Sec4Fun, we go beyond merely identifying vulnerabilities. We work closelywith your team to prioritize and remediate the risks, offering strategic guidance to strengthen your overall security posture. Our commitment is to empower your organization with the knowledge and tools necessary to defend against sophisticated cyber threats.

With Sec4Fun as your security partner, you can be confident that your organization's digital assets are well-protected, and your operations aresecure against potential cyberattacks

Scope of Work

The scope of this penetration test is restricted to the **web applications** hosted under the domain **juice-shop.com** and its associated subdomains. This includes:

1. **Primary Web Application**: The main web application accessible at https://juice-shop.com.
2. **Subdomain Web Applications**: Any web applications hosted on subdomains of **juice-shop.com**
3. **APIs**: Any publicly accessible APIs associated with **juice-shop.com** and its subdomains.
4. **Web Services**: Web services or microservices that are part of or support the webapplications on the example.com domain.

Out of Scope

- Non-web applications, including backend systems and network infrastructure.
- Any domains or subdomains not hosting a web application.
- Internal or private web applications not publicly accessible.

Findings Classifications

Each vulnerability or risk identified has been labeled as a Finding and categorized as aCritical Risk, High Risk, Medium Risk, Low Risk, or Informational, which are defined as:

1- Critical Risk Issues [C]

These vulnerabilities should be addressed as soon as possible as they may pose an immediate danger to the security of the networks, systems, or data. Exploitation doesnot require advanced tools or techniques or special knowledge of the target.

2- High Risk Issues [H]

These vulnerabilities should be addressed promptly as they may pose a significant danger to the security of the networks, systems, or data. The issue is commonly more difficult to exploit but could allow for elevated permissions, loss of data, or system downtime.

3- Medium Risk Issues [M]

These vulnerabilities should be addressed in a timely manner. Exploitation is often difficult and requires social engineering, existing access, or exceptional circumstances.

4- Low Risk Issues [L]

The vulnerabilities should be noted and addressed at a later date. These issues offerlittle opportunity or information to an attacker and may not pose an actual threat.

5- Informational Issues [I]

These issues are for informational purposes only and likely do not represent an actual threat.

Our Findings:

Critical Risk Issues [C]:-

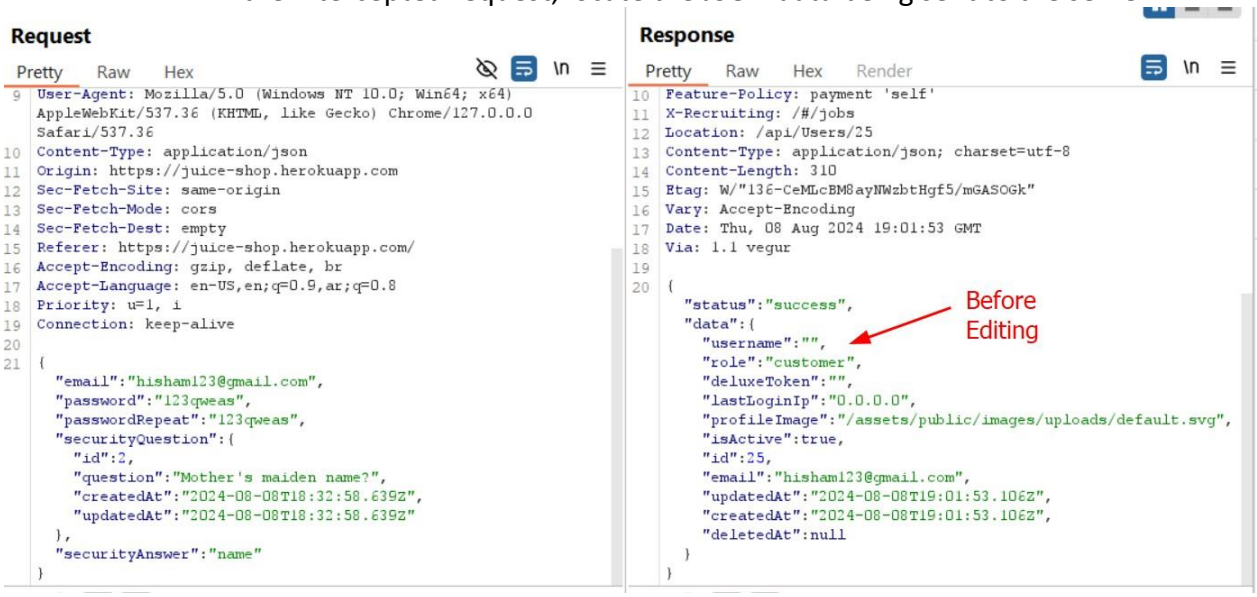
(C001) Improper Input Validation Allowing User Registration with Administrator Privileges

Weakness Type: Broken Access ControlDescription

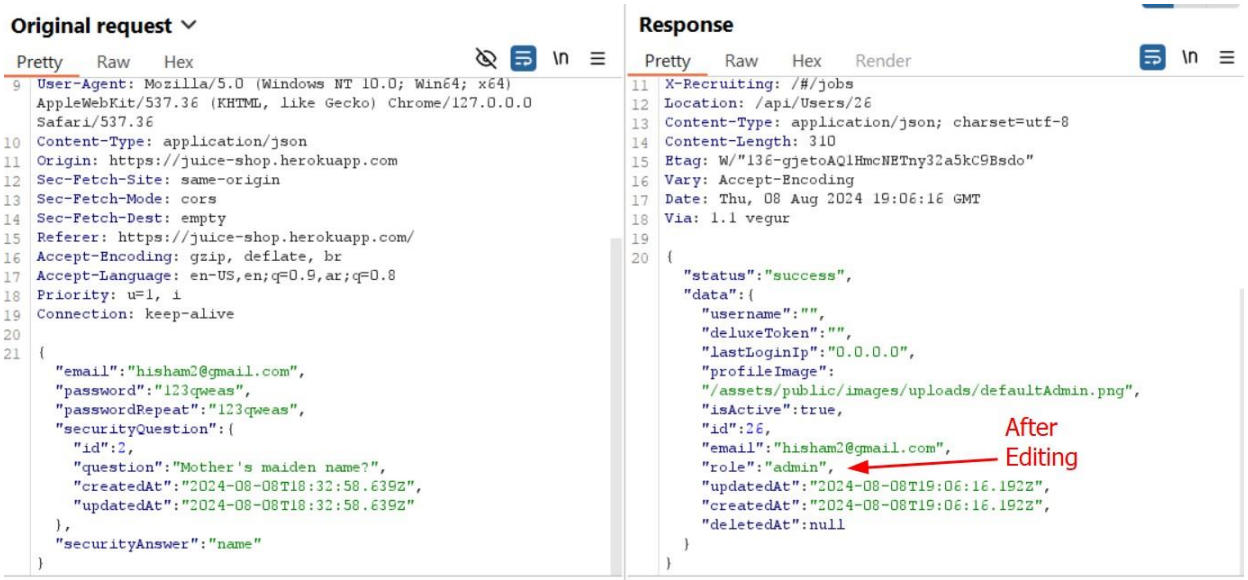
An improper input validation vulnerability exists on the user registration endpoint of the application. By manipulating the registration request, an attacker can elevate their privileges tothat of an administrator. This allows unauthorized users to access and perform actions restricted to administrators, leading to potential full compromise of the application.

Steps to Reproduce:

1. **Navigate to Registration Page:**
    - Go to https://juice-shop.com/#/register.
2. **Intercept the Registration Request:**
    - Open a proxy tool like Burp Suite.
    - Fill in the registration form with the required details (e.g., username, password,etc.).
    - Before submitting the form, intercept the outgoing request using your proxytool.
3. **Modify the Request:**
    - In the intercepted request, locate the JSON data being sent to the server.



Add a new key-value pair: **"role":"admin"**.



4. **Forward the Modified Request:**
    - Send the modified request to the server.
    - The registration proceeded and the new account will have administratorprivileges.
5. **Verify Admin Privileges:**
    - Log in with the newly created account.
    - Access the admin panel or perform actions restricted to administrators toconfirm the elevated privileges.

Impact

Exploiting this vulnerability allows an attacker to register as a user with administrative privileges. This would enable the attacker to perform privileged actions such as managing useraccounts, accessing sensitive data, and altering application settings, leading to a complete takeover of the application.

Recommended Fix

- Implement strict server-side validation to ensure that user input cannot bemanipulated to grant unauthorized roles.
- Specifically, the role attribute should be controlled exclusively by the server andshould not be modifiable by the client.
- Ensure that only authorized processes (e.g., administrators assigning roles) canalter the role of a user.
- Conduct regular security testing to identify and remediate similar issues

(C002) SQL Injection in Login Page Leading toAdministrator Access

Weakness Type: SQL InjectionDescription

An SQL Injection vulnerability was identified on the login page of the application, allowing unauthorized users to bypass authentication and gain access as an administrator. The issue arises due to improper handling of user-supplied input within SQL queries.

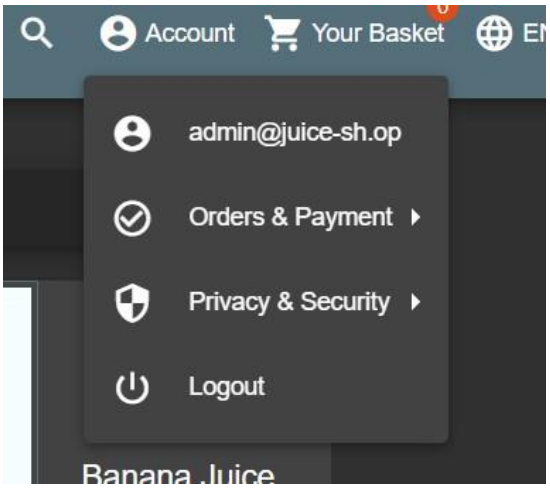
Steps to Reproduce:

1- Navigate to the application's login page https://juice-shop.herokuapp.com/#/login2- In the username field, enter the following payload:

```
attacker ' or 1=1 --
```

3- Leave the password field empty or enter any random value and Click on the "Login" button.

4- The system successfully logs in as the user admin without requiring a validpassword.



Impact

Exploiting this vulnerability, an attacker can gain unauthorized access to the application with administrator rights. This could lead to complete control over the application, including viewing, modifying, or deleting sensitive data, compromising the integrity, confidentiality, andavailability of the entire system.

Recommended Fix

1) **Use Parameterized Queries:** Implement parameterized queries to avoid directinclusion of user inputs in SQL statements.
2) **Input Validation:** Validate all user inputs to ensure they meet expected formats.
3) **Limit Error Messages:** Use generic error messages to prevent revealing detailsabout SQL queries.
4) **Regular Security Testing:** Regularly conduct security assessments to identify andfix SQL injection vulnerabilities.

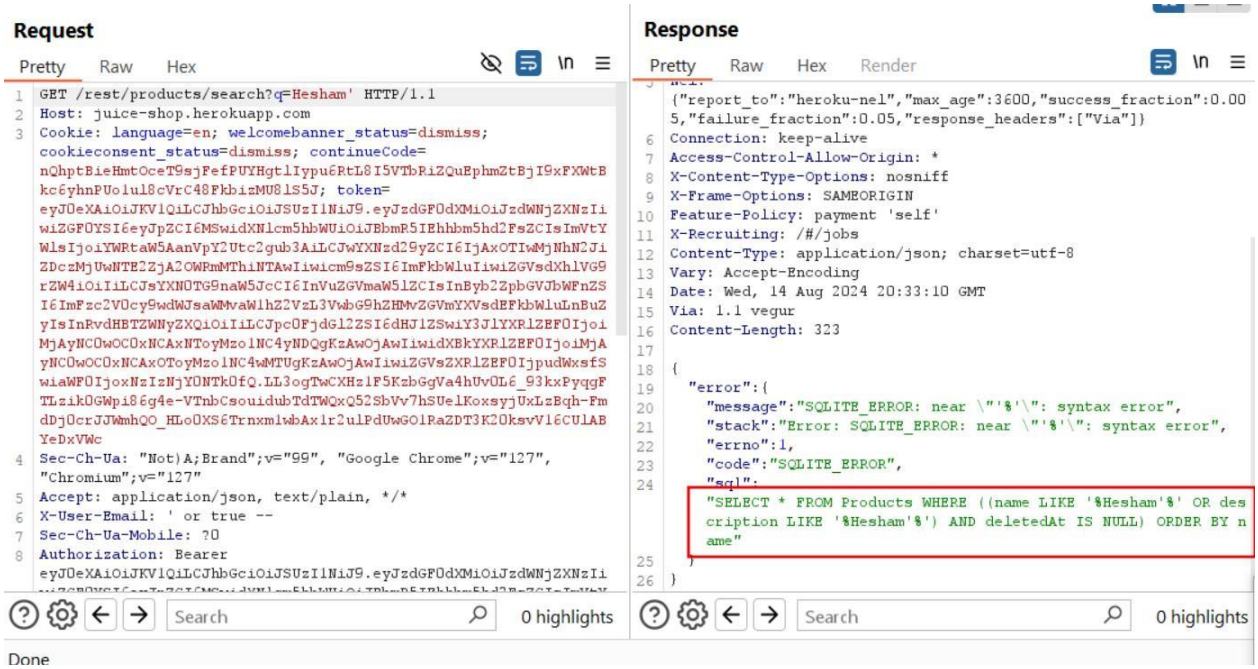
(C003) SQL Injection in Search Bar Leads to Retrieval of All User Credentials

Weakness Type: SQL InjectionDescription

An SQL injection vulnerability was discovered in the search bar functionality of the target application. By manipulating the input data sent to the server, it is possible to execute arbitrarySQL queries. This allows an attacker to retrieve sensitive information from the database, including the credentials of all users.

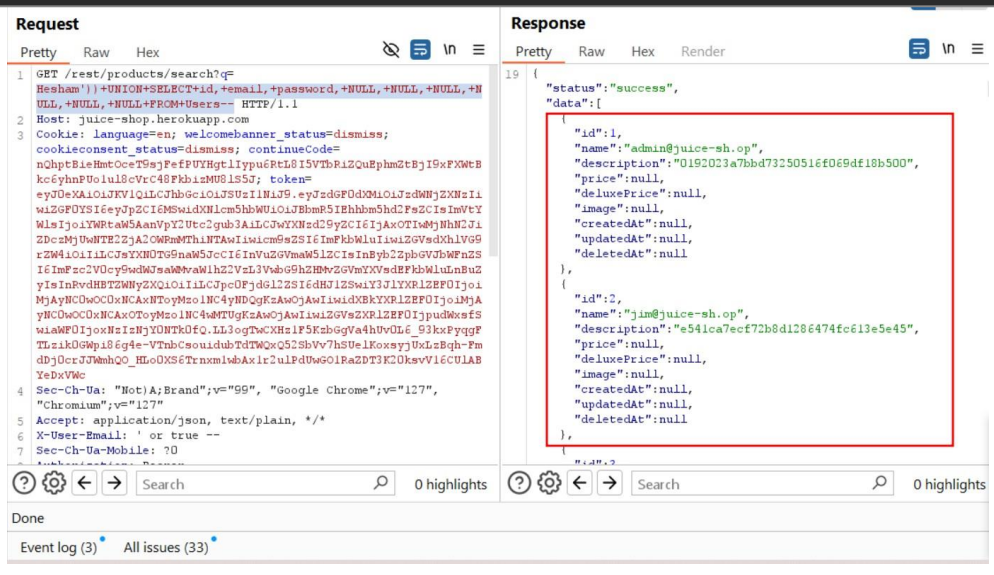
Steps to Reproduce:

1. **Navigate to https://juice-shop.com/#/search**
2. **Inject ' to test for SQLI . SQL error has been returned with the Query.**



3. **Inject this SQL query**

4. Hesham')) UNION SELECT id, email, password, NULL, NULL, NULL, NULL, NULL, NULL FROM Users--



Impact

An attacker can leverage this vulnerability to gain unauthorized access to the database, retrieveall user credentials, and potentially use this information for further attacks, including account takeover. The exposure of sensitive data, such as usernames, emails, and passwords, poses a significant security risk to both the affected users and the organization.

Recommended Fix

1) **Use Parameterized Queries:** Implement parameterized queries to avoid directinclusion of user inputs in SQL statements.
2) **Input Validation:** Validate all user inputs to ensure they meet expected formats.
3) **Limit Error Messages:** Use generic error messages to prevent revealing detailsabout SQL queries.
4) **Regular Security Testing:** Regularly conduct security assessments to identify andfix SQL injection vulnerabilities.

High Risk Issues [H]:-

(H001) Insecure Direct Object Reference (IDOR) inUserId Parameter (Feedback Function)

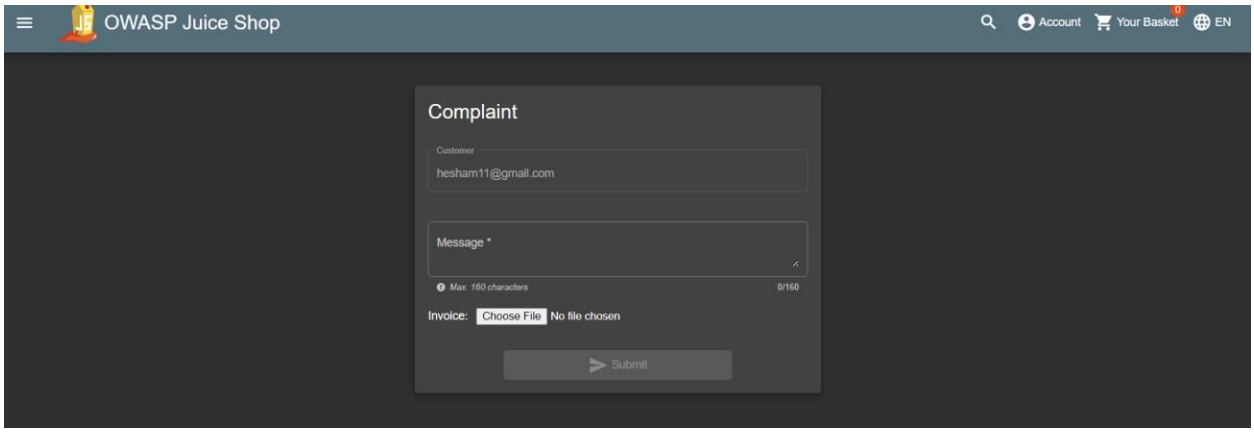
Weakness Type: Broken Access Control

Description

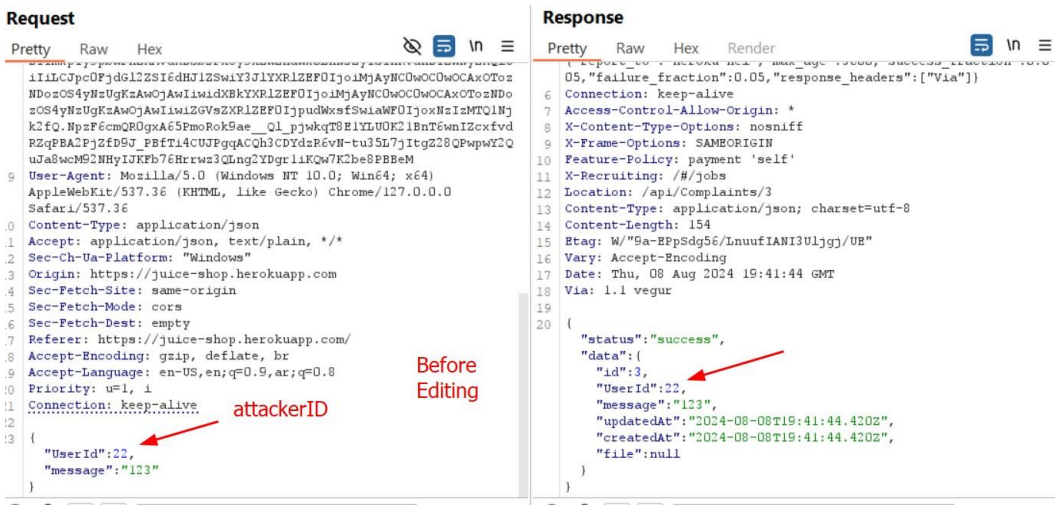
An Insecure Direct Object Reference (IDOR) vulnerability was discovered in the **UserId** parameter of the request sent to the **/complain** endpoint. This vulnerability allows an unauthenticated attacker to modify the **UserId** parameter to another user's ID, enabling theattacker to access or manipulate the data of other users without proper authorization.

Steps to Reproduce:

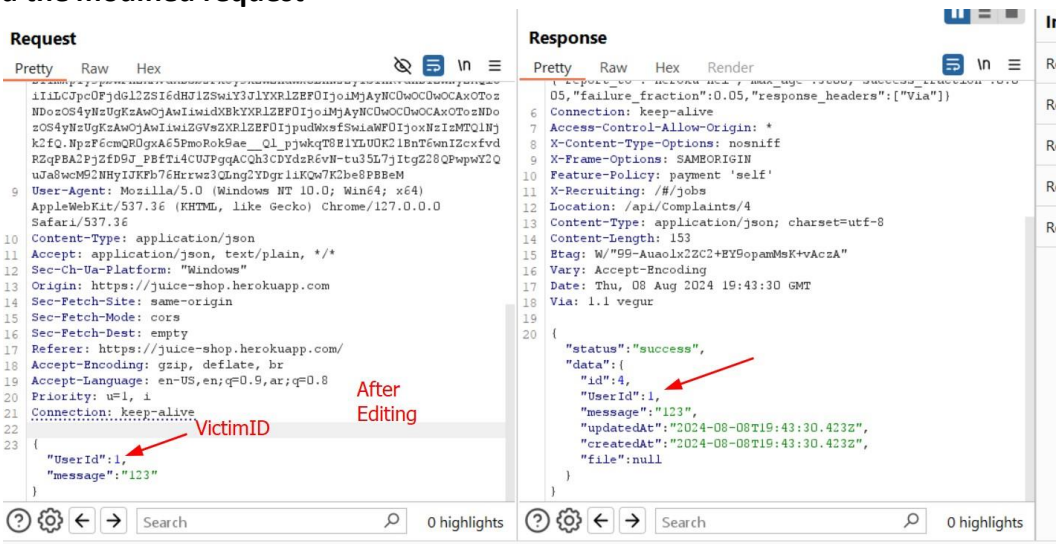
1. **Navigate to https://juice-shop.com/#/complain.**



2. **Intercept the request using a web proxy tool (e.g., Burp Suite).**
3. **Fill out the message field and send the request.**
4. **In the proxy tab, observe the UserId parameter sent in the request.**
5. **Move the intercepted request to the repeater tool.**



6. **Edit the UserId value to a different user's ID (e.g., replace with VictimId).**
7. **Send the modified request**



Impact

The impact of this vulnerability is significant, as it can lead to unauthorized access to sensitive user data. In this specific instance, an attacker could access another user's complaints by modifying the UserId in the request. This could expose personal information, such as complaintdetails, to unauthorized parties.

Recommended Fix

1. **Implement Authorization Checks:**
   - Ensure that access control logic is applied server-side to validate whether the user is authorized to access or modify the resource identified by the **UserId** parameter. This canoften be done by checking if the authenticated user is the owner of the resource.
2. **Restrict Parameter Modification:**
   - Use server-side validation to prevent unauthorized changes to sensitive parameterssuch as **UserId**. Ensure that only authorized users can modify or view the data associated with their own ID.

(H002) Insecure Direct Object Reference (IDOR)Leads to Sensitive Data Exposure

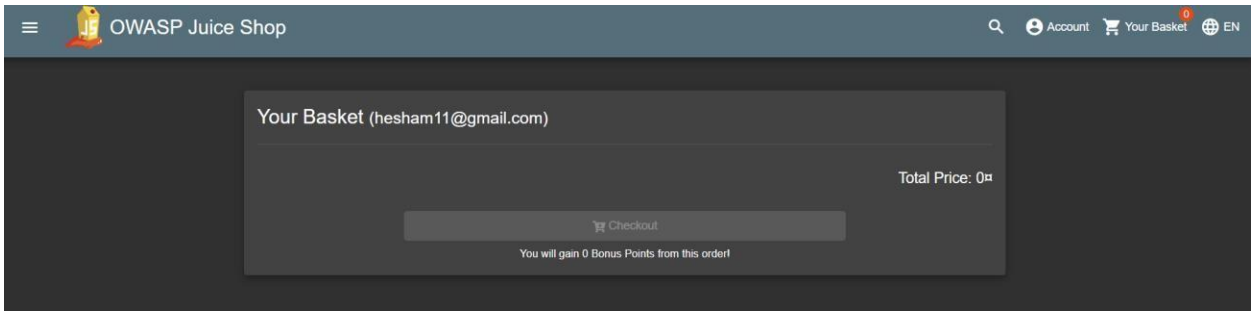
Weakness Type:

IDOR Description

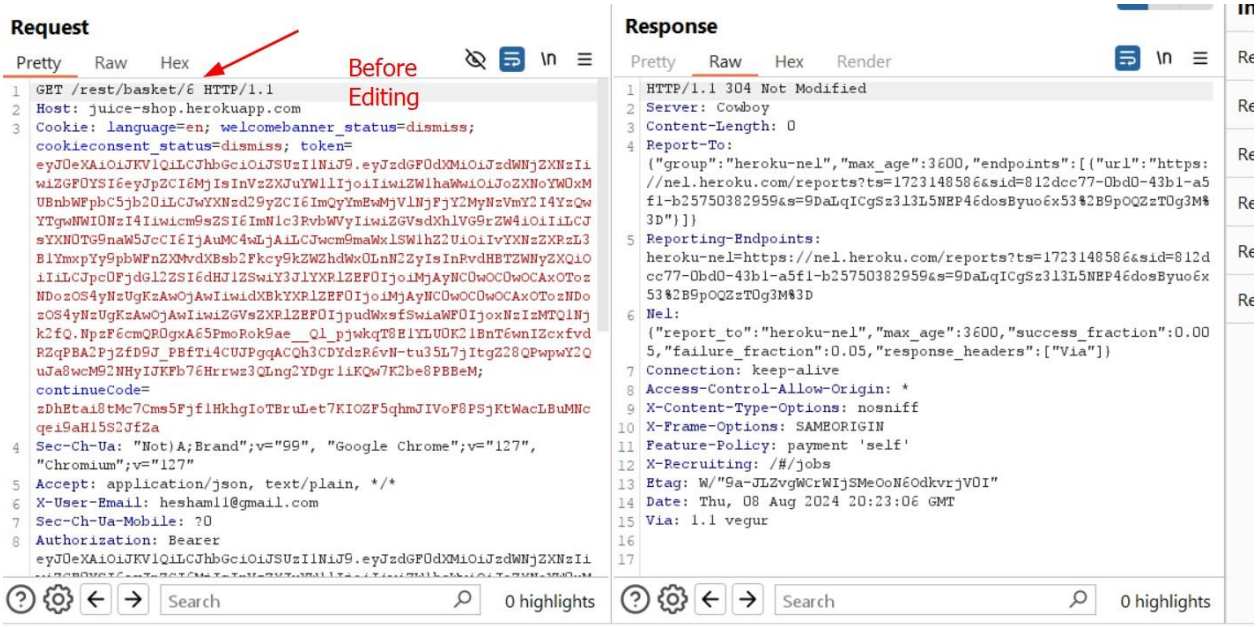
An Insecure Direct Object Reference (IDOR) vulnerability was identified in the Juice Shop application, specifically in the basket functionality. This issue allows an attacker to access sensitive data of other users by manipulating the request parameter. The vulnerability exists due to improper authorization checks on the resource being accessed, leading to unauthorizeddata exposure.

Steps to Reproduce:

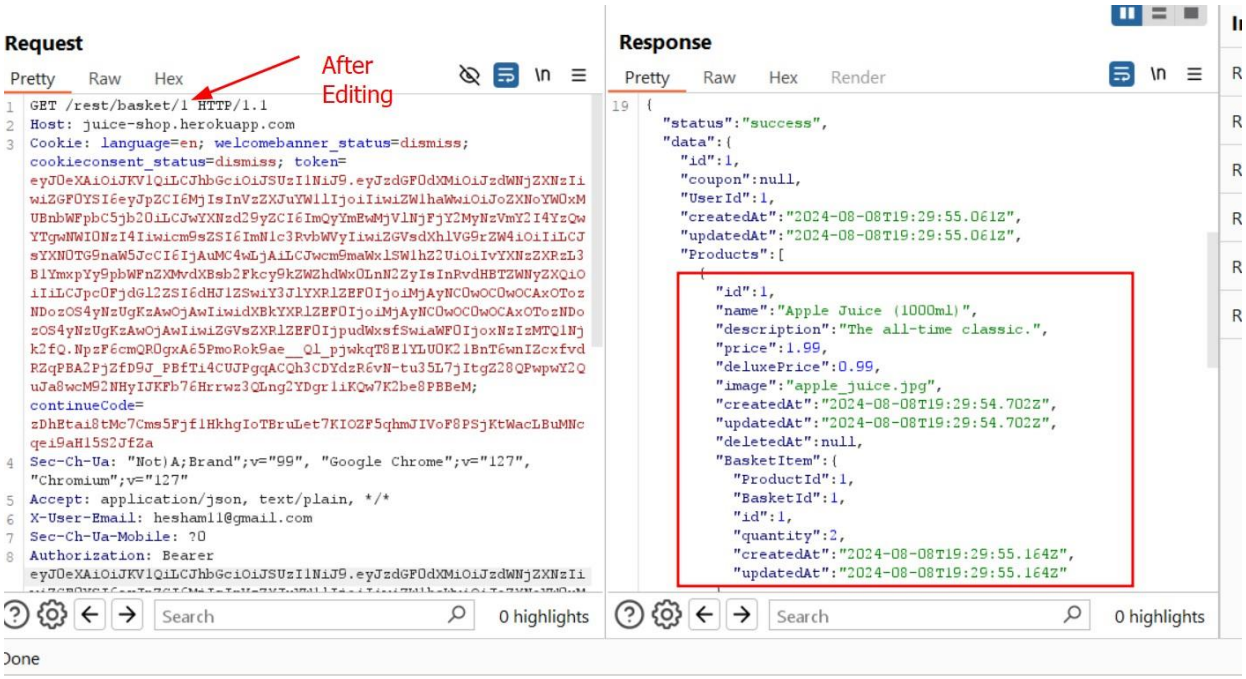
1) Navigate to the Juice Shop application at https://juice-shop.com/#/basket/{basketID}.



2) Intercept the HTTP request using Burp Suite.
3) Send the intercepted request to the Repeater tool.



4) Modify the **{basketID}** in the request to a different value (which corresponds to anotheruser's data).
5) Send the modified request.



6) Observe that the response contains Basket data from the user associated with the new **basketID**, indicating successful access to unauthorized data.

Impact

This vulnerability allows attackers to access and view sensitive data belonging to other users, potentially exposing personal information and other confidential details. The unauthorized access can lead to privacy violations and could be leveraged for further attacks.

Recommended Fix

Implement proper authorization checks to ensure users can only access their own data. The application should verify the identity of the requesting user and ensure that they have permission to access the requested resource. Additionally, consider applying the principle ofleast privilege and data access controls to mitigate similar issues.

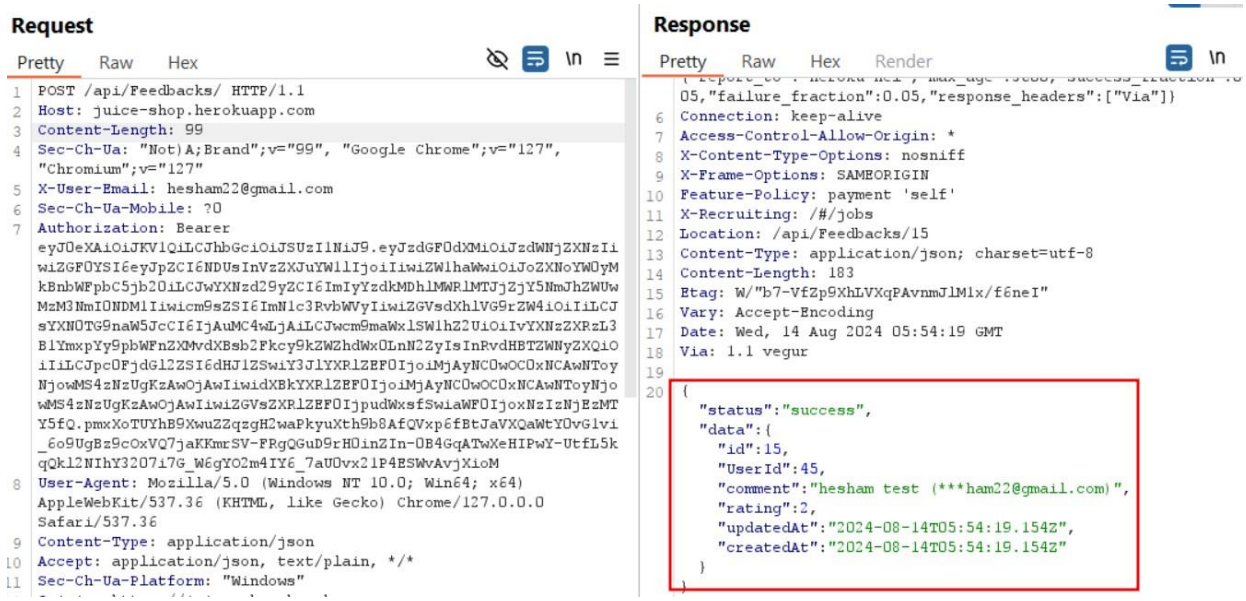
(H003) Insecure Direct Object Reference (IDOR) Leadto Unauthorized access to all feedbacks

Weakness Type: Broken Access Control

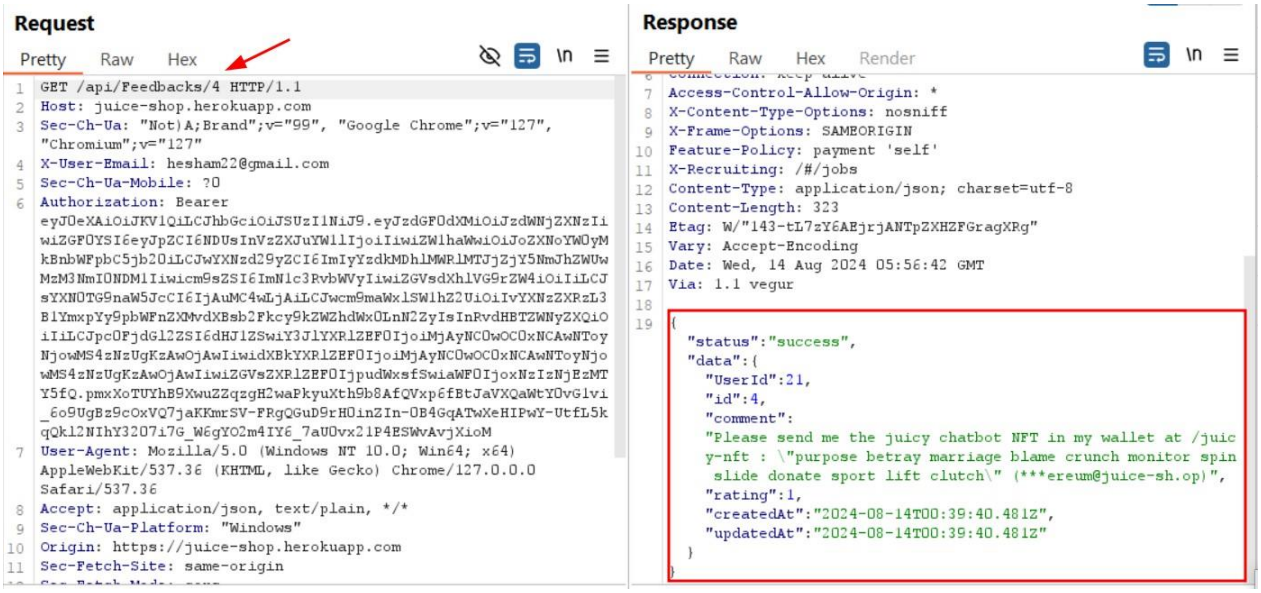
An Insecure Direct Object Reference (IDOR) vulnerability was discovered in the feedback management section of the application. This vulnerability allows an authenticated user to access feedback records belonging to other users by directly manipulating the feedback ID inthe URL or API request. As a result, any user can gain unauthorized access to all feedback entries within the system, potentially exposing sensitive user information

Steps to Reproduce:

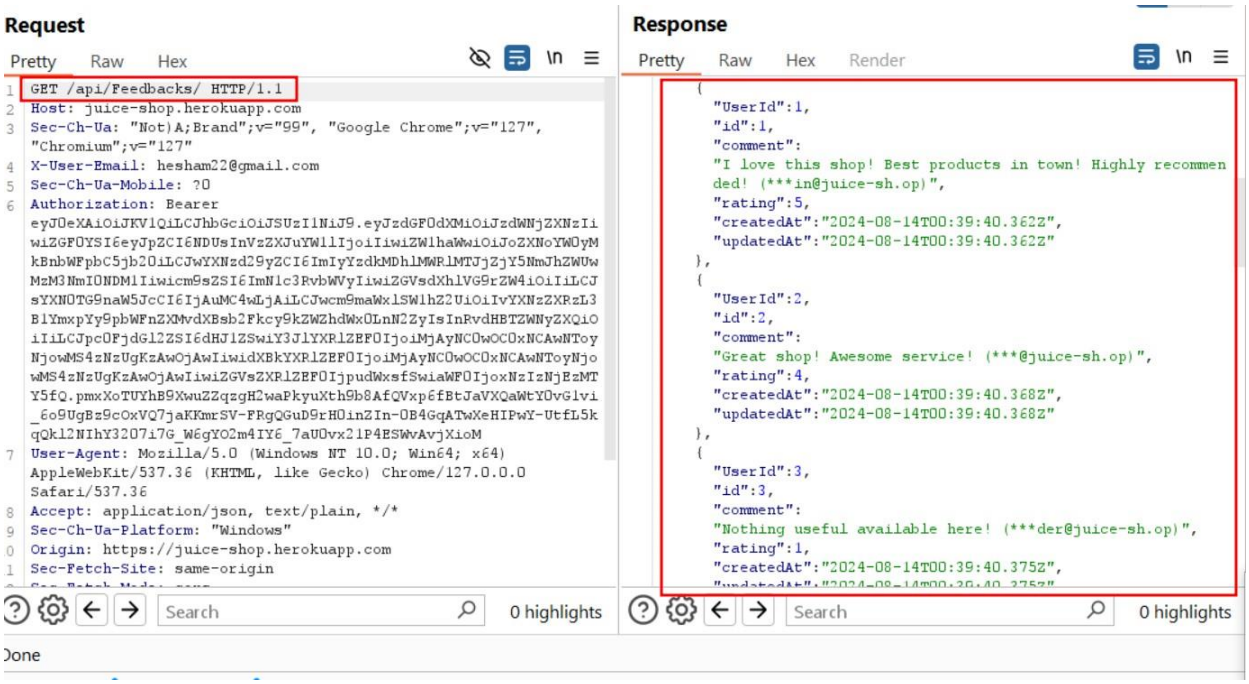
1) **Intercept proxy via burpsuite**
2) **Go to https://juice-shop.herokuapp.com/#/contact and submit any feedback thenshow the request.**
3) **Try to delete a cookie and send request. Observe that the returned response showfeedback without checking if the user is authenticated or not.**



4) **Change method and add any id after the endpoint. It will return feedback that itbelongs to this id.**



5) **Also without adding any id, we can retrieve all feedbacks by change method to GETonly.**



Impact

- The impact of this vulnerability is significant, as it allows unauthorized users to view andpossibly alter feedback entries that do not belong to them. This can lead to a breach of privacy, leakage of sensitive information, and loss of trust in the platform.

Recommended Fix

1. **Implement Object-Level Authorization:** Ensure users can only access feedback entriesthey own by validating the ownership of the feedback ID.
2. **Use Indirect Object References**: Replace direct references in URLs or API requests withnon-guessable identifiers like UUIDs

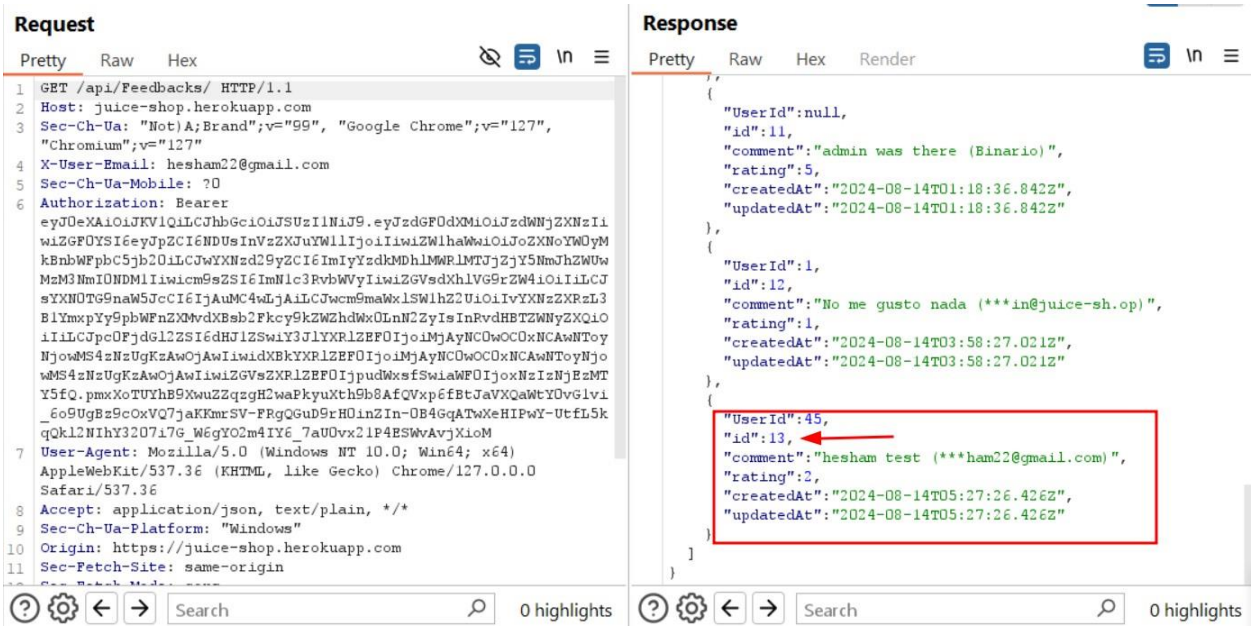
(H004) Insecure Direct Object Reference (IDOR)Leads to Deletion of Any Feedback

Weakness Type: Broken Access Control

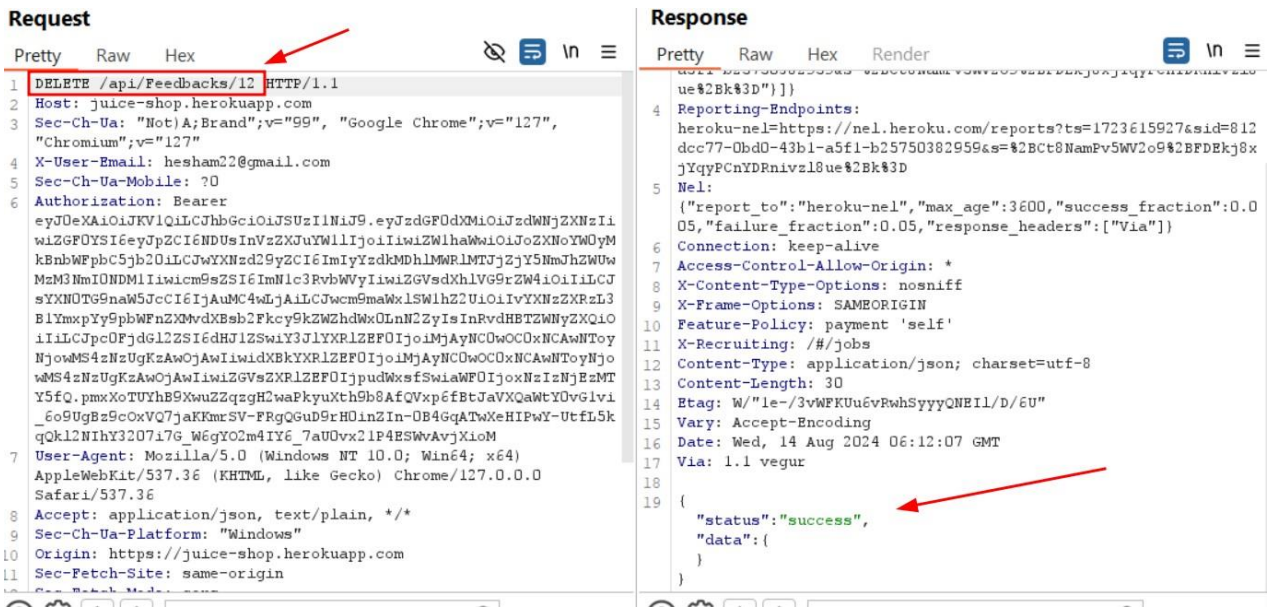
An Insecure Direct Object Reference (IDOR) vulnerability was discovered in the feedback management system of the application. This vulnerability allows an authenticated user to delete feedback entries that they do not own, simply by altering the feedback ID parameter in the DELETE request. The lack of proper authorization checks on the server side permits users todirectly access and manipulate feedback entries that they should not have access to.

Steps to Reproduce:

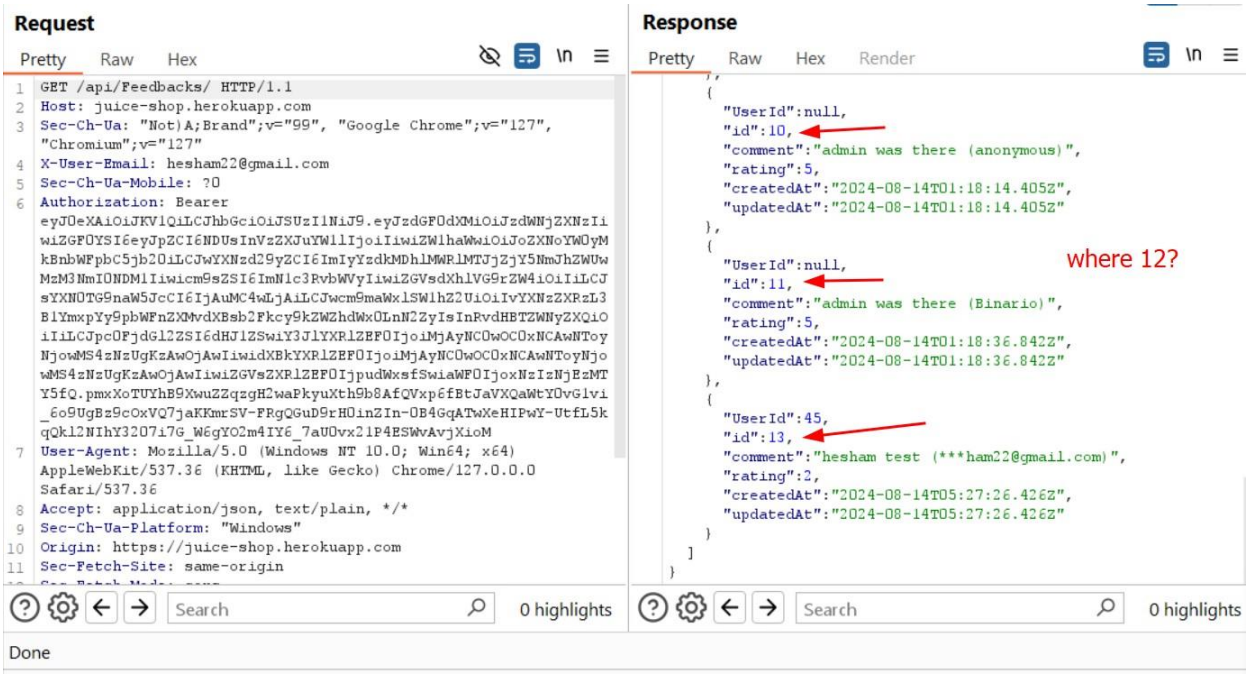
1) **Intercept proxy via burpsuite**
2) **Go to https://juice-shop.herokuapp.com/#/contact and submit any feedback thenshow the request.**
3) **By exploiting last vulnerability change method to GET to know your feedback id.**



4) **My feedback id is 13. Now I will delete feedback with id 12 by changing method toDELETE and add the id after endpoint.**



5) **View all feedback and observe that the feedback with id 12 has been deleted.**



Impact

An attacker can delete any feedback, including those submitted by other users, without properauthorization. This can result in data loss and a breach of the integrity of the feedback system, potentially leading to loss of trust and operational issues.

Recommended Fix

- **Implement authorization checks: Verify that the user has the right to delete thespecific feedback.**
- **Use indirect references: Replace direct IDs with tokens or hashed IDs.**

(H005) IDOR Leading to Unauthorized Comment Submission on Any Post with Any Email

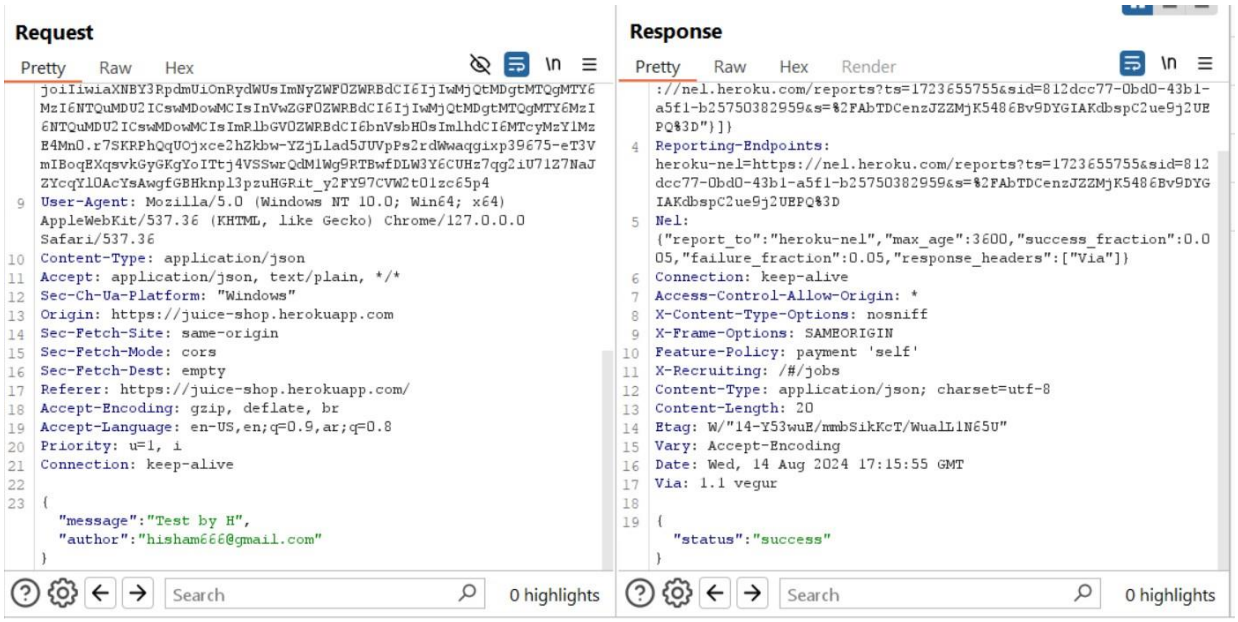
Weakness Type: Authorization Bypass Through User-Controlled Key

Description

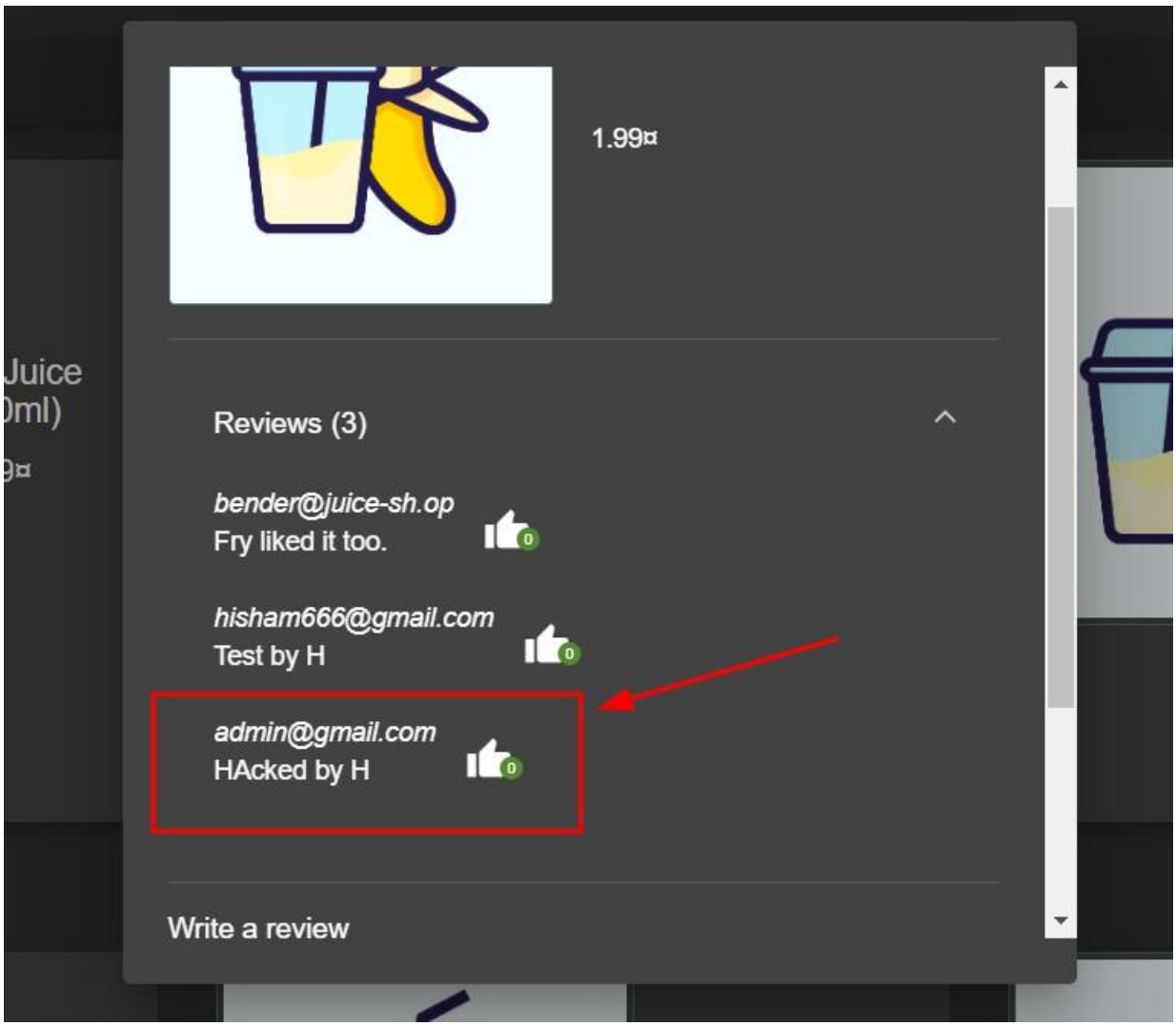
An Insecure Direct Object Reference (IDOR) vulnerability was identified in the comment submission functionality. This flaw allows an attacker to submit comments on any post usingany email address, bypassing authorization mechanisms. The vulnerability exists because theserver fails to properly validate the user's authorization to post comments on behalf of a specific email address.

Steps to Reproduce:

1) **Intercept proxy via burpsuite**
2) **Go to any post and submit any comment then show the request in repeater.**



3) **Try to edit author value to admin email. Email has been submitted with admin email**



Impact

The IDOR vulnerability can lead to the following impacts:

- **Impersonation:** Attackers can submit comments using any email address, making itappear that someone else made the comment.
- **Reputation Damage:** Fake or malicious comments can damage the reputation of theusers whose emails are misused.
- **Trust Issues:** Users might lose trust in the platform if they notice comments submittedunder their email address without their consent.

Recommended Fix

1. **Server-Side Validation:** Ensure that the server validates the email address associatedwith the logged-in user. The server should reject any request where the email parameter does not match the authenticated user's email.
2. **Authorization Checks:** Implement proper authorization checks to ensure that users canonly submit comments on behalf of their own accounts.

(H006) Email Creation Without Any VerificationWeakness

Type: Improper Verification of Cryptographic Signature.

Description

The application allows users to create accounts using an email address without requiring any form of verification. This can lead to various security risks, including account abuse, spam, andphishing attacks. An attacker can register an email account and use it for malicious activities without ever confirming that they have access to the specified email address.

Steps to Reproduce:

1) **Go to: https://juice-shop.com/#/register**
2) **Fill up Data and click Register.**
3) **Email Will be created without any Verification.**

Impact

- **Account Abuse**: Unverified accounts can be used for spam or other malicious activities.
- **Phishing Risk**: Attackers may impersonate legitimate users with unverified emails.

Recommended Fix

- **Implement Email Verification**: Require users to verify their email before accessing theiraccount.

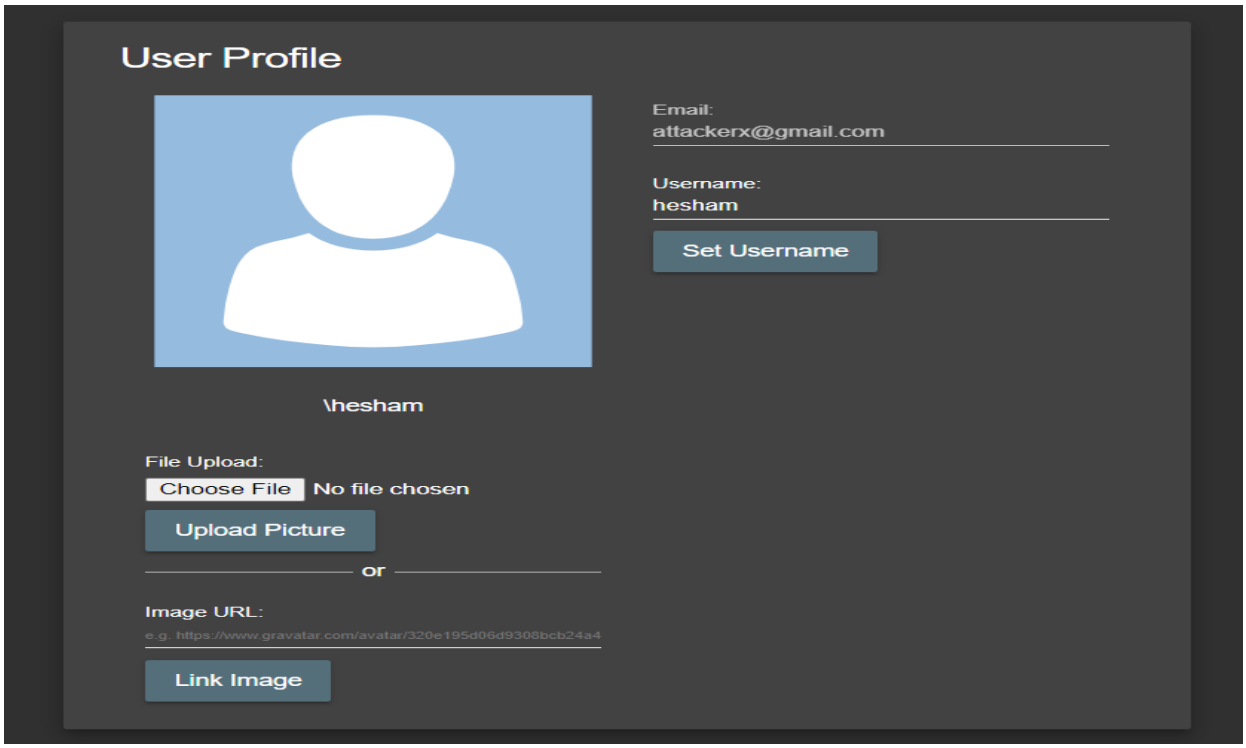
(H007) Cross-Site Request Forgery (CSRF) inUsername Update Functionality

Weakness Type: Cross-Site Request Forgery (CSRF)Description

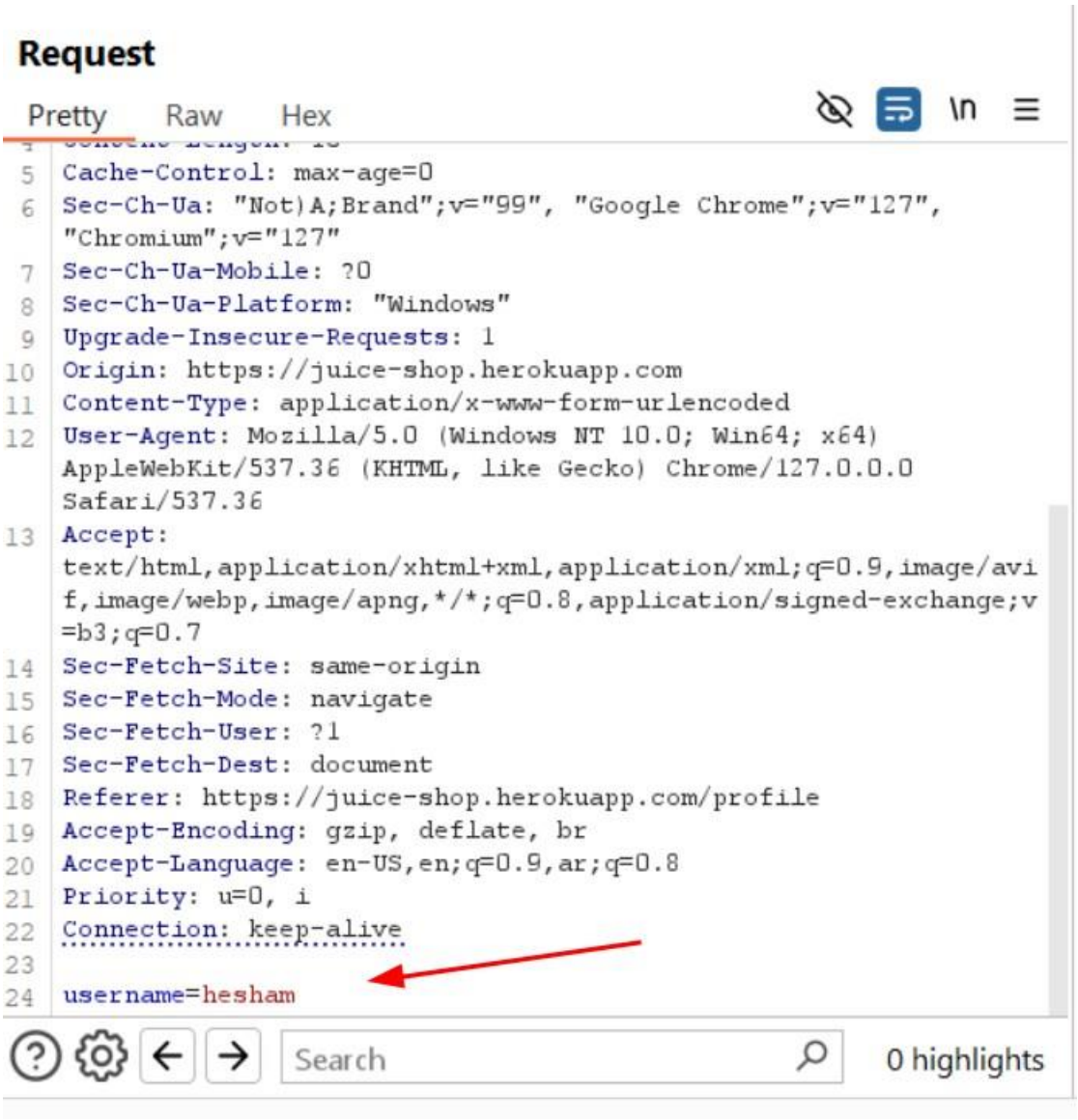
A Cross-Site Request Forgery (CSRF) vulnerability exists in the username update functionality ofthe application. This vulnerability allows an attacker to force authenticated users to change their username without their consent. The issue arises due to the lack of anti-CSRF tokens or other protective measures in the username update request, enabling unauthorized actions onbehalf of the victim.

Steps to Reproduce:

1. Go to https://juice-shop.herokuapp.com/profile

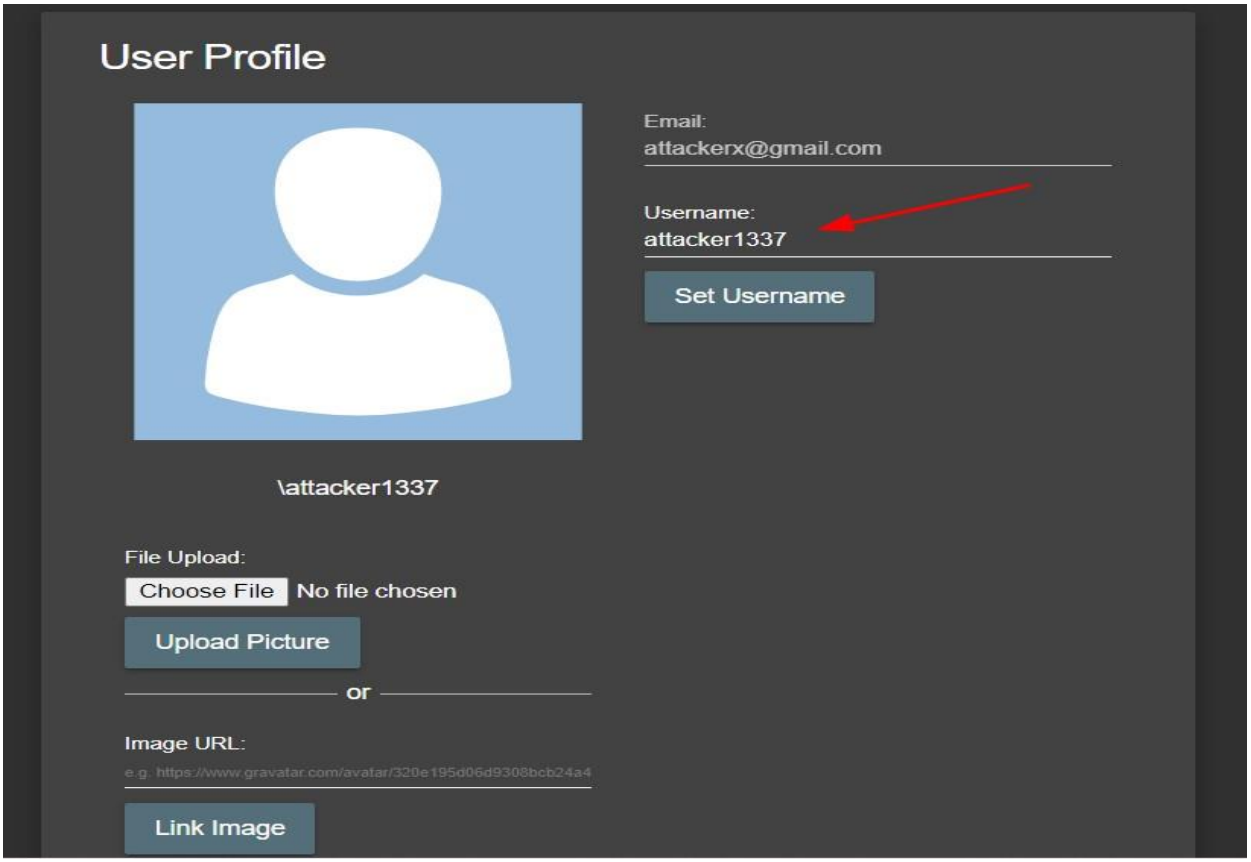


2. Capture the request that it change username. Observe that the request didn't has any csrf protection.



3. Directly make a POC => Username has been changed.

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <form action="https://juice-shop.herokuapp.com/profile" method="POST">
      <input type="hidden" name="username" value="attacker1337" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```



Impact

The vulnerability could be exploited to perform unauthorized changes to a user's username.This can lead to user confusion, potential denial of service (e.g., if usernames are changed tonon-usable formats), or further exploitation if the username is used in security checks or logging. The attacker could also leverage this to disrupt user identification processes.

Recommended Fix

1. Implement Anti-CSRF Tokens in the username update form and validate them server-side.
2. Set SameSite Attribute on session cookies to Strict or Lax.
3. Require User Interaction (e.g., password re-entry) for username changes.

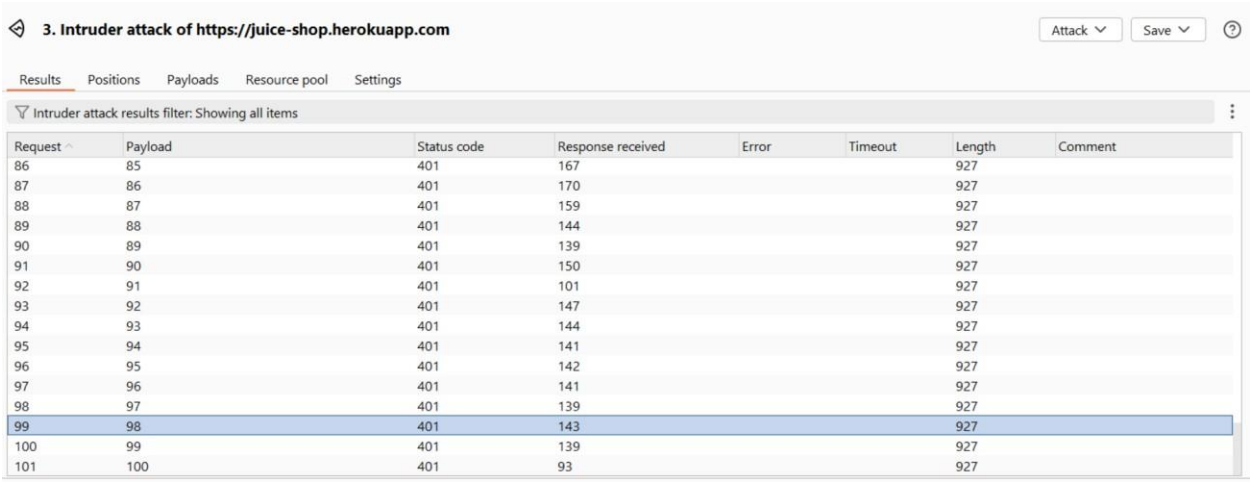
(H008) No Rate Limiting in Login Page Weakness Type: Improper Control of Interaction FrequencyDescription

The login page of the application lacks rate limiting, allowing an attacker to perform brute-forceattacks. Without rate limiting, an attacker can automate login attempts to guess user credentials by trying multiple password combinations in quick succession. This vulnerability significantly increases the risk of unauthorized access to user accounts, especially when weak passwords are in use.

Steps to Reproduce:

1- **Go to https://juice-shop.herokuapp.com/#/login and submit any email and invalidpassword.**

2- **Send request to intruder and brute-force password.**



Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
86	85	401	167			927	
87	86	401	170			927	
88	87	401	159			927	
89	88	401	144			927	
90	89	401	139			927	
91	90	401	150			927	
92	91	401	101			927	
93	92	401	147			927	
94	93	401	144			927	
95	94	401	141			927	
96	95	401	142			927	
97	96	401	141			927	
98	97	401	139			927	
99	98	401	143			927	
100	99	401	139			927	
101	100	401	93			927	

3- **I didn’t get any rate limit or block.**

Impact

The absence of rate limiting exposes user accounts to brute-force attacks. An attacker couldpotentially:

1. Compromise user accounts by successfully guessing credentials.
2. Gain unauthorized access to sensitive user information.
3. Elevate privileges if higher-privileged accounts are compromised.
4. Cause account lockouts or denial of service by exhausting login attempts.

Recommended Fix

1. **Implement Rate Limiting**: Limit the number of login attempts per IP or accountwithin a short timeframe.
2. **Account Lockout**: Temporarily lock accounts after a set number of failedattempts.
3. **Add CAPTCHA**: Introduce CAPTCHA after a defined number of failed attempts.

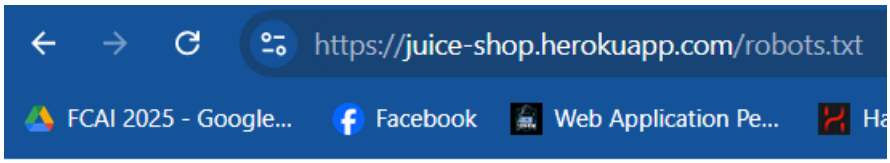
(H009) Publicly Accessible Important File ExposesSensitive Information

Weakness Type: Information DisclosureDescription

An important file containing sensitive information was found to be publicly accessible on the server. This file includes details that could be exploited by an attacker to gain unauthorized access to the system, potentially leading to further compromise.

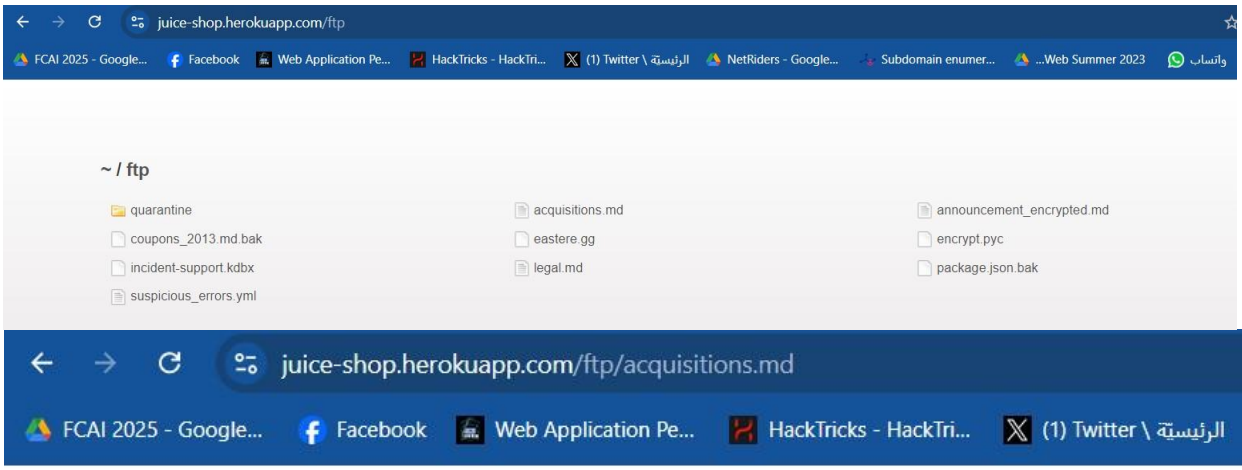
Steps to Reproduce:

1- **Go to https://juice-shop.herokuapp.com/robots.txt**




```
User-agent: *
Disallow: /ftp
```

2- **When accessing this file, Important files disclosed.**




```
# Planned Acquisitions

> This document is confidential! Do not distribute!

Our company plans to acquire several competitors within the next year.
This will have a significant stock market impact as we will elaborate in
detail in the following paragraph:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet
clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
ipsum dolor sit amet.

Our shareholders will be excited. It's true. No fake news.
```

Impact

An attacker with access to this file could extract sensitive information, such as credentials, configuration details, or other critical data, which could be used to further attack the system,escalate privileges, or disrupt operations.

Recommended Fix

1. Restrict access to the file by implementing proper access controls. Ensure that onlyauthorized users or services can access the file.
2. Move sensitive data into secure storage solutions, such as environment variables orencrypted configuration files, which are not publicly accessible.

(H010) Username and Passwords Leaked in SourceCode Weakness Type: Use of Hard-coded CredentialsDescription

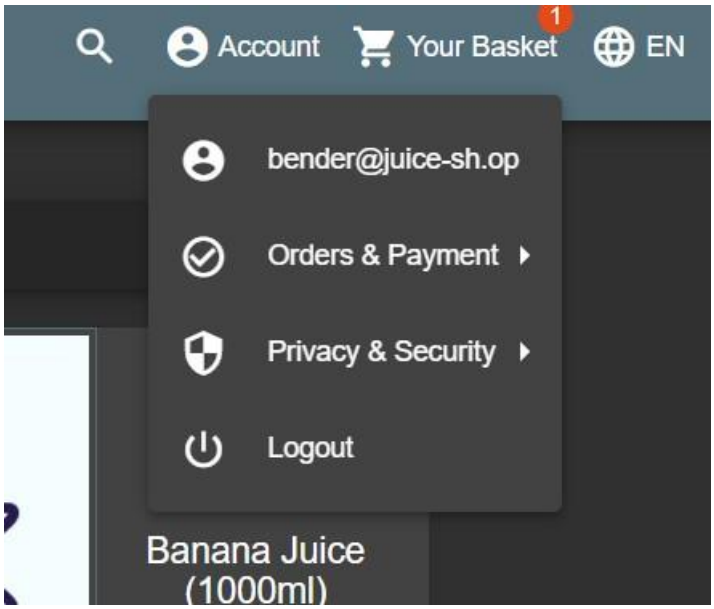
A critical vulnerability was discovered where sensitive information, specifically usernames andpasswords, were exposed in the application's source code repository. The issue stems from improper handling of credentials, which were inadvertently committed and pushed to a public-facing repository. This allows unauthorized individuals to access these credentials and potentially exploit the associated services or systems.

Steps to Reproduce:

1- **Go to https://juice-shop.herokuapp.com/main.js**

2- **Search for password in source code. Emails and passwords are disclosed in sourcecode.**

3. Select email and try to login with it.



Impact

Exposure of usernames and passwords in the source code can lead to unauthorized access to critical systems, databases, or APIs. Attackers with access to this information could impersonatelegitimate users, steal sensitive data, modify system configurations, or carry out further attacks within the network. The severity of this issue is high, as it affects the confidentiality of sensitive information and could lead to significant security breaches.

Recommended Fix

1. **Remove Hardcoded Credentials**: Immediately remove the hardcoded usernames andpasswords from the source code and use environment variables instead.
2. **Rotate Credentials**: Change the exposed credentials to invalidate any potential misuse.
3. **Audit Repository**: Scan the repository for other potential exposures using tools like git-secrets or truffleHog.
4. **Update .gitignore**: Ensure sensitive files are excluded from commits using .gitigno

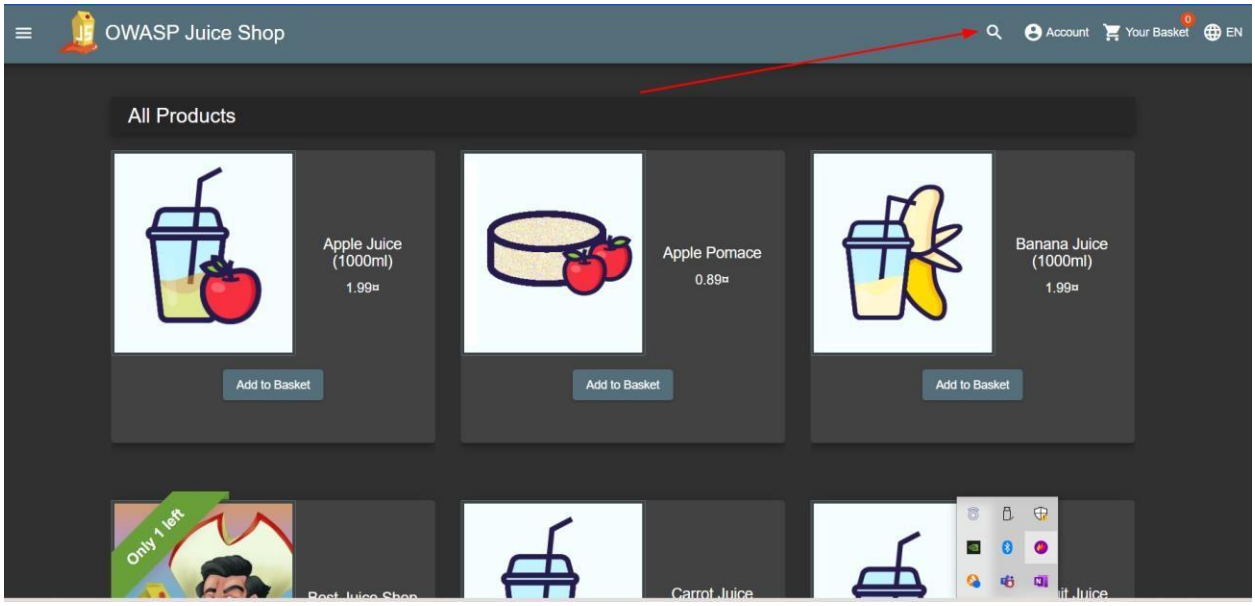
(H011) DOM-based Cross-Site Scripting (XSS) inSearch Parameter (q)

Weakness Type: Cross-site Scripting (XSS)Description

A DOM-based Cross-Site Scripting (XSS) vulnerability was identified in the search functionality of the Juice Shop web application. The application dynamically generates content based on userinput without properly sanitizing or escaping it, allowing an attacker to inject and execute arbitrary JavaScript code in the context of the victim's browser.

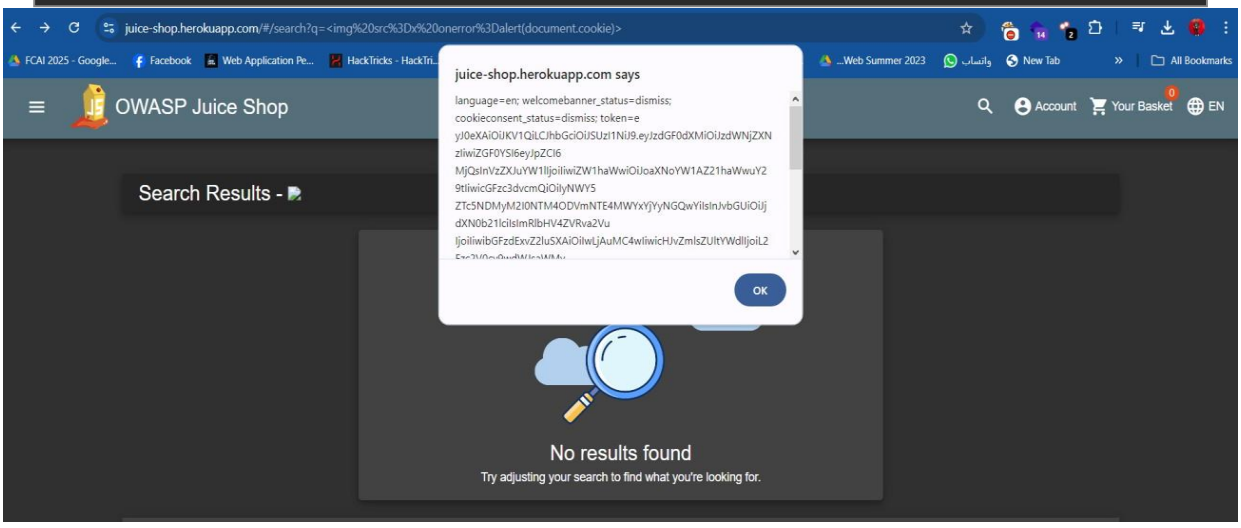
Steps to Reproduce:

1- **Go to https://juice-shop.herokuapp.com/#/search**
2- **Locate the Search Bar , find the search bar where you can input text to search forproducts.**



3- **Inject Malicious JavaScript Code**

```
<img src=x onerror=alert(document.cookie)>
```



Impact

1. Stealing sensitive information (e.g., session cookies, local storage data).
2. Performing unauthorized actions on behalf of the victim.
3. Modifying the content of the webpage.
4. Phishing attacks by presenting malicious content to the user.

This could lead to unauthorized access to the victim's account or other maliciousactivities performed under the victim's session.

Recommended Fix

1- **Sanitize Input:** Validate and sanitize all user inputs.
2- **Escape Output:** Properly escape any user-generated content before inserting it into **the** DOM.
3- **Use a Secure Framework:** Consider frameworks like Angular or React that prevent directDOM manipulation.
4- **Implement CSP:** Apply a strict Content Security Policy (CSP) to block inline scriptexecution.

(H012) SSRF Leading to Access Internal Resources

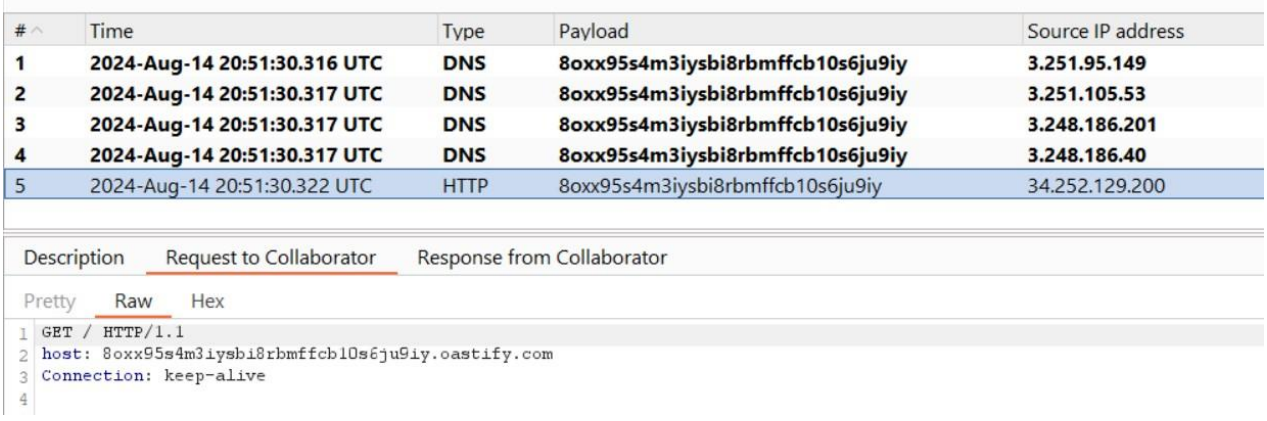
Weakness Type: Server-Side Request Forgery (SSRF)

Description

A Server-Side Request Forgery (SSRF) vulnerability was identified in the application, allowing anattacker to send crafted requests from the server. This flaw enables an attacker to access internal resources that are not directly exposed to the internet. SSRF occurs when an application fetches a remote resource based on user-controlled input without proper validation.

Steps to Reproduce:

1- **Go to https://juice-shop.herokuapp.com/profile**
2- **Update image url with your server ip.**
3- **Go to logs and observing requests.**



#	Time	Type	Payload	Source IP address
1	2024-Aug-14 20:51:30.316 UTC	DNS	8oxx95s4m3iysbi8rbmffcb10s6ju9iy	3.251.95.149
2	2024-Aug-14 20:51:30.317 UTC	DNS	8oxx95s4m3iysbi8rbmffcb10s6ju9iy	3.251.105.53
3	2024-Aug-14 20:51:30.317 UTC	DNS	8oxx95s4m3iysbi8rbmffcb10s6ju9iy	3.248.186.201
4	2024-Aug-14 20:51:30.317 UTC	DNS	8oxx95s4m3iysbi8rbmffcb10s6ju9iy	3.248.186.40
5	2024-Aug-14 20:51:30.322 UTC	HTTP	8oxx95s4m3iysbi8rbmffcb10s6ju9iy	34.252.129.200

Description | Request to Collaborator | Response from Collaborator

Pretty | Raw | Hex

```
1 GET / HTTP/1.1
2 host: 8oxx95s4m3iysbi8rbmffcb10s6ju9iy.oastify.com
3 Connection: keep-alive
4
```

Impact

The impact of this SSRF vulnerability is significant, as it allows an attacker to leverage the serverto perform unauthorized actions within the internal network. This could include:

1. Accessing internal services and resources that are otherwise inaccessible from the publicinternet.
2. Exfiltrating sensitive data from internal endpoints.
3. Potentially escalating to further attacks such as Remote Code Execution (RCE)depending on the internal services exposed.

Recommended Fix

1. **Input Validation:** Only allow external URLs from a whitelist. Block requests to private IPranges (e.g., 127.0.0.0/8, 10.0.0.0/8).
2. **DNS Resolution:** Ensure URLs resolve to public IP addresses only.
3. **Network Segmentation:** Isolate internal services from public-facing interfaces.
4. **Logging:** Monitor and log server requests to detect SSRF attempts.

Medium Risk Issues [M]:-

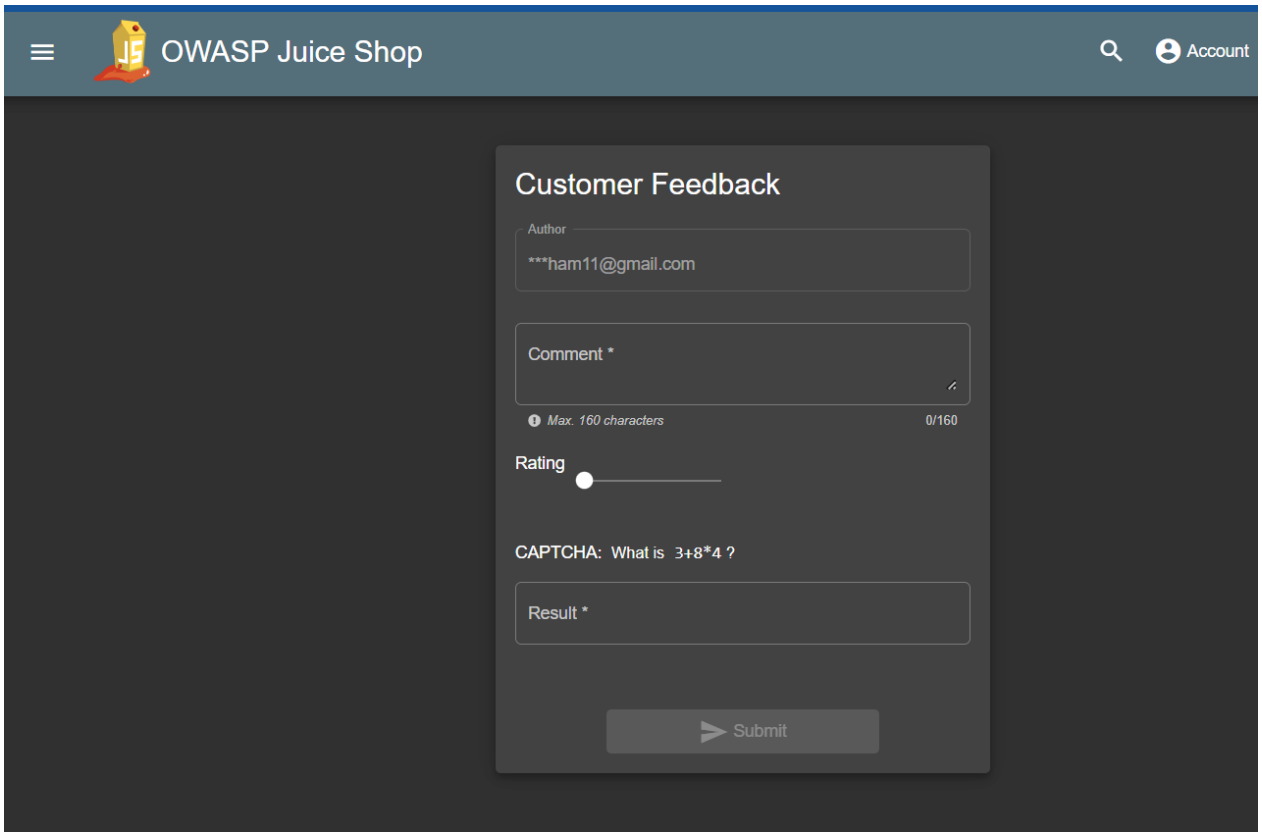
(M001) Insecure Direct Object Reference (IDOR) in UserId Parameter (Complain Function)

Weakness Type: Broken Access ControlDescription

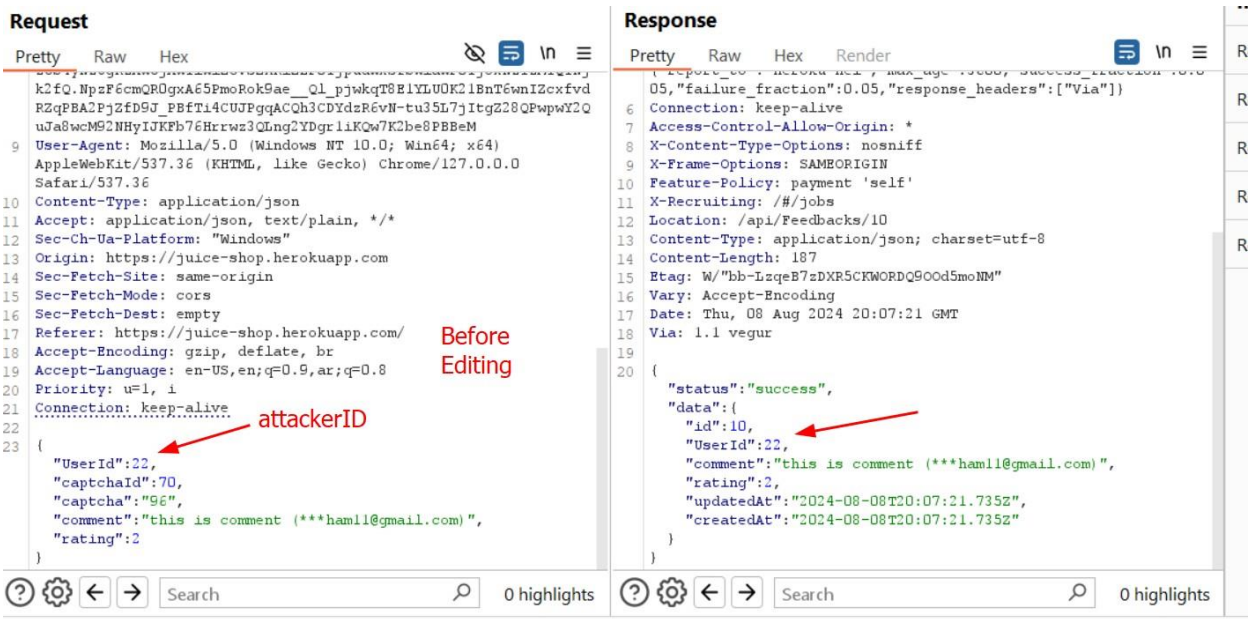
An Insecure Direct Object Reference (IDOR) vulnerability was discovered in the **UserId** parameter of the request sent to the **/contact** endpoint. This vulnerability allows an attacker tomanipulate the **UserId** parameter to gain access to or alter information of other users.

Steps to Reproduce:

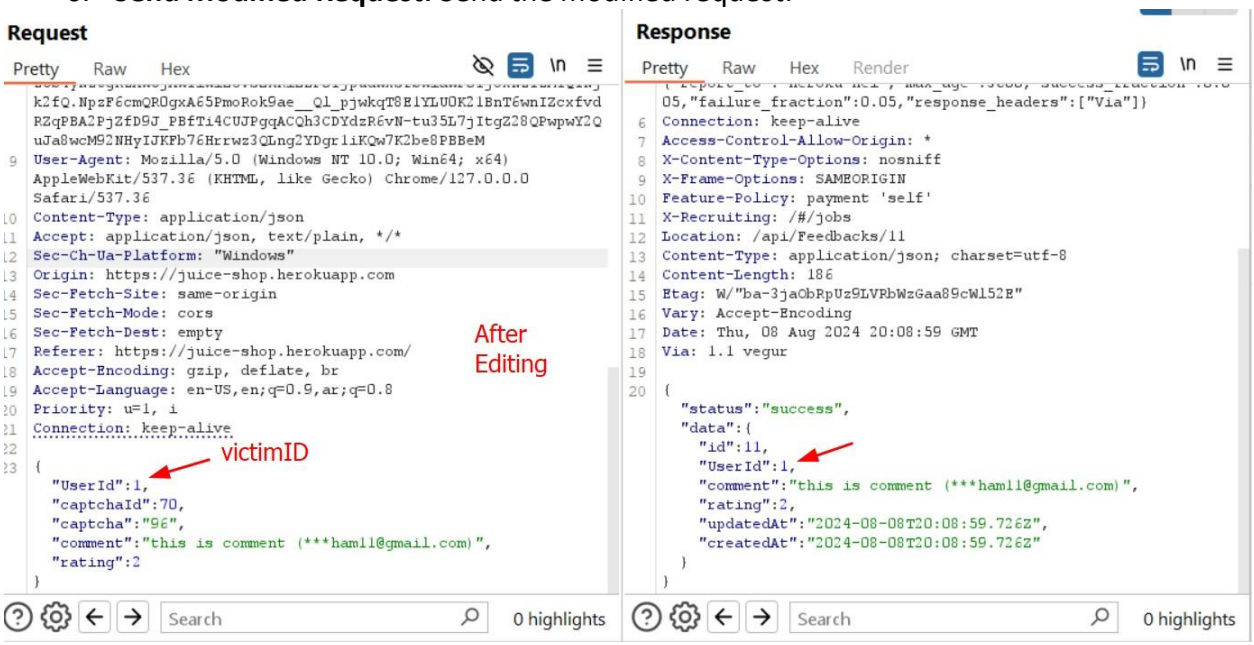
1. **Go to:** https://juice-shop.com/#/contact
2. **Intercept Request:** Use Burp Suite to intercept the request sent when you submit afeedback message.



3. **Submit Feedback:** Fill out the feedback form with a message and send the request.
4. **Observe Request:** In the intercepted request, find the UserId parameter.



5. **Modify UserId:** Move the intercepted request to the Repeater tab in Burp Suite. Edit theUserId value to that of a victim user (replace UserId with VictimId).
6. **Send Modified Request:** Send the modified request.



Impact

An attacker can exploit this vulnerability to access or modify feedback messages associated with other users. This could lead to unauthorized information disclosure or tampering with feedback data. In the worst-case scenario, it could allow attackers to impersonate other usersor disrupt user feedback integrity.

Recommended Fix

Implement proper access control checks on the server-side to ensure that users can only accessor modify their own feedback data. Ensure that user access permissions are validated against authenticated session information before processing requests involving user-specific data.

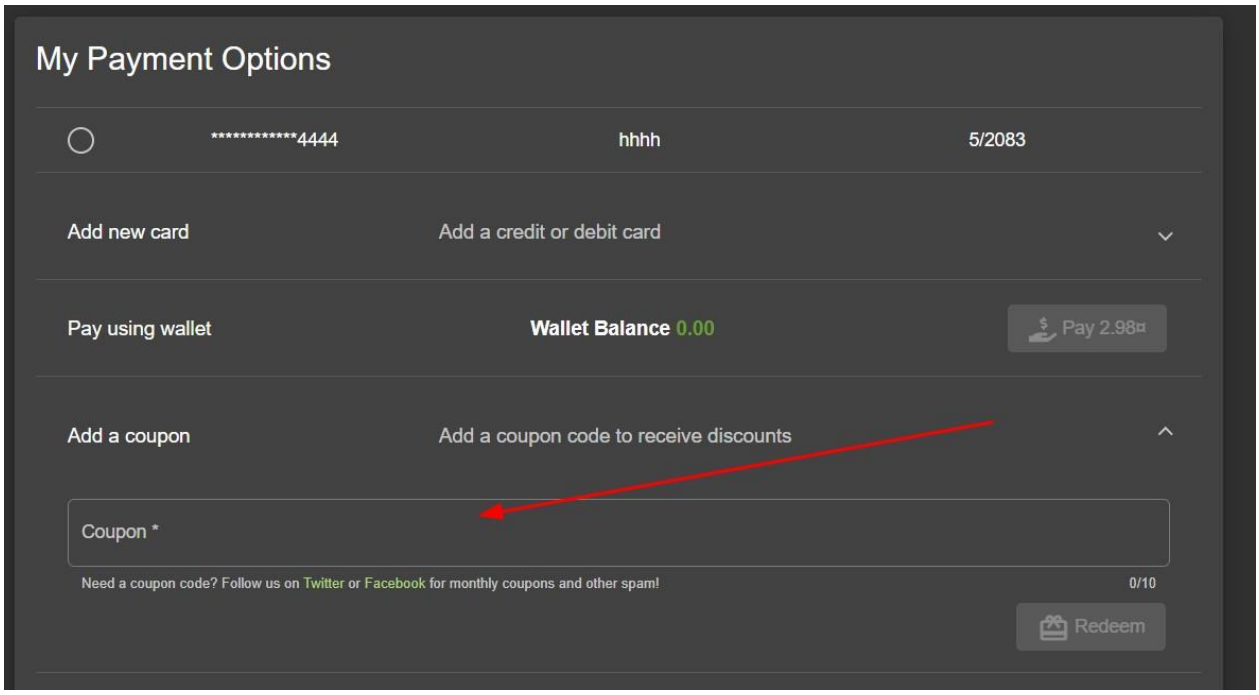
(M002) Absence of Rate Limiting in Coupon Redemption Function

Weakness Type: Lack of Rate LimitingDescription

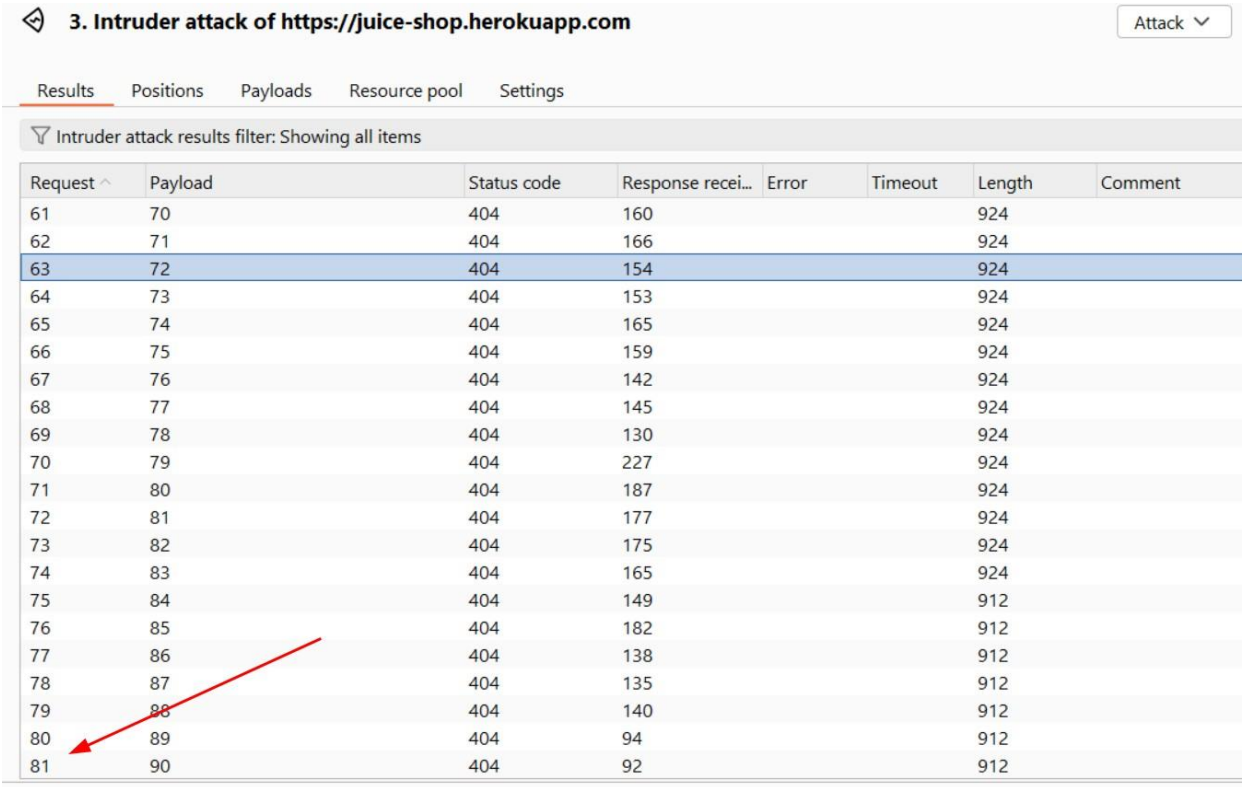
The coupon redemption function within the application lacks rate limiting controls. This absence allows an attacker to make an excessive number of coupon redemption requests in ashort period. Rate limiting is essential to prevent abuse and ensure fair use of resources.
Without it, the system is vulnerable to exploitation, potentially leading to denial of service orother unintended consequences.

Steps to Reproduce:

1. **Add Item to Basket**
    - Go to the Juice Shop website and add an item to your basket.
2. **Proceed to Checkout:**
    - Complete the checkout process by entering the required details (name, address,credit card information).
3. **Access Coupon Input Field:**
    - After completing the checkout, you will be redirected to the payment page at:https://juice-shop.herokuapp.com/#/payment/shop.



4. **Test Coupon Input Field:**
    - On the payment page, locate the coupon input field.
    - Begin entering various coupon codes rapidly.
    - Note that the application does not enforce rate limiting, allowing for anexcessive number of submissions in a short timeframe.



Request	Payload	Status code	Response recei...	Error	Timeout	Length	Comment
61	70	404	160			924	
62	71	404	166			924	
63	72	404	154			924	
64	73	404	153			924	
65	74	404	165			924	
66	75	404	159			924	
67	76	404	142			924	
68	77	404	145			924	
69	78	404	130			924	
70	79	404	227			924	
71	80	404	187			924	
72	81	404	177			924	
73	82	404	175			924	
74	83	404	165			924	
75	84	404	149			912	
76	85	404	182			912	
77	86	404	138			912	
78	87	404	135			912	
79	88	404	140			912	
80	89	404	94			912	
81	90	404	92			912	

Impact

1. **Denial of Service (DoS**): An attacker could overwhelm the system with rapid coupon redemption requests, causing legitimate users to experience degraded performance orloss of functionality.
2. **Abuse of Coupons:** Attackers could exploit the lack of rate limiting to redeem multiplecoupons fraudulently, potentially resulting in financial loss or inventory issues.

Recommended Fix

Implement rate limiting on the coupon redemption function to restrict the number of requestsper user or IP within a given timeframe.

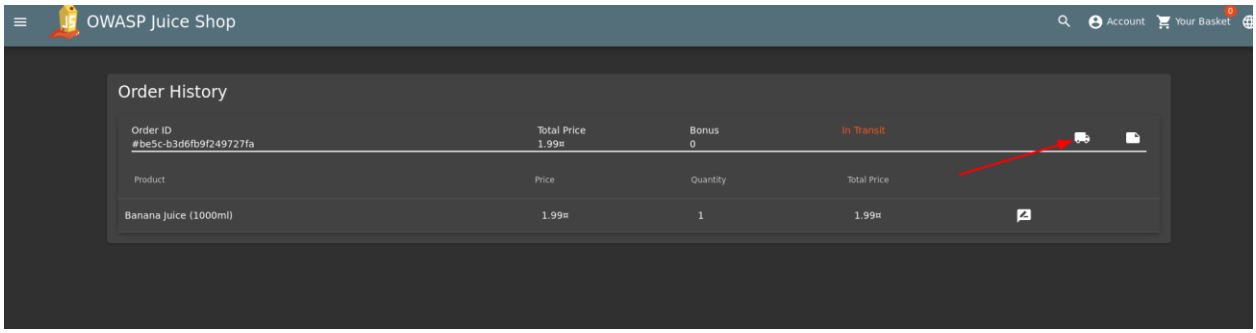
(M003) Reflected Cross-Site Scripting (XSS) in id Parameter in Delivery Section

Weakness Type: Cross-site Scripting

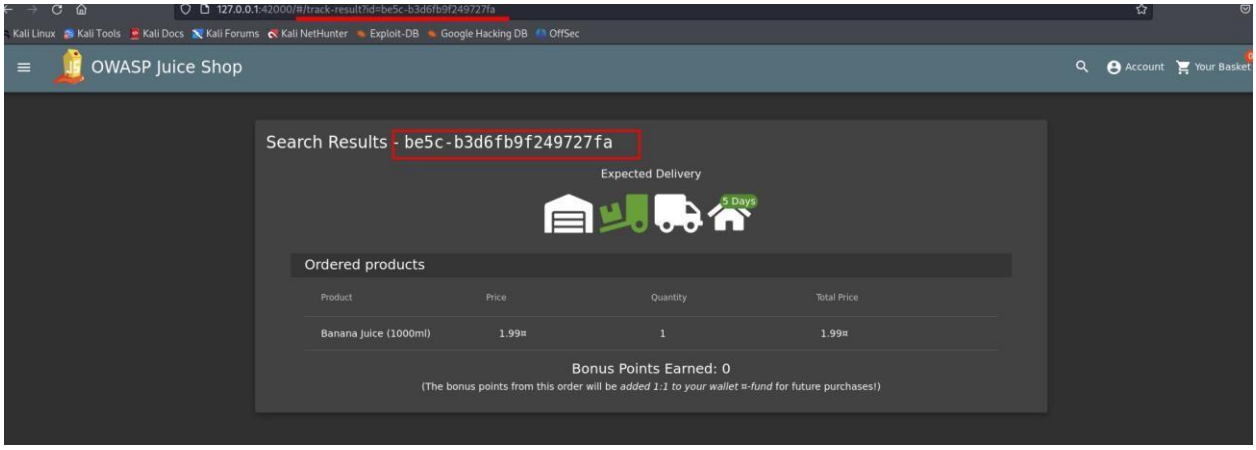
A reflected XSS vulnerability exists in the application where user input in the id parameter is notproperly sanitized before being reflected back to the user. An attacker can exploit this by crafting a malicious URL containing a JavaScript payload. When a victim clicks on this link, the malicious script executes in the context of the user's browser, potentially leading to session hijacking, defacement, or other malicious actions.

Steps to Reproduce:

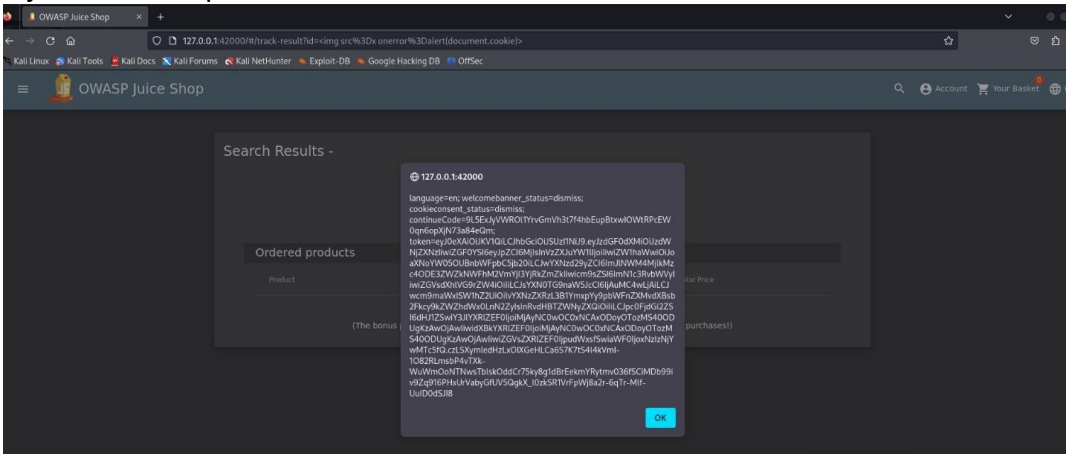
1- After making an invoice, Go to https://juice-shop.herokuapp.com/#/order-history .



2- Click to Delivery Method. The Invoice id reflected in source code.



3- Inject a Javascript code => XSS.



Impact

An attacker can execute arbitrary JavaScript in the context of the victim's browser. This can lead to various attacks such as:

1. Stealing session cookies, allowing the attacker to impersonate the user.
2. Redirecting the victim to a malicious site.
3. Displaying fake login forms to capture user credentials.
4. Defacing the web application by modifying the displayed content.

Recommended Fix

1. Sanitize all user inputs by escaping or removing potentially dangerous characters such as <, >, &, ", and '.
2. Implement a Content Security Policy (CSP) to restrict the sources from which scripts canbe loaded.
3. Use security libraries or frameworks that provide built-in protection against XSS.

(M004) Client-Side Validation Bypass in Password Parameter in Registration Form

Weakness Type: Client-Side Enforcement of Server-SideSecurity

Description

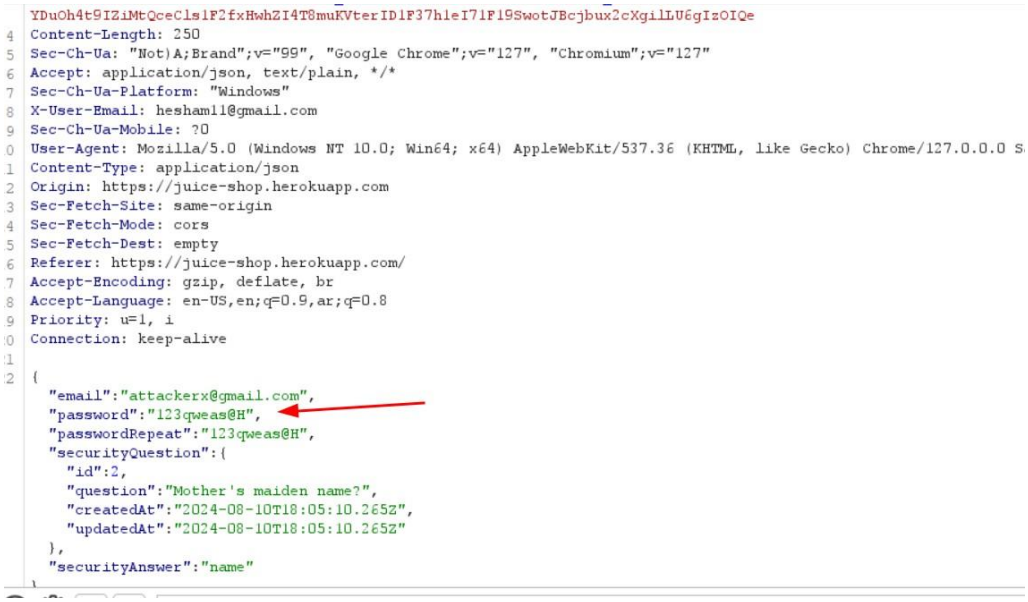
The registration form on the application implements client-side validation for the password parameter. However, this validation can be bypassed by manipulating the client-side code or intercepting and modifying the request before it reaches the server. This allows an attacker tosubmit weak or non-compliant passwords that do not adhere to the security policies intendedby the application, such as minimum length or complexity requirements.

Steps to Reproduce:

1- Go to Registration page https://juice-shop.herokuapp.com/#/register

2- Intercept request via burpsuite.

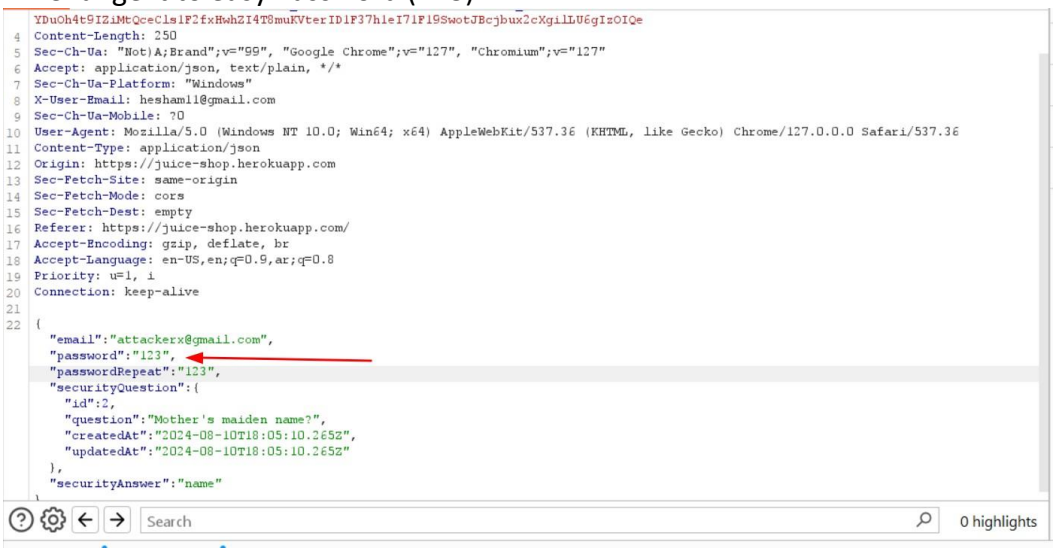
3- Enter a Dummy Data and Show Request.



```
YDuOh4t9IZiMtQceCls1F2fxHwhZI4T8muKVterID1F37hleI71F19SwotJBcjbux2cXgilLU6gIzOIQe
Content-Length: 250
Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
Accept: application/json, text/plain, */*
Sec-Ch-Ua-Platform: "Windows"
X-User-Email: hesham11@gmail.com
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 S
Content-Type: application/json
Origin: https://juice-shop.herokuapp.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://juice-shop.herokuapp.com/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Priority: u=1, i
Connection: keep-alive

{
  "email":"attackerx@gmail.com",
  "password":"123qweas@H",
  "passwordRepeat":"123qweas@H",
  "securityQuestion":{
    "id":2,
    "question":"Mother's maiden name?",
    "createdAt":"2024-08-10T18:05:10.265Z",
    "updatedAt":"2024-08-10T18:05:10.265Z"
  },
  "securityAnswer":"name"
}
```

4- Change it to easy Password (123).

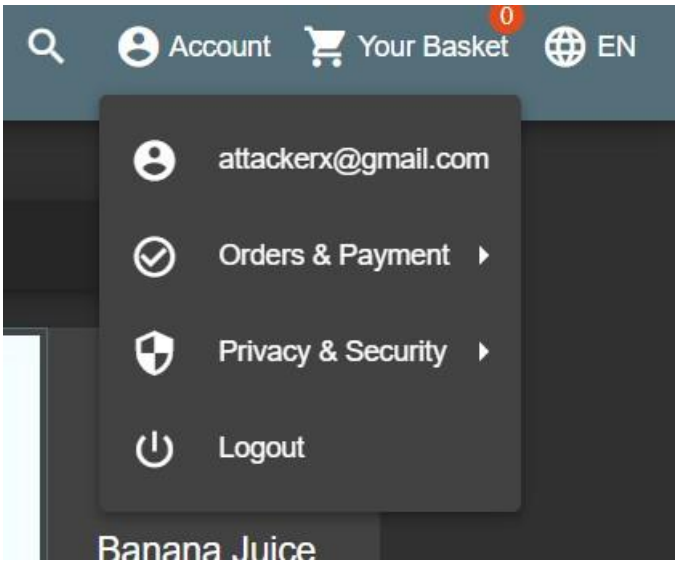


```
YDuOh4t9IZiMtQceCls1F2fxHwhZI4T8muKVterID1F37hleI71F19SwotJBcjbux2cXgilLU6gIzOIQe
Content-Length: 250
Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
Accept: application/json, text/plain, */*
Sec-Ch-Ua-Platform: "Windows"
X-User-Email: hesham11@gmail.com
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Content-Type: application/json
Origin: https://juice-shop.herokuapp.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://juice-shop.herokuapp.com/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Priority: u=1, i
Connection: keep-alive

{
  "email":"attackerx@gmail.com",
  "password":"123",
  "passwordRepeat":"123",
  "securityQuestion":{
    "id":2,
    "question":"Mother's maiden name?",
    "createdAt":"2024-08-10T18:05:10.265Z",
    "updatedAt":"2024-08-10T18:05:10.265Z"
  },
  "securityAnswer":"name"
}
```

Search 0 highlights

5- Email Registered.



Impact

By bypassing the client-side validation, an attacker can create accounts with weak passwords, which increases the risk of those accounts being compromised. If attackers gain access to theseaccounts, they could potentially escalate their privileges, access sensitive information, or perform unauthorized actions within the application. Additionally, it undermines the integrity of the security controls in place, leading to a false sense of security for users and administrators.

Recommended Fix

1) **Enforce Server-Side Validation:** Ensure that password requirements (length, complexity,etc.) are validated on the server-side.
2) **Harden Client-Side Validation:** Retain client-side validation for user experience but donot rely on it for security.
3) **Implement Strong Password Policies:** Require strong passwords server-side, includinglength and complexity.

(M005) Improper Security Questions Allow for Account Takeover

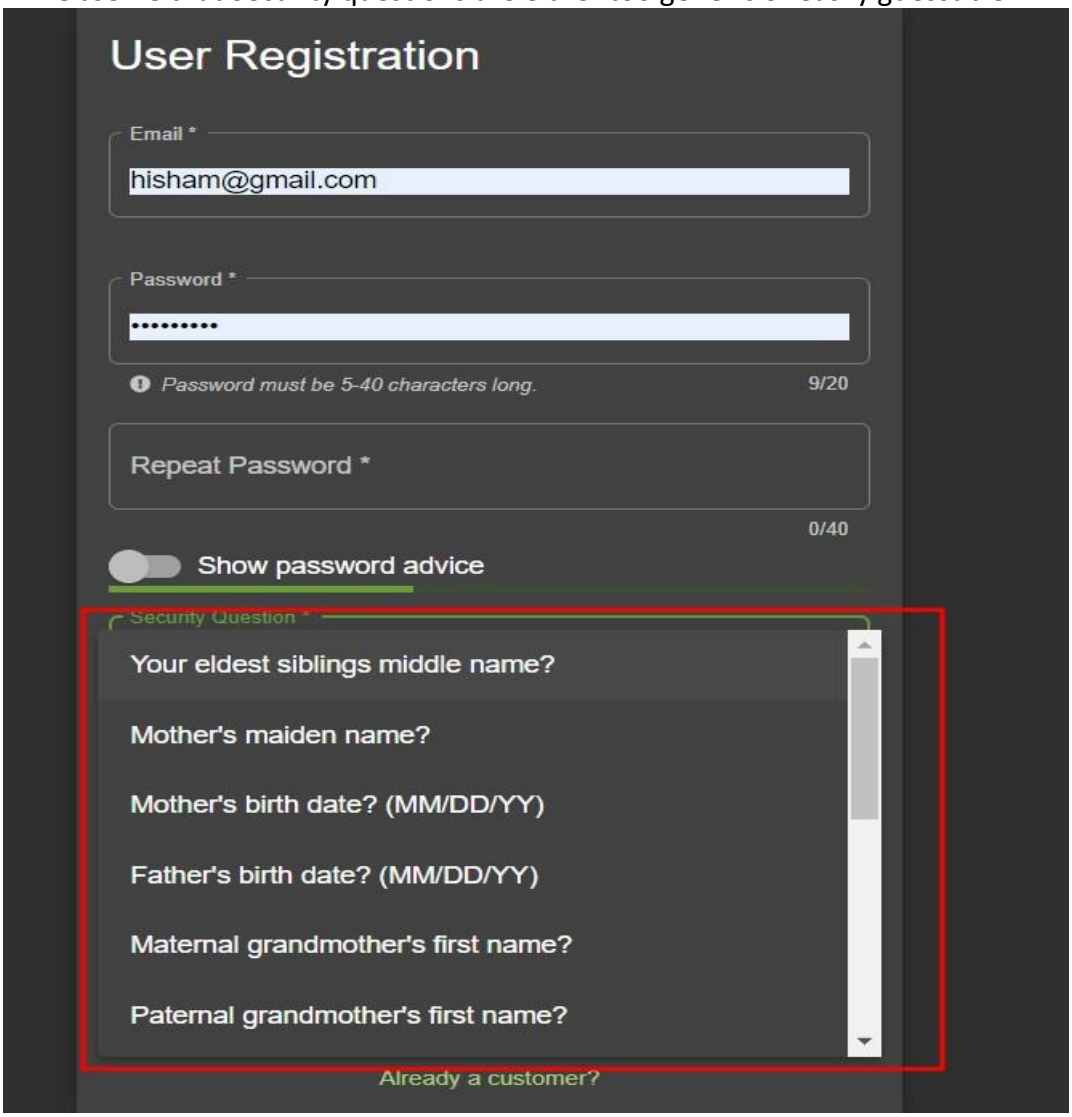
Weakness Type: Weak Password Recovery Mechanism

Description

The application employs security questions as a method for password recovery. However, thesesecurity questions are either too generic or easily guessable, allowing attackers to bypass authentication mechanisms by answering these questions correctly. Examples of weak questions include "What is your mother's maiden name?" or "What is your first pet's name?" which can often be discovered through social engineering or public information sources.

Steps to Reproduce:

1- Go to Registration page https://juice-shop.herokuapp.com/#/register
2- Observe that Security questions are either too generic or easily guessable



Impact

An attacker with minimal information about the user can successfully answer the security questions and gain unauthorized access to the user's account. This can lead to a full account takeover, exposing sensitive user information and allowing malicious activities to be performedunder the user's identity.

Recommended Fix

1. **Implement Multi-Factor Authentication (MFA):** Require MFA for account recovery.
2. **Use Stronger Security Questions:** Use questions that are not easily guessable or publiclyavailable.
3. **Limit Attempts:** Restrict the number of attempts for answering security questions.

(M006) Weak Reset Password Implementation withWeak Security Question

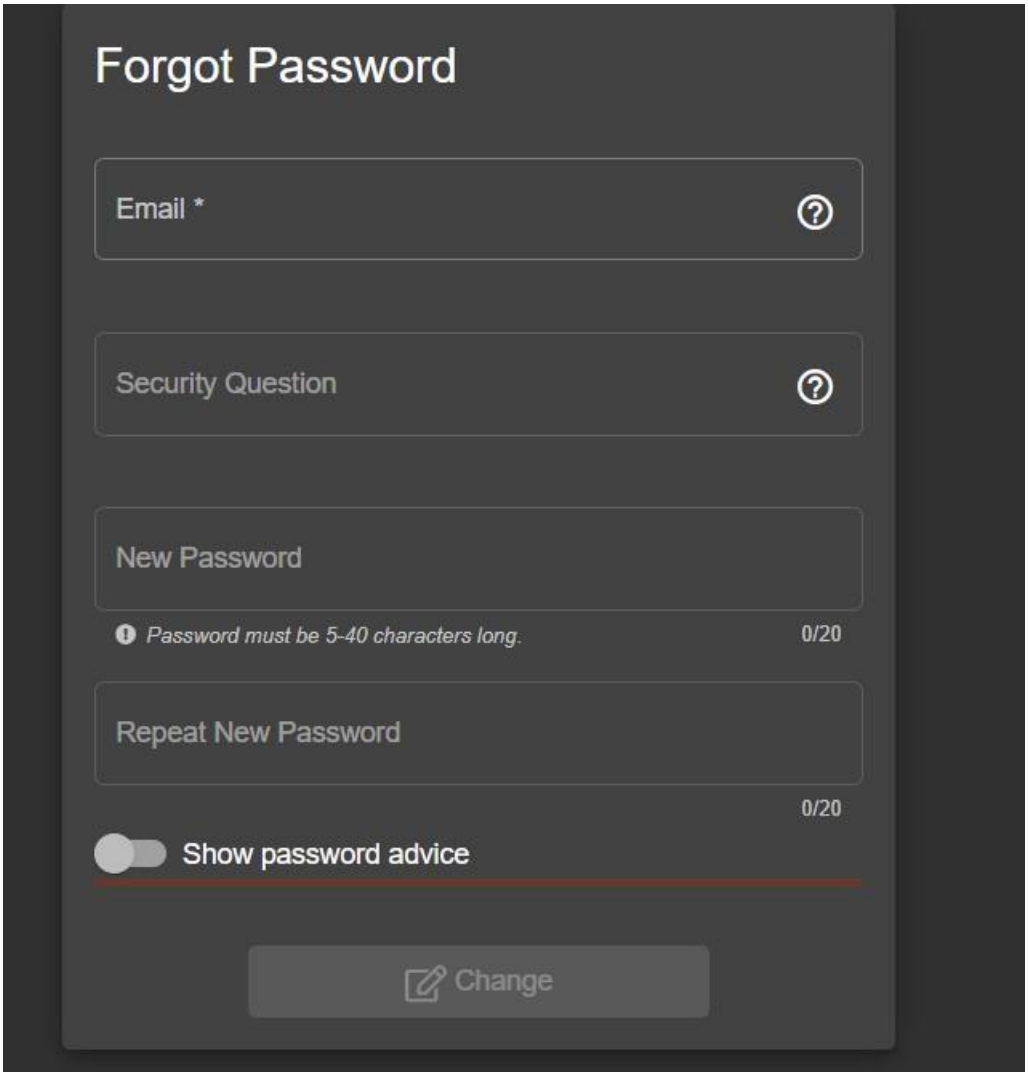
Weakness Type: Weak Password Recovery Mechanism forForgotten Password

Description

The application's password reset functionality allows users to reset their password by answering a security question. However, the security question used is weak and can be easilyguessed or obtained through social engineering, leading to unauthorized account access.

Steps to Reproduce:

1- Go to https://juice-shop.herokuapp.com/#/forgot-password and observing thatif user need to reset his password, he will should answer security question only.



Impact

An attacker who knows or guesses the answer to the security question can reset the victim's password and gain unauthorized access to the account. This can lead to exposure of sensitiveinformation, unauthorized actions on behalf of the user, and potential account takeover.

Recommended Fix

1. **Strengthen Security Questions**: Use security questions that are not easily guessable.
2. **Implement Additional Verification**: Require an additional verification step, such as acode sent to the user's email or phone number.
3. **Rate Limiting**: Limit the number of attempts to answer the security question to preventbrute-force attacks.

Low Risk Issues [L]:-

(L001) GET Method Used in Change Password FunctionWeakness

Type: Sensitive Data Exposure

Description

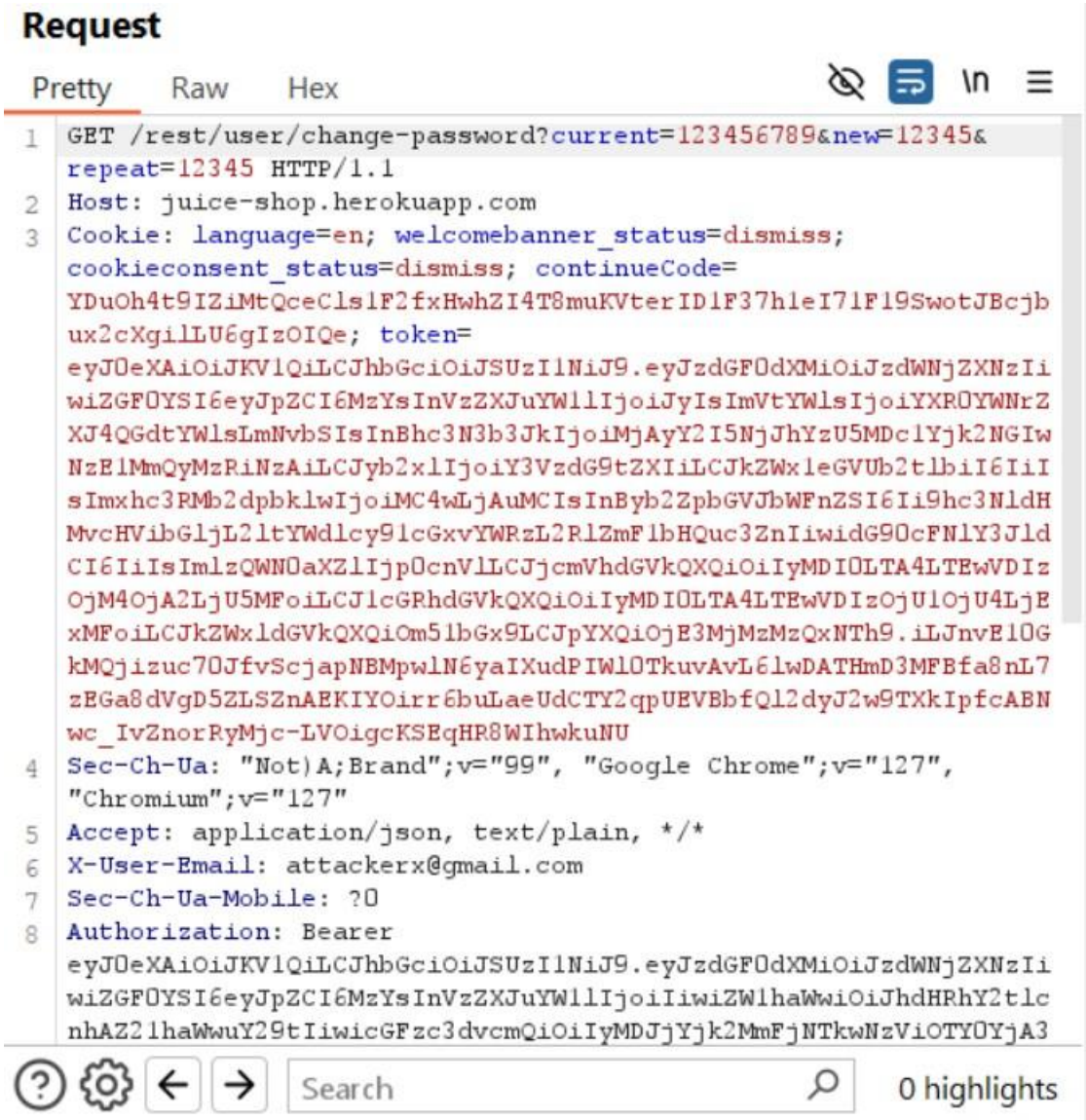
The GET method is used to handle password change requests, which can expose sensitive datain URLs. Using GET requests for sensitive operations like password changes is problematic because URLs can be logged in various places (browser history, server logs, etc.), potentiallyexposing sensitive information.

Steps to Reproduce:

1- Go to (https://juice-shop.herokuapp.com/#/privacy-security/change-password)

2- 2- Change your old password and go to history.

3- You will find that the request sent with GET method.



Impact

1. **Data Exposure:** The password change operation may expose sensitive information, suchas tokens or parameters, in browser history or server logs.
2. **Replay Attacks:** If the request URL is intercepted, an attacker could potentially reuse theURL to change the password if the same parameters are still valid.
3. **Insecure Logging:** The URL with sensitive data might be logged by web servers orproxies, leading to unintentional exposure of sensitive information.

Recommended Fix

1. **Use POST Method:** Switch the password change operation to use the POST method insteadof GET. This will ensure that sensitive information is included in the request body and not exposed in the URL.
2. **Implement CSRF Protection:** Ensure that Cross-Site Request Forgery (CSRF) protection isimplemented to prevent unauthorized password changes**.**
3. **Secure Logging Practices:** Review and secure logging practices to ensure that sensitiveinformation is not logged in plain text.
4. **Encrypt Communication:** Use HTTPS to encrypt all communication between the client andserver to protect against data interception.

(L002) Weak Password Policy in Change PasswordFunction

Weakness Type: Weak Password

RequirementsDescription

The application allows users to change their passwords through a "Change Password" function. However, the function does not enforce a strong password policy. This weak password policy permits users to set easily guessable or insecure passwords, such as "password123" or "123456". Such passwords are vulnerable to brute-force attacks and can be easily compromisedby attackers, leading to unauthorized access to user accounts.

Steps to Reproduce:

1) **Go to https://juice-shop.herokuapp.com/#/privacy-security/change-password .**
2) **Change password to 12345.**

Impact

Weak password policies significantly increase the risk of unauthorized access to user accounts.If an attacker can successfully guess or brute-force a user's password, they can gain full accessto the account, potentially leading to data theft, account takeover, or further exploitation of the application.

Recommended Fix

Implement a strong password policy in the "Change Password" function. The policy should include the following requirements:

- Minimum password length (e.g., at least 12 characters).
- Inclusion of both uppercase and lowercase letters.
- Inclusion of at least one numerical digit.
- Inclusion of at least one special character (e.g., @, #, $, etc.).
- Prohibition of common passwords (e.g., "password123", "123456", etc.).
- Enforce password history to prevent reuse of previous passwords.

(L003) Email Enumeration via Post Comment FeatureWeakness Type:

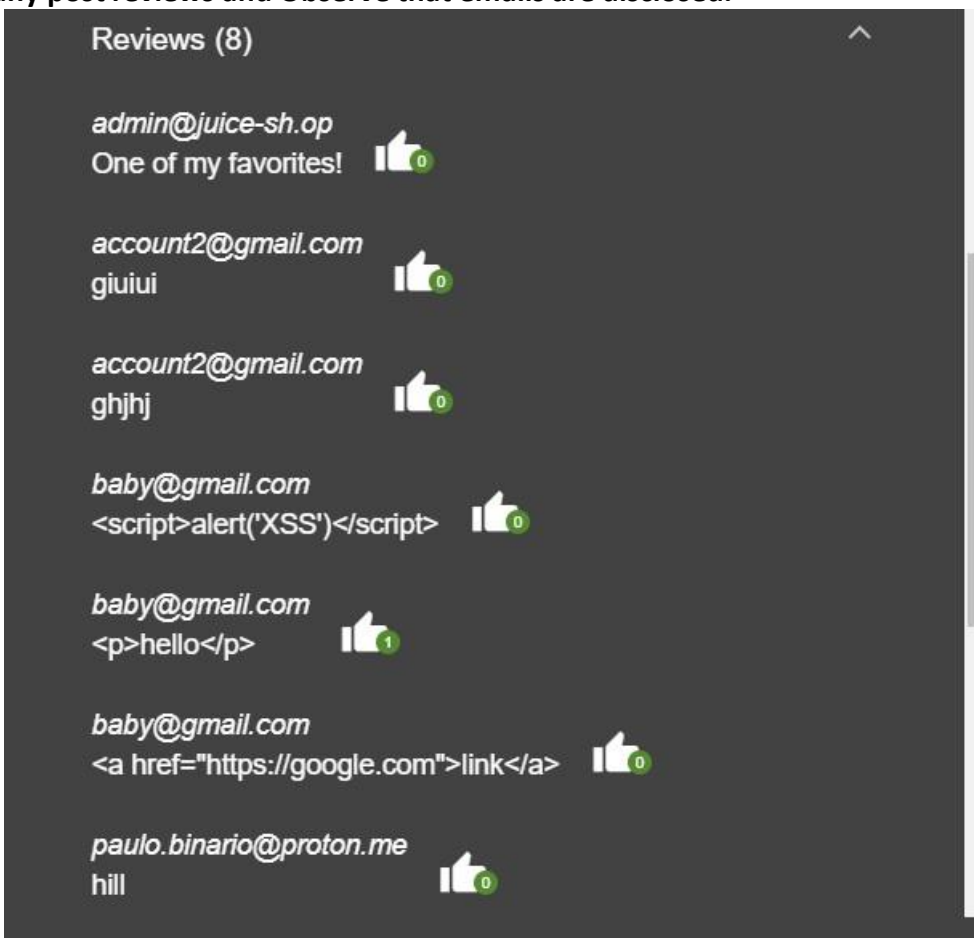
Information Disclosure

Description

The post comment feature on the website discloses registered email addresses is in the system.An attacker can use this to enumerate valid email addresses associated with user accounts on the platform.

Steps to Reproduce:

1. **Go to any post reviews and Observe that emails are disclosed.**



Impact

This vulnerability allows an attacker to enumerate valid email addresses, which can then be used for further attacks such as targeted phishing, social engineering, or brute-force attacks. The exposure of email addresses can compromise the privacy of the users and potentially leadto account takeovers if exploited in combination with other vulnerabilities.

Recommended Fix

Comments should be with names not email