SECURITY ASSESSMENT

Juice Shop Team 2 Report

Submitted to : The Development Team

Security Analyst: Team2:

1- Ahmed Tarek Elbahgy
2- Mazen Ahmed Mahmoud
3- Ahmed Mohamed Samir
4- Yehya Shapan Mahmoud
5- Shahd Ahmed Goda

Date Of Testing : 10-10-2024

Date of Report Delivery: 24-10-2024

Table of Contents

Contents

Security Engagement Summary

Engagement Overview

- Who requested the engagement and why?
    - The Development Team that is responsible for the organization's web-applications has requested a vulnerability assessment of a legacy web-application ,to help them understand what security risk the web-application is posing to the organization, and what mitigations are possible to increase the security posture and reduce the risk to the organization
- What are the engagement's goals?
    - Detect security weaknesses in the application that could be exploited by attackers, and Evaluate the potential impact of identified vulnerabilities on the application and the organization.
- Who is complete the engagement?
    - Team 2 :
        - Ahmed Tarek Elbahgy
        - Mazen Ahmed Mahmoud
        - Ahmed Mohamed Samir
        - Yehya Shapan Mahmoud
        - Shahd Ahmed Goda

Scope

The scope of this engagement was the OWASP Juice Shop's web application, specifically targeting its functionalities, and authentication mechanisms.

The allowed scope for this engagement was the following:

OWASP Juice Shop: http://localhost/

The testing team was not provided accounts for testing

Executive Risk Analysis

The vulnerabilities discovered have been categorized based on their potential impact and exploitability:

- **High Risk**: These vulnerabilities present a critical threat to the application, allowing unauthorized access, data breaches, or full system compromise.
- **Medium Risk**: These vulnerabilities could be exploited by attackers with some effort and may lead to security breaches under specific conditions.
- **Low Risk**: These vulnerabilities are less likely to be exploited or have limited impact on the application.

Executive Recommendation

- Implement strong input validation, particularly in sensitive areas like login pages and feedback forms.
- Enforce strict role-based access control (RBAC) to prevent unauthorized access to administrative sections.
- Regularly update and patch web applications to address known vulnerabilities.

Significant Vulnerability Summary

Vulnerability	Severity
SQL Injection in Login Page	Critical
Admin registration (Privilege escalation)	Critical
Access to the Administration Section (Broken Access Control)	Critical
Captcha Bypass in Feedback Submission	High
Payback Time (Place an Order that Makes You Rich)	High
Broken Access Control (View Basket)	High
Upload Type (Improper Input Validation)	High
DOM XSS	High
Manipulate Basket (Broken Access Control)	High
Improper Input Validation (Zero Stars)	Medium
Broken Access Control (Five-Star Feedback)	Medium
Security Misconfiguration (Error Handling)	Medium
Privacy Policy Auto-Solve (Information Disclosure)	Medium
Upload Size (Improper Input Validation)	Medium
Forged Feedback	Medium
Information Disclosure	Medium
Repetitive Registration (Improper Input Validation)	Medium
Negative quantity value (Improper Input Validation)	Medium

Significant Vulnerability Detail

1. SQL Injection in Login Page

Description:

Login page injection is a type of injection attack that exploits vulnerabilities in a web application's login system. The OWASP Juice Shop's login page is vulnerable to SQL Injection, where an attacker can inject malicious SQL code to bypass authentication mechanisms.

Impact:

- **Authentication Bypass**: Attackers can use malicious inputs to gain unauthorized access to user accounts, including administrative accounts.
- **Data Exposure**: Attackers can retrieve sensitive data like usernames, passwords, and other personally identifiable information (PII).
- **System Compromise**: Once authenticated, attackers may further exploit the system or escalate privileges.

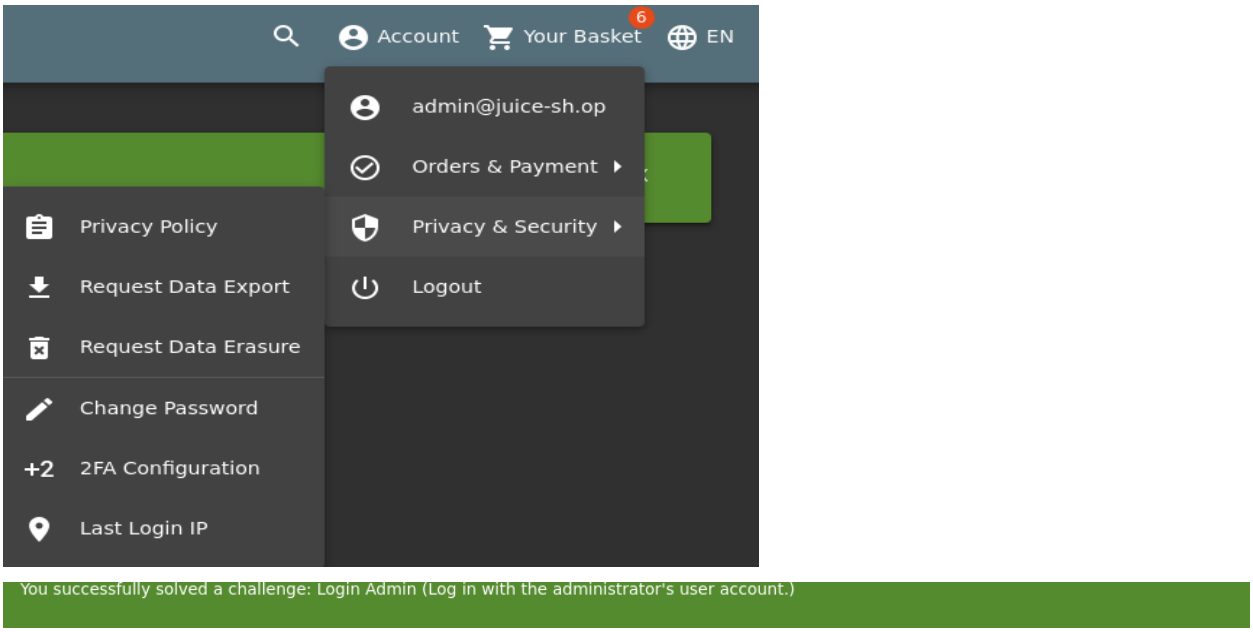
Exploitation Scenario:

An attacker might input admin' OR '1'='1 into the username and password fields, resulting in a successful login with admin credentials because the SQL query becomes always true. This can lead to unauthorized access to the system without proper credentials.

Mitigation:

- Implement parameterized queries or prepared statements to prevent SQL code injection.
- Use input validation and output encoding.
- Ensure error messages do not reveal sensitive information that could aid an attacker.

POC:



2. Captcha Bypass in Feedback Submission

Description:

CAPTCHA mechanisms are used to prevent automated bots from interacting with web forms, including feedback submission forms. However, the CAPTCHA implementation in OWASP Juice Shop is susceptible to bypass attacks due to weak or improperly validated mechanisms.

Impact:

- **Automated Spam**: Bots could flood the feedback system with fake or malicious submissions.
- **Denial of Service (DoS)**: Attackers could automate feedback submissions, overwhelming the server and leading to performance degradation or downtime.
- **Loss of Integrity**: Inaccurate or excessive feedback could affect the integrity of the feedback system.

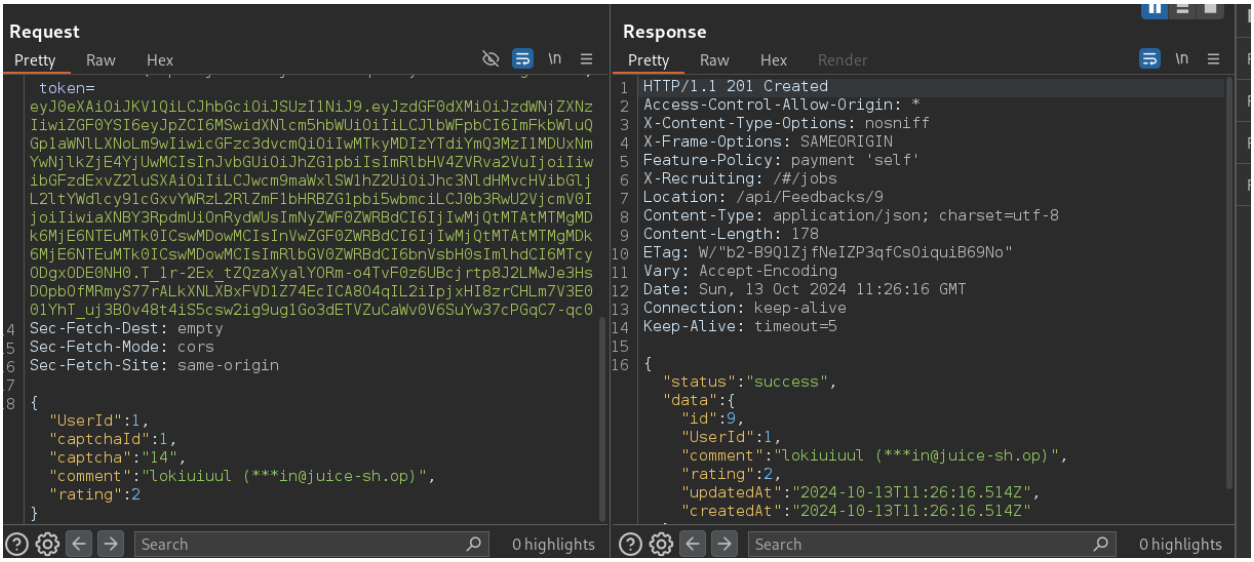
Exploitation Scenario:

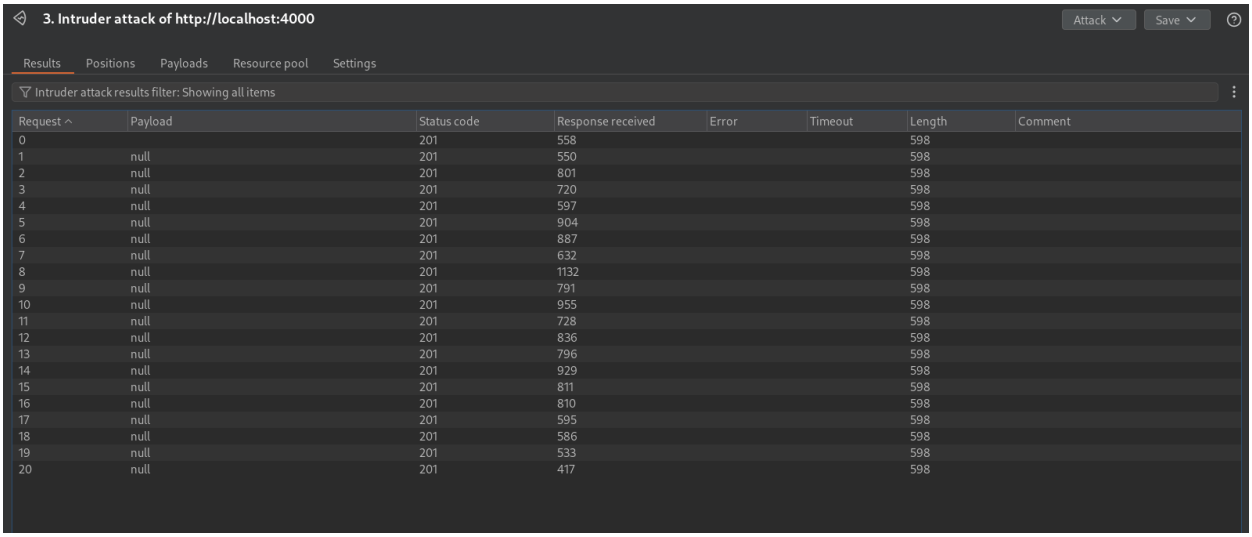
1. **Identify the Vulnerability**: The attacker discovers that the feedback submission form in the OWASP Juice Shop uses a CAPTCHA to prevent automated submissions. However, they notice that CAPTCHA validation might be weak or performed only on the client-side.
2. **Capture the Request**: The attacker submits a feedback form and captures the HTTP request using a tool like Burp Suite. The request contains the CAPTCHA token and other form fields such as name, email, and comment.
3. **Use Repeater for Automation**: The attacker sends the captured request to Burp Suite's **Repeater** module, where they modify. They then begin sending multiple requests with minimal changes to see if the server will accept one of them without CAPTCHA validation.
4. **Analyze Server Response**: After sending multiple requests, the attacker eventually receives a valid response (201), indicating that the feedback was accepted by the server. This response confirms that the CAPTCHA mechanism has been bypassed.
5. **Automate the Process**: The attacker could then script this process to automate submitting large volumes of fake feedback, overwhelming the system, and potentially causing a Denial of Service (DoS) or injecting harmful or malicious content.

Mitigation:

- Use more robust CAPTCHA systems like reCAPTCHA v3, which relies on behavior analysis.
- Ensure CAPTCHA validation occurs on both the client and server sides.
- Implement rate limiting for feedback submissions to reduce the risk of automation.

POC:





3. Access to the Administration Section (Broken Access Control)

Description:

The OWASP Juice Shop suffers from Broken Access Control in its administrative section. This vulnerability allows users who should not have access to administrative functionalities to gain entry, potentially leading to unauthorized system changes and data breaches.

Impact:

- **Privilege Escalation**: Users without administrative rights can access the administration panel, potentially leading to data modification, system configuration changes, and more.
- **Data Loss or Tampering**: Unauthorized users can alter or delete sensitive information within the application.
- **Complete System Takeover**: If administrative functionality is compromised, the entire system could be under the control of malicious actors.

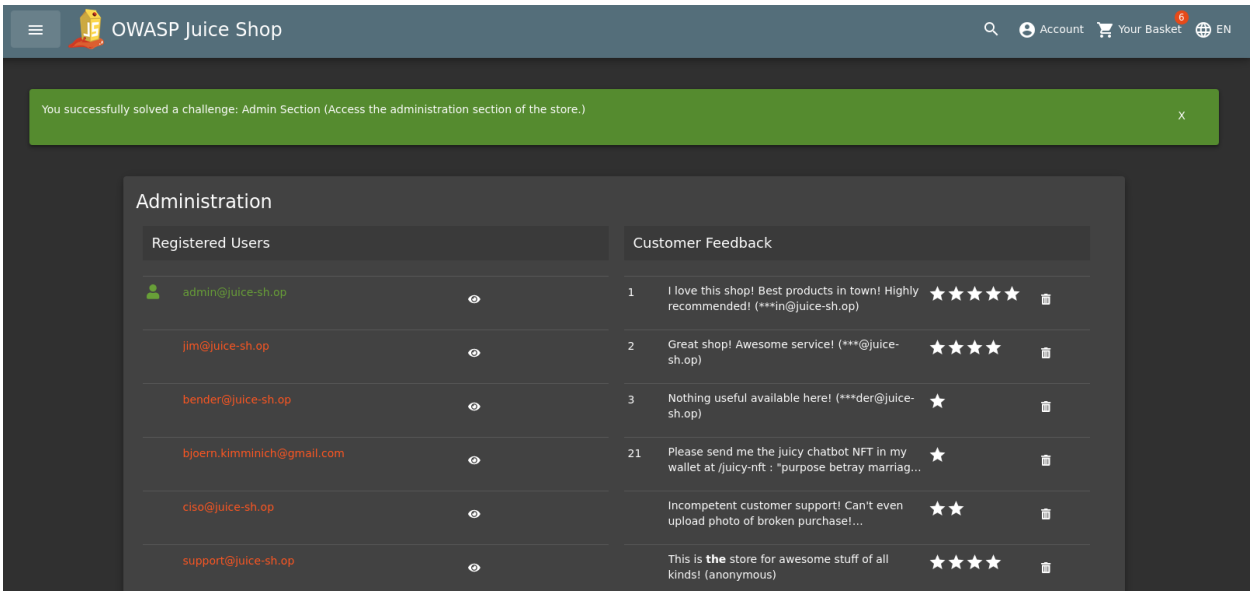
Exploitation Scenario:

When I logged in as an admin I opened the inspector and tried to find a hidden pages then i found ADMINISTRATION page

Mitigation:

- Enforce strict role-based access control (RBAC) policies.
- Implement secure session management and validation techniques.
- Continuously monitor access logs for any signs of unauthorized access attempts.

POC:



4. Discovery of Hidden Page - "Score-board"

Description:

The OWASP Juice Shop contains a hidden page called **"Score-board"**, which serves as a dashboard for tracking progress in the security challenges offered by the application. Although this page is not linked in the main navigation or accessible through standard user interaction, it can still be accessed through specific means, indicating a vulnerability known as **Unprotected or Hidden Resource Exposure**.

Steps Taken by the Attacker:

1. **Analyze Web Application**: The attacker explores the OWASP Juice Shop application by navigating through all visible pages and analyzing the structure of the URLs. The attacker observes that most internal URLs follow a common pattern (e.g., /#/login, /#/register, etc.).
2. **Guessing the URL**: Since many web applications name their pages and routes using human-readable identifiers, the attacker guesses that the **Score-board** page might be hidden under a simple URL such as /#/score-board.
3. **Direct URL Access**: The attacker manually inputs http://<domain_name>/#/score-board in the browser's address bar, trying to directly access the hidden page.

4. **Successful Discovery**: Upon entering the guessed URL, the **"Score-board"** page successfully loads, revealing the attacker's progress in solving various challenges. The attacker now has access to information not intended for casual users without having solved other challenges first.

Impact:

While the "Score-board" page is not necessarily dangerous in itself, accessing it prematurely exposes information about the application's security challenges. This can lead to:

- **Challenge Spoiling:** The attacker gains an advantage by knowing the list of challenges and their descriptions without having discovered them naturally through solving earlier challenges.
- **Exposure of Metadata:** The "Score-board" page may contain metadata or identifiers that could help an attacker infer other hidden elements within the application.
- **User Experience Manipulation:** If an attacker were to manipulate or abuse this page, it could spoil the learning experience for legitimate users aiming to discover and solve the challenges on their own.

4. Mitigation:

1. **Authentication and Authorization Controls:**
    - Ensure that pages intended to be hidden or restricted (such as the "Score-board") require appropriate user authentication and authorization before access.
    - Implement role-based access control (RBAC) to ensure that only authorized users can view certain resources.
2. **Obfuscation of URLs:**
    - Avoid using predictable URL patterns for sensitive or hidden resources. Instead, use random or unique identifiers for internal pages to make URL guessing attacks less effective.
3. **Robust Input Validation:**
    - Monitor and validate incoming requests for direct access to hidden or sensitive URLs. Implement logging and alerting mechanisms to detect when unauthorized access attempts are made.
4. **Disable Directory Listing and Prevent Exposure of Site Map:**
    - Disable directory listing on the server to prevent attackers from discovering hidden pages by exploring unlinked resources.
    - Ensure that the robots.txt file does not reveal sensitive or hidden pages that should remain undiscoverable.

POC:

5. Upload Size (Improper Input Validation)

Description:

The OWASP Juice Shop application allows users to upload files as part of its functionality. However, the application does not properly validate the size of the uploaded files. This vulnerability can be exploited by an attacker to upload files larger t han the intended limit, potentially leading to various security issues such as server resource exhaustion, denial of service (DoS) attacks, or even arbitrary code execution.

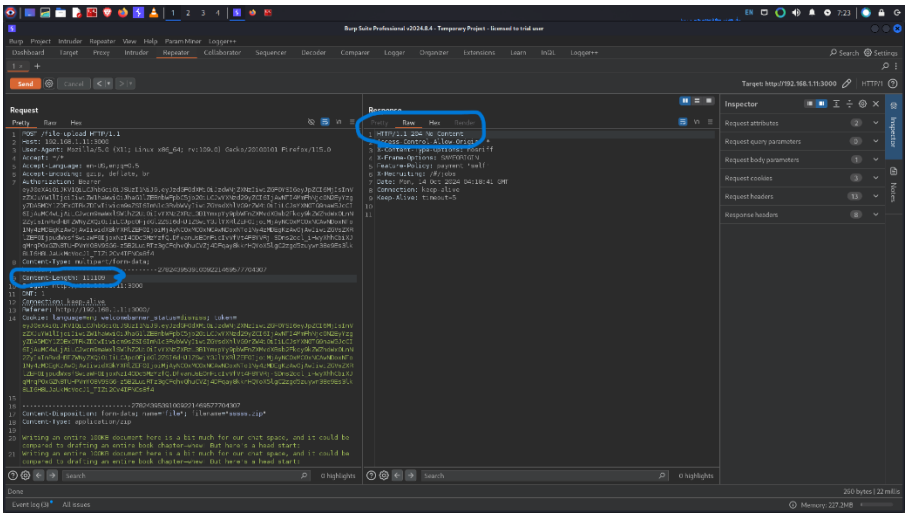
Impact:

- **Server Resource Exhaustion:** Large file uploads can consume excessive server resources, leading to degraded performance or server crashes.
- **Denial of Service (DoS):** Attackers can intentionally upload large files to overwhelm the server, causing service d isruptions for legitimate users.
- **Arbitrary Code Execution:** In some cases, improper input validation can be exploited to execute arbitrary code o n the server, leading to complete compromise of the application.

Mitigation:

- **Implement Proper Input Validation:** Ensure that the application validates the size of uploaded files before proce ssing them. Set a strict limit on the maximum allowed file size and reject any files that exceed this limit.

POC:



6. Payback Time (Place an Order that Makes You Rich)

Description:

The OWASP Juice Shop application includes a feature called "Payback Time," which allows users to place orders. Howev er, the application does not properly validate the input for order amounts in this feature. This vulnerability can be exploited by an attacker to input excessively large or negative values for orders, potentially leading to various security issues such as financial manipulation, denial of service (DoS) attacks.

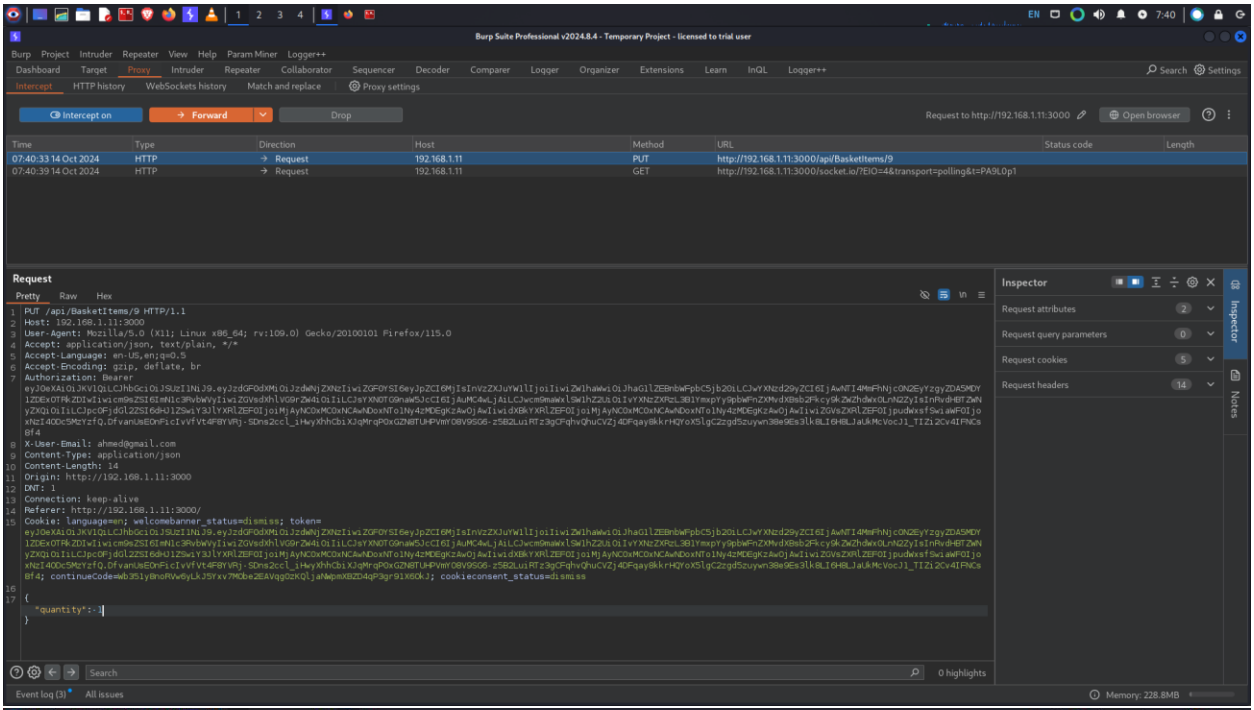
Impact:

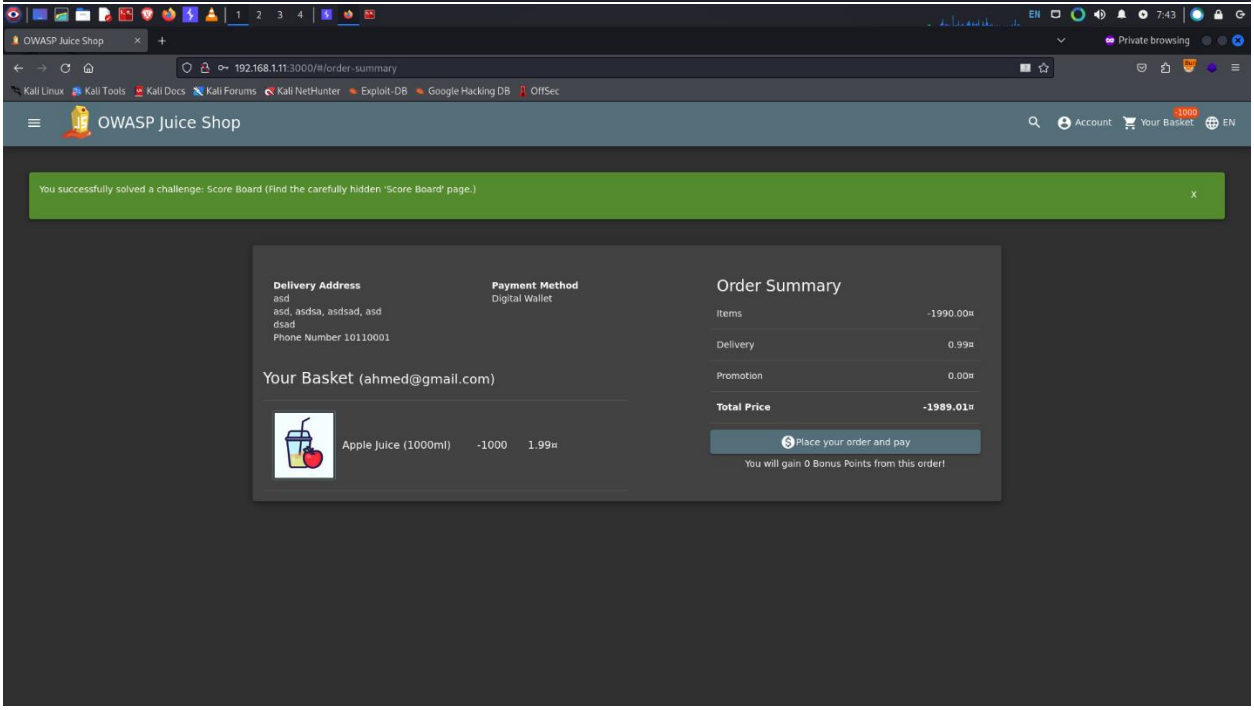
- **Financial Manipulation:** Attackers can exploit this vulnerability to place orders with negative or extremely large v alues, resulting in incorrect financial transactions or gaining unauthorized funds.
- **Denial of Service (DoS):** Large or negative order amounts can consume excessive server resources, leading to degraded performance or server crashes.

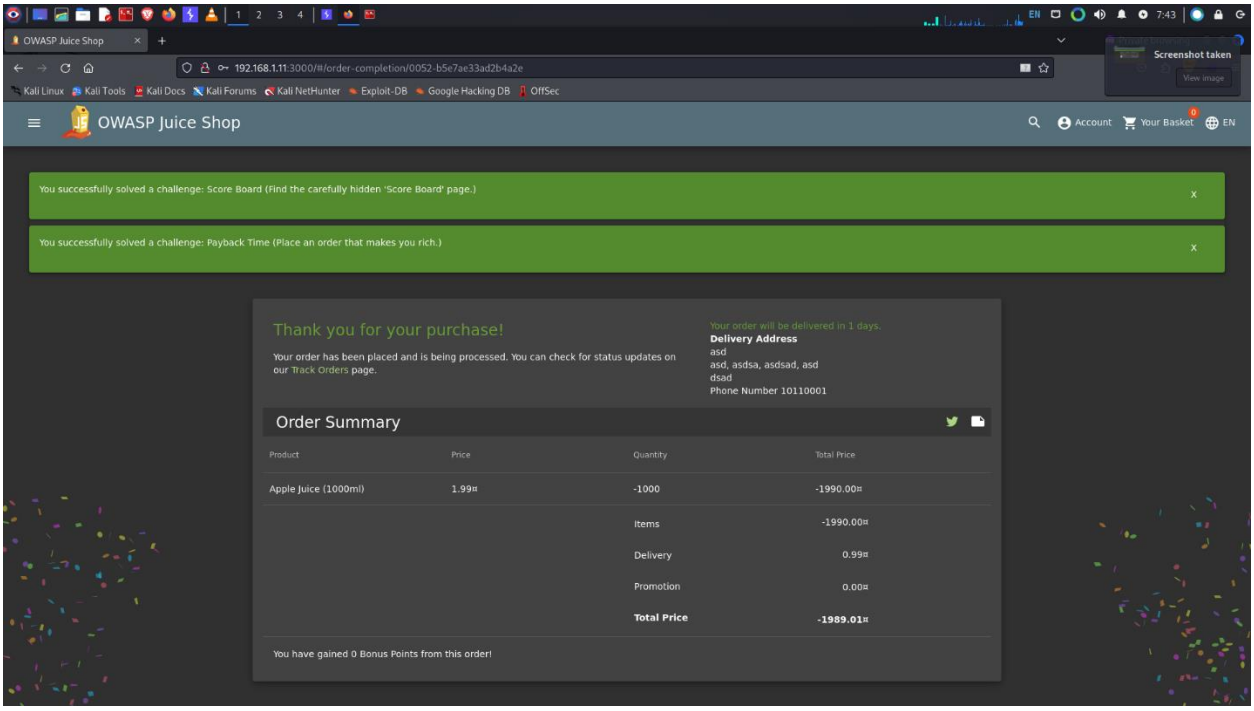
Mitigation:

- **Implement Proper Input Validation:** Ensure that the application validates the input for order amounts before pro cessing them. Set strict limits on the acceptable range of values and reject any inputs that fall outside this range.

POC:


Burp Suite Professional v2024.8.4 - Temporary Project - licensed to trial user
Request
Inspector
Request attributes
Request query parameters
Request cookies
Request headers
PUT /api/BasketItems/9 HTTP/1.1
Host: 192.168.1.11:3000
{
"quantity":-3
}


OWASP Juice Shop
192.168.1.11:3000/#/order-summary
You successfully solved a challenge: Score Board (Find the carefully hidden 'Score Board' page.)
Delivery Address
asd
asd, asdsa, asdsad, asd
dsad
Phone Number 10110001
Payment Method
Digital Wallet
Order Summary
Items -1990.00¤
Delivery 0.99¤
Promotion 0.00¤
Total Price -1989.01¤
Your Basket (ahmed@gmail.com)
Apple Juice (1000ml) -1000 1.99¤
Place your order and pay
You will gain 0 Bonus Points from this order!



7.Forged Feedback

Description:

The OWASP Juice Shop application includes a feedback feature that allows users to leave reviews and comments. Howe ver, the application does not properly validate the authenticity of the feedback submissions. This vulnerability can be explo ited by an attacker to submit forged feedback, potentially leading to various security issues such as misleading other user s, manipulating the application's reputation, or even injecting malicious content.

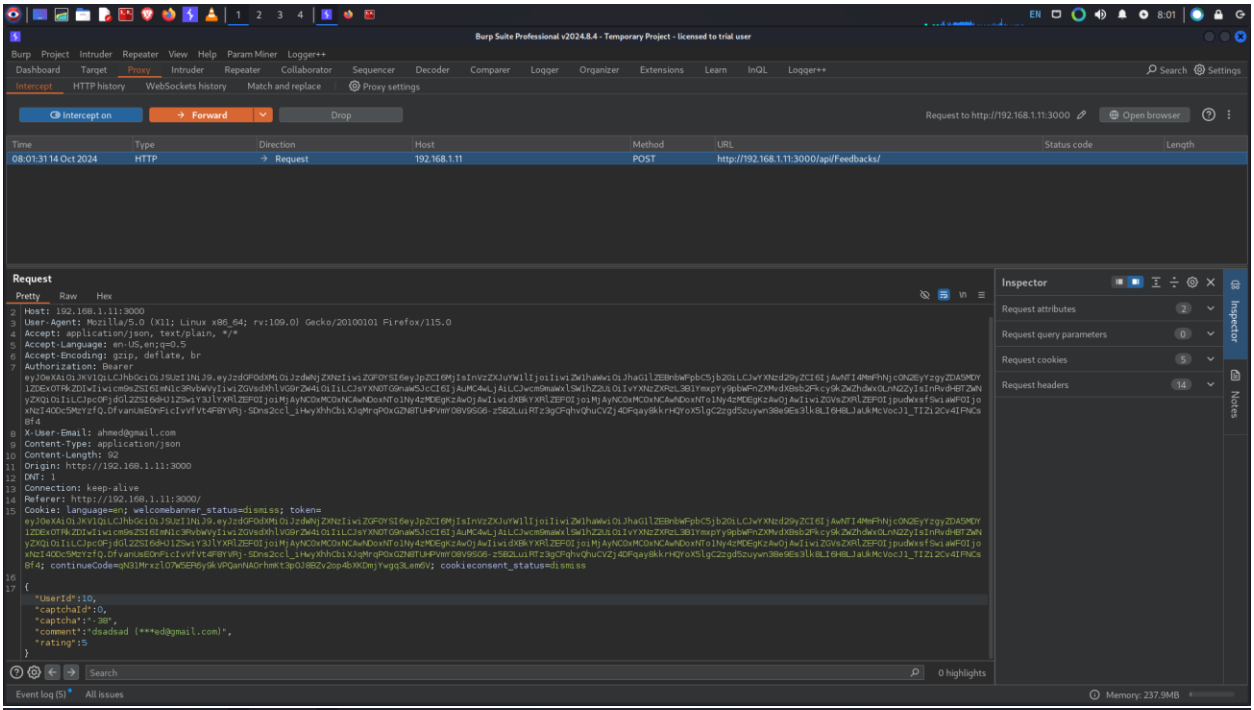
Impact:

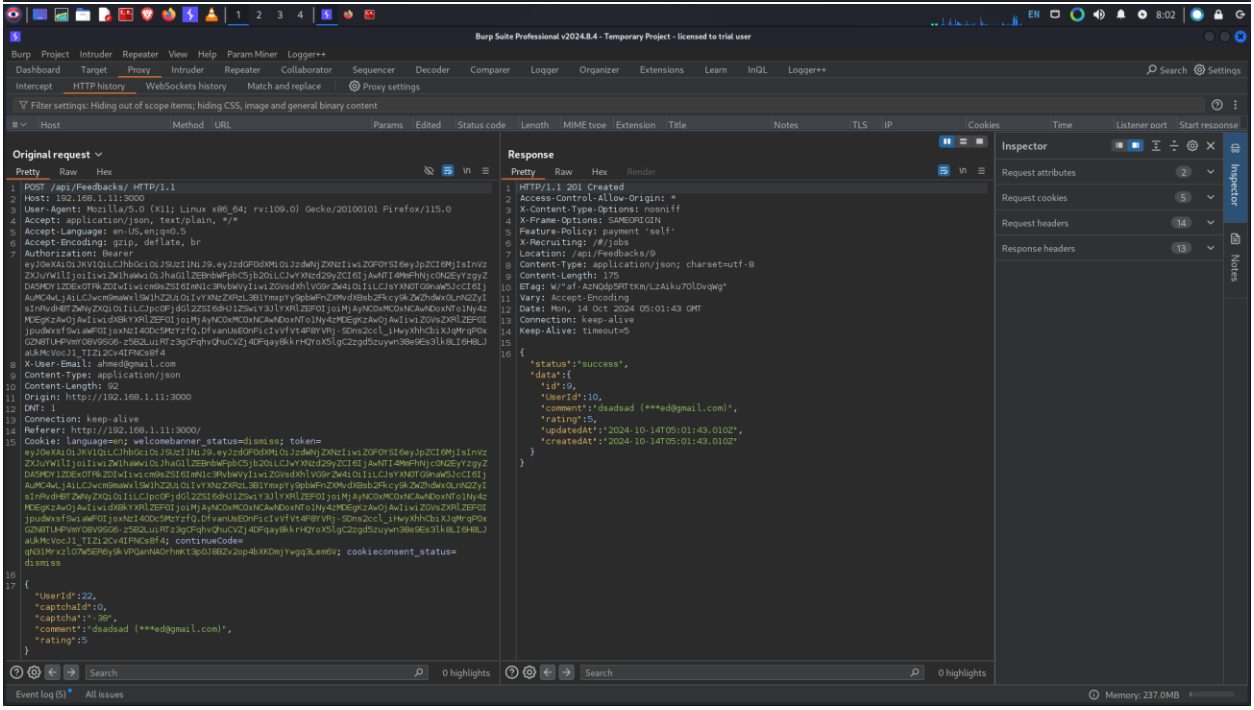
- **Misleading Information:** Attackers can submit fake reviews or comments, misleading other users about the quali ty and reliability of the products and services offered by the application.
- **Reputation Manipulation:**
  Forged feedback can be used to manipulate the application's reputation, either positively or negatively, affecting it s overall trustworthiness.

Mitigation:

- **Implement Proper Input Validation:** Ensure that the application validates the authenticity of feedback submissio ns. Use techniques such as user authentication

POC:


Burp Suite Professional v2024.8.4 - Temporary Project - licensed to trial user
Intercept on
Forward
Drop
Request to http://192.168.1.11:3000
Open browser
Time
Type
Direction
Host
Method
URL
Status code
Length
08:01:31 14 Oct 2024
HTTP
Request
192.168.1.11
POST
http://192.168.1.11:3000/api/Feedbacks/
Request
Pretty
Raw
Hex
Inspector
Request attributes
Request query parameters
Request cookies
Request headers
Notes
Search
0 highlights
Event log (5)
All issues
Memory: 237.9MB


Burp Suite Professional v2024.8.4 - Temporary Project - licensed to trial user
Filter settings: Hiding out of scope items; hiding CSS, image and general binary content
Original request
Response
Pretty
Raw
Hex
Render
Inspector
Request attributes
Request cookies
Request headers
Response headers
Notes
Search
0 highlights
Event log (5)
All issues
Memory: 237.0MB

8. Information Disclosure

Description:

The OWASP Juice Shop application contains a JavaScript file named main.js that includes references to various paths wit hin the application. However, this file inadvertently exposes the path to the administration panel, which is typically restricte d to authorized users only. This vulnerability can be exploited by an attacker to gain unauthorized access to the admin pa nel, potentially leading to various security issues.

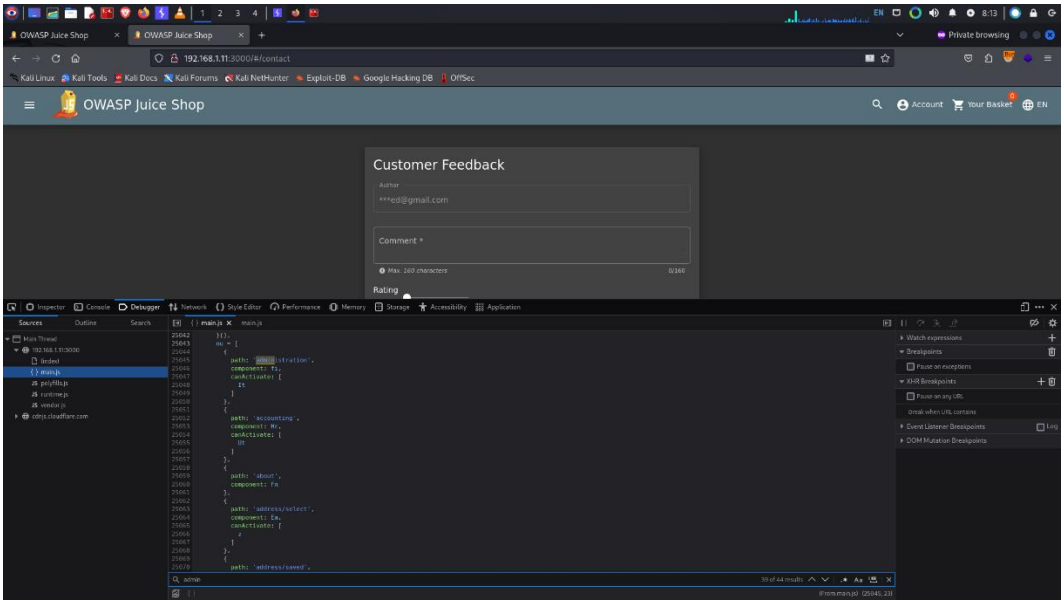
Impact:

- **Unauthorized Access:** Attackers can use the exposed admin path to gain access to the administration panel wit hout proper authentication, allowing them to perform administrative actions.
- **Data Manipulation:**
  Once inside the admin panel, attackers can modify or delete sensitive data, leading to potential data breaches or l oss of data integrity.

Mitigation:

- **Remove Path References from JavaScript Files: :** Ensure that sensitive paths, such as those to the admin pan el, are not included in JavaScript files or other client-side code. Use server-
  side logic to handle path references securely.
- **Implement Proper Access Controls:**
  Ensure that access to the admin panel is properly restricted and requires strong authentication mechanisms. Use techniques such as multi-factor authentication (MFA) and role-
  based access controls (RBAC) to enhance security.

POC:



9. Broken Access Control (View Basket)

Description:
This issue happens when users can see or change things they shouldn't be allowed to. In the View Baske example, someone might be able to view or mess with another person's shopping basket when they shouldn't have access to it.

Impact:

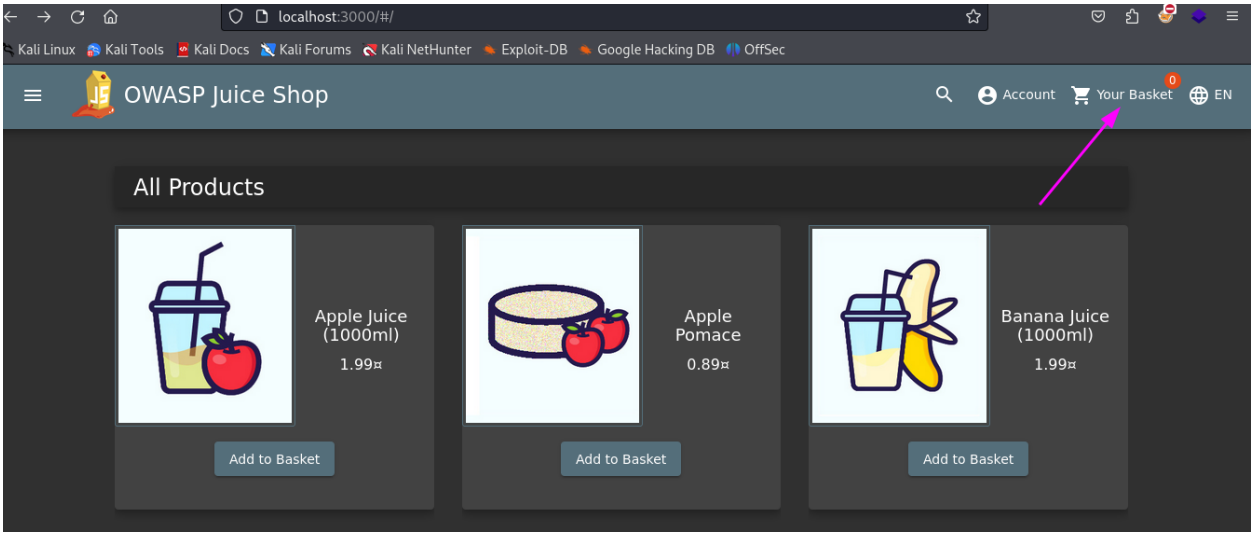
- **Sensitive information, like what other people are buying could be exposed.**

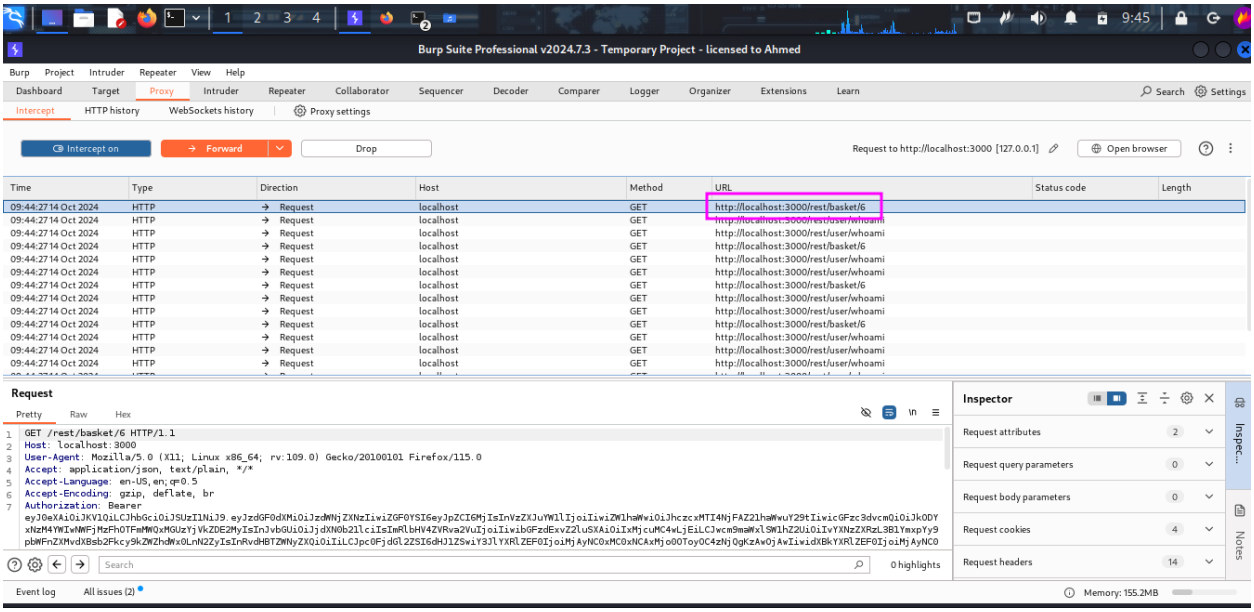
- **Hackers could change order details or prices.**
- **It might lead to more serious issues if they can get control over higher level accounts.**

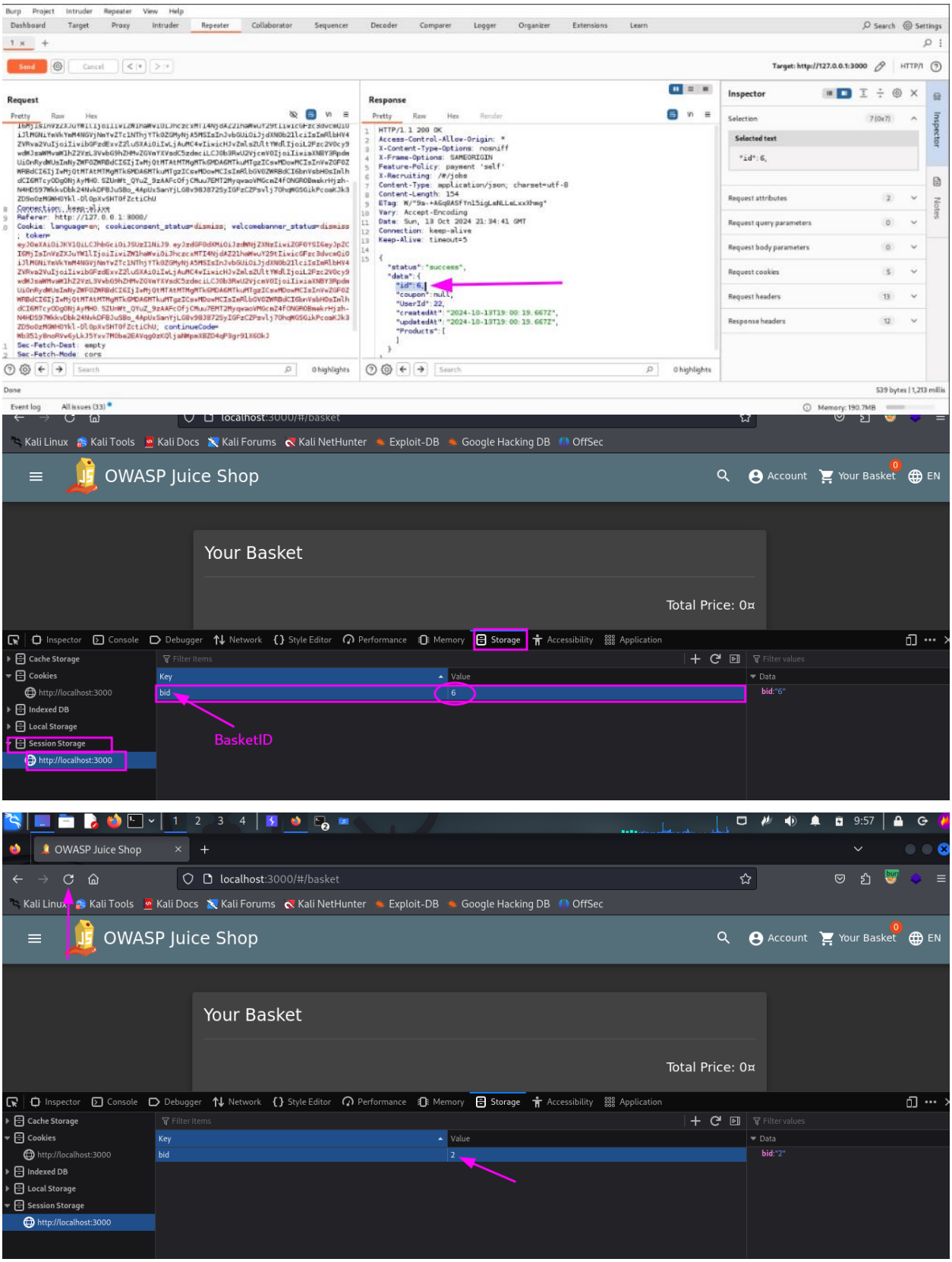
Mitigation:

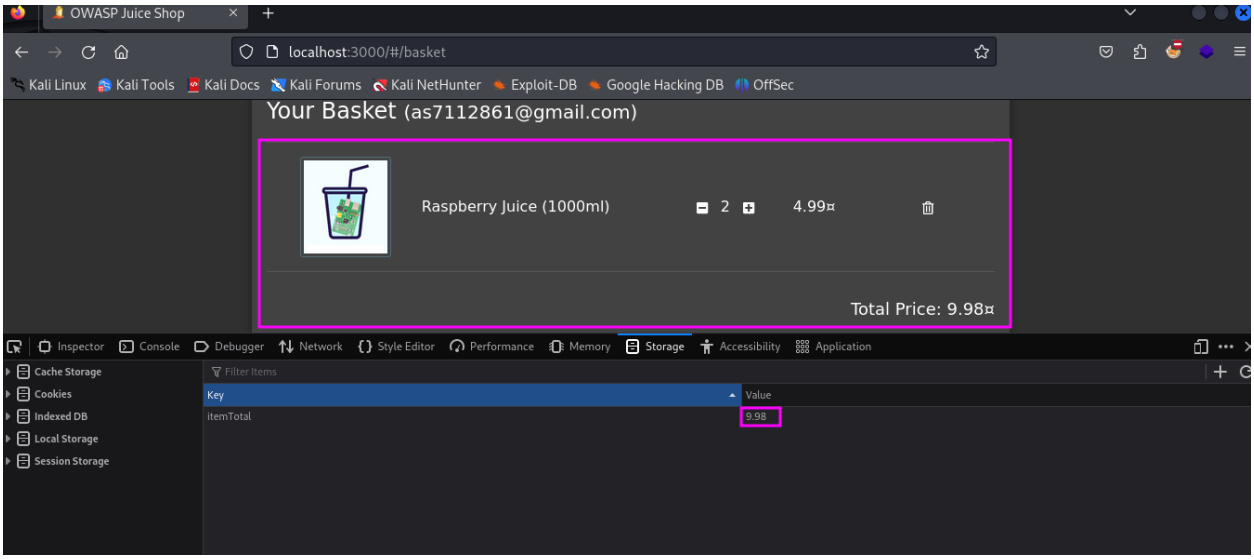
- **Make sure that only the right users can access their own baskets by checking permissions both on the website and on the server.**
- **Regularly use security tools to test for weak spots in access control.**

POC:






Request
Response
Inspector
Your Basket
Total Price: 0¤
BasketID
OWASP Juice Shop
Account
Your Basket
EN
Storage
Cookies
Session Storage
Local Storage
Indexed DB
Cache Storage
http://localhost:3000
bid
6
2



10. Broken Access Control (Five-Star Feedback)

Description:

This issue lets admins delete five star feedback, attackers can take advantage of this by logging as an admin by any way like SQL injection in juice shop and removing positive reviews which can impact the rating system.

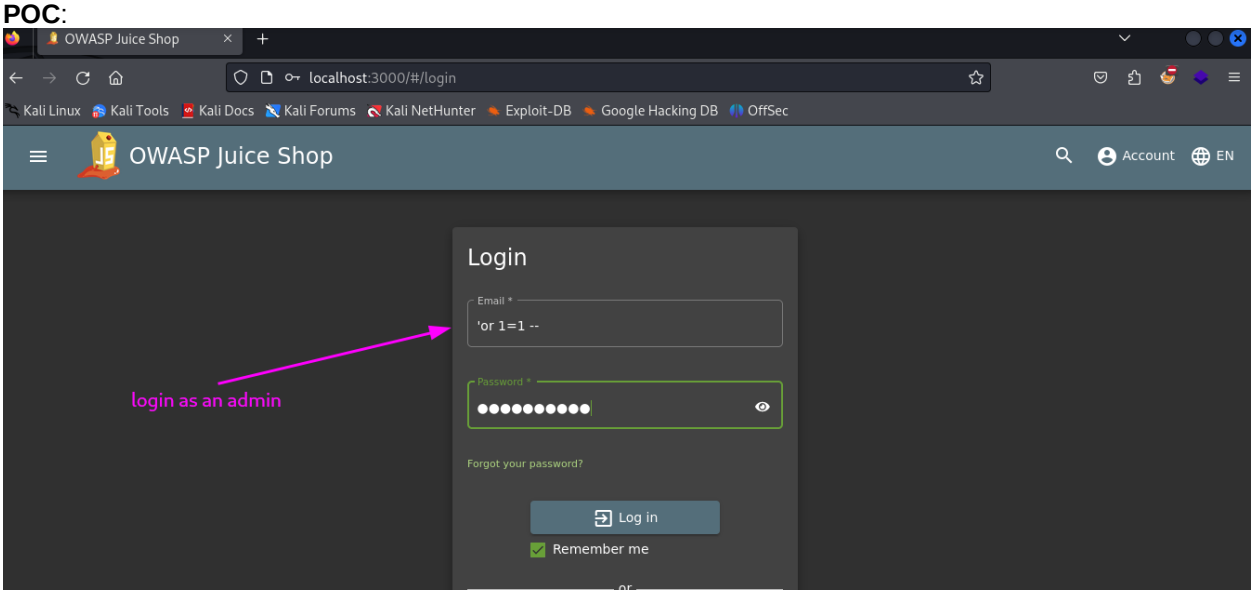
Impact:

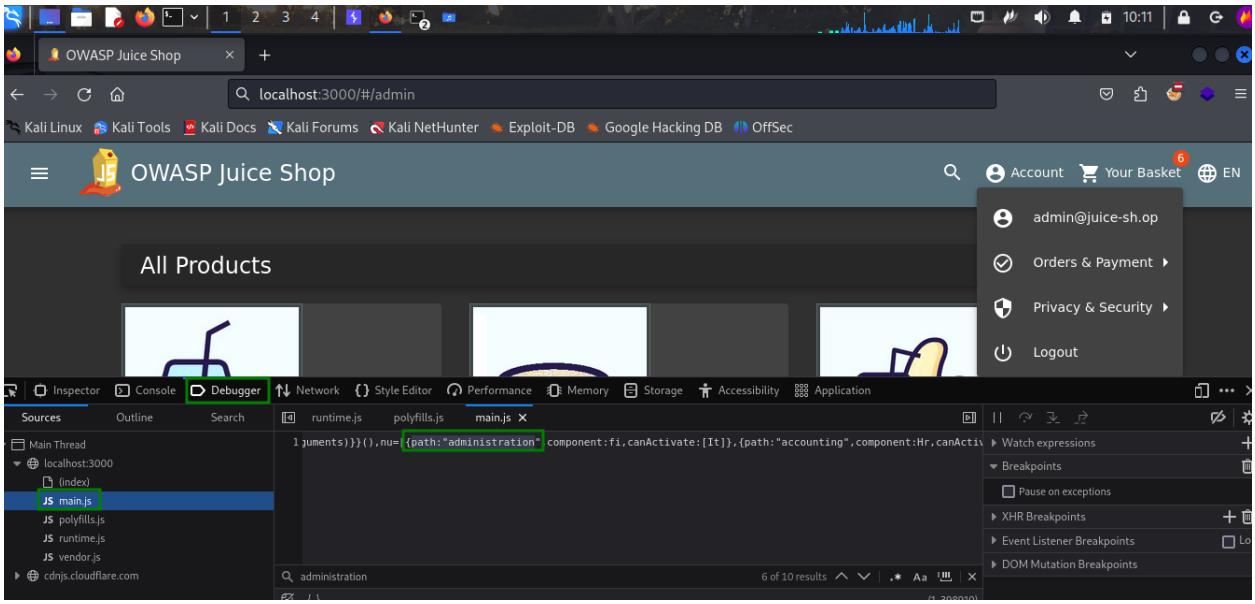
- **Genuine positive reviews can be deleted, damaging the reputation of products or services.**
- **Bad actors can manipulate ratings, making competitors look bad or skewing reviews to their advantage.**
- **The platform's trustworthiness is weakened as users start to lose confidence in the feedback system.**

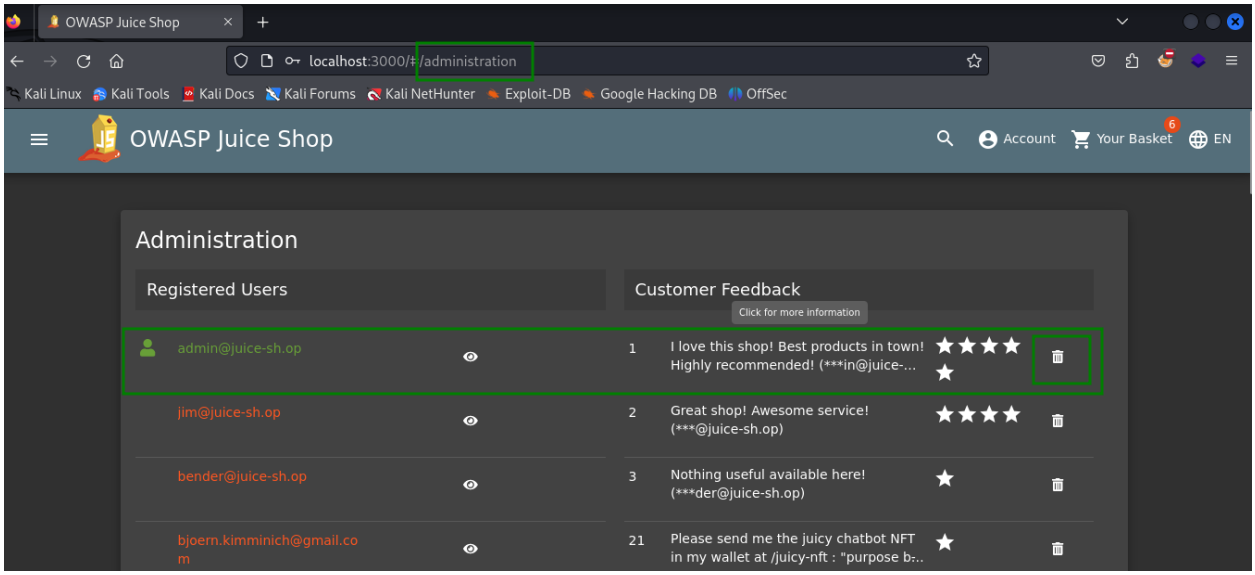
Mitigation:

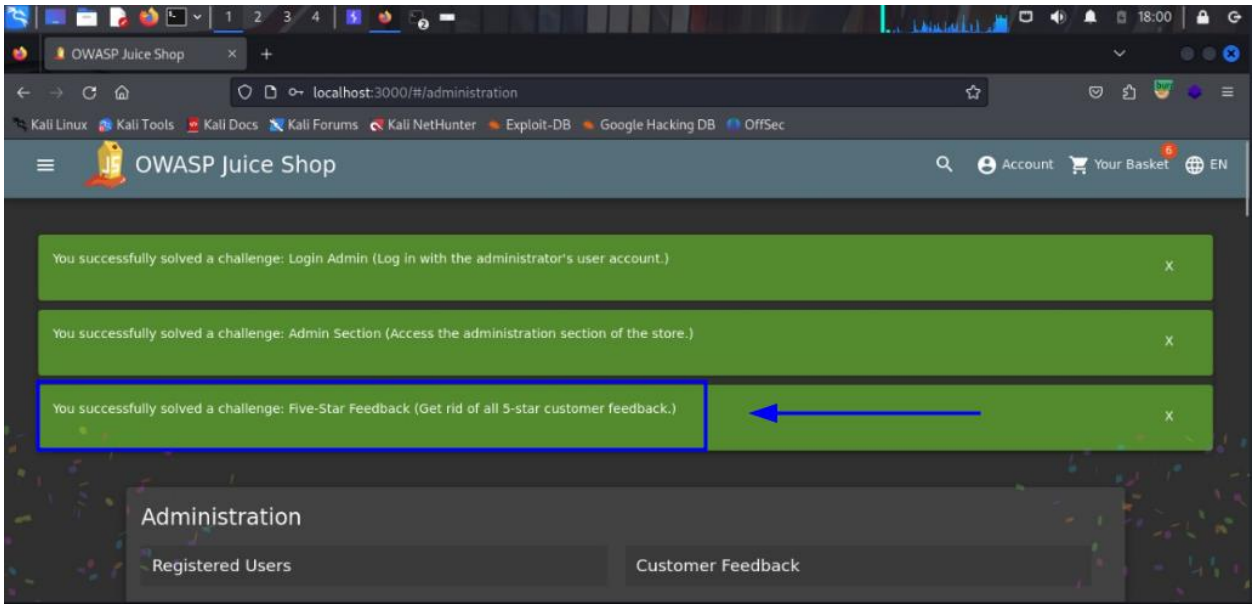
- **Keep track of feedback deletions by logging activities, so you can spot and prevent any misuse quickly.**
- **If you notice errors when trying to delete feedback, check the JavaScript console for any issues and fix them.**

POC:




OWASP Juice Shop
localhost:3000/#/admin
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
OWASP Juice Shop
Account
Your Basket
EN
admin@juice-sh.op
Orders & Payment
Privacy & Security
Logout
All Products
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application
Sources Outline Search
runtime.js polyfills.js main.js
Main Thread
localhost:3000
(index)
main.js
polyfills.js
runtime.js
vendor.js
cdnjs.cloudflare.com
Watch expressions
Breakpoints
Pause on exceptions
XHR Breakpoints
Event Listener Breakpoints
DOM Mutation Breakpoints
administration
6 of 10 results


OWASP Juice Shop
localhost:3000/#/administration
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
OWASP Juice Shop
Account
Your Basket
EN
Administration
Registered Users
Customer Feedback
Click for more information
admin@juice-sh.op
1
I love this shop! Best products in town! Highly recommended! (***in@juice-...
jim@juice-sh.op
2
Great shop! Awesome service! (***@juice-sh.op)
bender@juice-sh.op
3
Nothing useful available here! (***der@juice-sh.op)
bjoern.kimminich@gmail.com
21
Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose b...


OWASP Juice Shop
localhost:3000/#/administration
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
OWASP Juice Shop
Account
Your Basket
EN
You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)
You successfully solved a challenge: Admin Section (Access the administration section of the store.)
You successfully solved a challenge: Five-Star Feedback (Get rid of all 5-star customer feedback.)
Administration
Registered Users
Customer Feedback

11. Improper Input Validation (Zero Stars)

Description:
Improper input validation means the site is not correctly checking user input. For example, allowing a rating of “zero stars” when the system should only allow 1 to 5 stars. This can be exploited to break or manipulate the site.

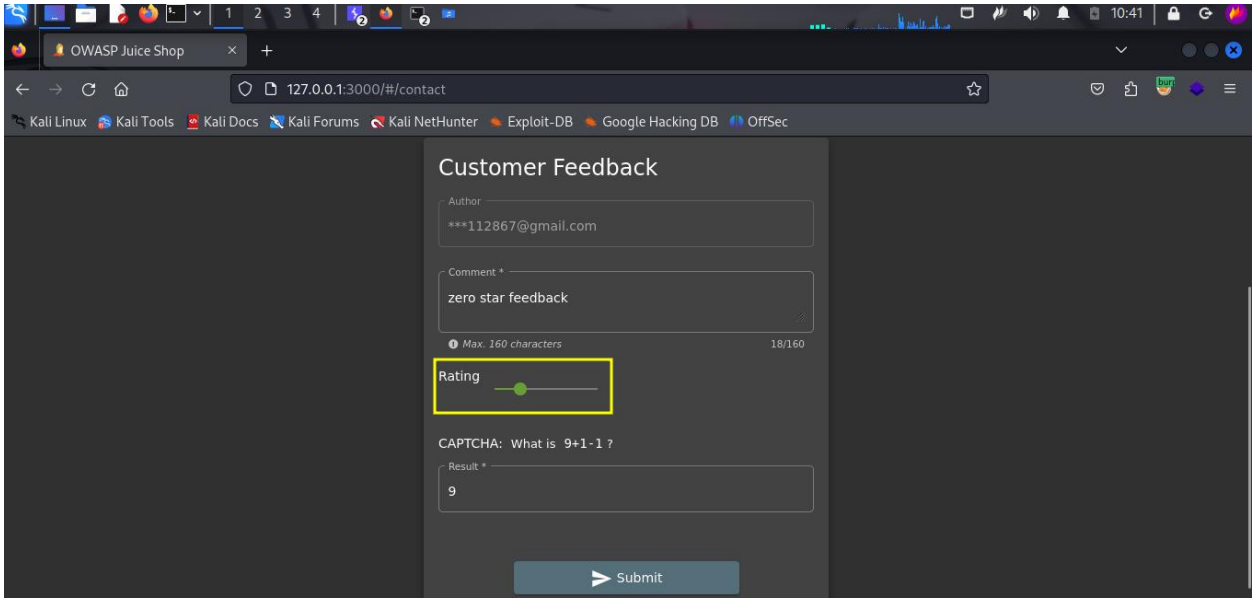
Impact:

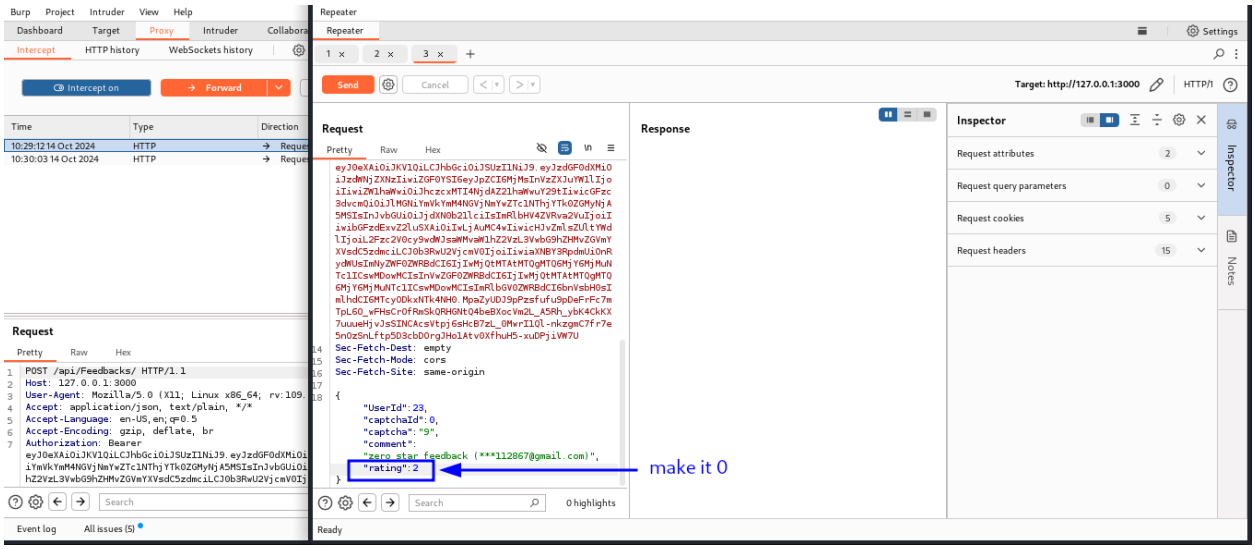
- **It can make the rating system unreliable, causing users to doubt the fairness of the reviews.**
- **Attackers might exploit this vulnerability to break other parts of the site.**

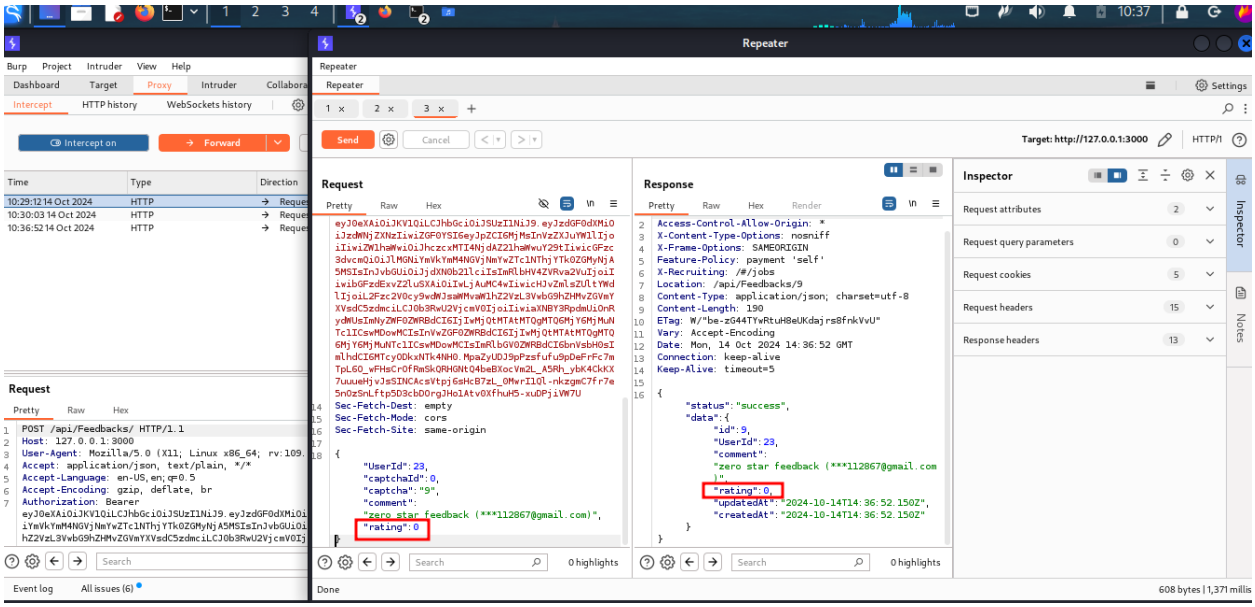
Mitigation:

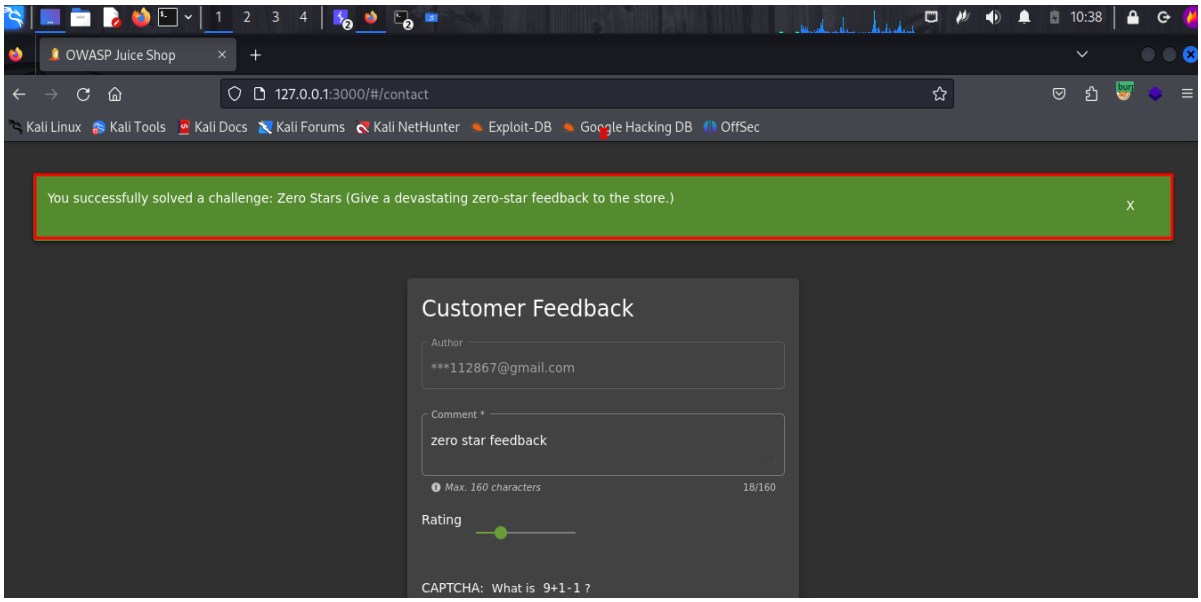
- **Ensure that the system checks input properly and only accepts valid ratings (1-5 stars).**
- **Sanitize all input data to filter out anything unexpected or malicious.**

POC:









12. DOM XSS

Description:
DOM-based XSS happens when malicious code gets injected into the site's JavaScript through user inputs. The browser then runs this bad code, which can allow an attacker to take over someone's session.

Impact:

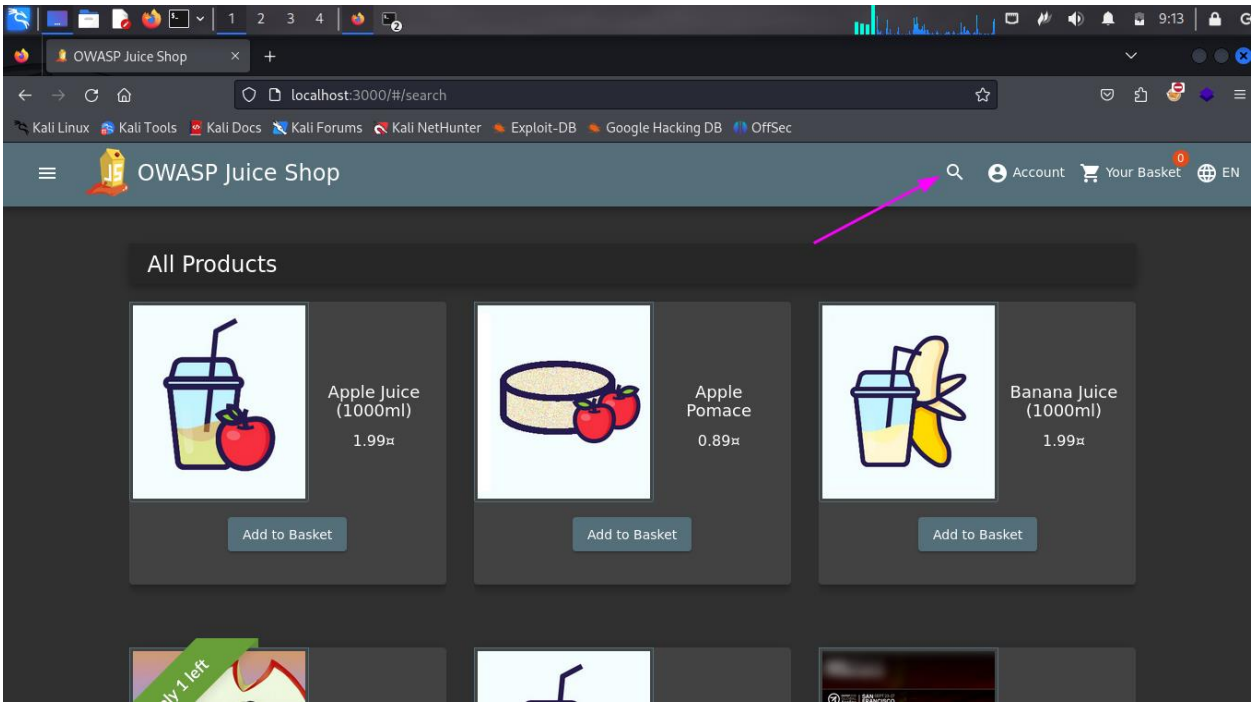
- **The attacker could steal session cookies and gain control over user accounts.**
- **They could redirect users to malicious websites or display fake content (phishing attacks).**
- **Sensitive data could be exposed.**

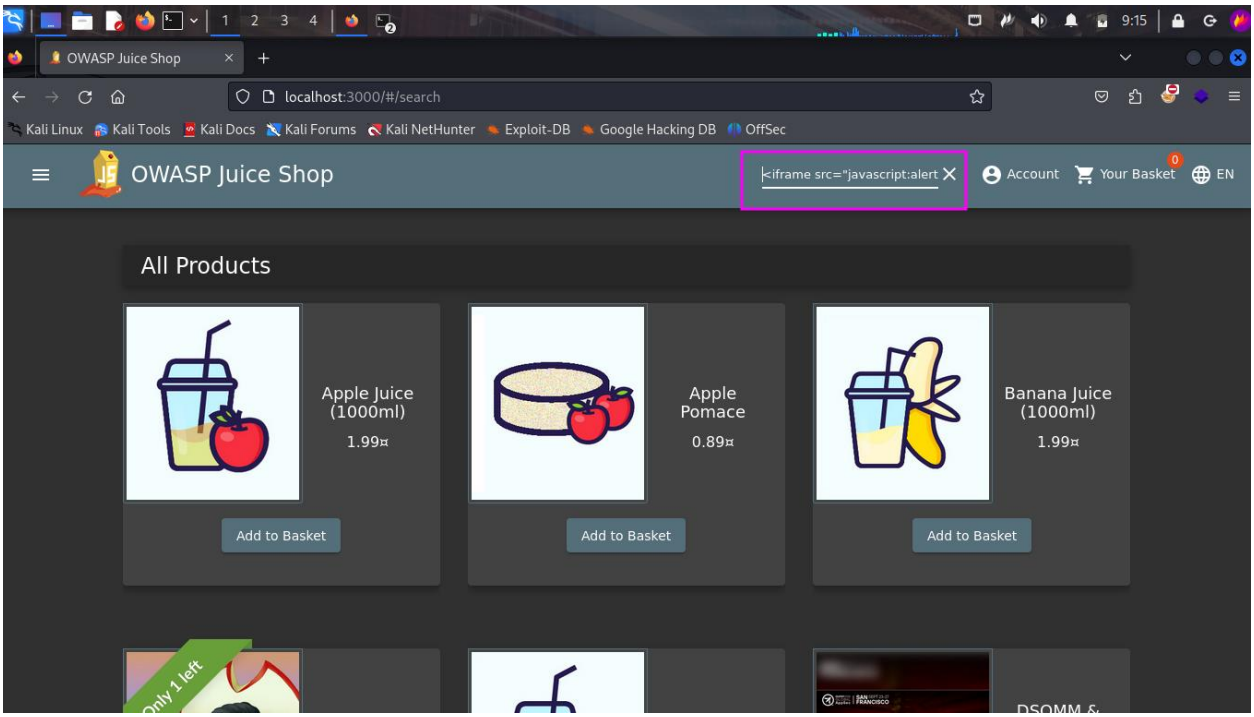
Mitigation:

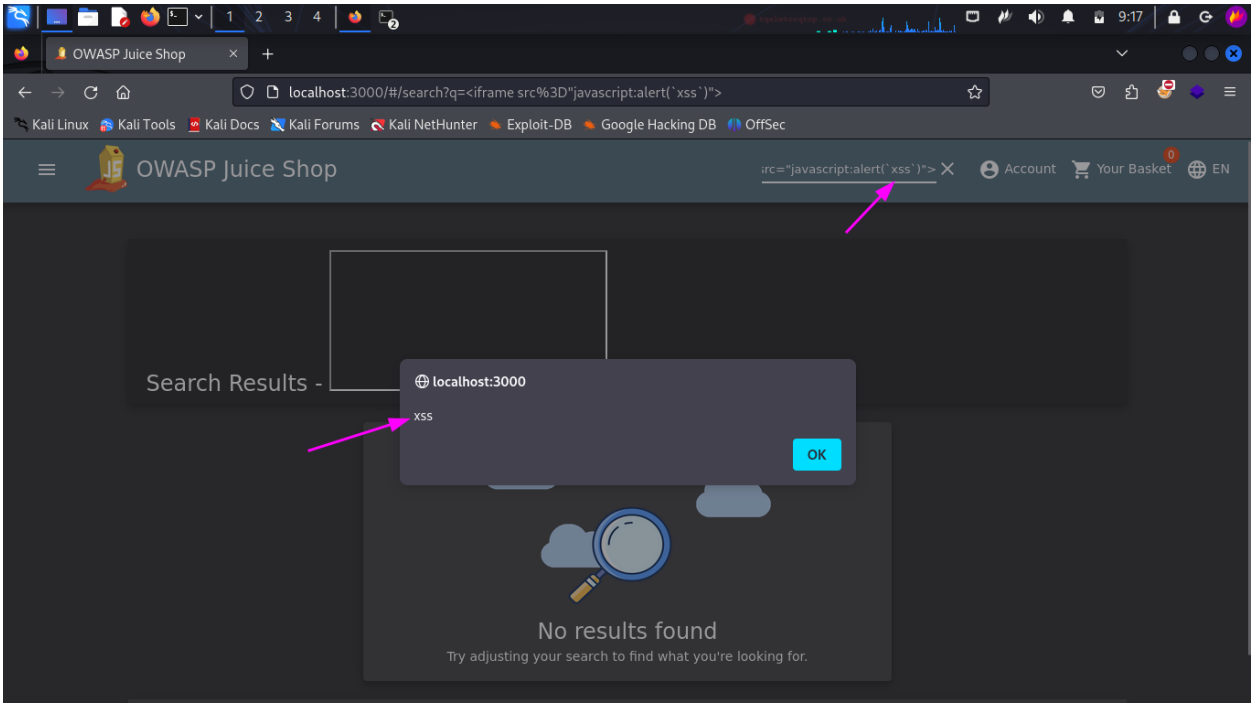
- **Never insert user input directly into the page's DOM without cleaning it up first.**
- **sanitize inputs before JavaScript handles them.**
- **Regularly review and audit your code for potential vulnerabilities.**

- **DON’T TRUST USER INPUT**

POC:







13. Security Misconfiguration (Error Handling)

Description:
This happens when the application gives too much information through error messages. For example, showing detailed error reports to users when something goes wrong can reveal sensitive details about your system.

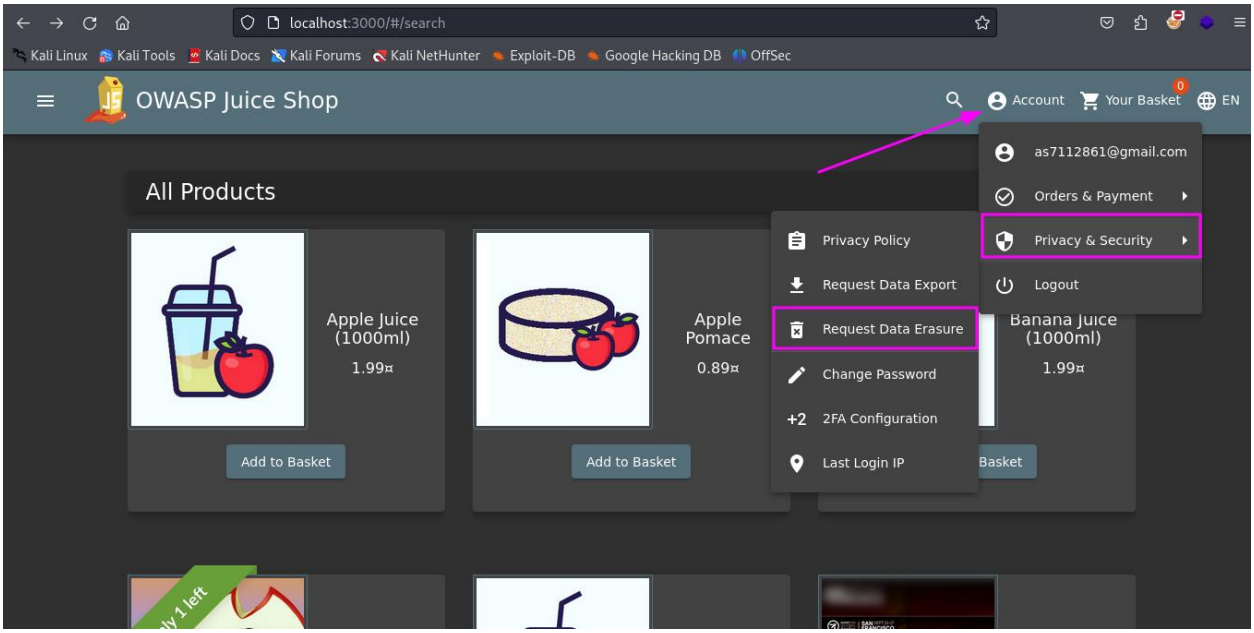
Impact:

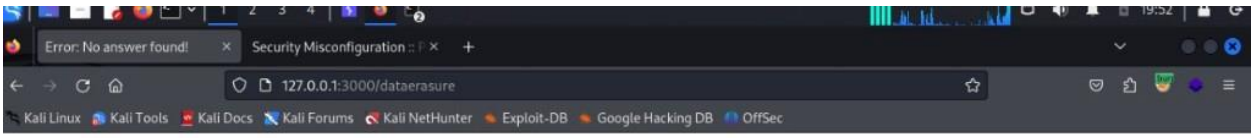
- **The file path /home/kali/juice-shop/build/routes/dataErasure.js:36:19 gives specific details about the structure of the server, the operating system Kali, the location of the problem and the exact line of code where the error occurred (js:36:19) all these pieces of information can be used by the attacker.**
- **500 error: the attacker knows that the problem is associated with the server**
- **The version of the Express framework: once the attacker knows the version he will look for known vulnerabilities associated with it.**
- **Once they have these pieces of information, they can exploit it to launch more serious attacks.**

Mitigation:

- **Show users simple, generic error messages like “Something went wrong”.**
- **Regularly check your server and application settings to prevent sensitive information from leaking.**

POC:


localhost:3000/#/search
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
OWASP Juice Shop
Account
Your Basket
EN
All Products
Apple Juice (1000ml)
1.99¤
Add to Basket
Apple Pomace
0.89¤
Add to Basket
Banana Juice (1000ml)
1.99¤
Basket
as7112861@gmail.com
Orders & Payment
Privacy & Security
Logout
Privacy Policy
Request Data Export
Request Data Erasure
Change Password
+2 2FA Configuration
Last Login IP


Error: No answer found!
Security Misconfiguration
127.0.0.1:3000/dataerasure
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec


OWASP Juice Shop (Express ^4.17.1)
500 Error: No answer found!
at /home/kali/juice-shop/build/routes/dataErasure.js:36:19

14. Admin registration (Privilege escalation)

Description:

The Juice Shop allows privilege escalation through a flaw in the user registration process, which permits non-admin users to register as admin by manipulating certain parameters. Attackers can gain administrative privileges without proper authorization.

Exploitation Scenario:

1. **Registration**: The attacker creates a new account as if he is a normal user.
2. **Capture the Request**: The attacker captures the POST request using Burp Suite.
3. **Use Repeater for Automation**: The attacker sends the captured request to Burp Suite's **Repeater** module, where he can edit and resend it again.
4. **Editing the request:** An attacker discovers a way to modify registration parameters (such as a hidden admin flag) during the registration process. The attacker edits the email in the request to create a new user's account, then edits the role of the user to admin.
5. **Analyze Server Response**: After resending the request, the attacker receives a valid response (201) Created, indicating that the new user was accepted by the server as an admin. The attacker will then login with the created admin user, and he will get a successful login.

Impact:

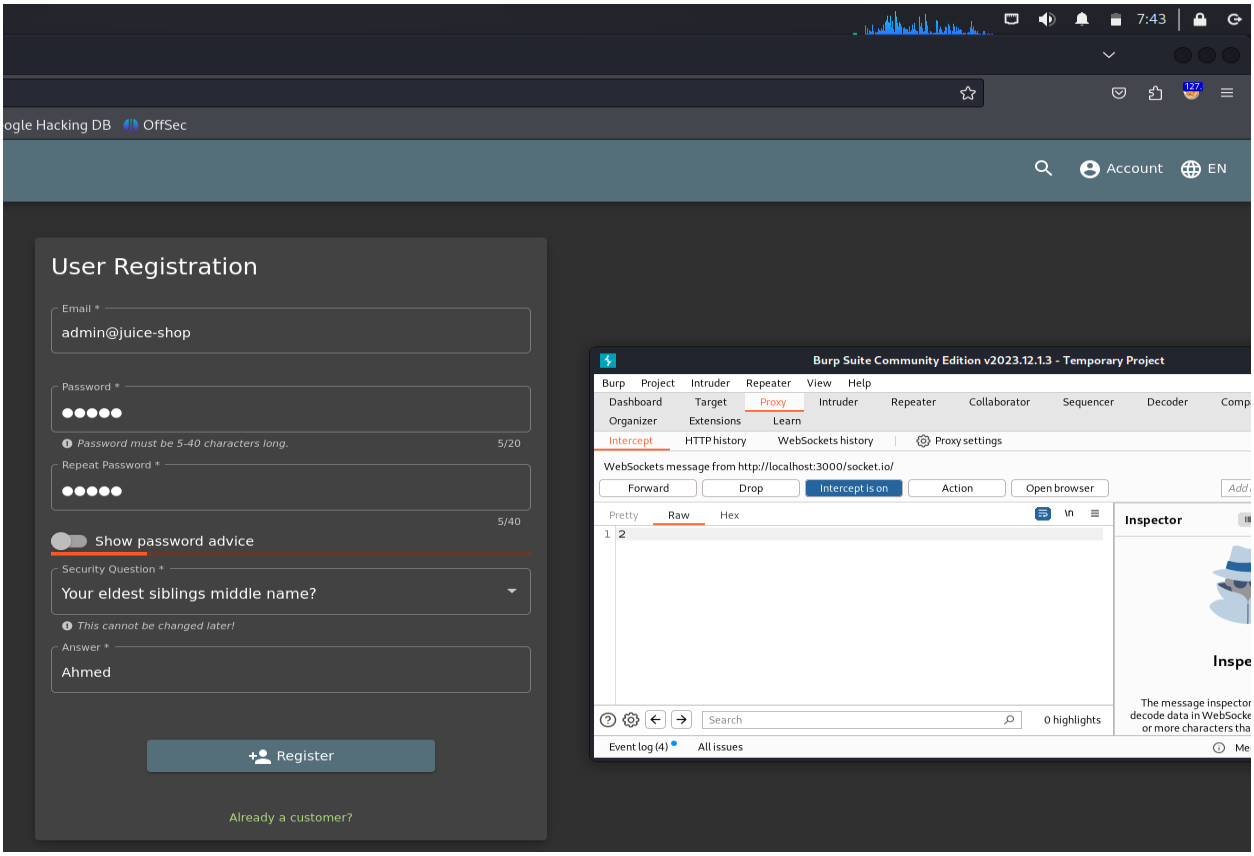
- **Unauthorized Admin Access**: Regular users can elevate their privileges to gain full control over the application.
- **System Compromise:** Once attackers have admin rights, they can change system settings, manipulate data, or access sensitive data.
- **Data Loss or Tampering:** Admin-level access can lead to the deletion or modification of critical data.

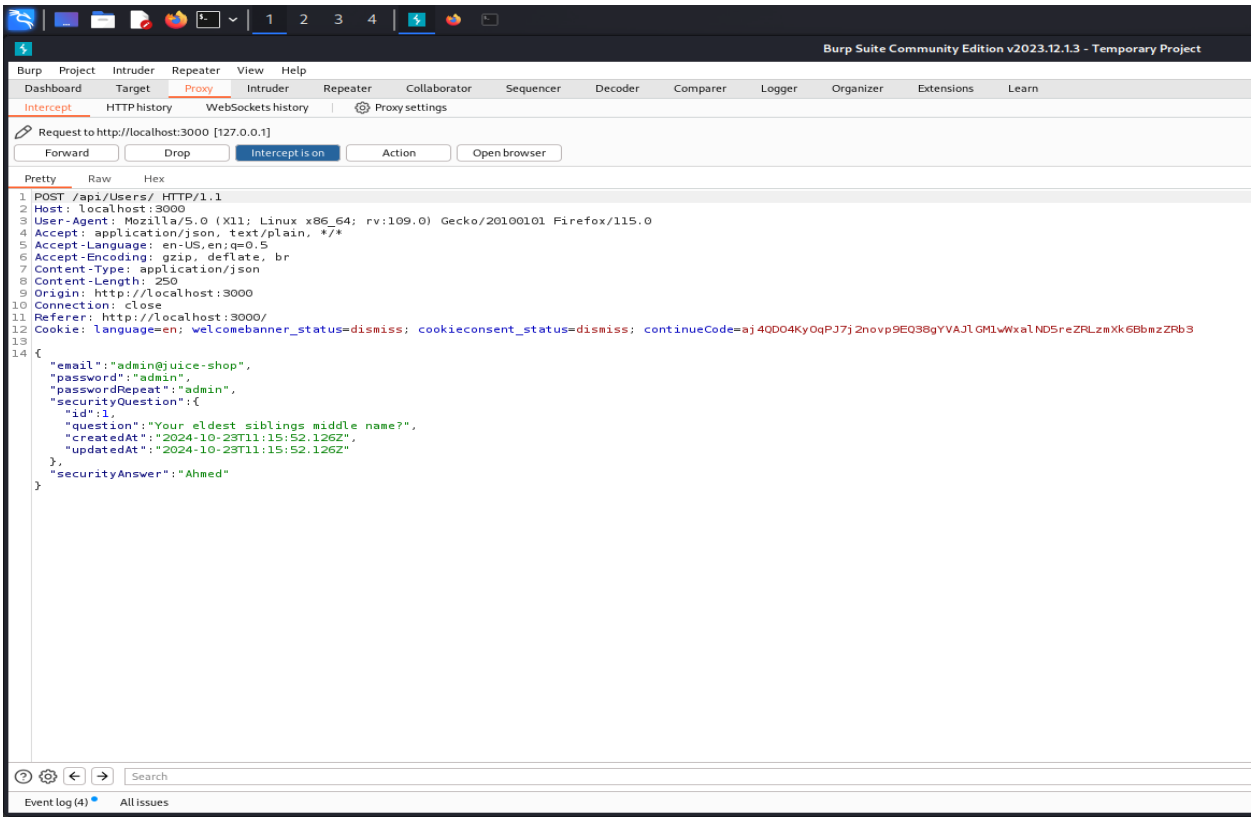
Mitigation:

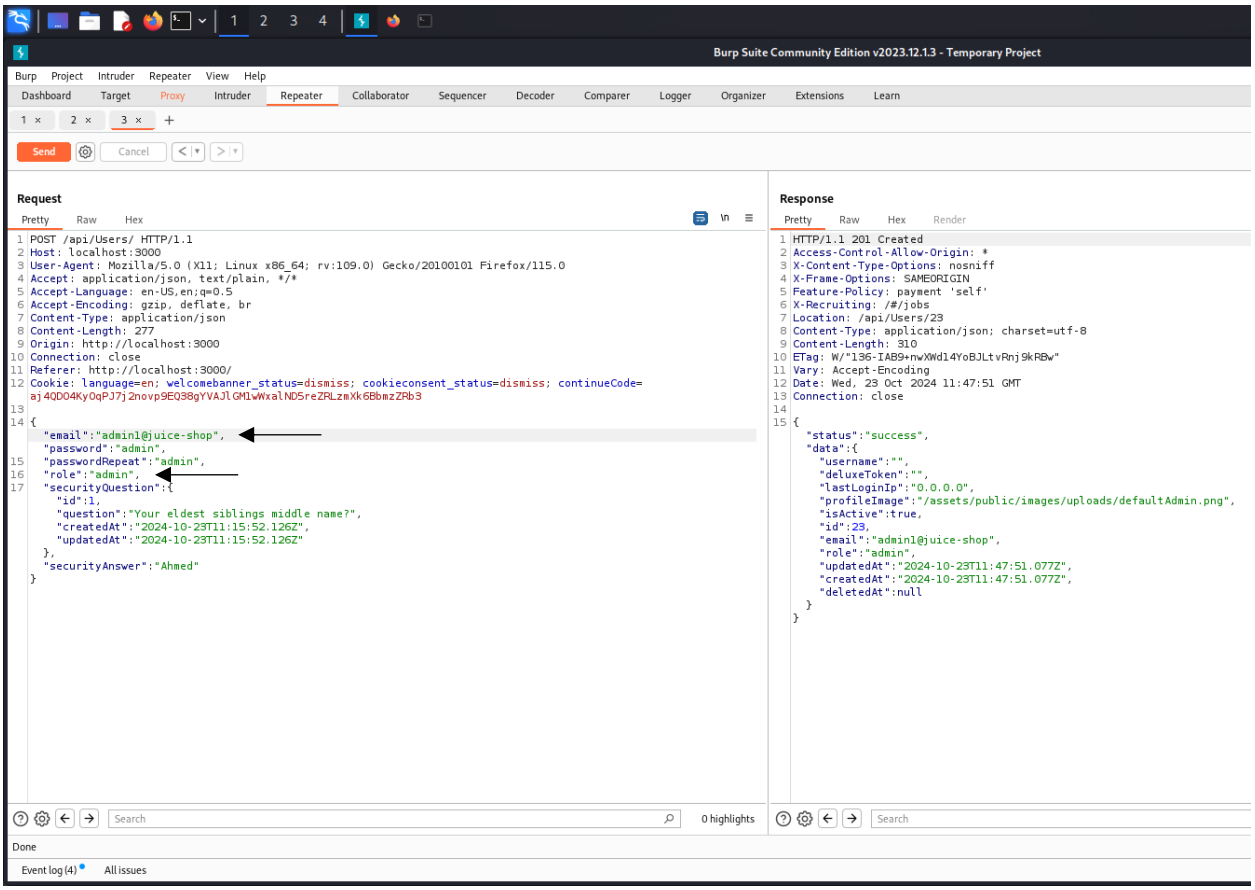
Apply least Privilege Principle:

- **Limit user permissions**: Ensure that users and applications only have the minimum privileges needed to perform their tasks.
- **Role-based access control**: Assign permissions based on roles rather than to individual users to simplify privilege management.
- **Privilege separation**: Ensure that administrative tasks and normal user tasks are performed under separate accounts. Apply strict checks to avoid role escalation during user registration.

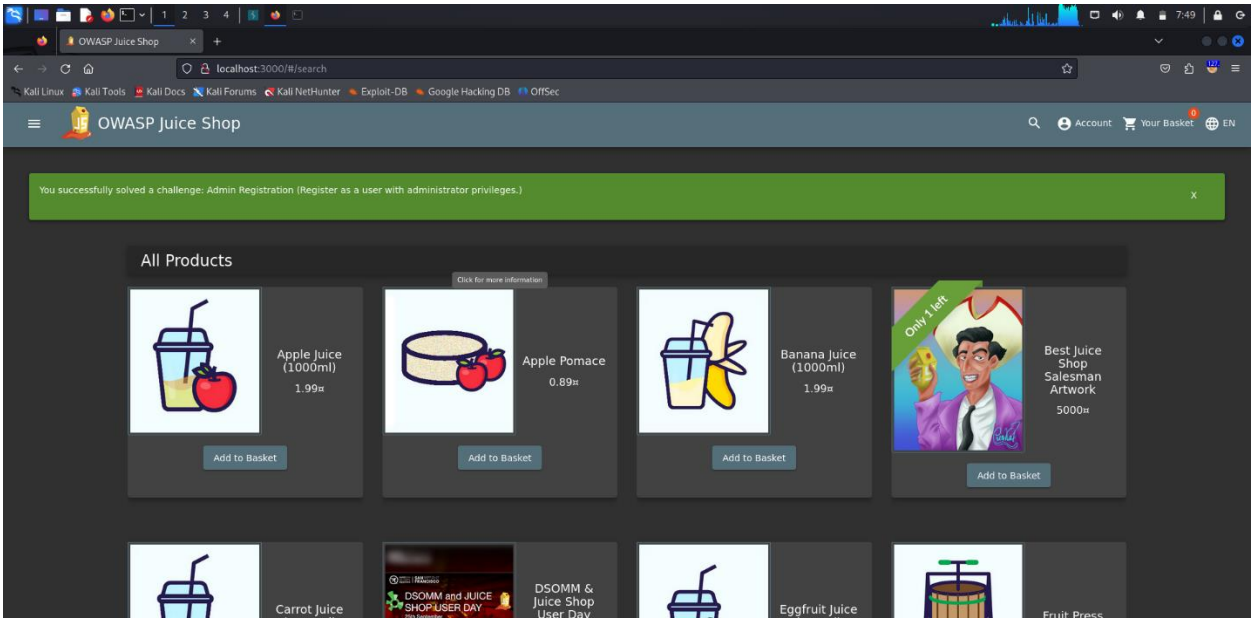
POC:


ogle Hacking DB
OffSec
Account
EN
User Registration
Email *
admin@juice-shop
Password *
Password must be 5-40 characters long.
5/20
Repeat Password *
5/40
Show password advice
Security Question *
Your eldest siblings middle name?
This cannot be changed later!
Answer *
Ahmed
Register
Already a customer?
Burp Suite Community Edition v2023.12.1.3 - Temporary Project
WebSockets message from http://localhost:3000/socket.io/
Forward
Drop
Intercept is on
Action
Open browser
Inspector
0 highlights
Event log (4)
All issues


Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Request to http://localhost:3000 [127.0.0.1]
Forward
Drop
Intercept is on
Action
Open browser
POST /api/Users/ HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 250
Origin: http://localhost:3000
Connection: close
Referer: http://localhost:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=aj4QDO4KyOqPJ7j2novp9EQ38gYVAJlGM1wWxalND5reZRLzmXk6BbmzZRb3
{
"email":"admin@juice-shop",
"password":"admin",
"passwordRepeat":"admin",
"securityQuestion":{
"id":1,
"question":"Your eldest siblings middle name?",
"createdAt":"2024-10-23T11:15:52.126Z",
"updatedAt":"2024-10-23T11:15:52.126Z"
},
"securityAnswer":"Ahmed"
}
Event log (4)
All issues



Successfully logged in as an admin



15. Privacy Policy Auto-Solve (Information Disclosure)

Description:
Upon entering the application, it was found that the "Privacy Policy" challenge was already marked as solved without user interaction, indicating an information disclosure or misconfiguration within the challenge system.

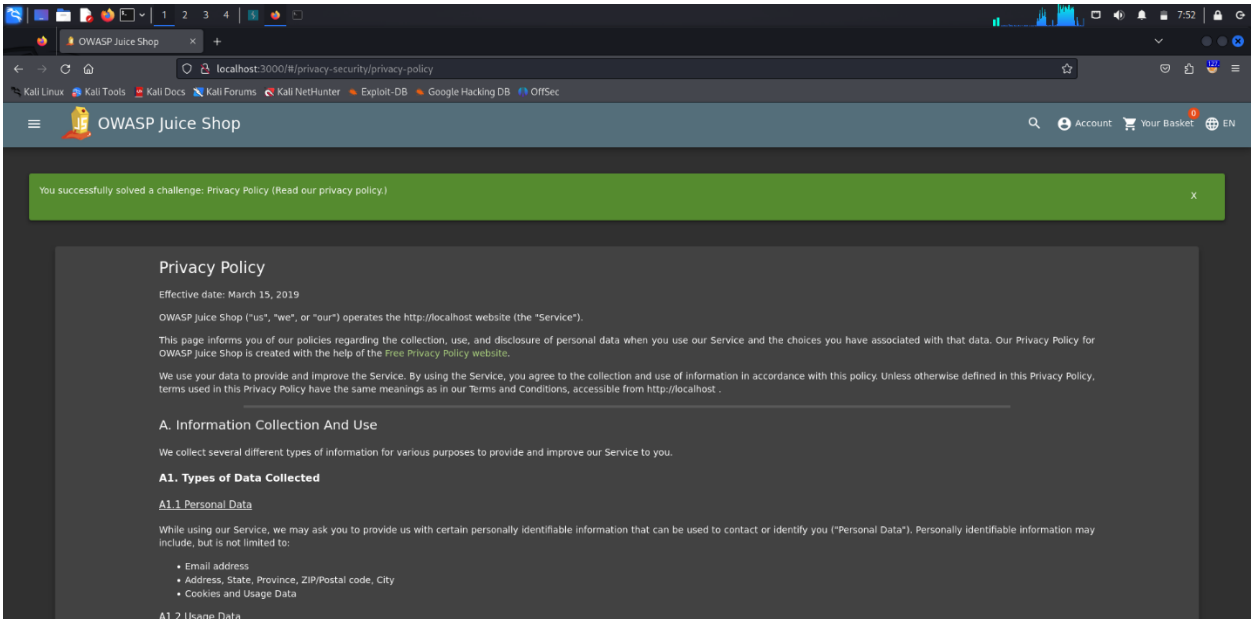
Impact:

- **Incorrect Challenge Completion:** Users may bypass certain challenges without solving them.
- **Potential Data Exposure:** If the Privacy Policy or related content is unintentionally exposed, sensitive information might be revealed.

Mitigation:

- Review and fix the challenge logic to ensure proper validation before marking challenges as complete.
- Perform regular audits on challenge functionality to avoid automatic completion or information leakage.

POC:



16. Upload Type (Improper Input Validation)

Description:
The Juice Shop does not properly validate file types during the upload process. This allows attackers to upload malicious files, which could lead to code execution or other security risks.

Exploitation Scenario:
An attacker attempts to upload a file with a non-permitted file extension, like .jpeg, bypassing file-type validation. The server accepts the file, potentially executing malicious code.

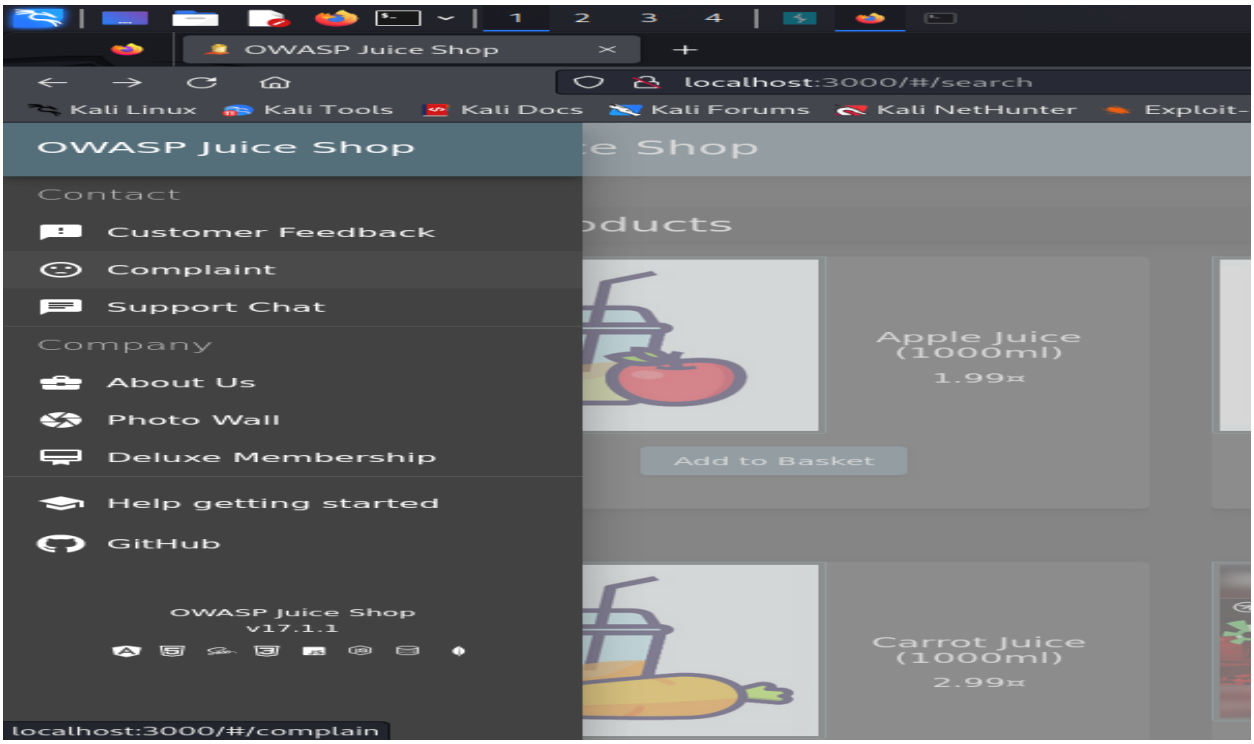
Impact:

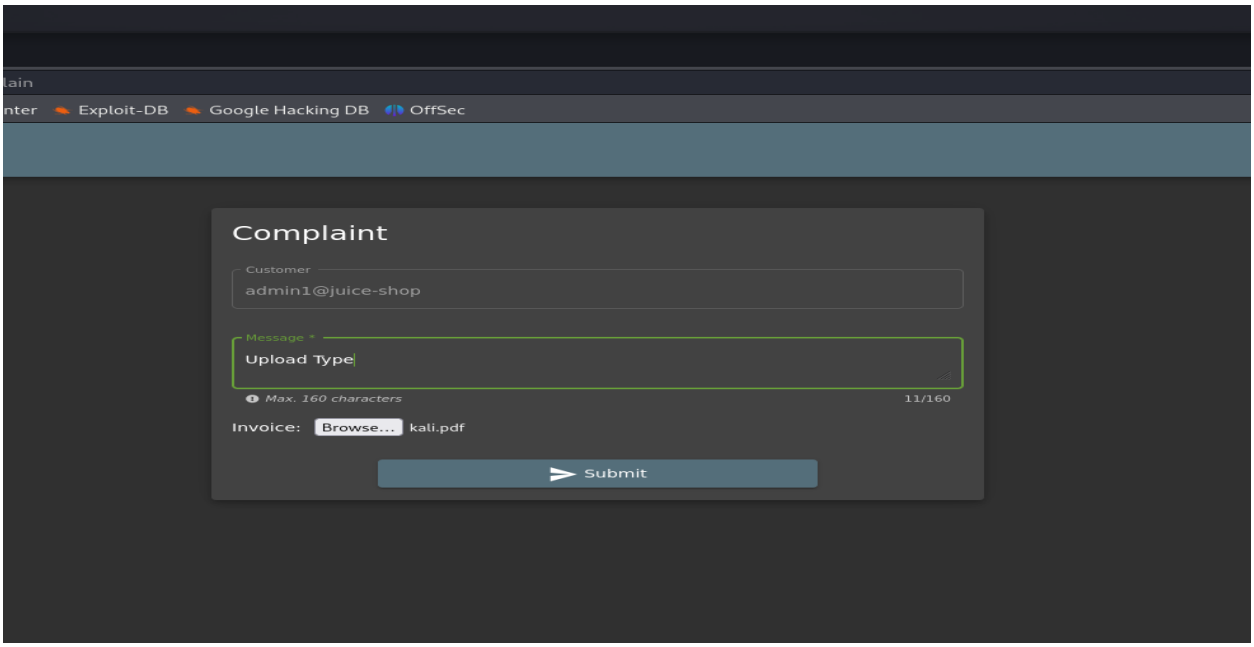
- **Malware Injection:** Attackers could upload scripts or malicious files that could be executed on the server, leading to system compromise.
- **Denial of Service (DoS):** Improperly handled file types could cause the server to crash or degrade performance.
- **Security Bypass:** Uploading files not intended for the application could allow attackers to bypass certain security controls.

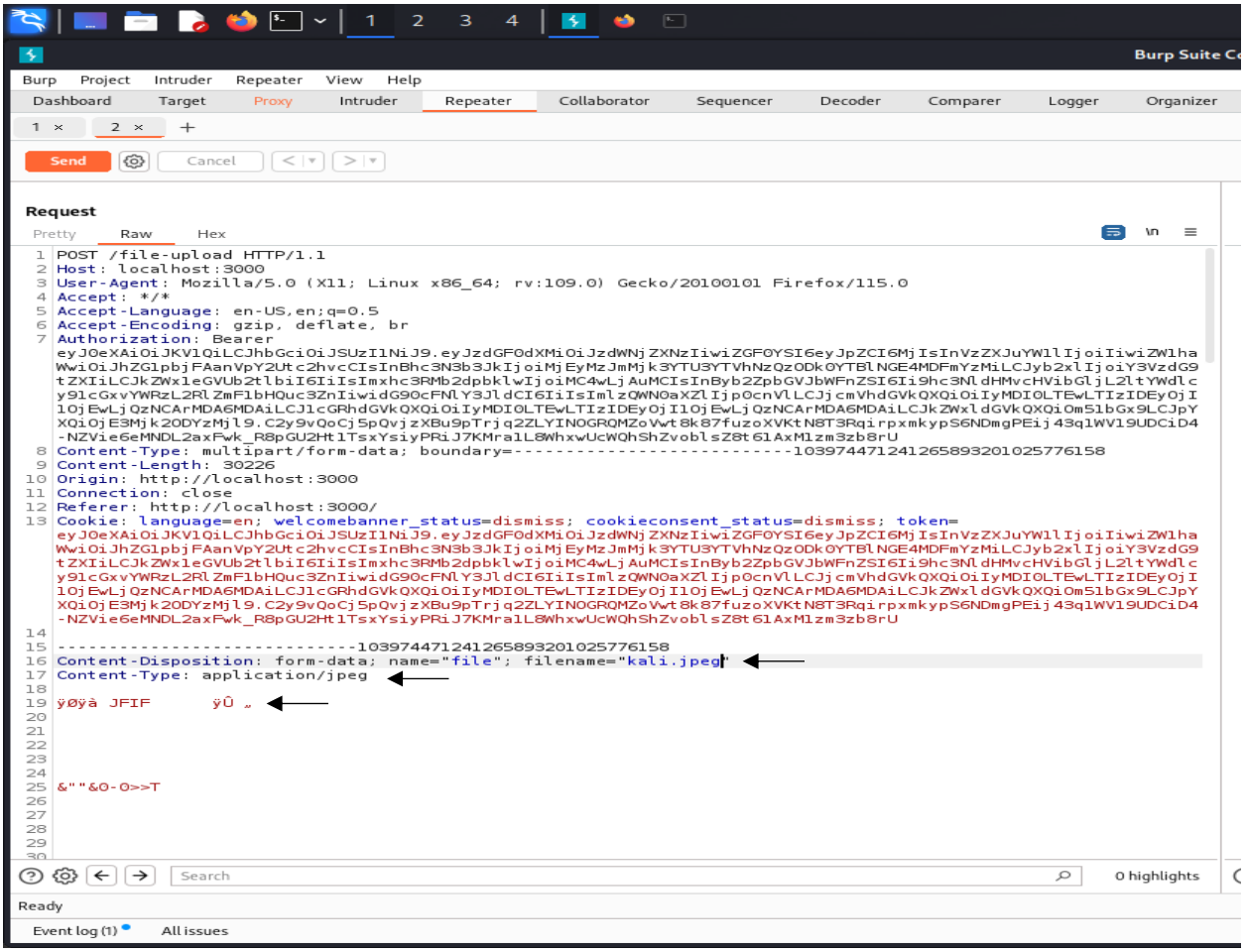
Mitigation:

- Implement strict server-side file type validation, ensuring only permitted types (e.g., .jpeg, .png) are accepted.
- Store uploaded files in a non-executable directory.

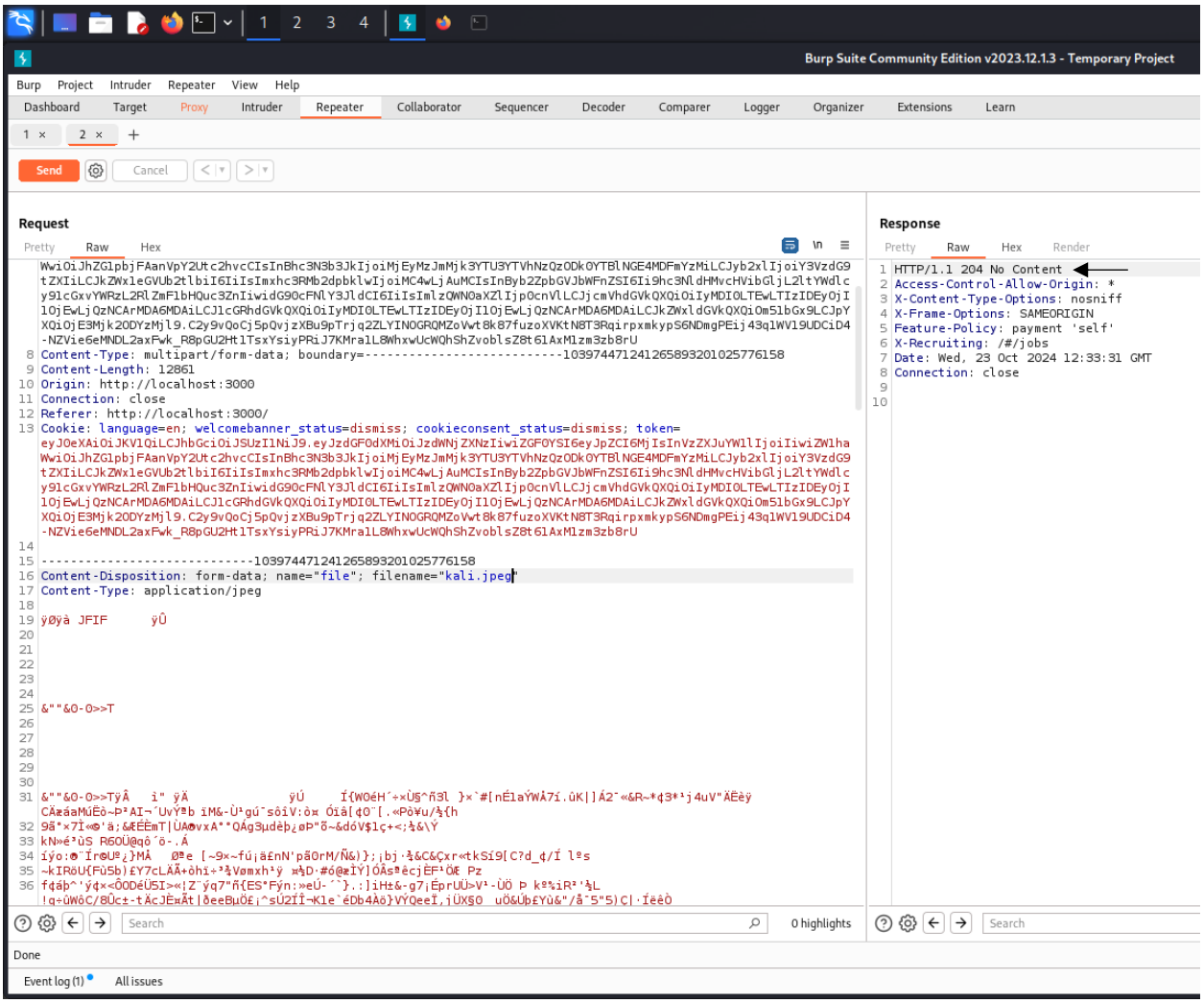
POC:

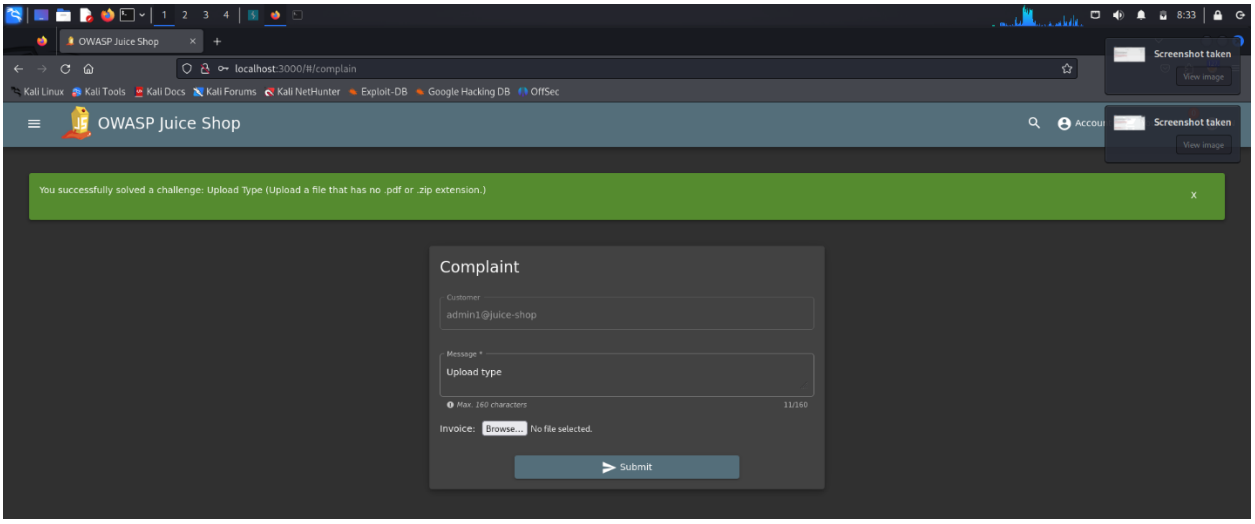






File uploaded successfully





17. Manipulate Basket (Broken Access Control)

Description:
This vulnerability occurs when users can manipulate the contents of their shopping basket by altering certain parameters. An attacker could add or remove items or change quantities without proper authorization checks.

Exploitation Scenario:
An attacker captures and modifies the parameters of a "basket" HTTP request using a tool like Burp Suite, altering item quantities or prices without authorization. The server accepts these changes without validating user permissions.

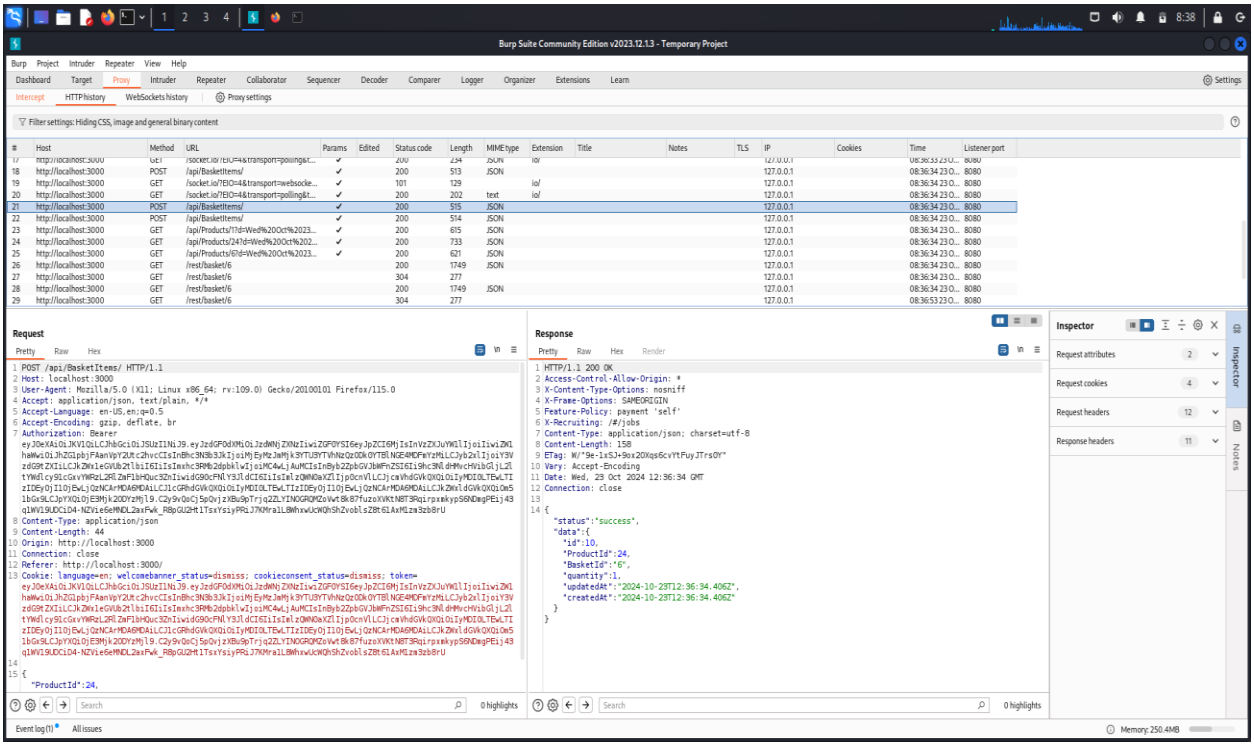
Impact:

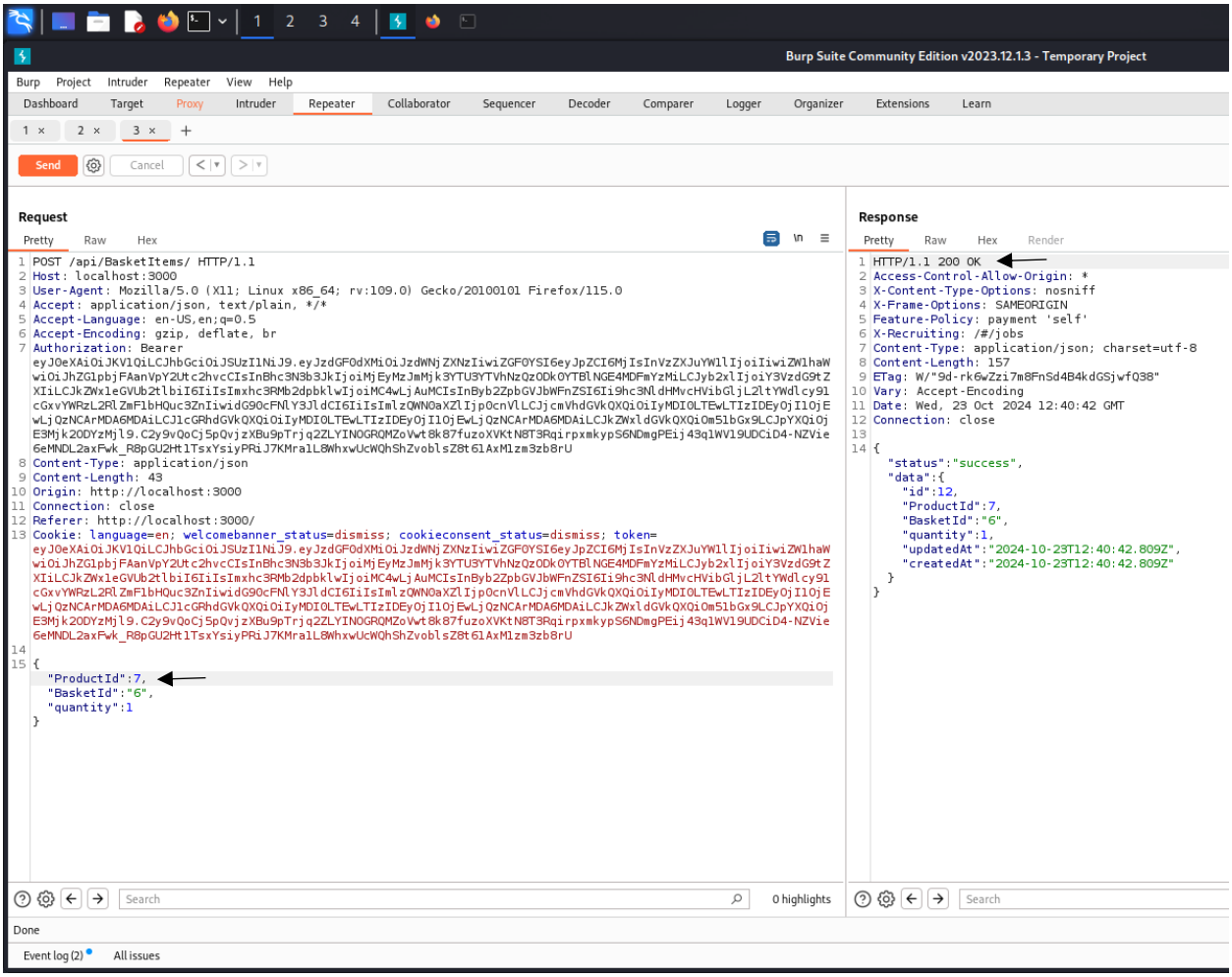
- **Financial Exploitation:** Attackers could manipulate the basket to pay less or avoid charges.
- **Data Integrity Issues:** Unauthorized manipulation of basket contents could lead to incorrect records of user transactions.
- **Unauthorized Access:** Attackers may view or modify the baskets of other users, exposing personal information.

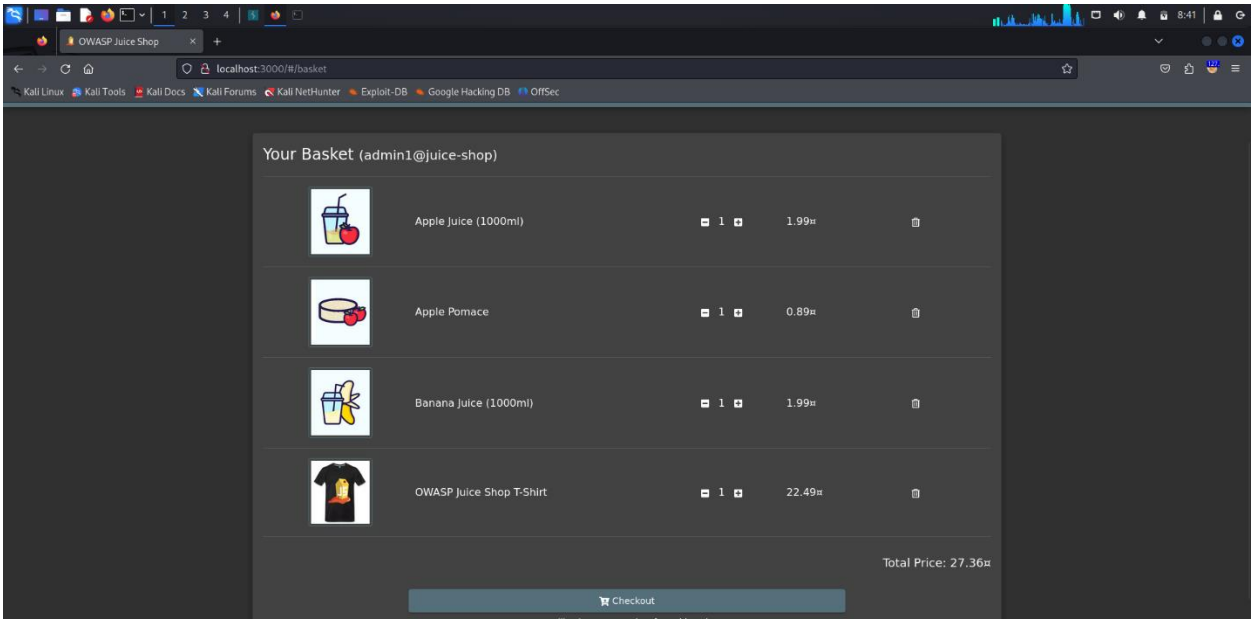
Mitigation:

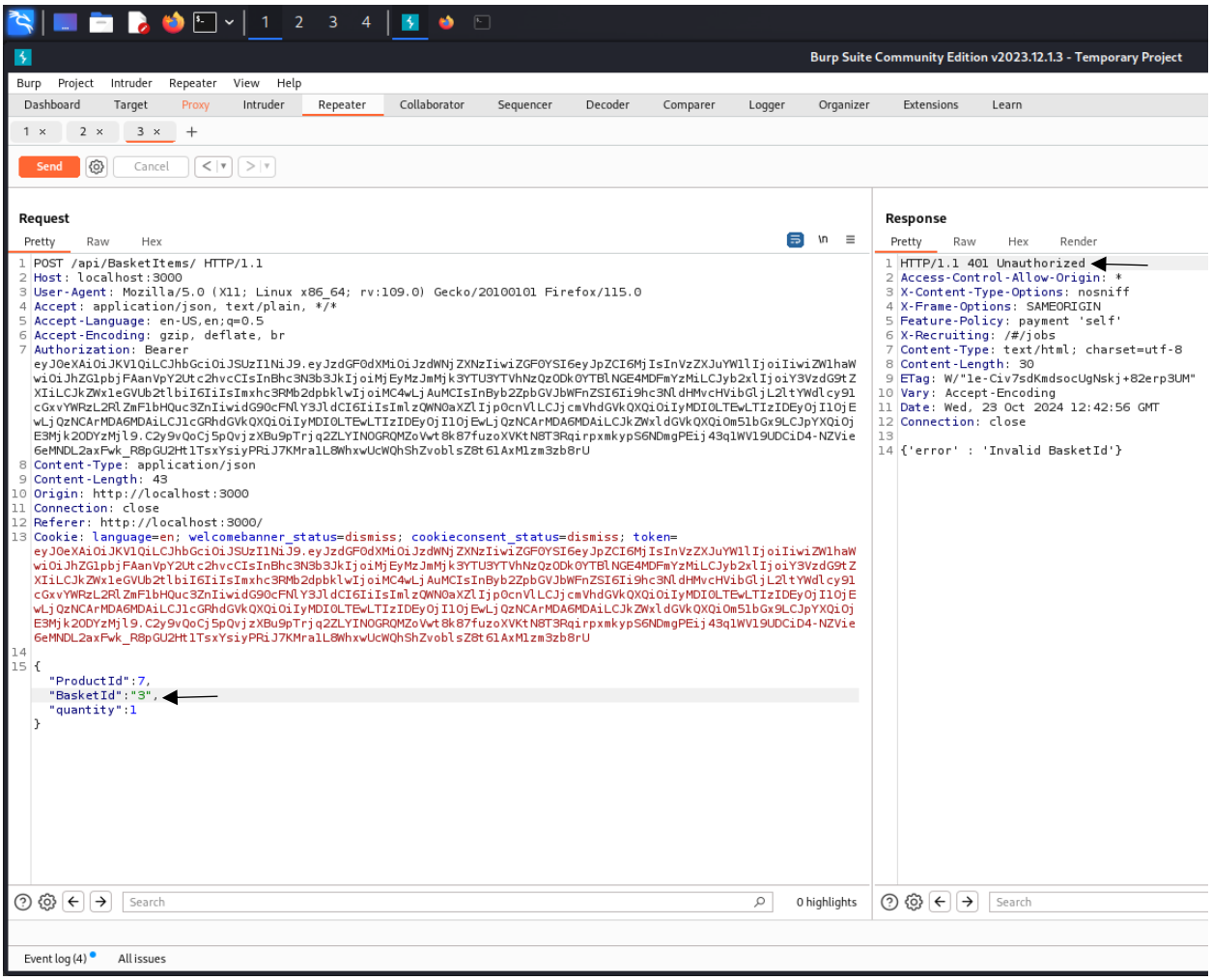
- Enforce proper access control mechanisms to ensure users can only modify their own baskets.
- Validate all parameters related to basket manipulation both client-side and server-side.

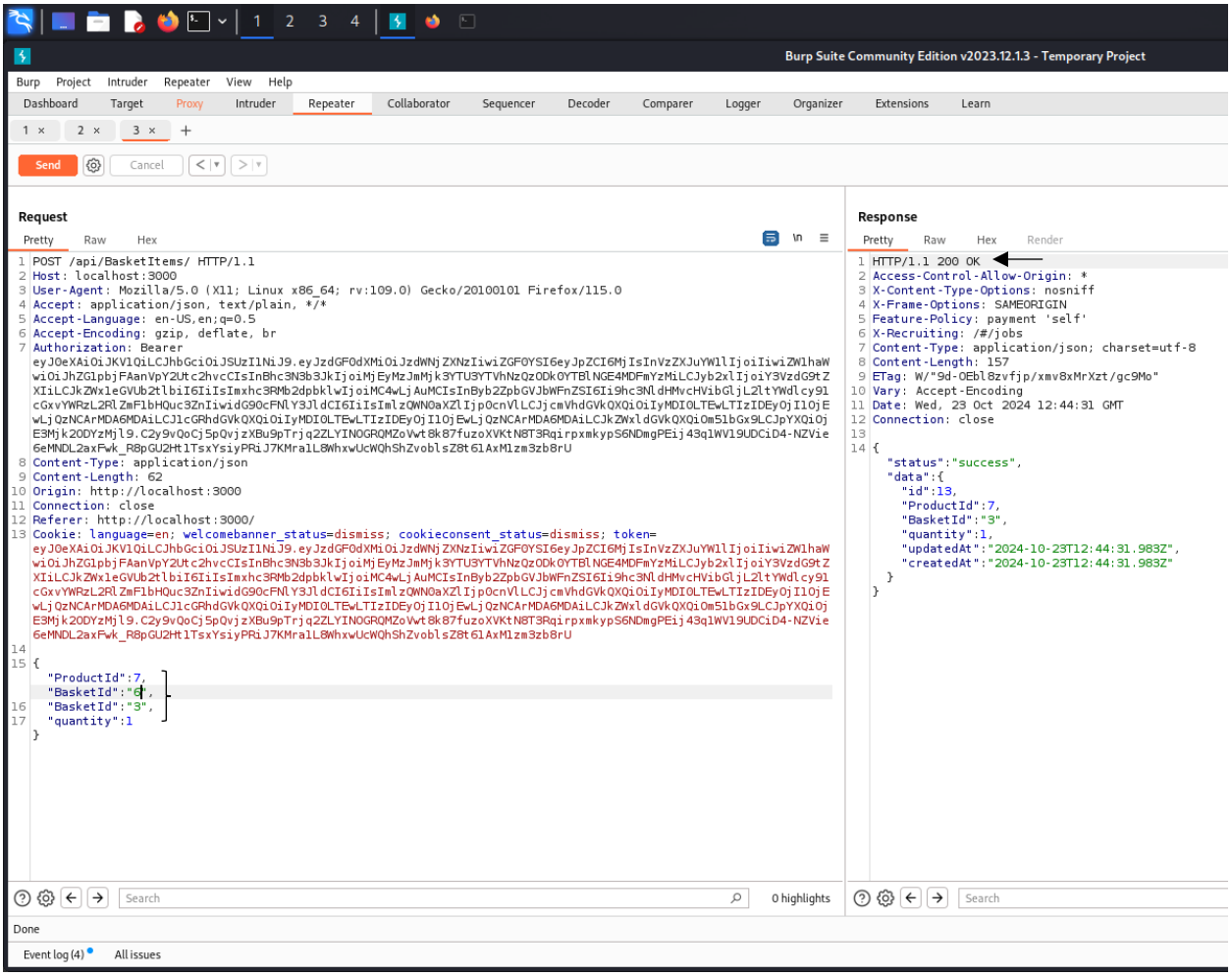
POC:

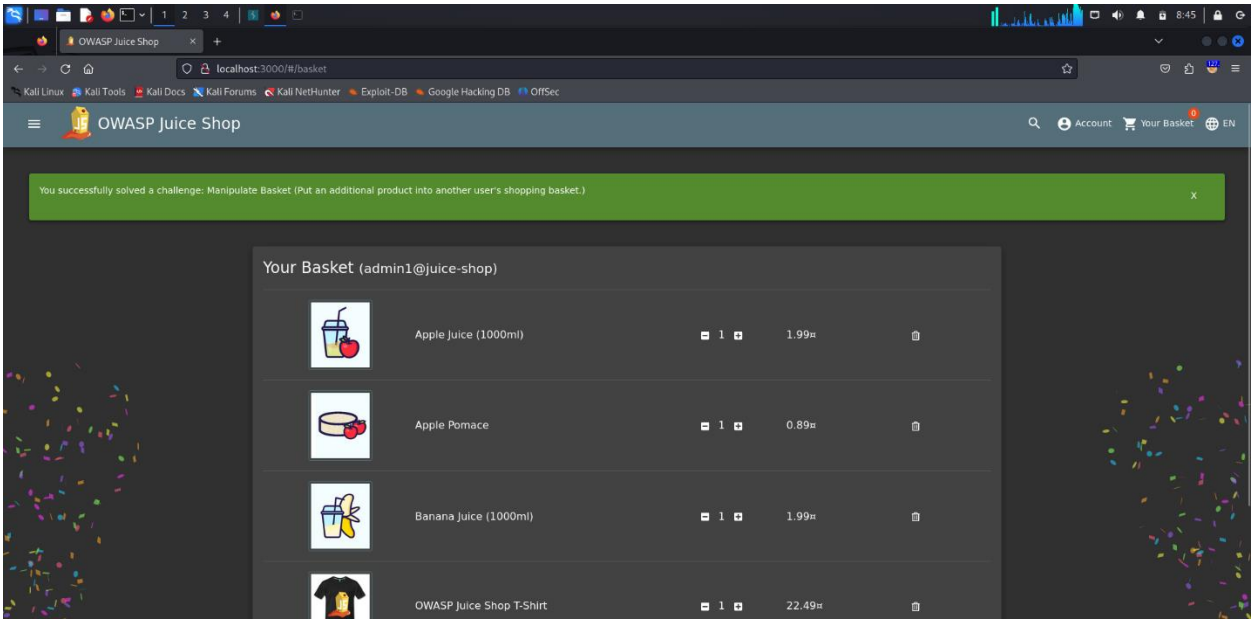



Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Request
Response
POST /api/BasketItems/ HTTP/1.1
HTTP/1.1 200 OK
"ProductId":7,
"BasketId":"6",
"quantity":1


OWASP Juice Shop
localhost:3000/#/basket
Your Basket (admin1@juice-shop)
Apple Juice (1000ml)
Apple Pomace
Banana Juice (1000ml)
OWASP Juice Shop T-Shirt
Total Price: 27.36¤
Checkout


Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Send Cancel
Request
Pretty Raw Hex
POST /api/BasketItems/ HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjIsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhZG1pbjFAanVpY2Utc2hvcCIsInBhc3N3b3JkIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI0LTEwLTIzIDEyOjI1OjEwLjQzNCArMDA6MDAiLCJ1cGRhdGVkQXQiOiIyMDI0LTEwLTIzIDEyOjI1OjEwLjQzNCArMDA6MDAiLCJkZWxldGVkQXQiOm5lbGx9LCJpYXQiOjE3Mjk20DYzMjl9.C2y9vQoCj5pQvjzXBu9pTrjq2ZLYINOGRQMZoVwt8k87fuzoXVKtN8T3RqirpxmkypS6NDmgPEij43q1WV19UDCiD4-NZVie6eMNDL2axFwk_R8pGU2Ht1TsxYsiyPRiJ7KMralL8WhxwUcWQhShZvoblsZ8t61AxM1zm3zb8rU
Content-Type: application/json
Content-Length: 43
Origin: http://localhost:3000
Connection: close
Referer: http://localhost:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjIsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhZG1pbjFAanVpY2Utc2hvcCIsInBhc3N3b3JkIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIyMDI0LTEwLTIzIDEyOjI1OjEwLjQzNCArMDA6MDAiLCJkZWxldGVkQXQiOm5lbGx9LCJpYXQiOjE3Mjk20DYzMjl9.C2y9vQoCj5pQvjzXBu9pTrjq2ZLYINOGRQMZoVwt8k87fuzoXVKtN8T3RqirpxmkypS6NDmgPEij43q1WV19UDCiD4-NZVie6eMNDL2axFwk_R8pGU2Ht1TsxYsiyPRiJ7KMralL8WhxwUcWQhShZvoblsZ8t61AxM1zm3zb8rU
{
"ProductId":7,
"BasketId":"3",
"quantity":1
}
Response
Pretty Raw Hex Render
HTTP/1.1 401 Unauthorized
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: 30
ETag: W/"1e-Civ7sdKmdsocUgNskj+82erp3UM"
Vary: Accept-Encoding
Date: Wed, 23 Oct 2024 12:42:56 GMT
Connection: close
{'error' : 'Invalid BasketId'}
Search
0 highlights
Search
Event log (4)
All issues





18. Repetitive Registration (Improper Input Validation)

Description:

The application allows users to register using different passwords, due to improper input validation in the registration process. This can lead to user confusion or abuse by attackers.

Exploitation Scenario:
An attacker registers with an email address, creating an account using a password the differs from the password confirmation field without being blocked by the system. This can lead to the attacker flooding the system with duplicate registrations.

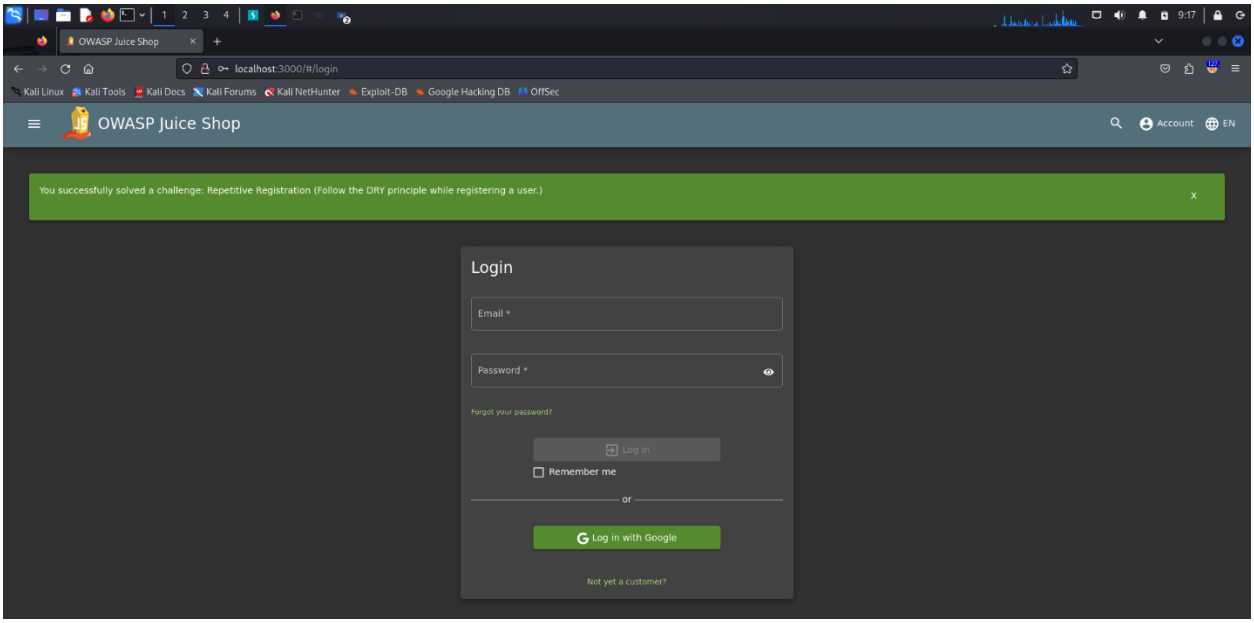
Impact:

- **Account Duplication:** Attackers can create duplicate accounts using different passwords, which can be used for spam or fraudulent activities.
- **Denial of Service (DoS):** Repetitive registrations can exhaust system resources, leading to degraded performance or downtime.

Mitigation:

- Implement strict input validation to prevent duplicate account creation
- Enforce email verification before allowing account creation.

POC:



19. Negative quantity value (Improper Input Validation)

Description:
The basket functionality is vulnerable to improper input validation. By entering a negative quantity in the basket, the application allows the user to proceed with the purchase, resulting in a negative total price.

Exploitation Scenario:
An attacker manipulates a system by inputting negative values where the system expects positive ones. This can lead to unintended outcomes, causing financial or operational damage.

Impact:

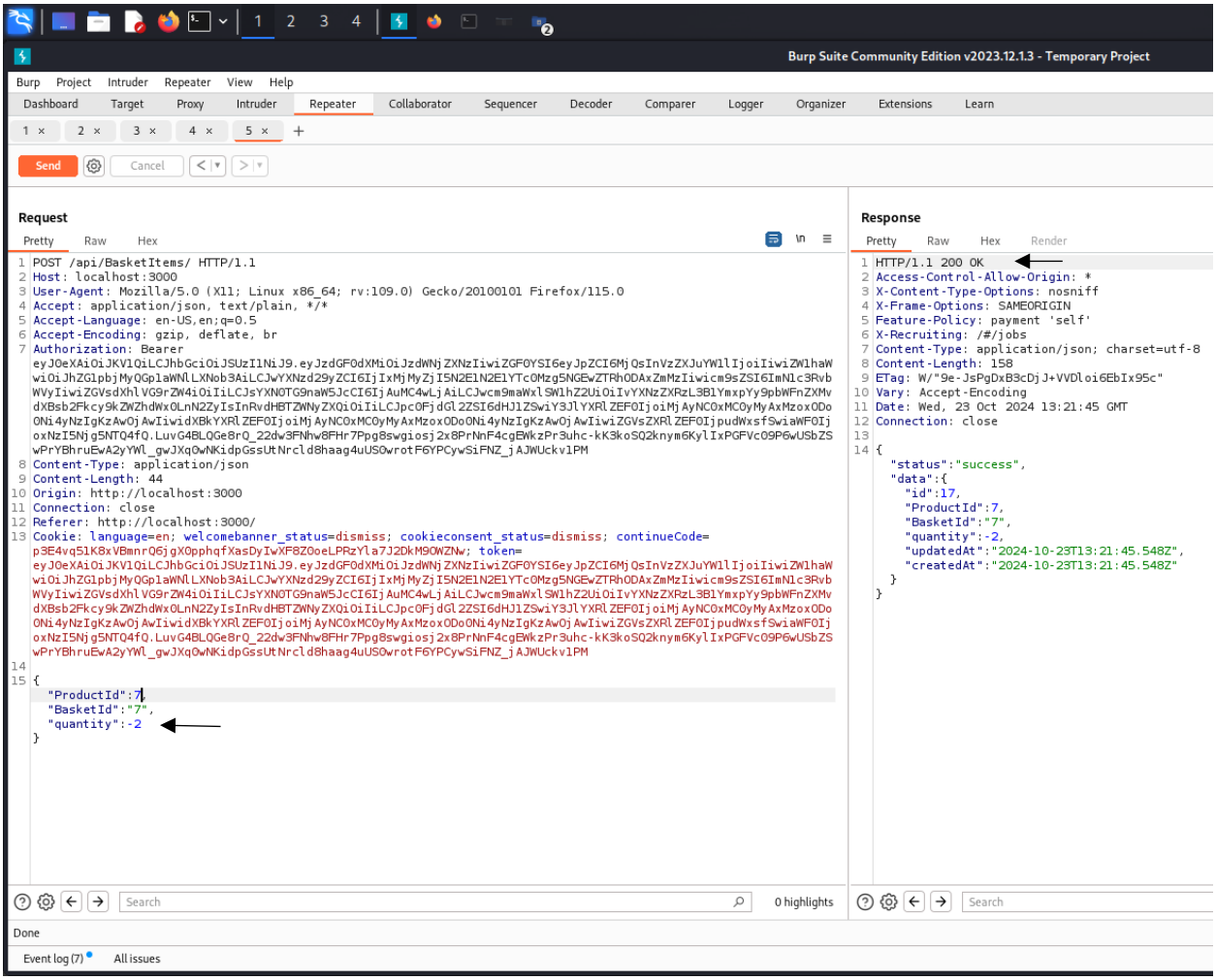
- **Adding items to the basket with negative quantity**
- **Negative total price:** as the quantity is added in negative value, the price is also calculated by negative values.

- **Adding money to users:** By checking out with a negative quantity using digital wallet functionality, the user receives money instead of paying for the items.

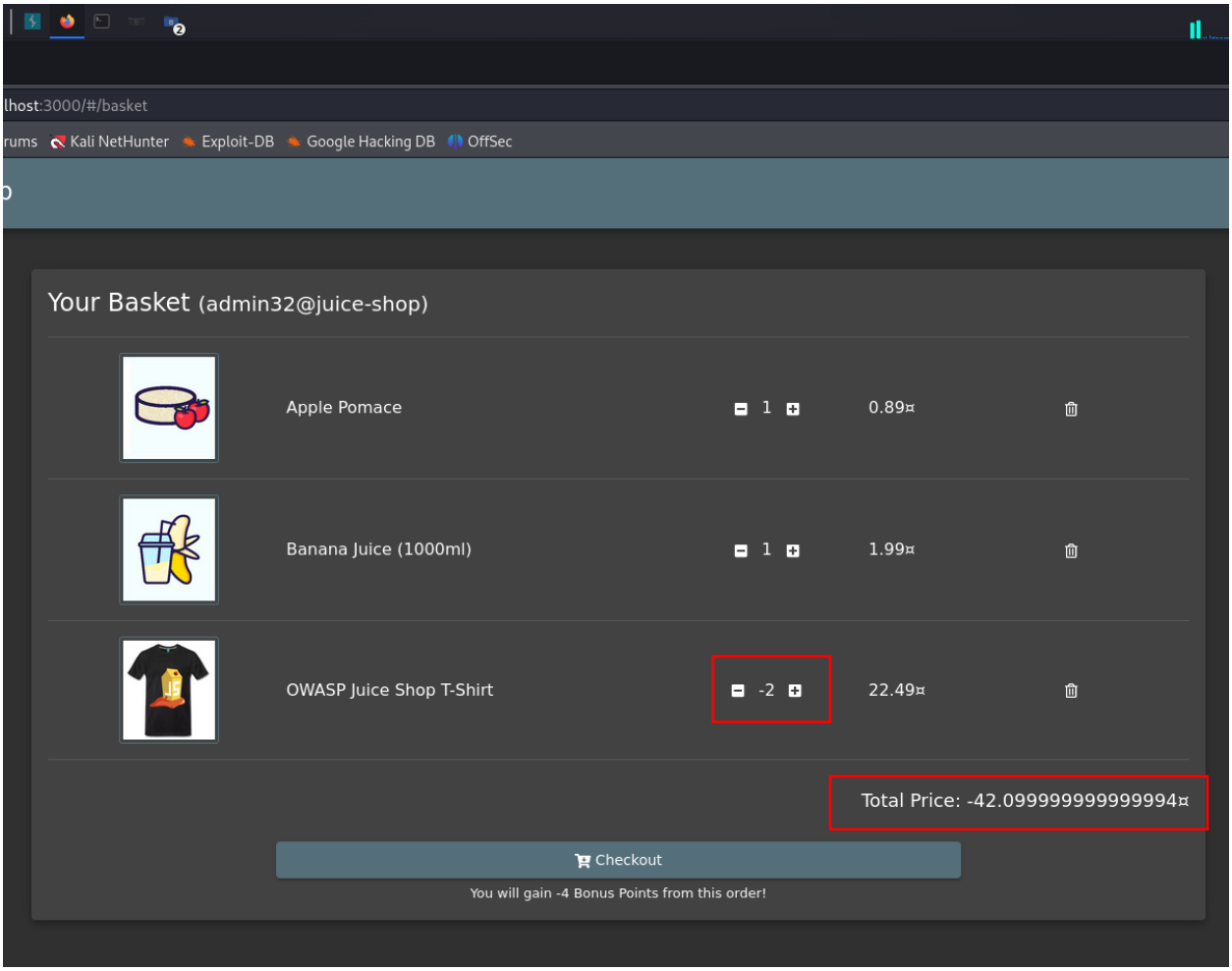
Mitigation:

- Implement strict input validation to prevent negative values in quantity field.

POC:



The application allowed the purchase of a negative quantity of items, resulting in a negative total price.



Methodology

Starting on 10-10-2024 the Penetration testing team engaged
on a penetration test of the Juice Shop Service. All of the testing was performed
with the following methodology:
1. Discovery
2. Scanning
3. Exploitation
4. Reporting
Along this report the team has provided screenshots and important files used during
the assessment

Assessment Toolset Selection

1-Burpsuit
2-SQLMap
Browser Developer Tools
Ffuf (Fuzz Faster U Fool) or DirBuster (for fuzzing)

Assessment Methodology Detail

1. SQL Injection in Login Page

Objective: Identify SQL injection vulnerabilities in the login form.

Tools Used:

- Burp Suite
- SQLMap (optional)

Steps:

1. **Intercept the Login Request**:
    - Use Burp Suite to intercept the HTTP request after attempting to log in.
    - Submit login credentials like admin' OR '1'='1.
2. **Modify the Input**:
    - Modify the username field to include ' OR '1'='1 in the intercepted request.
    - Observe if the server bypasses authentication and grants access.
3. **Validate**:
    - Check if administrative access is achieved or sensitive data is exposed.

Manual Validation:

- Attempt using different payloads to assess if the vulnerability is present on other fields such as the password reset form.

Conclusion: The vulnerability is confirmed if unauthorized access is granted. It suggests the server lacks proper input sanitization.

---

2. Captcha Bypass in Feedback Submission

Objective: Test if CAPTCHA mechanisms can be bypassed, allowing spam submissions.

Tools Used:

- Burp Suite

Steps:

1. **Submit Feedback**:
    - Navigate to the feedback form and capture the request with CAPTCHA.
2. **Modify the Request**:
    - Intercept the HTTP POST request in Burp Suite.
    - Remove or tamper with the CAPTCHA token.
3. **Automate Bypass**:
    - Use Burp Suite's Repeater to send multiple requests without CAPTCHA.

Manual Validation:

- If submissions are accepted without CAPTCHA validation, confirm the vulnerability by submitting multiple entries.

Conclusion: A successful bypass reveals the CAPTCHA implementation is insufficient, exposing the application to automated attacks.

---

3. Broken Access Control in Admin Panel (Including Fuzzing)

Objective: Gain unauthorized access to the admin panel through broken access controls, and explore hidden endpoints using fuzzing.

Tools Used:

- Burp Suite
- Browser Developer Tools
- Ffuf (Fuzz Faster U Fool) or DirBuster (for fuzzing)

Steps:

1. **Analyze the Application**:
    - Open the browser Developer Tools to inspect JavaScript files and HTML code.
    - Search for references to admin or hidden URLs, such as /admin or /score-board.
2. **Direct URL Access**:

- After discovering potential hidden URLs (e.g., /admin, /score-board), attempt to access them directly by entering them into the browser.

3. **Fuzzing for Hidden Endpoints**:
    - **Use a Fuzzing Tool**: Run a directory or file discovery tool like **Ffuf** or **DirBuster** to identify unlisted but accessible admin pages or other restricted resources.
        - Example command for **Ffuf**: ffuf -u http://<domain>/FUZZ -w /path/to/wordlist.txt -mc 200
    - **Analyze Fuzzing Results**: Check the output to identify any administrative paths or API endpoints that may not be directly visible to regular users.
4. **Intercept Requests and Modify Parameters**:
    - Use Burp Suite to capture requests when accessing protected pages.
    - Modify the session token or user role parameter in the request to escalate privileges and gain unauthorized admin access.
5. **Validate Permissions**:
    - If access is granted to restricted resources or admin functionalities, confirm that role-based access control is not enforced properly.

Manual Validation:

- Use the browser Developer Tools and Burp Suite to confirm access to hidden or restricted resources.
- Test different HTTP methods (GET, POST) for bypassing restrictions.
- Review access logs and session management if possible to detect any anomalies.

Fuzzing Validation:

- Review the fuzzing results to ensure no sensitive paths are missed.
- Test discovered endpoints to see if they allow access to admin functionalities.

Conclusion:

- Gaining access to hidden or restricted URLs (e.g., admin panel) through fuzzing or direct access demonstrates broken access controls.
- Lack of proper authorization checks in the application reveals severe security weaknesses, exposing sensitive areas of the web application.

---

4. Upload Size (Improper Input Validation)

Objective: Identify improper input validation by uploading a file that exceeds the allowed size limit.

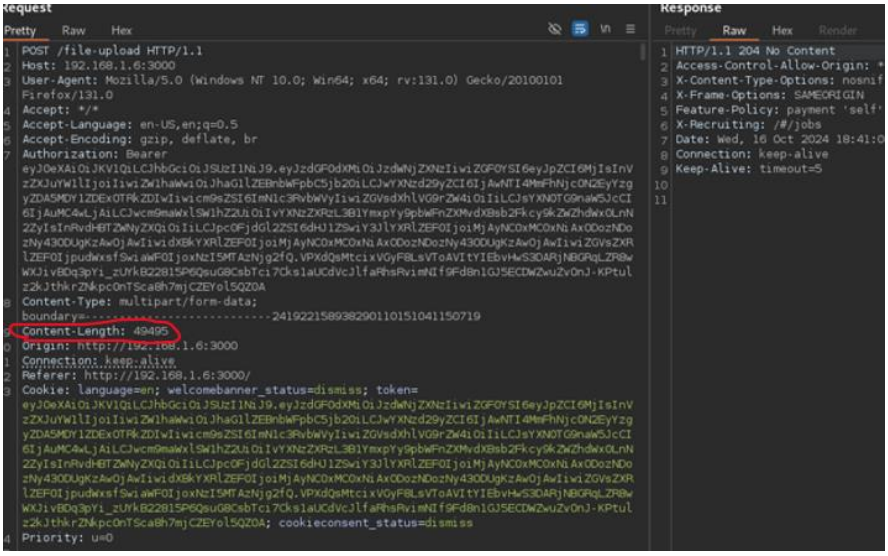
Methodology:

1. **Tool Used:** Burp Suite Pro
2. **Steps:**
    1. **Intercept the request:**
        - Open Burp Suite and intercept the traffic.

- Navigate to the file upload page in the OWASP Juice Shop.
- Upload a file within the allowed size limit .
- Burp Suite will intercept the POST request containing the file.

2. **Modify the Request:**
   - In the intercepted POST request, modify the Content-Length header to a value exceeding the upload limit
   - You can also upload a larger file and modify the size limit in the form data.
3. **Send the request:**
   - Forward the modified request to the server.
   - Observe if the server validates the size limit properly or accepts the file.

3. **Sample Burp Suite Output:**



4. **Manual Validation:**
   - If thea upload succeeds with an oversized file, the vulnerability is confirmed.
   - Use browser Developer Tools to confirm that client-side size limits were bypassed and validate that the server failed to perform proper size validation.
5. **Conclusion:**
   - Uploading files exceeding the allowed limit indicates improper input validation on the server side, which could potentially lead to denial of service or server resource exhaustion.

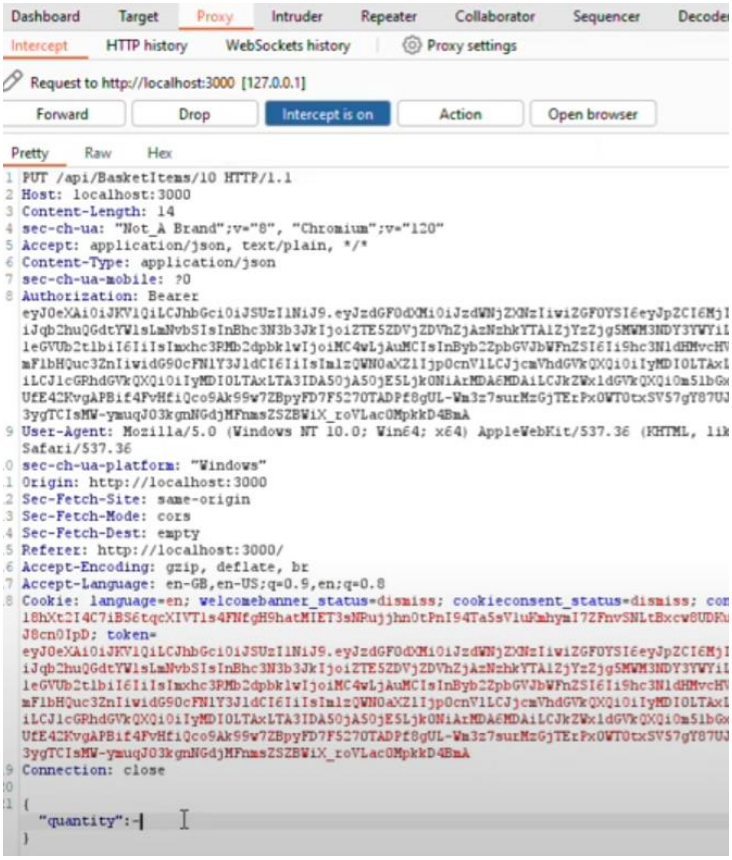
---

5. Payback Time (Place an Order that Makes You Rich)

Objective: Manipulate the order total to receive an amount of money instead of spending it.

Methodology:

1. **Tool Used:** Burp Suite pro.

2. **Steps:**
    1. **Add items to cart:**
        - add items to the shopping cart.
    2. **Intercept the purchase request:**
        - Enable Burp Suite's Intercept and click to purchase the items.
        - Capture the POST request containing the payment and order details.
    3. **Modify the total amount:**
        - In the intercepted request, modify the totalAmount or price field to a negative value or an extreme positive value (like -1000 or 9999999).
    4. **Send the request:**
        - Forward the modified request to the server and observe the response.
3. **Sample Burp Suite Output:**
    - 

4. **Manual Validation:**
    - Check the user's account balance or order summary after the request.
    - If the order results in a positive balance or refund instead of a charge, the vulnerability is confirmed.
5. **Conclusion:**
    - The ability to manipulate order totals demonstrates insufficient server-side validation, which could lead to financial loss or system abuse.

6. Forged Feedback (Improper Input Validation)

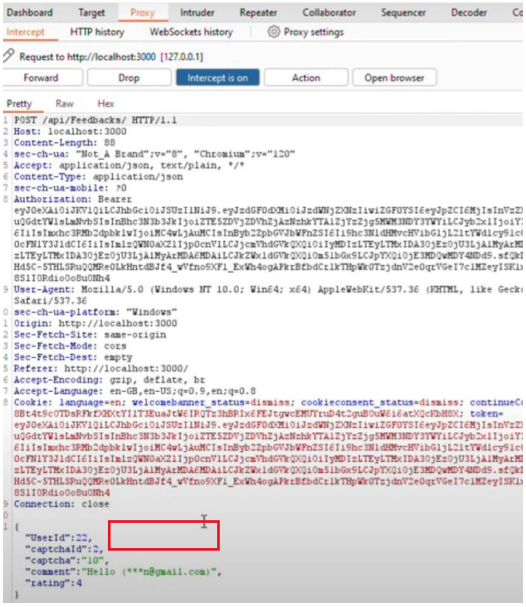
Vulnerability Description: Feedback can be submitted under another user's identity without authorization, demonstrating a lack of proper user validation.

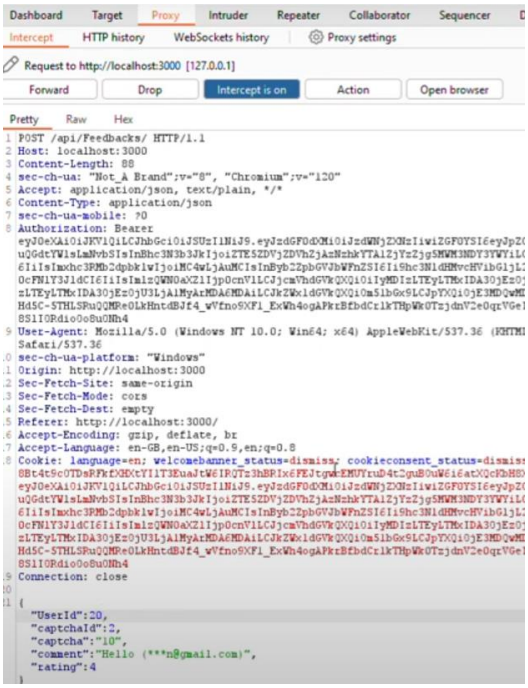
Tools Used:

- Burp Suite

Methodology:

1. **Submit Feedback:**
    - Navigate to the feedback submission form in the Juice Shop.
    - Submit a valid feedback entry, capturing the POST request in Burp Suite.
2. **Request Interception and Modification:**
    - In the intercepted request, modify the author or userId field to reflect a different user.
    - Forward the modified request to the server.
3. **Validation of Results:**
    - Check if the feedback appears under the different user's name.
    - Confirm whether the application allows feedback submission under unauthorized users without validation.
4. **Sample Burp Suite Output:**





Manual Validation:

- Check the feedback page to verify if the altered feedback appears under the spoofed user's name.
- Test with multiple user IDs to assess the scope of the vulnerability.

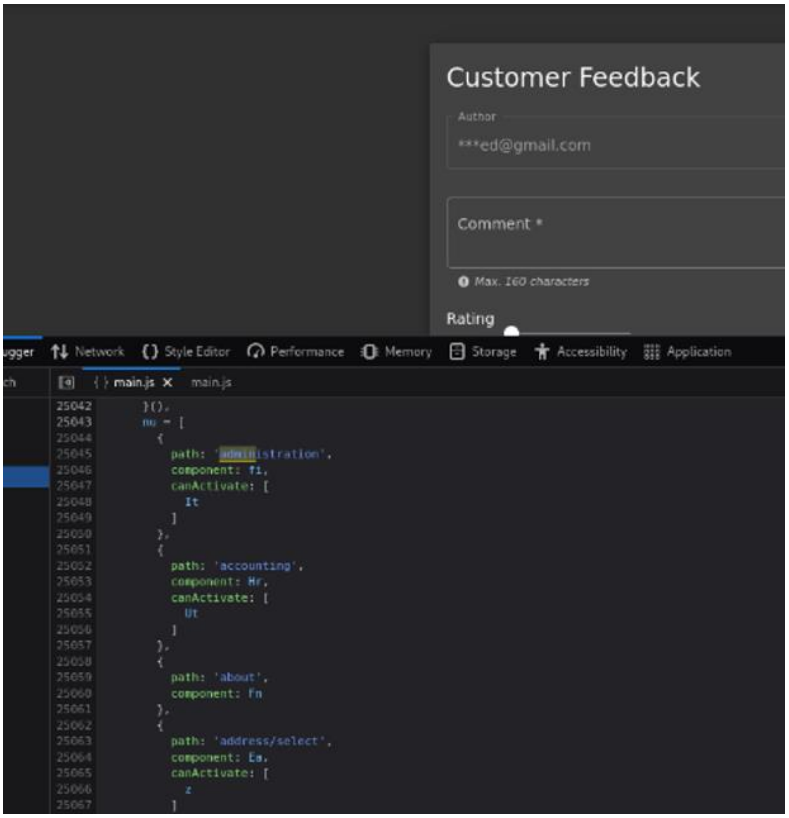
5. **Conclusion:**
   - The ability to forge feedback under different users reveals a critical input validation flaw on the server side, enabling impersonation.

7- **Information Disclosure (Place an Order that Makes You Rich)**

Objective: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

Methodology:

6. **Tool Used:** Browser Developer Tools
7. **Steps:**
   1. **navigate to main.js file**
   2. **search about admin**
   3. **you will find the administration path**
8. **Sample Burp Suite Output:**



9. **Manual Validation:**
    - Search about admin on the browser Developer Tool.
10. **Conclusion:**

Exposing sensitive data such as admin path in JavaScript files can allow attackers to discover backend infrastructure.

---

8. Broken Access Control (View Basket)

Objective:
We want to test if users can view someone else's basket which they shouldn't have access to.

Tools Used:

- Burp Suite

Steps:

1. Log in with a valid user account.
2. Intercept the request when clicking on the basket.
3. We will find in the response “id=6” (first hint).
4. In your browser, open developer tools.
5. Go to the Storage tab and click on session storage then click on localhost:3000.
6. We will find bid section which refers to “Basket ID”.
7. Change the value of the Basket ID.

8. Refresh the page to see if the contents of another user's basket are displayed.

Manual Validation:
If you can see or edit someone else's basket, that’s an access control flaw.

Conclusion:
This flaw would let anyone view or modify others' baskets, exposing sensitive details or even allowing changes to their orders.

---

9. Broken Access Control (Five-Star Feedback)

Objective:
We're testing to see if admins can delete positive feedback

Steps:

1. Login as an admin and go to the feedback section.
2. try to delete five star feedback.

Manual Validation:
If the five-star feedback gets deleted without proper authorization, so the system has a problem with the review process.

Conclusion:
Allowing easy deletion of positive feedback can be exploited to manipulate the reputation of products or services, which undermines the entire platform’s trustworthiness.

---

10. Improper Input Validation (Zero Stars)

Objective:
Check if the system allows invalid ratings, like "zero stars” which can be not good for the site’s rating system.

Tools Used:

- Burp Suite

Steps:

1. Submit any rating you want.
2. intercept the request by Burp Suite.
3. Modify the rating value to zero stars.
4. Then forward the request and see if the system accepts it.

Manual Validation:
If the system lets you submit a rating that’s outside the allowed range (like zero stars when it should be 1-5), it means input validation isn’t working correctly.

Conclusion:
This kind of flaw can weaken trust in the rating system, allowing malicious users to skew reviews and harm the integrity of the platform.

---

11. DOM-Based XSS

Objective:
We're testing if it's possible to insert malicious code through user inputs that could then be executed in the browser.

Steps:

1. Find a form or field that reflects user input back into the page.
2. Here our field is search bar
3. Enter a malicious script like <iframe src =”javascript:alert(‘xss’)”>, and submit it.
4. script is executed in the browser.

Manual Validation:
If you see the script execute (like a pop-up appears) then you've found a DOM XSS vulnerability.

Conclusion:
This flaw can be dangerous because it could allow attackers to steal users' session information or redirect them to malicious sites.

---

12. Security Misconfiguration (Error Handling)

Objective:
We want to see if the application leaks sensitive information through its error messages.

Steps:

1. trying to make an error by doing something invalid on the site.
2. By clicking on the account tab then privacy and security then request data erasure.
3. The juice shop sometimes gives an error message with a lot of details.
4. Look for any sensitive details, like server paths, software versions, or other technical data.

Manual Validation:
If the error message shows too much information like file paths or the exact software version that could help an attacker.

Conclusion:
Detailed error messages can give attackers the information they need to launch more advanced attacks. The system should only show simple error messages to avoid this.

13- Admin registration (Privilege escalation):

Objective:

Test if regular users can register as admins by exploiting weaknesses in the registration process.

Tools used:

Burp Suite

Steps:

6. **Registration**: create a new account as if you are a normal user.
7. **Capture the Request**: capture the POST request using Burp Suite.

8. **Use Repeater for Automation**: send the captured request to Burp Suite's **Repeater** module, where you can edit and resend it again.
9. **Editing the request:** discover a way to modify registration parameters (such as a hidden admin flag) during the registration process. edit the email in the request to create a new user's account, then edit the role of the user to admin.
10. **Analyze Server Response**: After resending the request, you will receive a valid response (201) Created, indicating that the new user was accepted by the server as an admin. You will then log in with the created admin user, and you will get a successful login.

Manual Validation:

If a regular user account can register with admin privileges by modifying the request, it indicates a privilege escalation vulnerability.

Conclusion:

This vulnerability allows attackers to gain administrative access, leading to full control over the system. It can result in unauthorized system changes, data breaches, and loss of sensitive information.

14- Privacy Policy Auto-Solve (Information Disclosure)

Objective:

Test if the "Privacy Policy" challenge is automatically marked as solved without user interaction.

Steps:

1. Log in to the application.
2. Observe the challenge status in the "Score-board".
3. Verify whether the "Privacy Policy" challenge is solved upon initial login, without any actions taken by the user

Manual Validation:

If the "Privacy Policy" challenge is marked as solved automatically, it indicates an issue with challenge validation.

Conclusion:

Automatically solving challenges compromises the integrity of the challenge system. This flaw could allow users to bypass learning objectives or potentially expose sensitive data.

15- Upload Type (Improper Input Validation)

Objective:

Test if the application allows file uploads of improper types, which could lead to code execution or other security risks.

Tools used:

Burp Suite

Steps:

1. **Visit the complaint section on the website**
2. **Upload a pdf file**
3. **Capture the request:** capture the submit request using Burp Suite
4. **Use Repeater for Automation**: send the captured request to Burp Suite's **Repeater** module, where you can edit and resend it again.
5. **Replace the file:** exchange the pdf file with jpeg content then change the file extension and send the request.
6. **Capture the Request**: capture the POST request using Burp Suite.
7. **Analyze Server Response**: After resending the request, you will receive a valid response (204) No content, indicating that the file is uploaded without any restricts.

Manual Validation:

If the server accepts files of improper types, it indicates a lack of proper input validation.

Conclusion:

Allowing dangerous file types to be uploaded can lead to malware injection or other security issues. This vulnerability exposes the server to risks of malicious file execution and Denial of Service (DoS) attacks.

16. Manipulate Basket (Broken Access Control)

Objective:

Test if users can manipulate their shopping basket contents by modifying parameters.

Tools used:

Burp Suite

Steps:

1. **Add items to your basket**
2. **Capture the request:** capture the add to basket request using Burp Suite

3. **Use Repeater for Automation**: send the captured request to Burp Suite's **Repeater** module, where you can edit and resend it again.
4. **Add item to your basket:** Add an item to your basket by editing the product ID and resend the request, you will get the product added successfully.
5. **Add an item to someone's basket:** Try to add an item to someone's basket by changing the basket ID, you will get a (401) Unauthorized response.
6. **Add an item to yourself first**: Try to add item to your basket and add another basket ID with different numbers, you will get a (200) OK response, and the item will be added successfully.

Manual Validation:

If basket content can be manipulated without proper validation, it indicates broken access control.

Conclusion:

This vulnerability allows users to manipulate their orders, potentially leading to financial exploitation and data integrity issues. Proper access control measures must be enforced to prevent unauthorized manipulation.

17. Repetitive Registration (Improper Input Validation)

Objective:

Test if the application improperly validates the password and confirmation password fields during user registration, allowing users to create an account with mismatched passwords.

Steps:

1. **Register a new user**
2. **Create two passwords:** Use different passwords in password and password confirmation fields.
3. **Sign in**: you will get a successful registration process.
4. **Log in**: If you logged in again using the password you have written in the password field, you will get a successful log in.

Manual Validation:

If the system allows registration despite mismatched passwords in the "Password" and "Confirm Password" fields, it indicates improper input validation. The system should ensure that both fields match before allowing account creation.

Conclusion:

This vulnerability allows users to create accounts with mismatched passwords, which weakens the security of the registration process. It can lead to user confusion and potential login issues, as the system may store the wrong password. Proper validation should ensure that the password and confirmation fields match before the account is created.

18. Negative quantity value (Improper Input Validation)

Objective:

Test if the application allows negative quantity values during the ordering process, which can be exploited to manipulate orders or cause errors in the system.

Tools used:

Burp Suite

Steps:

1. **Add items to your basket**
2. **Capture the Request**: capture the add to basket request using Burp Suite.
3. **Use Repeater for Automation**: send the captured request to Burp Suite's **Repeater** module, where you can edit and resend it again.
4. **Editing the request:** Edit the quantity value into a negative value.
5. **Analyze Server Response**: After sending the request, you will get a (200) OK request indicating that the item is added successfully in a negative quantity.
6. **Negative total price**: The application allowed the purchase of a negative quantity of items, resulting in a negative total price.

Manual Validation:

If the system processes the negative quantity without rejection, it indicates improper input validation.

Conclusion:

Allowing negative quantities can lead to order manipulation and financial exploitation. Attackers could reduce the price of an order or even earn credit from the system. Proper input validation should be implemented to ensure that only positive integers are allowed for quantities.

This concluded the vulnerability assessment methodology portion of this report.