رواد مصر الرقمية

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

OWASP Juice Shop-Report

Supervisor: Khaled Taha

Team-Members:

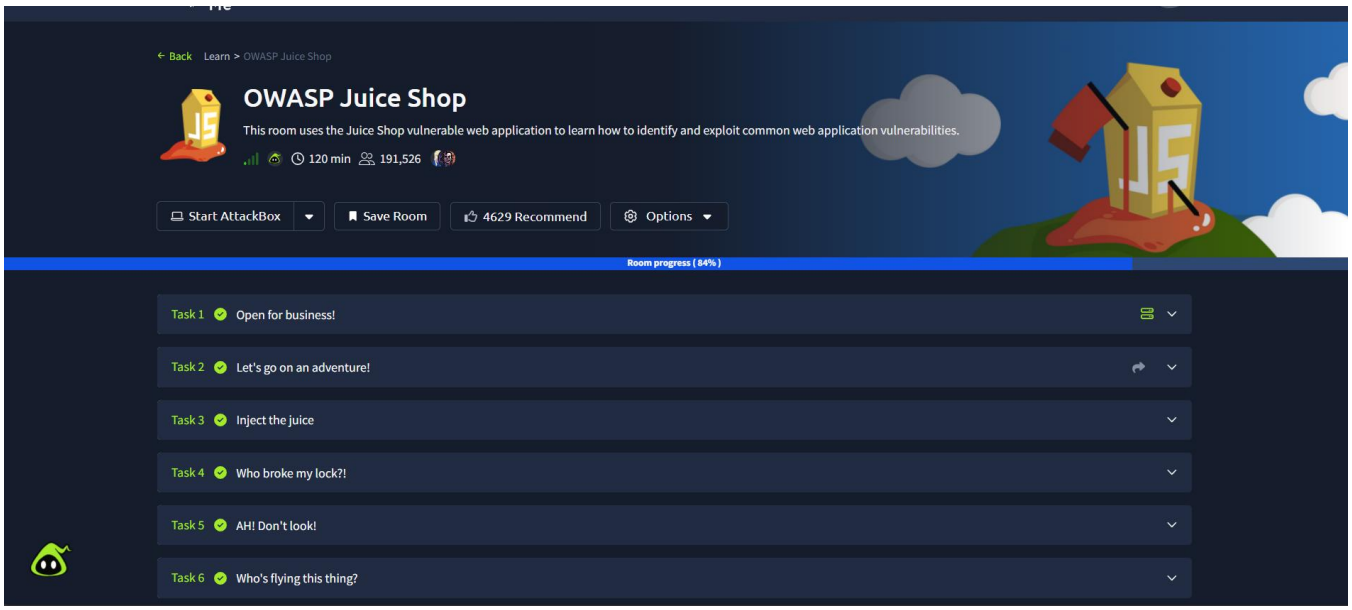
- Hatem Harby Mohamed (Leader)
- Mohamed Ahmed Othman Mahmoud
- Ammer Ahmed Mohammed Eid
- Abdul Rahman Nasr Rushdie
- Abdul Hamid Hani Mohamed
- Ahmed Ezzat

1. Executive Summary

This report presents the results of a penetration test performed on OWASP Juice Shop, a deliberately insecure web application used for security training.
The objective of the assessment was to evaluate the security posture of the application by identifying and exploiting common vulnerabilities that reflect real-world threats.

Several critical issues were discovered, including **SQL Injection**, **Broken Access Control**, **Cross-Site Scripting (XSS), Account takeover**, and **Security Misconfigurations**.
If exploited by an attacker, these vulnerabilities could lead to data leakage, unauthorized access, account takeover, and full compromise of the application.

Overall, the security posture of the application is considered **High Risk**.



2. Scope of Work

Target: OWASP Juice Shop (Local deployment)
URL: http://localhost:3000
Testing Type: Black-box / Manual
Allowed Techniques: Web vulnerability scanning, manual exploitation, injection attacks, enumeration
Tools Used:

- Burp Suite Community
- Dirsearch
- Browser Developer Tools

3. Methodology

The testing methodology followed the guidelines of:

- OWASP Top 10
- Manual testing and fuzzing

The approach consisted of:

1. **Reconnaissance** – Gathering information about the application
2. **Enumeration** – Identifying input fields, parameters, and hidden functionalities
3. **Vulnerability Testing** – Manual and automated scanning
4. **Exploitation** – Validating vulnerabilities
5. **Reporting** – Documenting findings and recommendations

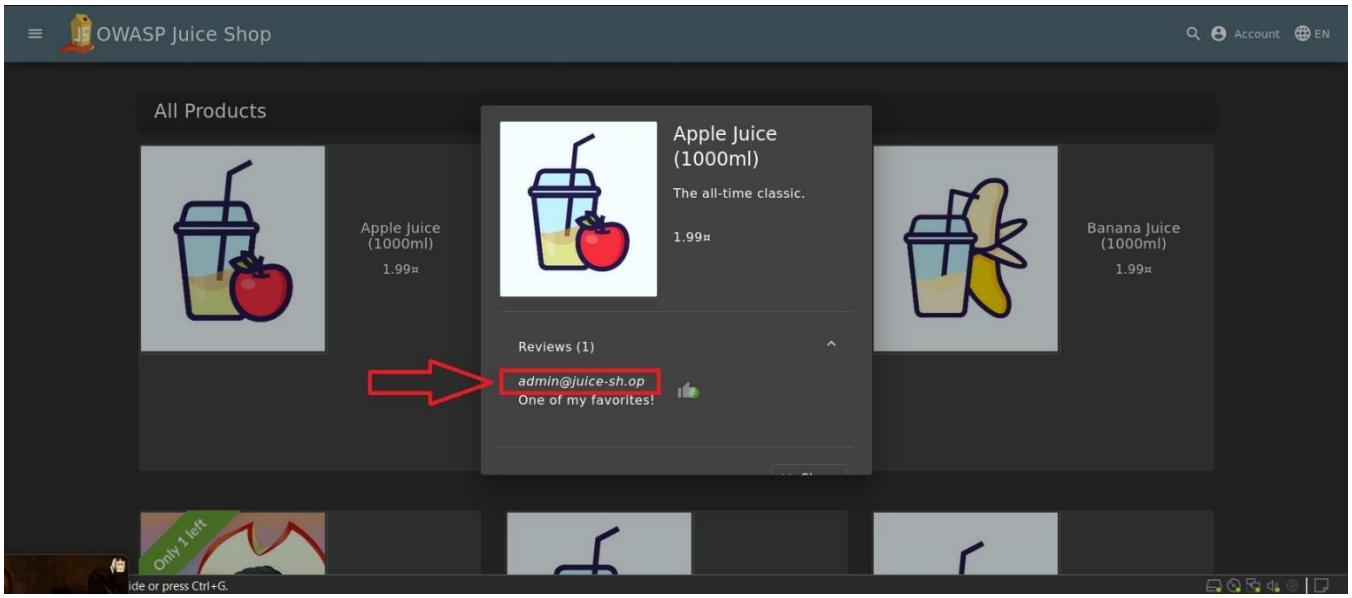
4. Findings Summary

Vulnerability	Risk Level	Status
SQL Injection	**High**	**Exploited**
Broken Authentication	**High**	**Exploited**
Broken Access Control	**High**	**Exploited**
Cross-Site Scripting (XSS)	**Medium**	**Exploited**
Sensitive Data Exposure	**Medium**	**Confirmed**
Logic Vulnerability	**High**	**Confirmed**
Cross-Site Request Forgery	Medium	**Confirmed**

Gathering information about the application:

During the information gathering, a collection of emails belonging to users was gathered, including sensitive emails such as the admin's email:

- admin@juice-sh.op
- bender@juice-sh.op
- stan@juice-sh.op
- uvogin@juice-sh.op
- jim@juice-sh.op
- mc.safesearch@juice-sh.op
- accountant@juice-sh.op
- bjoern@owasp.org
- morty@juice-sh.op

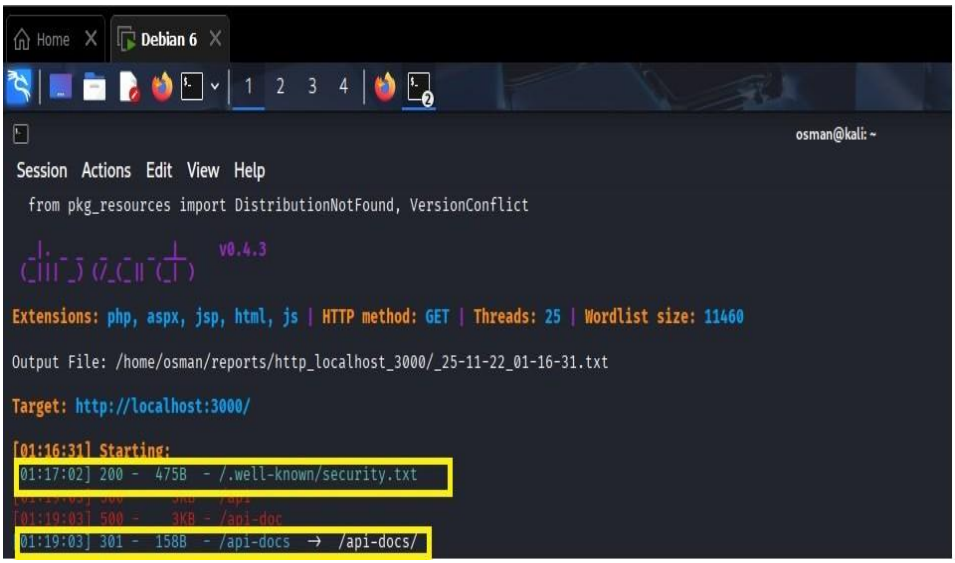


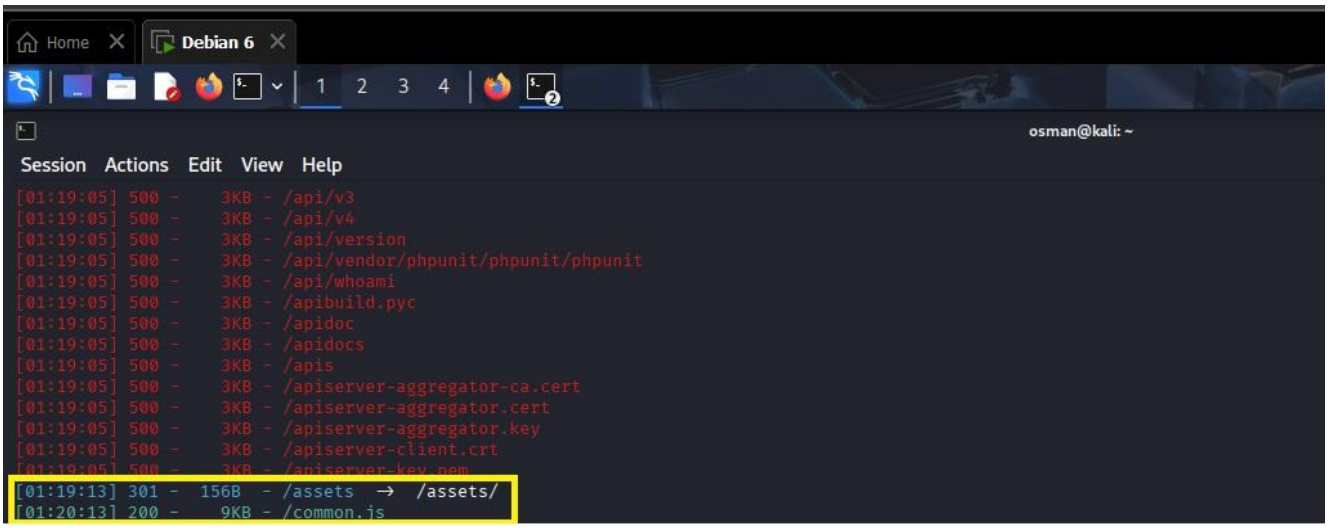
رواد مصر الرقمية

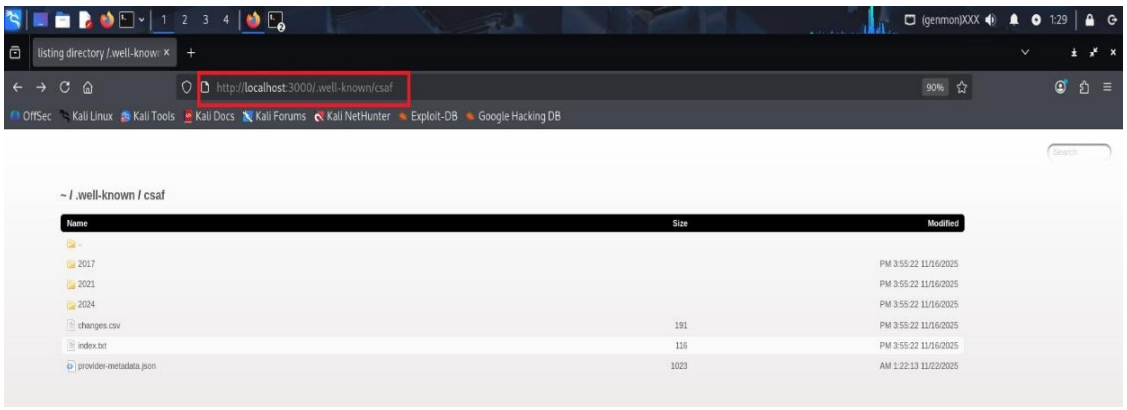
Gathering information about the application:

Using **dirsearch**, a set of paths was discovered :

- /.well-known/security.txt
- /api-docs/
- /assets/
- /common.js
- /ftp







Sensitive Data Exposure

Definition:

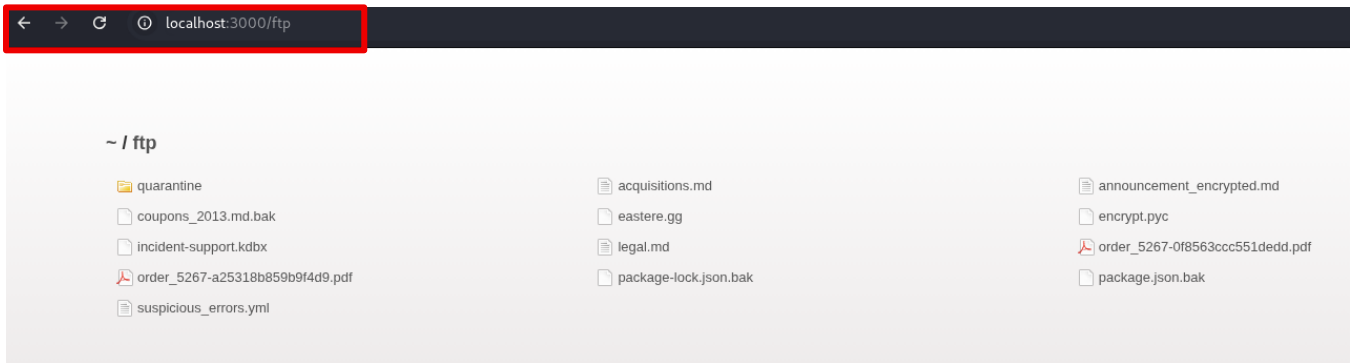
Sensitive Data Exposure occurs when an application accidentally reveals confidential information due to weak protections or misconfigurations.

Risk Level: Medium
Affected File: /ftp directory

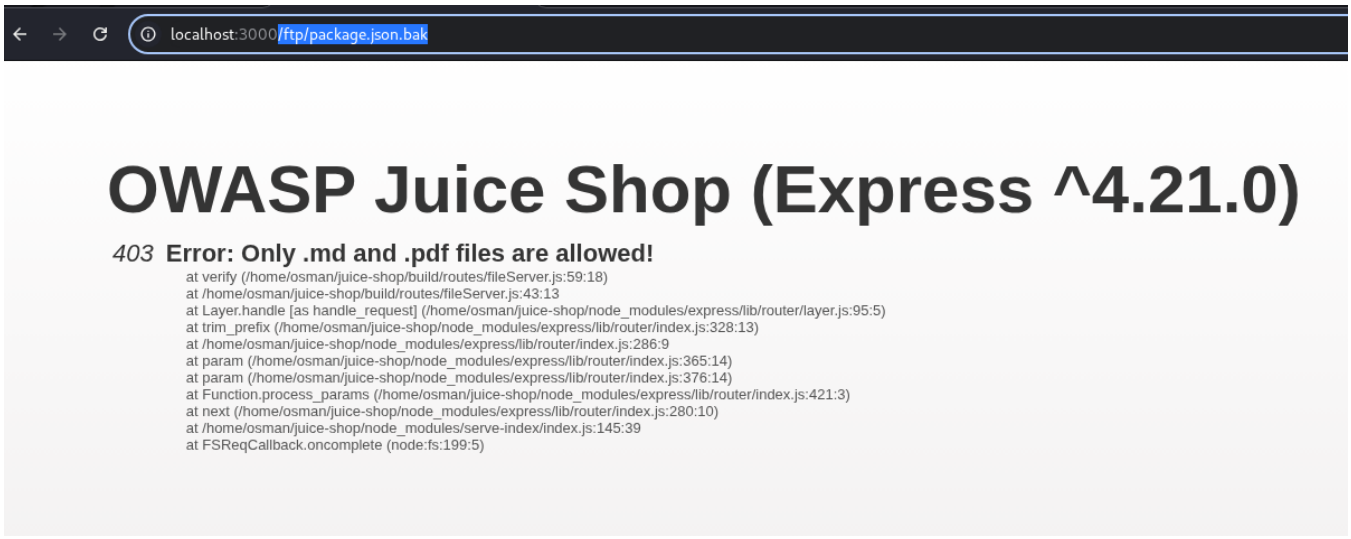
Description:

After Gathering information steps by dirsearch I try all endpoints to discover any important information

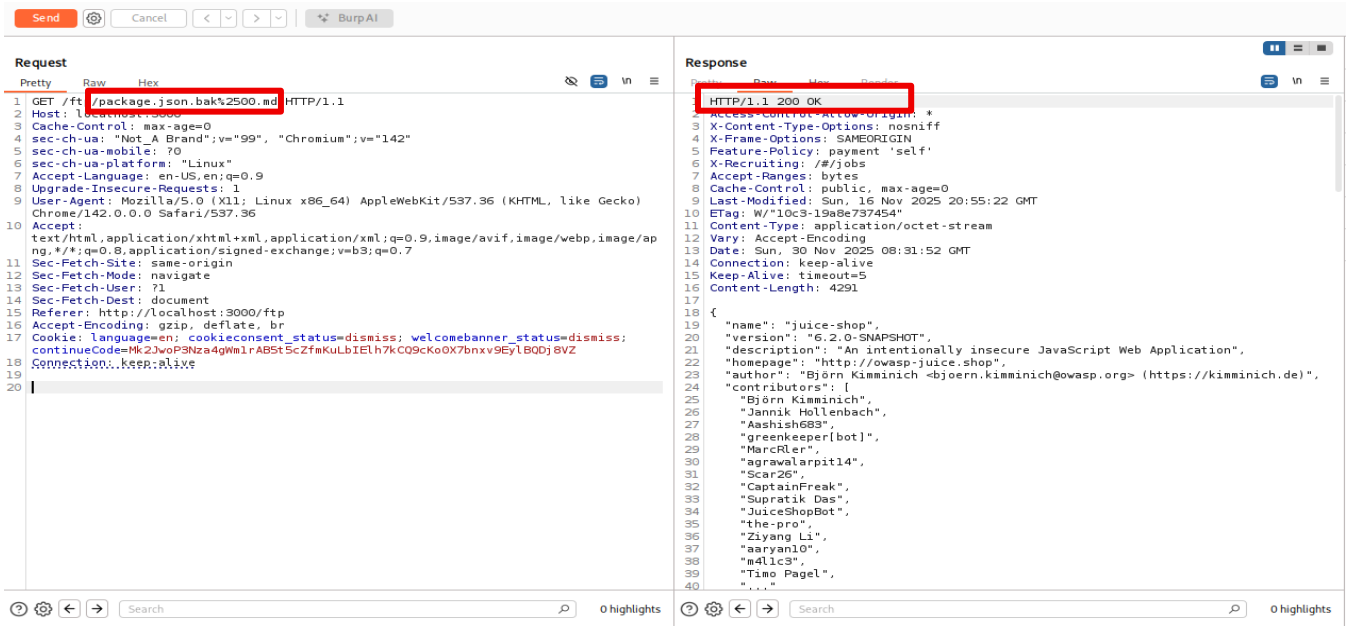
The application exposes backup files containing sensitive information

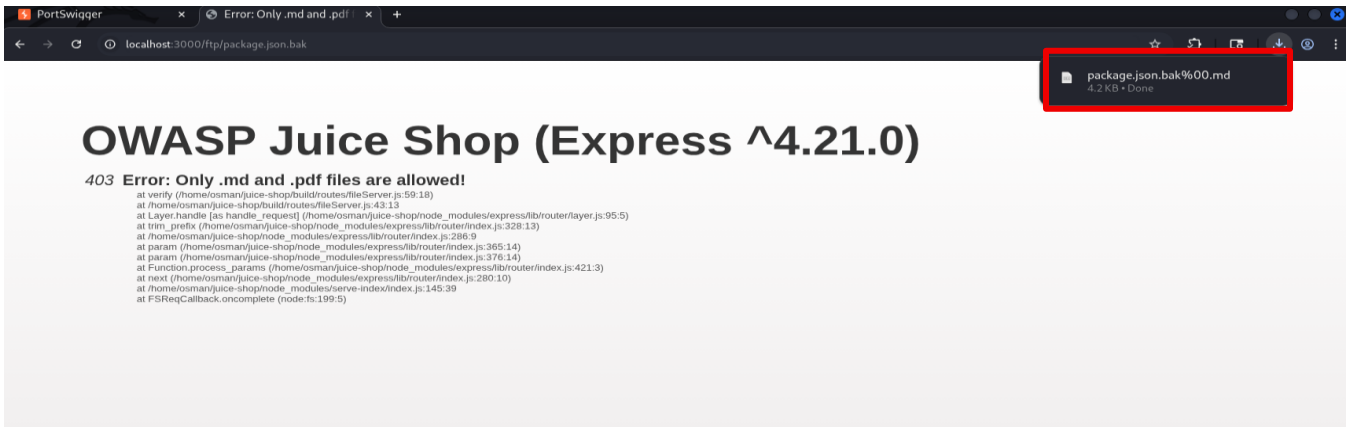


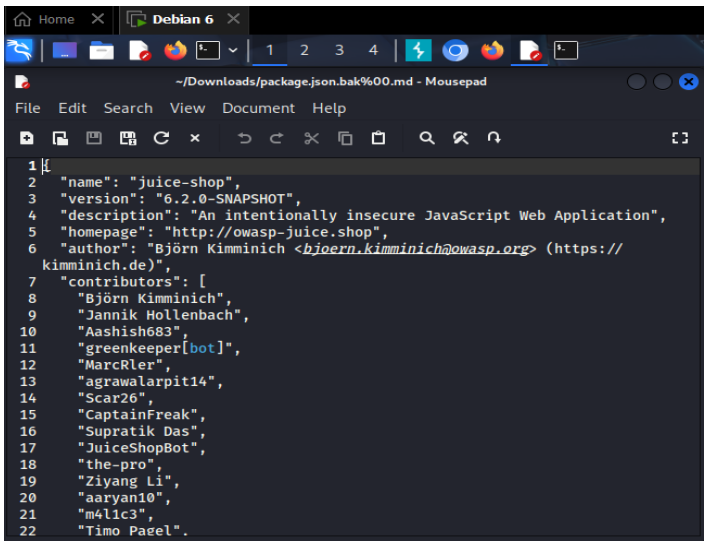
Now we try **/ftp/package.json.bak**



Now to download file we use nullbit:





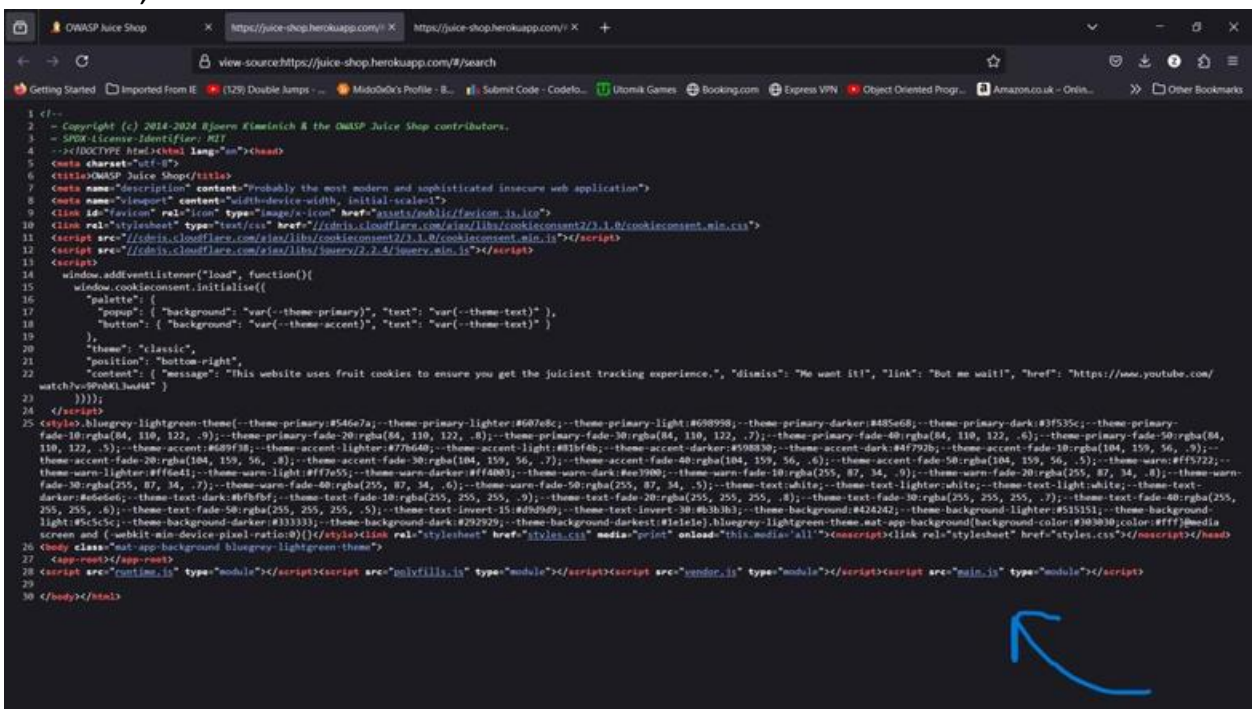


Data Leak refers to the unauthorized exposure or disclosure of sensitive information, such as personal data, financial records, credentials, or proprietary business information. This can occur through various means, including accidental publication, inadequate security measures, vulnerabilities in software applications, or intentional malicious actions.

Risk Level: critical

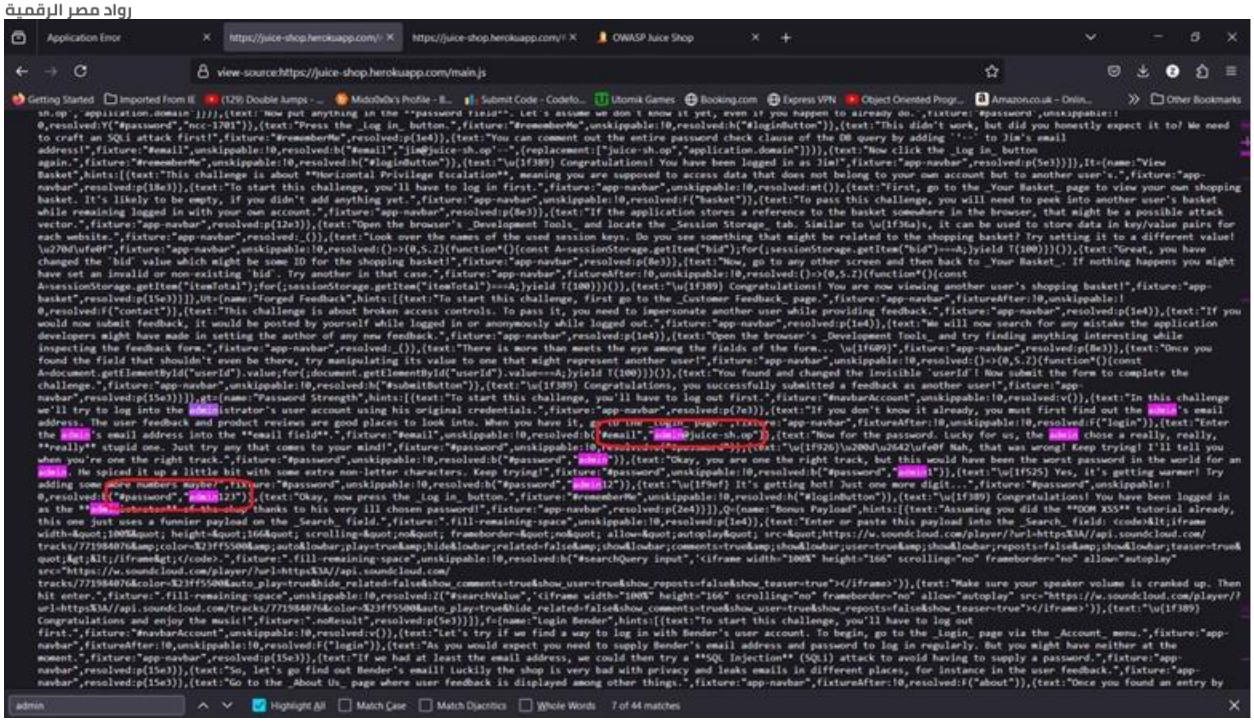
Access the main.js File

1. Open your web browser.
2. Navigate to the URL: https://juice-shop.herokuapp.com/main.js.
3. This file typically contains JavaScript code that may include various functions, variables, or even hardcoded credentials.



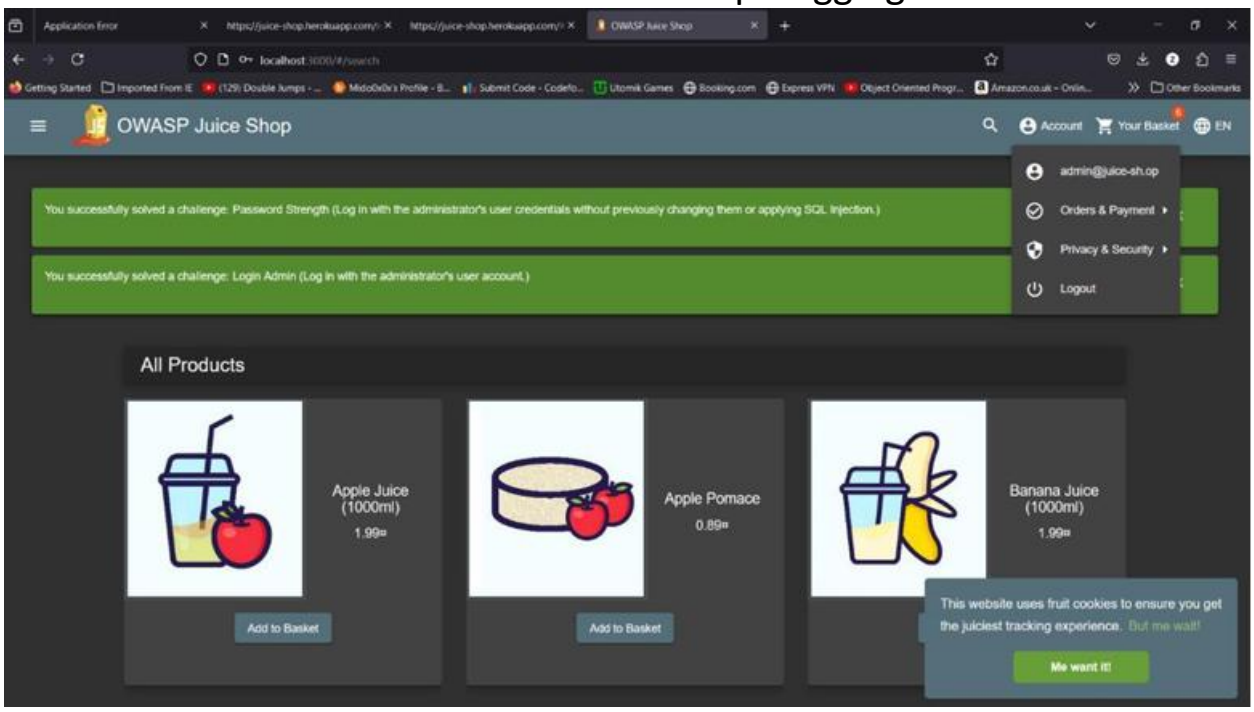
2: Search for Sensitive Information

- Once the main.js file is loaded, use your browser's search functionality (usually Ctrl + F or Cmd + F).
- Search for keywords that could indicate sensitive information, such as:
  - Password
  - Admin
  - Administrator
  - Email
  - credential
- Review any occurrences of these terms to identify potential admin credentials or other sensitive information.



3: Test the Found Admin Credentials:

- Navigate to the Juice Shop application login page or admin panel.
- Enter the found credentials to attempt logging in as an admin.



Findings

1. Hardcoded Credentials in JavaScript: The main.js file was found to contain hardcoded credentials, such as admin usernames and passwords. This practice exposes sensitive information directly in the client-side code, making it accessible to anyone who can view the source.

2. Risk of Unauthorized Access: The presence of these credentials poses a significant security risk, as unauthorized users can exploit this vulnerability to gain admin access to the application, potentially leading to data breaches or manipulation of sensitive information.

3. Insufficient Security Measures: The existence of hardcoded credentials indicates a lack of proper security measures and practices in the development lifecycle, including inadequate protection for sensitive data.

5. Detailed Findings:

5.1 SQL Injection:

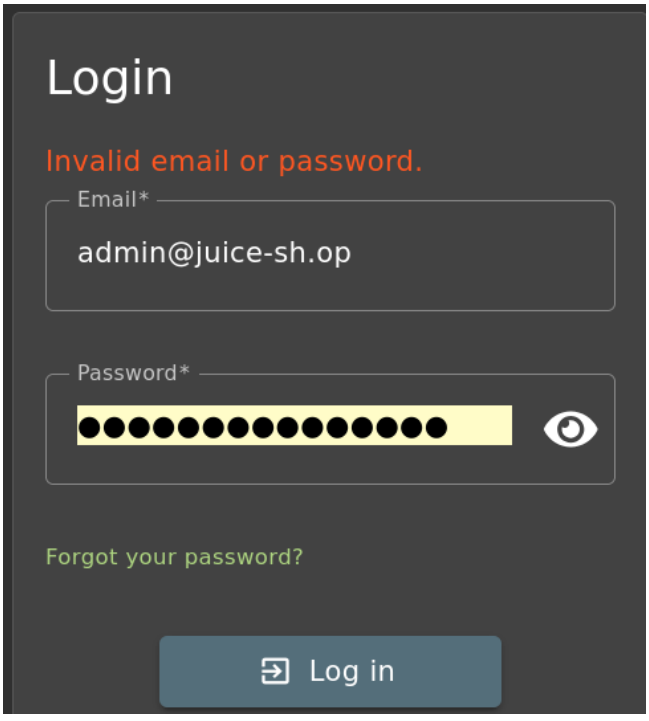
Definition:

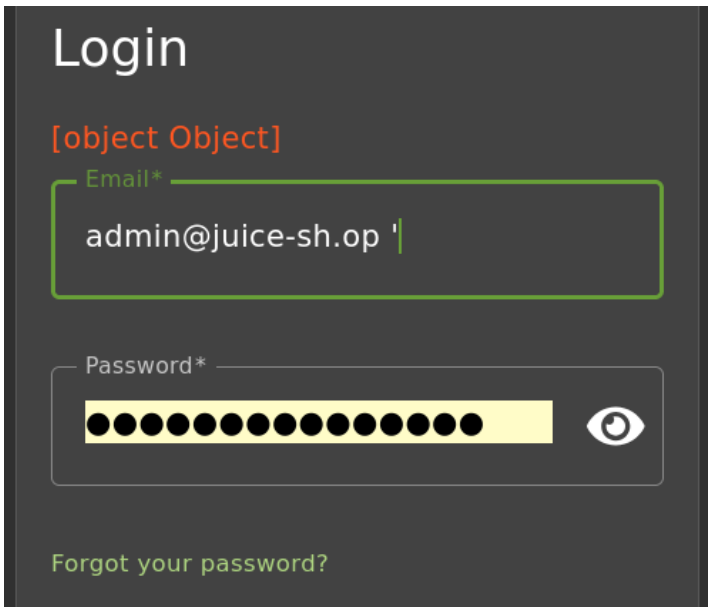
SQL Injection is a web vulnerability that allows an attacker to interfere with the queries an application makes to its database by injecting malicious SQL commands.

Risk Level: High
Affected URL: http://localhost:3000/login

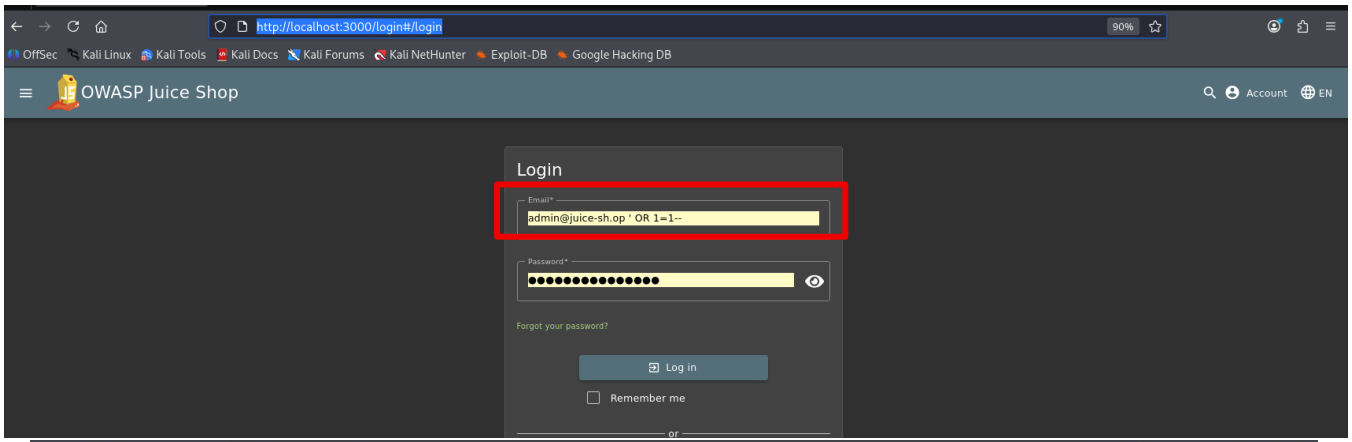
Description:

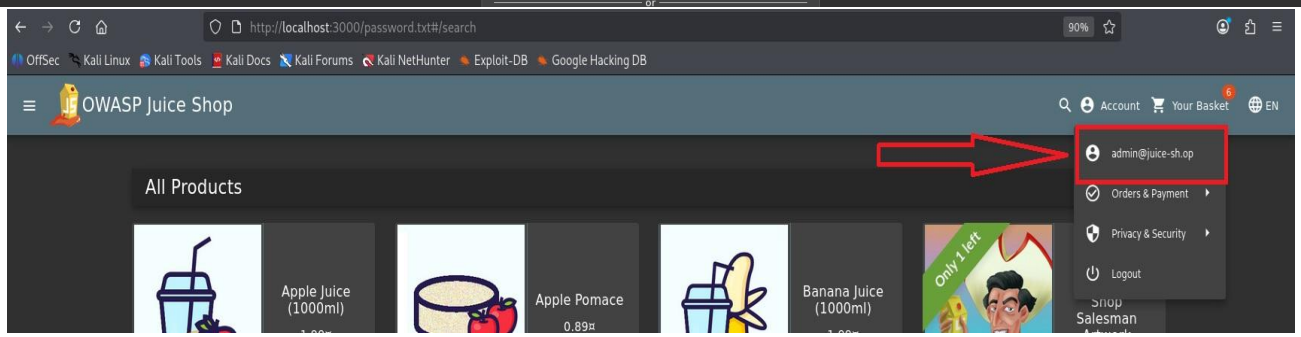
The login functionality is vulnerable to SQL Injection due to improper input validation (email and password)

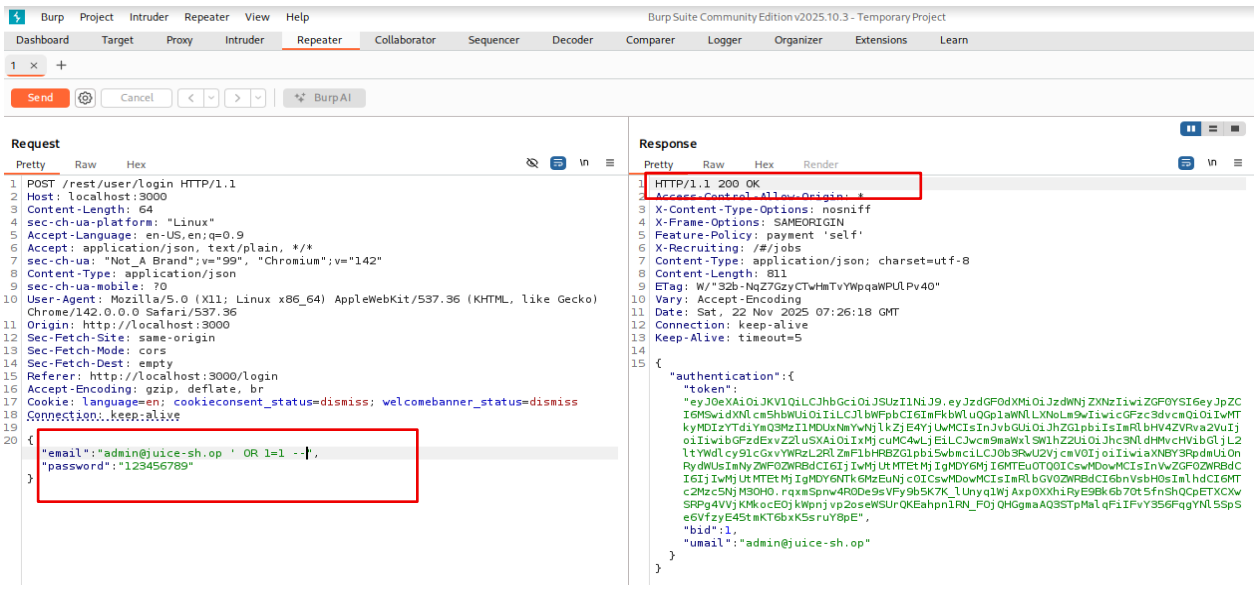




Proof of Concept (Payload): ' OR 1=1 --







Impact:

1-Possibility of extracting account's data

2-Can lead to full database compromise

Recommendation:

Use parameterized queries

Apply server-side input validation

Filter user input

5.2 Broken Authentication:

We were able to find a **Broken Authentication vulnerability** by :

- Brute Force Attack
- Account Takeover

Brute Force Attack:

Definition:

Brute Force is an attack technique where an attacker repeatedly tries different username and password combinations until the correct credentials are found, exploiting weak authentication controls.

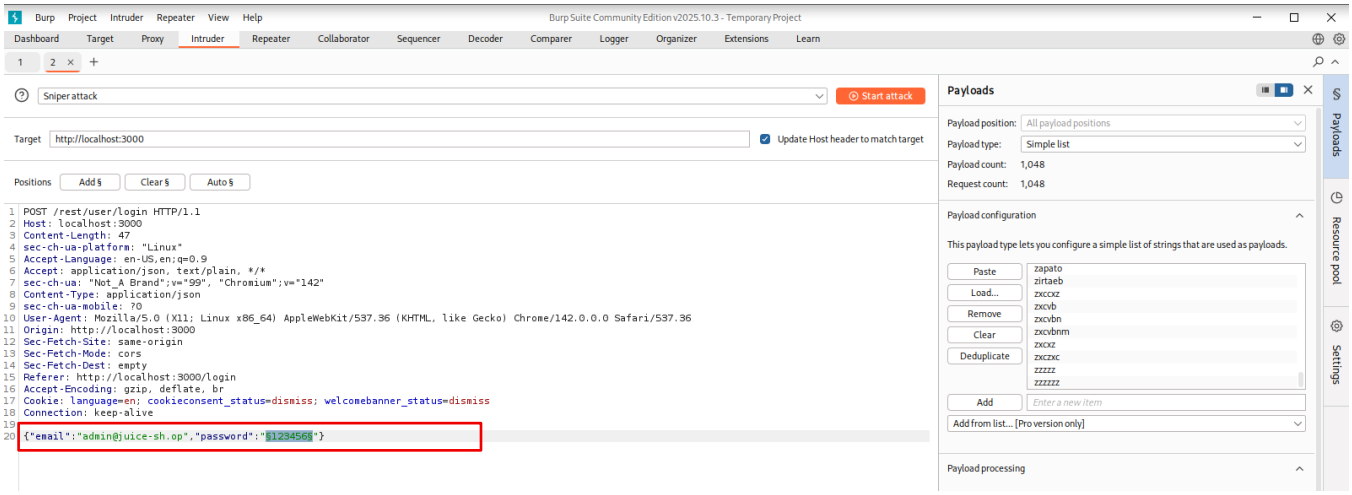
Risk Level: High

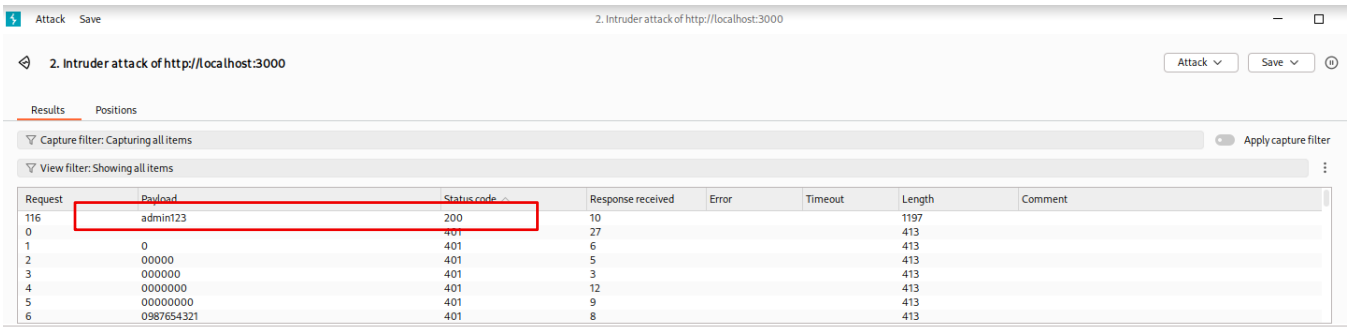
Affected Page: /login

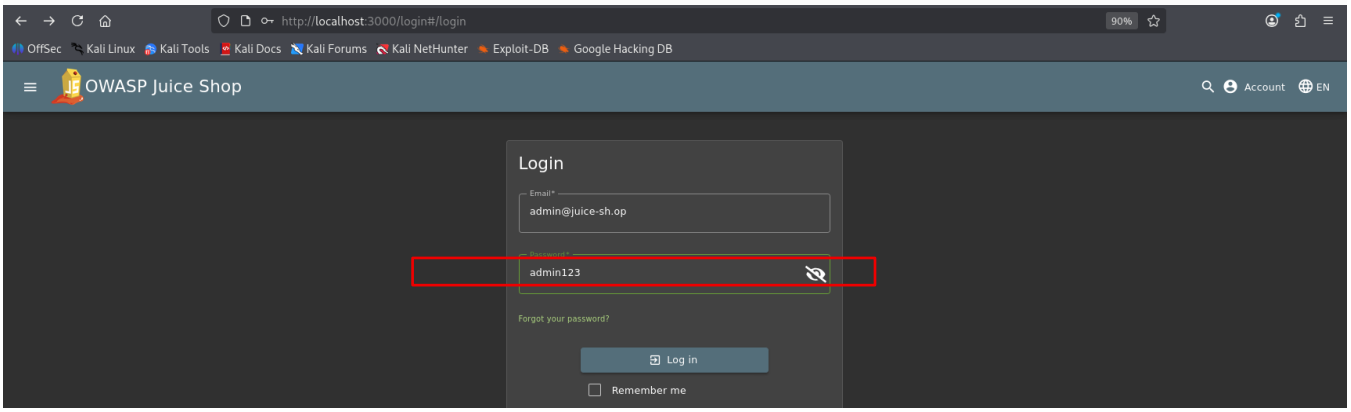
Description:

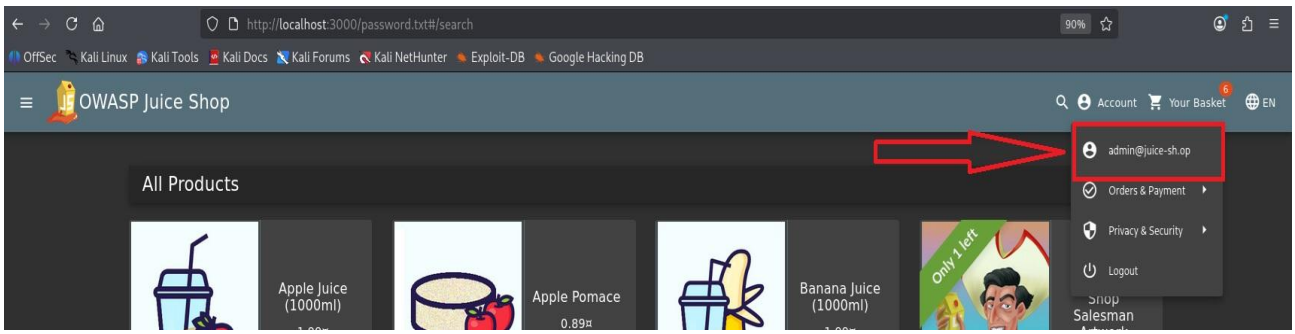
The login functionality is vulnerable to brute force attacks due to the absence of rate limiting, CAPTCHA, and account lockout mechanisms. An attacker can repeatedly attempt multiple username and password combinations without restriction, allowing unauthorized access through credential guessing.

Proof of Concept:









Impact:

- Unauthorized access to user accounts
- full account takeover
- Exposure of sensitive user information
- Ability for attackers to escalate privileges if high-value accounts (e.g., admin) are compromised
- Increased risk of automated credential stuffing attacks
- Compromise of the entire system if administrative credentials are guessed

---

Recommendation:

- Implement rate limiting (e.g., block or delay after multiple failed attempts)
- Add account lockout after repeated failed login attempts
- Enable Multi-Factor Authentication (MFA)
- Use CAPTCHA to prevent automated attacks
- Enforce strong password policies
- Monitor and alert on suspicious login attempts
- Limit login attempts per IP and per username

Account Takeover:

Definition:

Account Takeover (ATO) is a security breach where an attacker gains unauthorized access to a user's account by exploiting weaknesses in authentication or access control mechanisms.

Risk Level: High

Affected Page: /forgot-password

Description:

The application is vulnerable to Account Takeover through the password reset functionality. The security question used during the reset process is easily guessable, allowing an attacker to provide the correct answer without legitimate knowledge of the user. As a result, an attacker can reset the victim's password and gain full access to their account.

Proof of Concept:

From information gathering steps we use all emails we found to access any account and we collect information about each user from different resource like Wikipedia

We found information this email

- jim@juice-sh.op

Contents hide

(Top)
Depiction
Kelvin Timeline
Development
Conception and television
Films
Franchise "reboot"
Star Trek: Strange New Worlds
Reception
Cultural impact
Fan productions
Star Trek: New Voyages
Star Trek Continues
Legacy
References
External links

promoted to lieutenant junior grade and returned to Starfleet Academy as a student instructor.[4] According to a friend, students could either "think or sink" in his class, and Kirk himself was "a stack of books with legs".[6] Upon graduating in the top five percent, Kirk was promoted to lieutenant and served aboard the USS *Farragut*.[4] While assigned to the *Farragut*, Kirk commanded his first planetary survey and survived a deadly attack by a bizarre cloud-like creature that killed a large portion of the *Farragut*'s crew,[4] including his commanding officer, Captain Garrovick. Kirk blamed himself for years for hesitating to fire his assigned weapons upon seeing the threat until a later encounter with the creature showed that firing immediately with conventional weapons would have been useless.

Publicity photo of William Shatner as Kirk, alongside Leonard Nimoy as Mr. Spock

Kirk became Starfleet's youngest starship captain after receiving command of the USS *Enterprise* for a five-year mission,[4] three years of which are depicted in the original *Star Trek* series (1966–1969).[7] Kirk's most significant relationships in the television series are with first officer Spock and chief medical officer Dr. Leonard "Bones" McCoy.[8] McCoy is someone to whom Kirk unburdens himself and is a foil to Spock.[9] Robert Jewett and John Shelton Lawrence's *The Myth of the American Superhero* describes Kirk as "a hard-driving leader who pushes himself and his crew beyond human limits".[10] Terry J. Erdman and Paula M. Block, in their *Star Trek 101* primer, note that while "cunning, courageous and confident", Kirk also has a "tendency to ignore Starfleet regulations when he feels the end justifies the means"; he is "the quintessential officer, a man among men and a hero for the ages".[11]

Although Kirk throughout the series becomes romantically involved with various women, when confronted with a choice between a

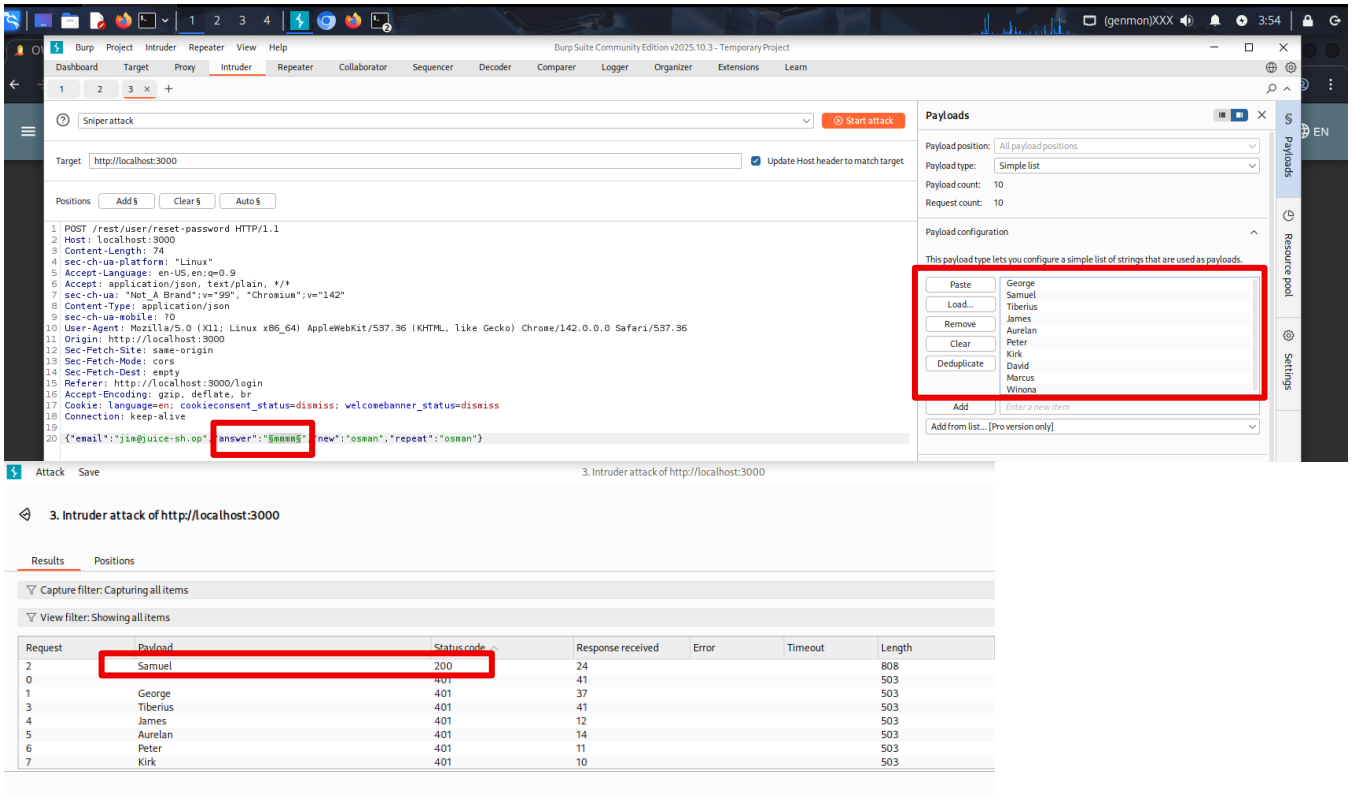
Nickname	Jim
Title	Cadet Ensign Lieutenant Commander Captain Admiral
Position	Chief of Starfleet Operations USS *Enterprise*: Commanding officer USS *Enterprise-A*: Commanding officer
Affiliation	United Federation of Planets Starfleet
Family	George Samuel Kirk Sr. (father) Winona Kirk (mother) George Samuel Kirk Jr. (brother) Tiberius Kirk (grandfather) James (maternal grandfather) Aurelan Kirk (sister-in-law) Peter Kirk (nephew) 2 other nephews
Children	David Marcus
Origin	Iowa, United States, Earth

Appearance hide

Text
Small
Standard
Large

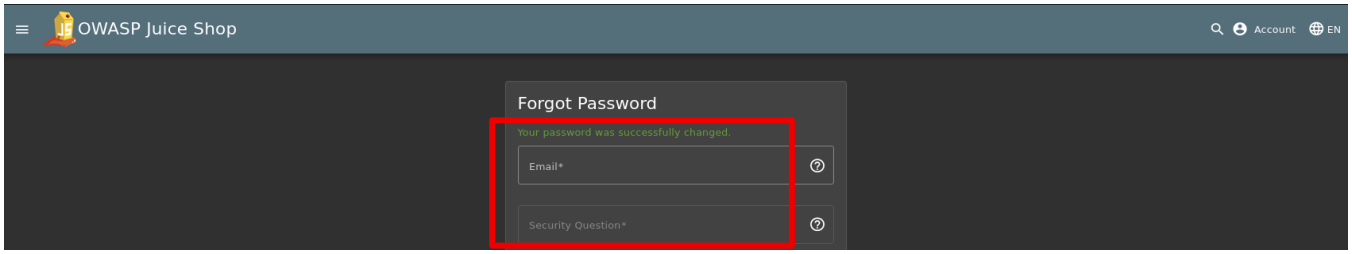
Width
Standard
Wide

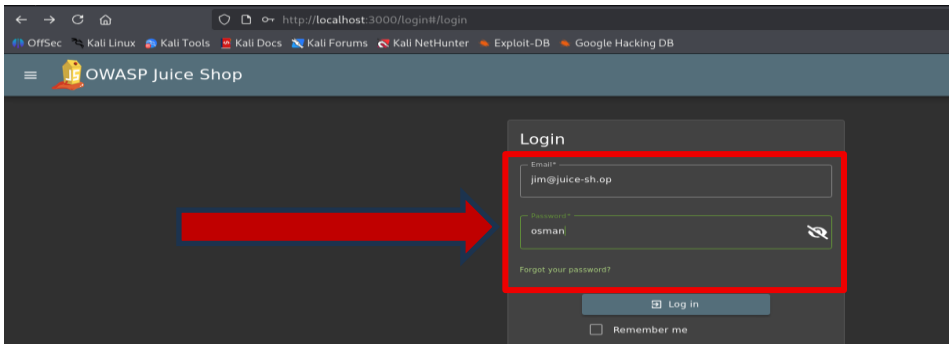
Color (beta)
Automatic
Light
Dark

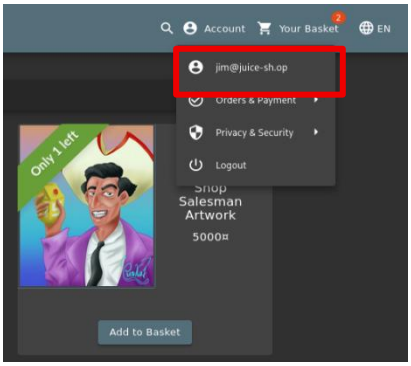


The answer of security question is: **Samuel**

New Password is : **osman**







Impact:

- Full compromise of victim user accounts
- Unauthorized access to personal or sensitive information
- Ability to perform actions on behalf of the victim
- Potential privilege escalation if admin or high-privilege accounts are taken over
- Fraud, data manipulation, or deletion of user data
- Loss of trust and severe security implications for the system

Recommendation:

- Strengthen authentication mechanisms (secure password reset, MFA, token validation)
- Enforce strict access control checks on all user actions
- Implement rate limiting and brute force protection
- Use email or phone verification when resetting passwords
- Encrypt and securely store credentials
- Monitor suspicious login behavior and notify users of unusual activit

5.3 Broken Access Control:

Definition:

Broken Access Control happens when an application does not correctly enforce user permissions, allowing attackers to access or modify data they shouldn't be able to.

Types:

- **Horizontal Privilege Escalation:**Occurs when a user can perform an action or access data of another user with the same level of permissions.
- **Vertical Privilege Escalation:**
  Occurs when a user can perform an action or access data of another user with a higher level of permissions.

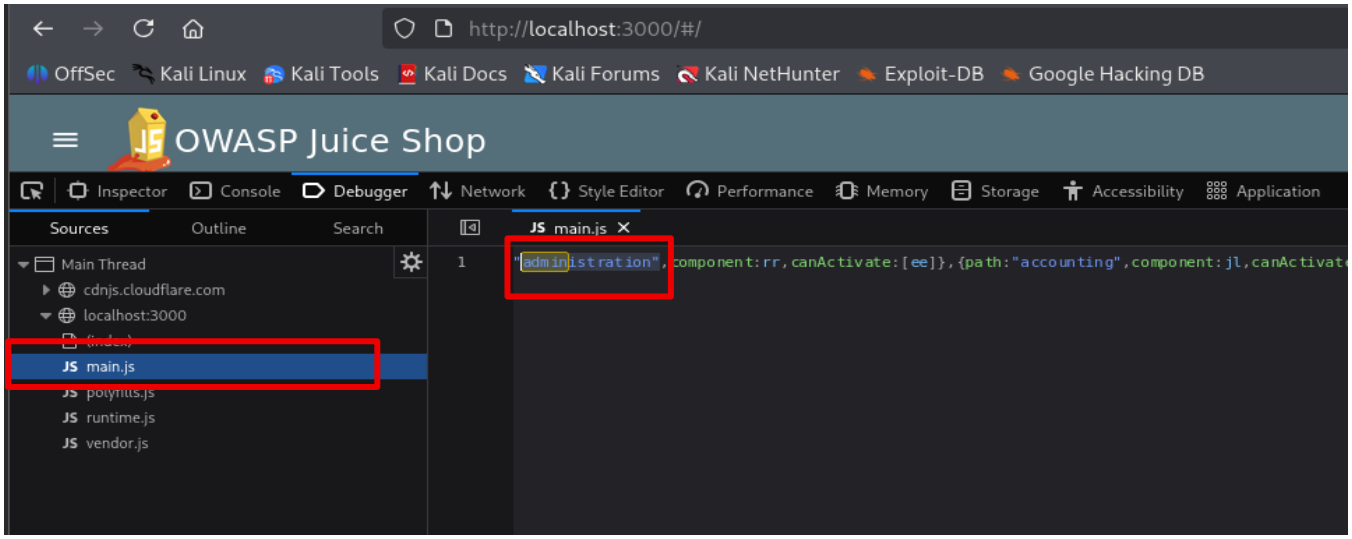
Risk Level: High

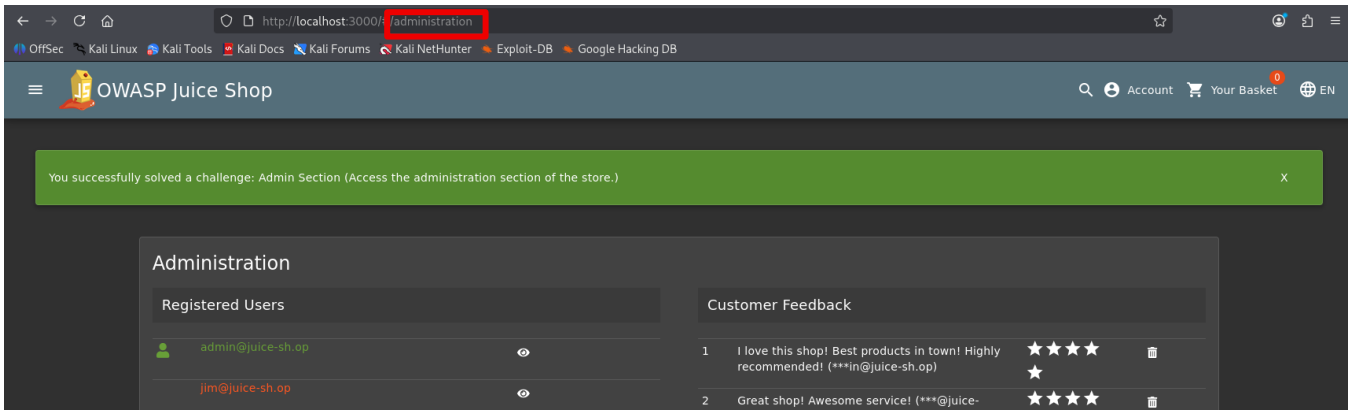
Description: data can be accessed without proper authorization check

Proof of Concept for Vertical Broken Access Control:

We user Browser tool:

- Inspect
- Debugger
- Main.js
- Search about "administration"



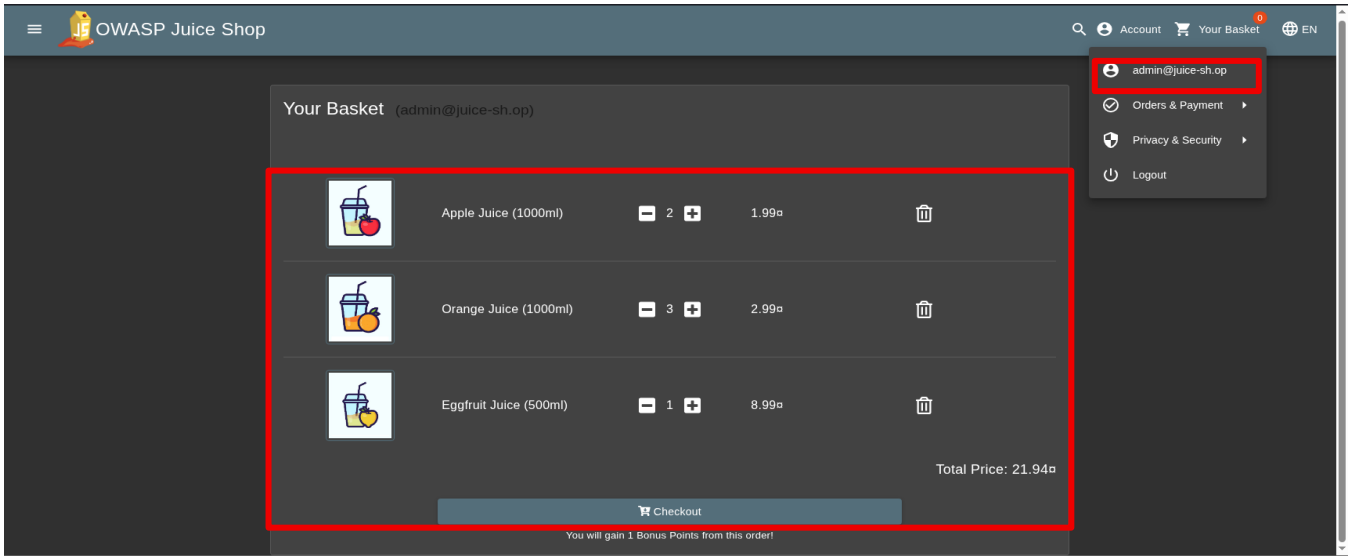


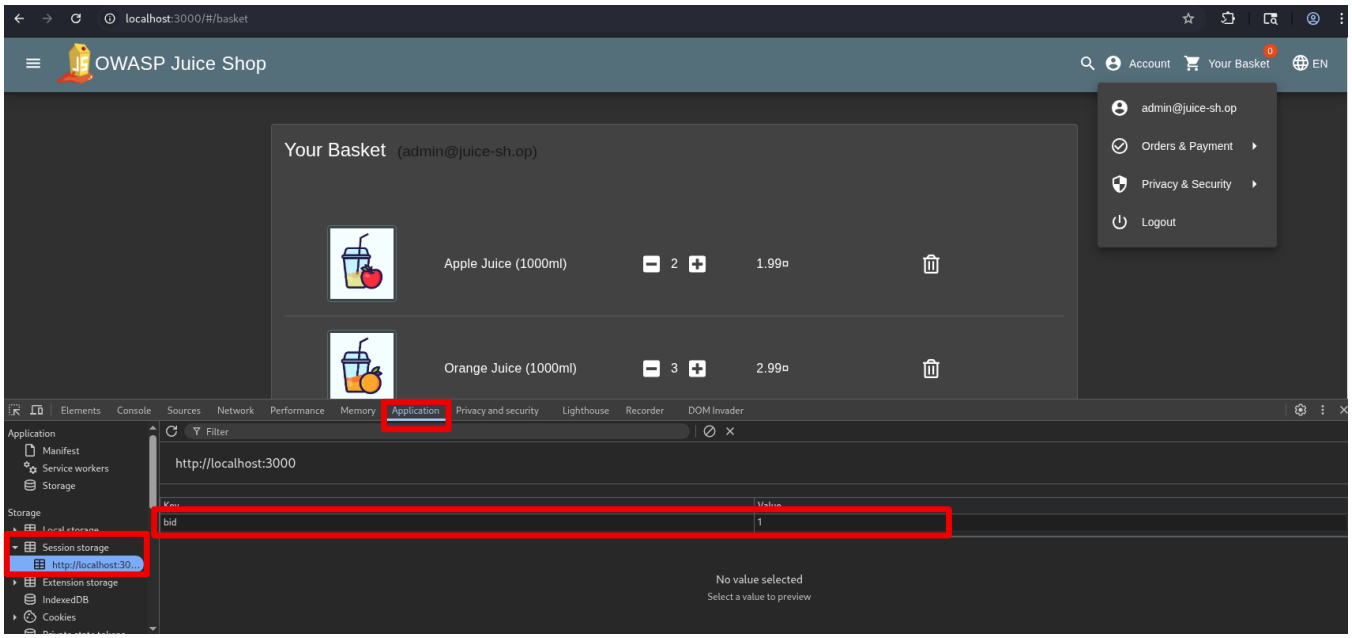
Proof of Concept for Horizontal Broken Access Control:

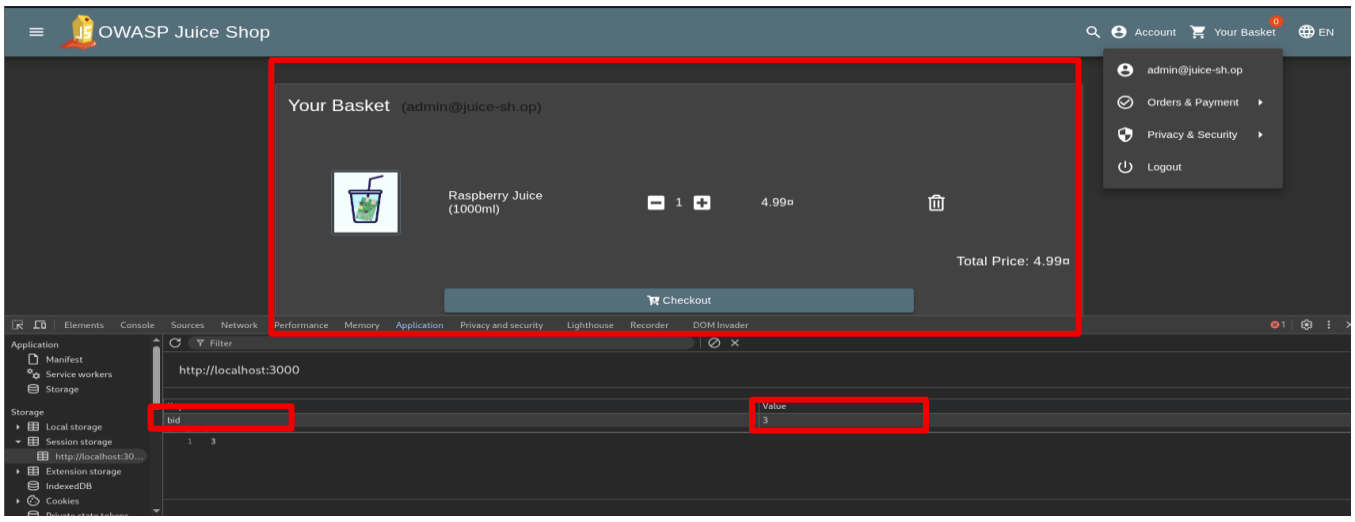
Endpoint: rest/basket/2

by Browser tool:

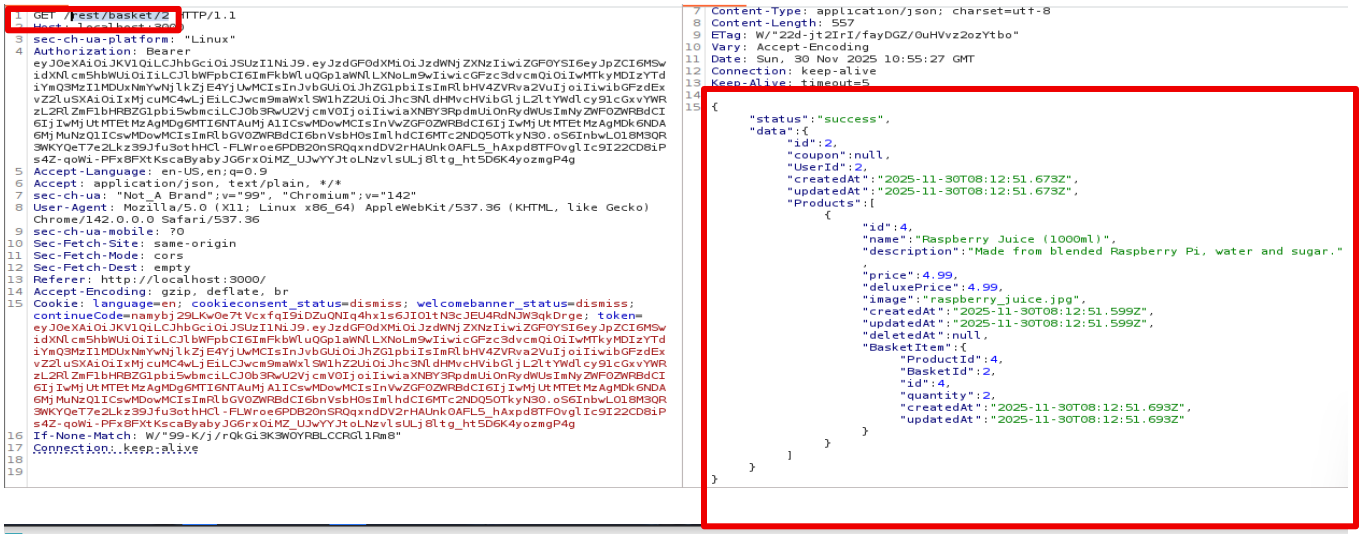
- Inspect
- Application
- Session storage
- bid

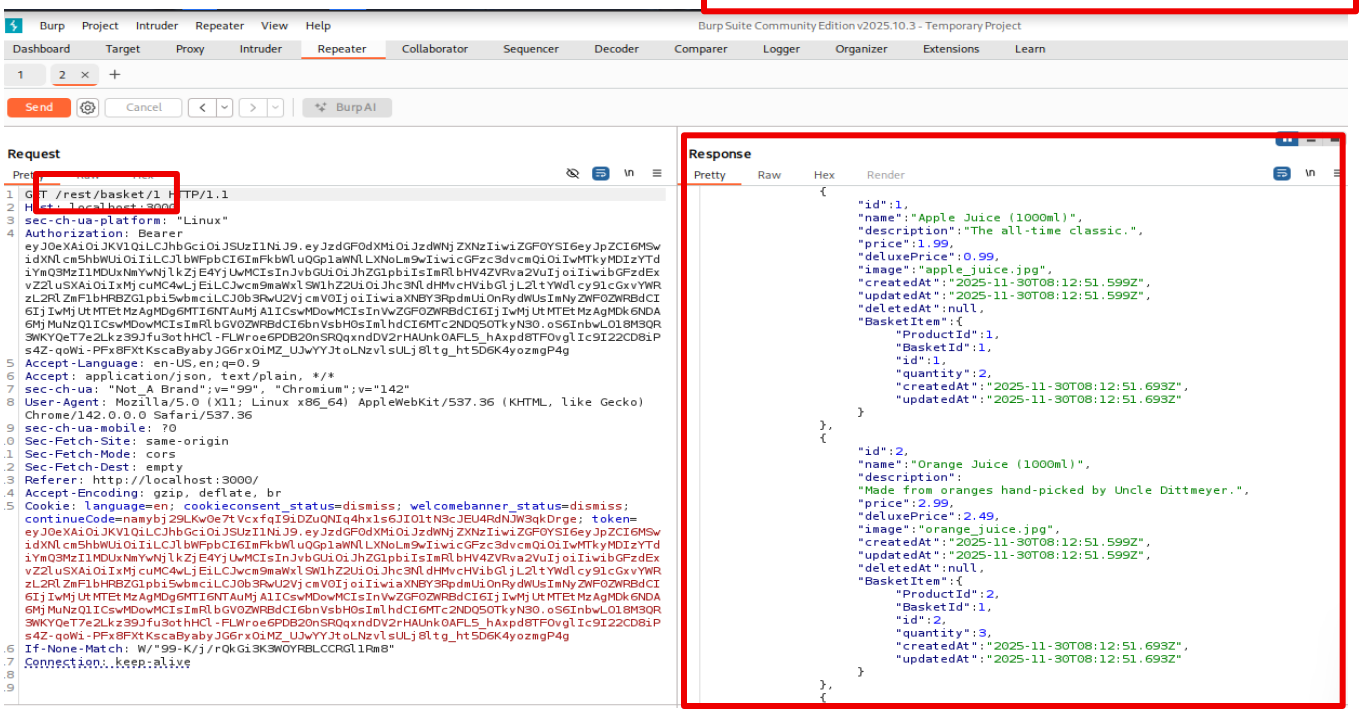

OWASP Juice Shop
Account
Your Basket
EN
admin@juice-sh.op
Orders & Payment
Privacy & Security
Logout
Your Basket (admin@juice-sh.op)
Apple Juice (1000ml) 2 1.99¤
Orange Juice (1000ml) 3 2.99¤
Eggfruit Juice (500ml) 1 8.99¤
Total Price: 21.94¤
Checkout
You will gain 1 Bonus Points from this order!


localhost:3000/#/basket
OWASP Juice Shop
Account
Your Basket
EN
admin@juice-sh.op
Orders & Payment
Privacy & Security
Logout
Your Basket (admin@juice-sh.op)
Apple Juice (1000ml) 2 1.99¤
Orange Juice (1000ml) 3 2.99¤
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
DOM Invader
Application
Manifest
Service workers
Storage
Storage
Session storage
Extension storage
IndexedDB
Cookies
http://localhost:3000
bid
1
No value selected
Select a value to preview


OWASP Juice Shop
Account
Your Basket
EN
admin@juice-sh.op
Orders & Payment
Privacy & Security
Logout
Your Basket (admin@juice-sh.op)
Raspberry Juice (1000ml) 1 4.99¤
Total Price: 4.99¤
Checkout
Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Lighthouse
Recorder
DOM Invader
Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
Extension storage
IndexedDB
Cookies
http://localhost:3000
bid
Value
3

By burp suite:





Impact:

- Leakage of all users' data
- Privilege escalation

Recommendation:

- Implement server-side access control
- Validate user roles on every request

5.4 Logic Vulnerability:

Definition:

A Logic Vulnerability occurs when the application's business rules are incorrectly implemented, allowing users to perform actions that violate the intended workflow or financial logic of the system.

Risk Level: High

Affected Functionality:

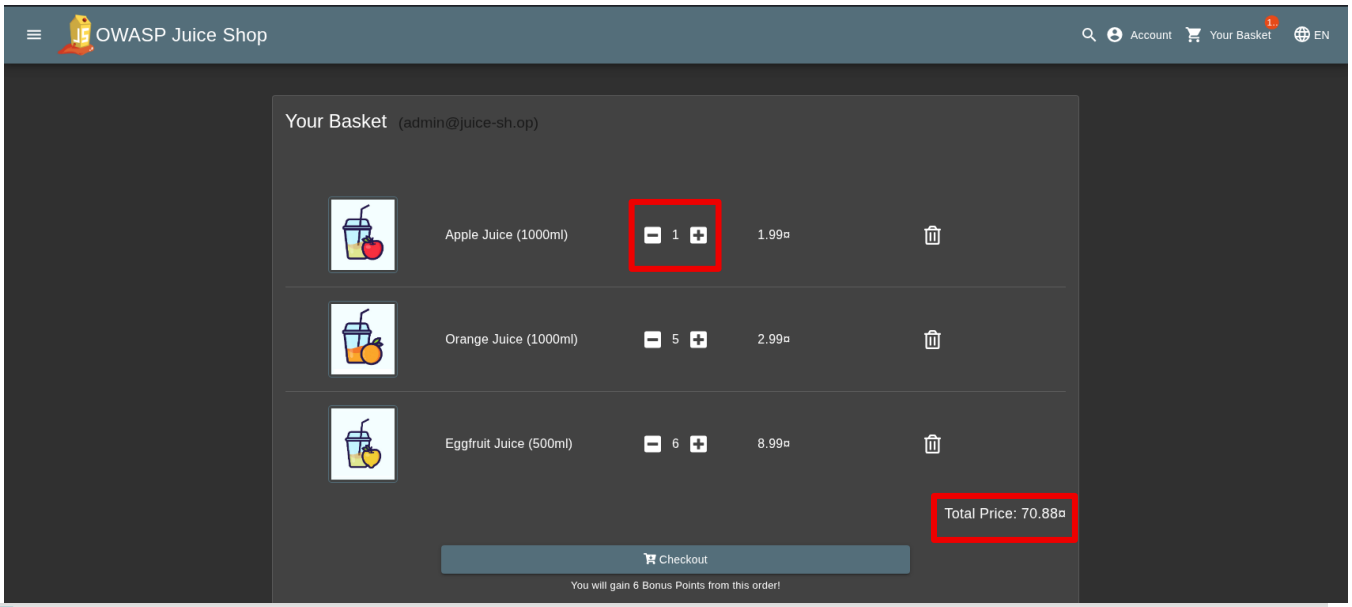
/api/BasketItems

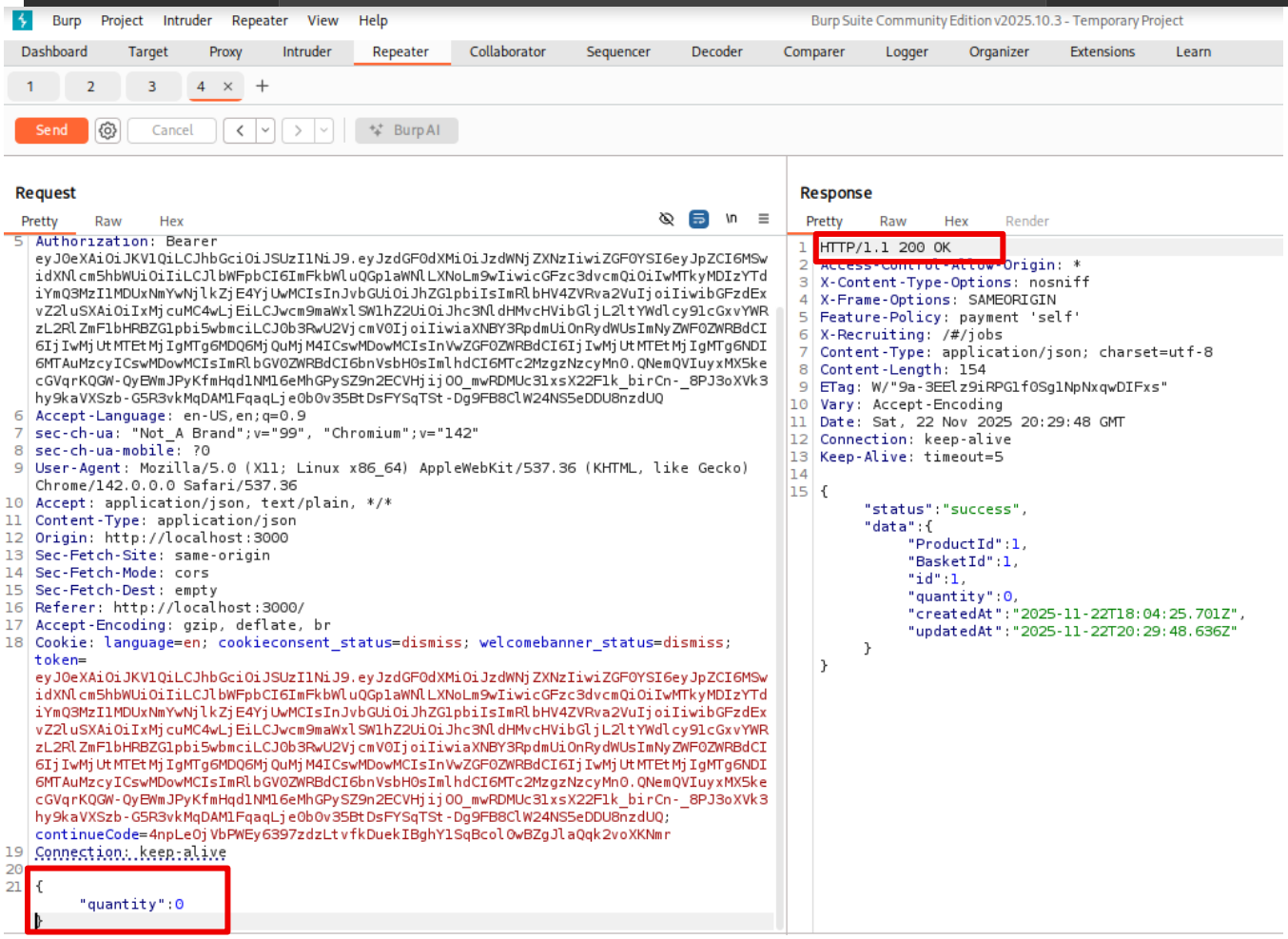
Description:

The application contains a business logic flaw that allows users to purchase multiple products while the total price is calculated as zero. Due to improper validation of pricing rules, the system fails to enforce the correct calculation of product costs during checkout. As a result, an attacker can exploit this flaw to obtain items without paying, bypassing the intended purchasing logic.

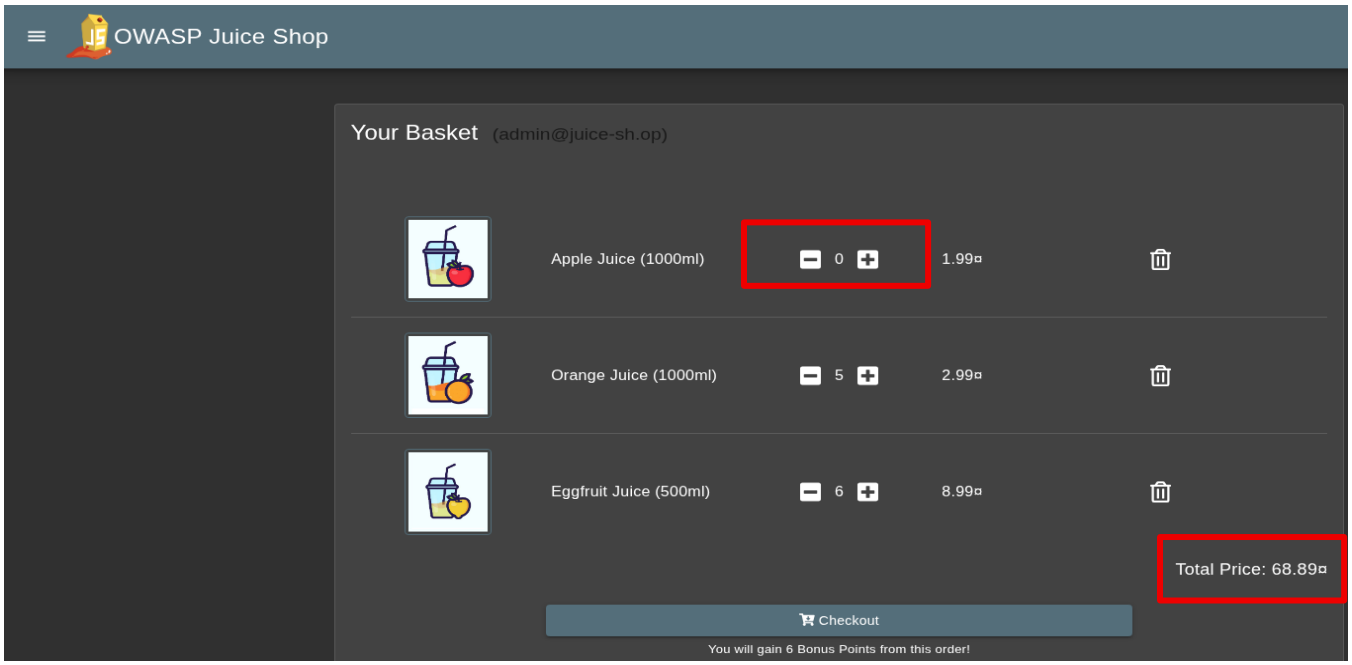
Proof of Concept:

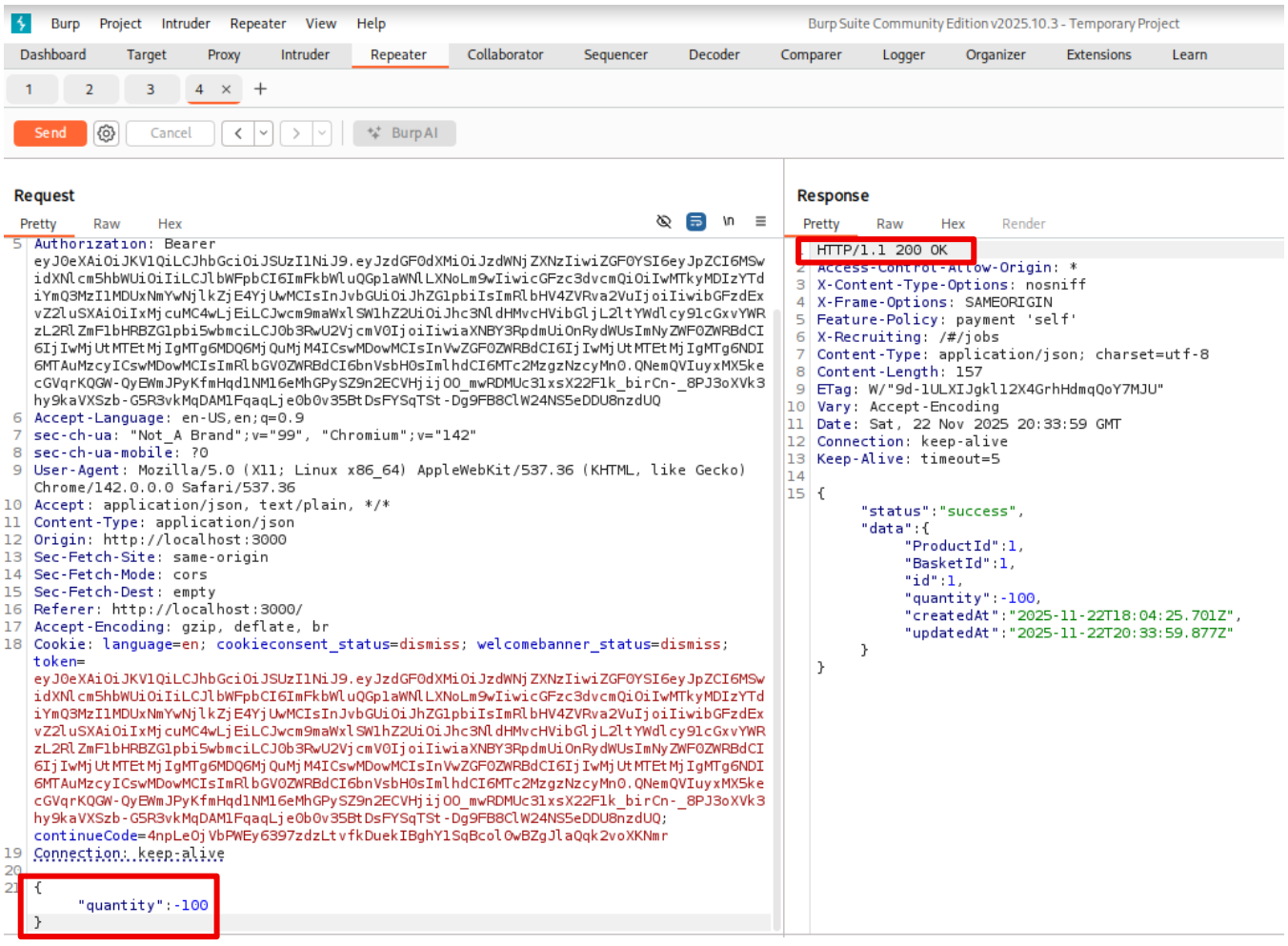
Cannot decrease item counter to 0

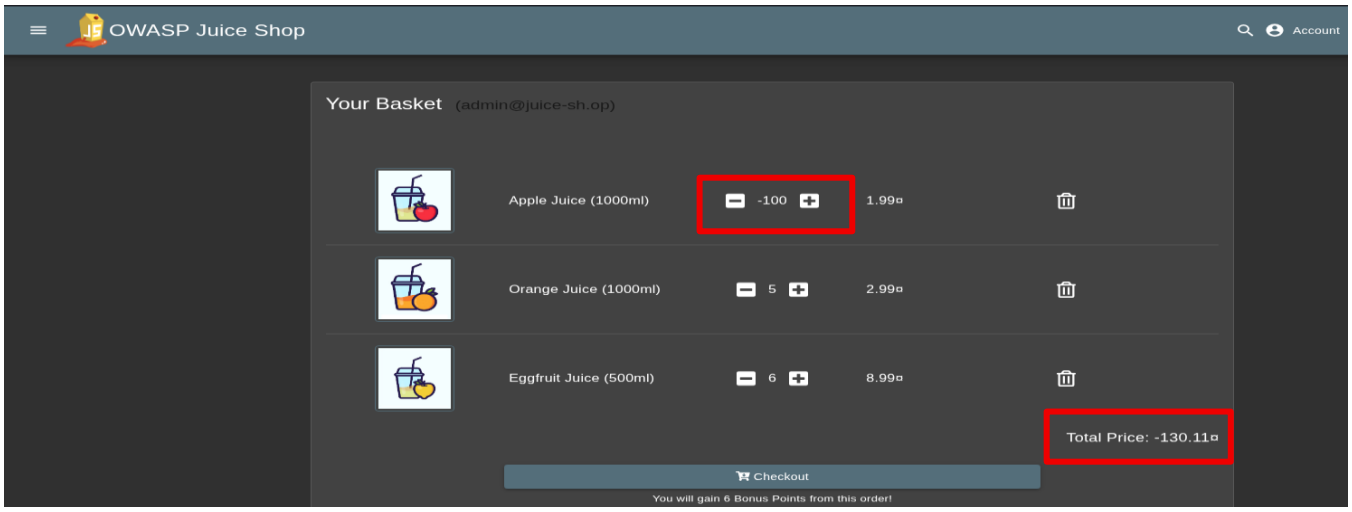




Now item counter is 0 and decrease total price







Impact:

- Allows attackers to complete orders with a total price of 0
- Direct financial loss to the system
- Bypasses core business rules and integrity
- Enables abuse of the e-commerce workflow
- Potential large-scale exploitation by automated scripts
- Loss of trust in the platform’s payment system

Recommendation:

- Recalculate product prices exclusively on the server side
- Do not trust any price or quantity values sent from the client
- Implement server-side validation for: item price , total cost , discounts, quantity
- Enforce integrity checks during checkout
- Ensure no order can be completed unless total price > 0
- Log and monitor suspicious discount or price manipulation behavior
- Add automated tests to validate business rules

5.5 Cross-Site Scripting (XSS):

Definition:

Cross-Site Scripting (XSS) is a vulnerability that allows attackers to inject malicious JavaScript into a web page viewed by other users.

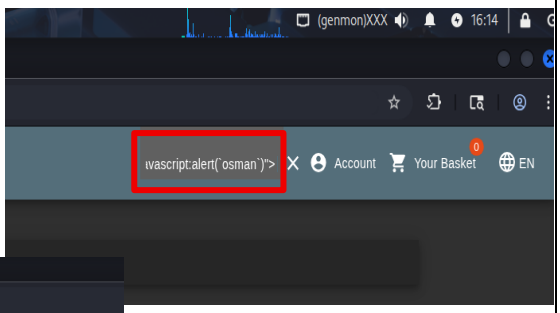
Risk Level: Medium

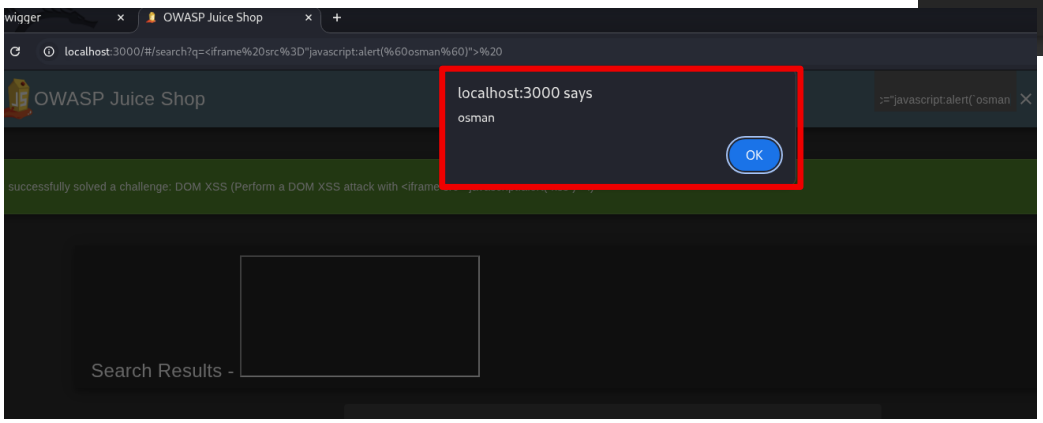
Types:

- **Dom**: Occurs when the malicious payload is executed entirely on the client side because the JavaScript code modifies the DOM without proper sanitization. The server is not involved in reflecting or storing the payload.
- **Stord :** Occurs when the attacker's payload is permanently stored on the server (e.g., in a database, comment, review, feedback). The script executes when any user views the infected content.
- **Reflect :** Occurs when the malicious input is immediately reflected by the server in the response (usually via URL parameters) without being stored. It executes when the victim clicks a crafted link.

Proof of Concept:

Payload : *<iframe src="javascript:alert(`xss`)">*



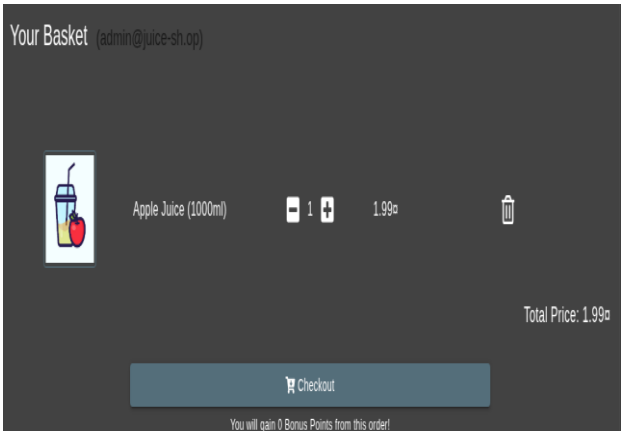


Payload 2:

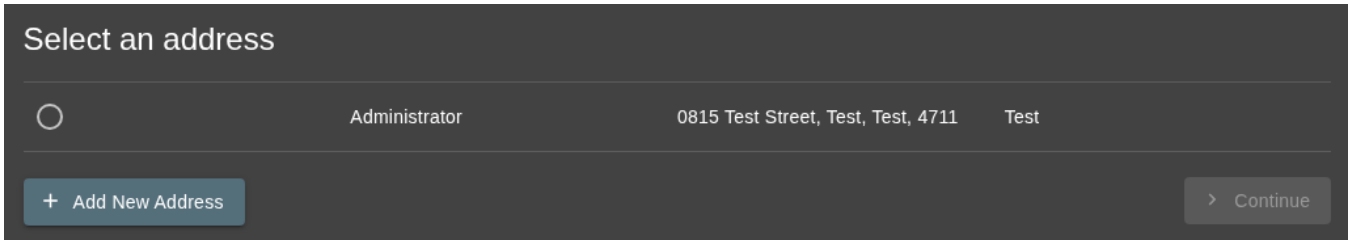
Steps: Place order and track order

1 choose product to buy

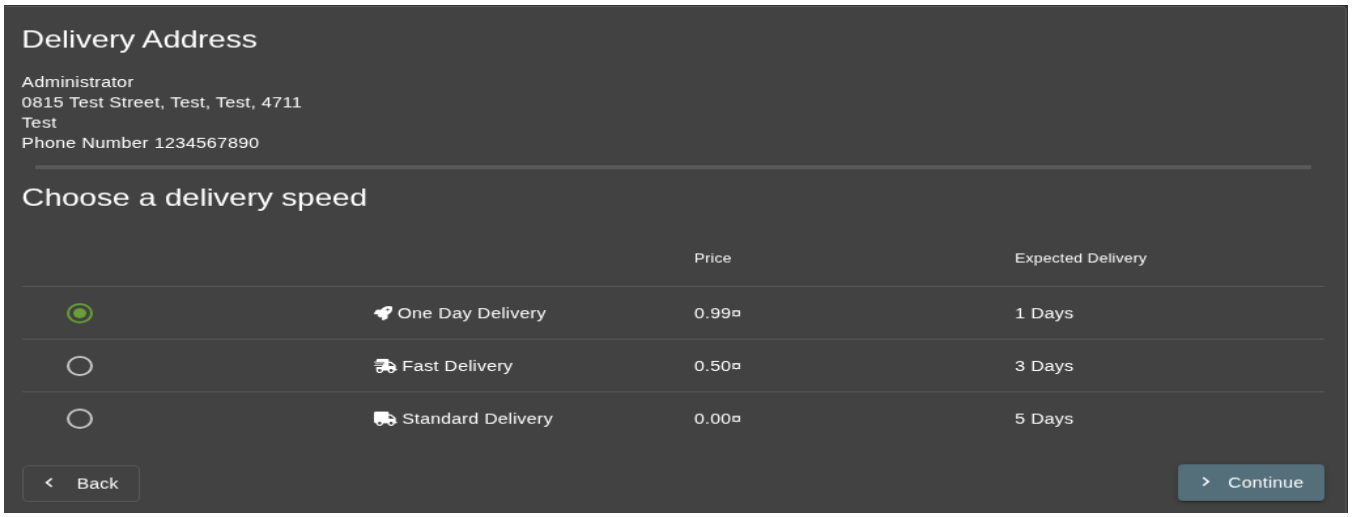
2 choose amount



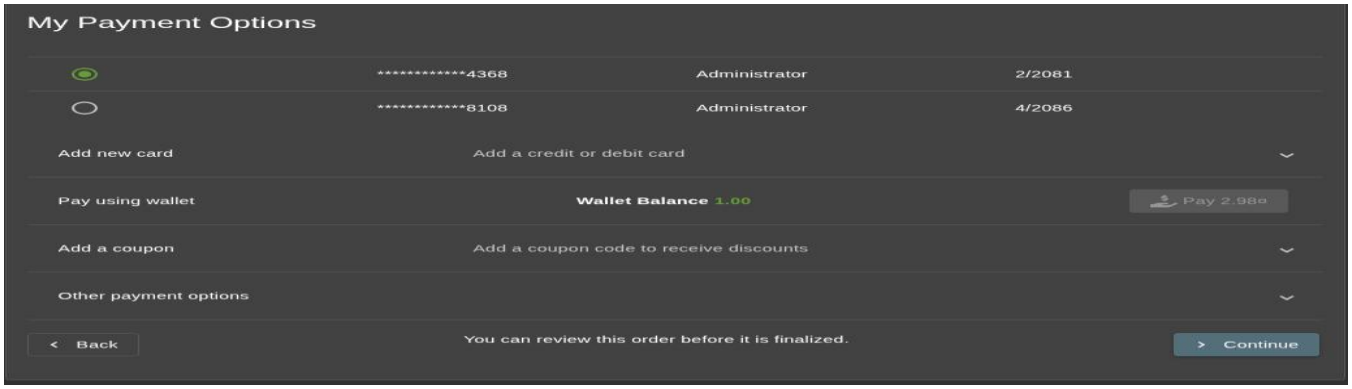
3 select address



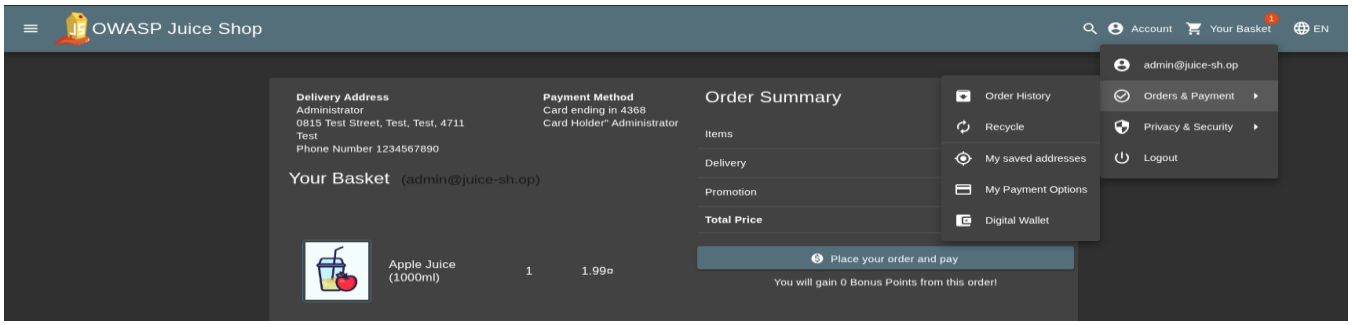
4 select delivery speed



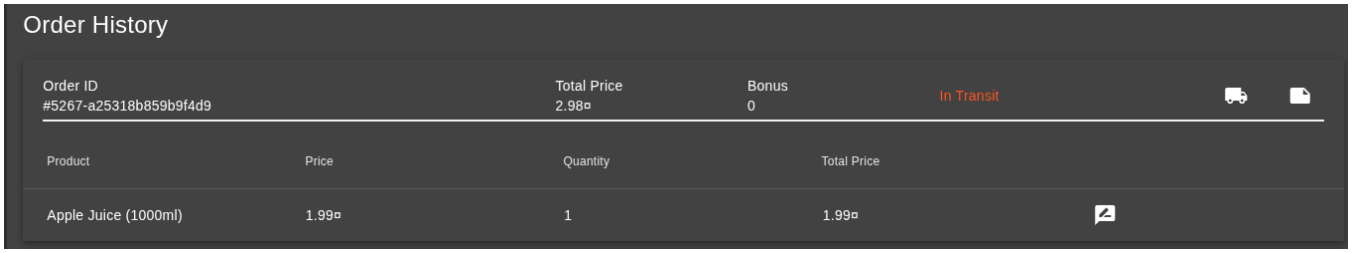
5 payment methods



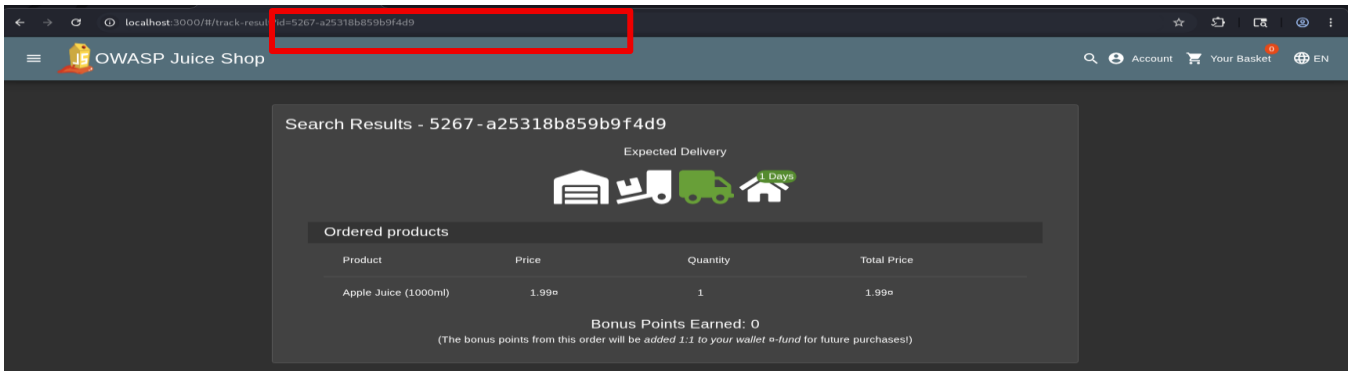
6



7 track product

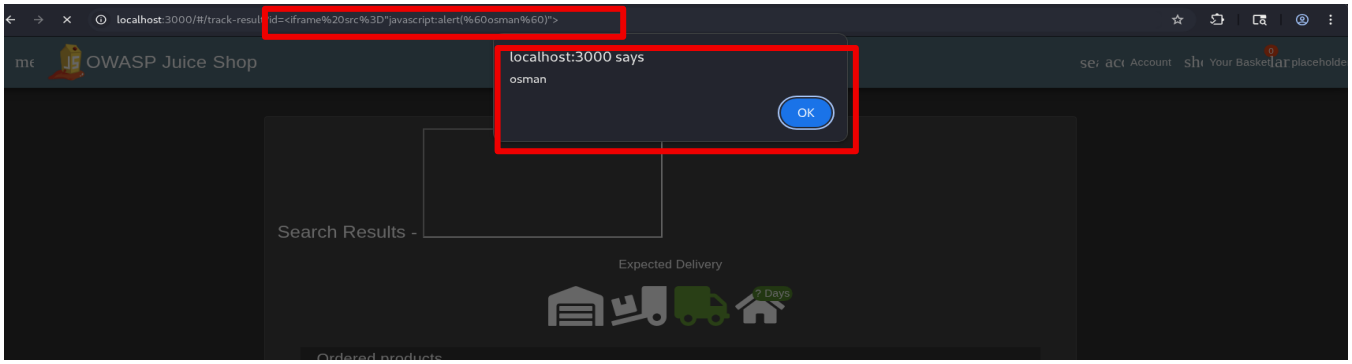


8



Now affected parameter is: /track-result?**id**=

<iframe src="javascript:alert(`xss`)">



XSS Impact :

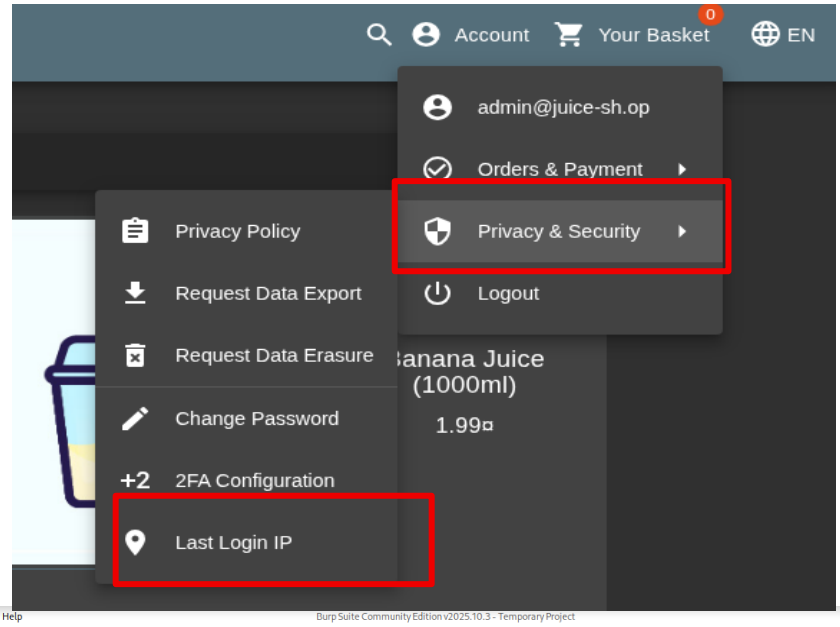
- Theft of user session cookies leading to account takeover.
- Execution of malicious JavaScript in the victim's browser.
- Performing unauthorized actions on behalf of the user.
- Redirecting users to phishing or malicious websites.
- Stealing sensitive data and form inputs (keylogging).
- Defacing pages or injecting unwanted content.
- Potential spread of stored XSS as a self-propagating worm.

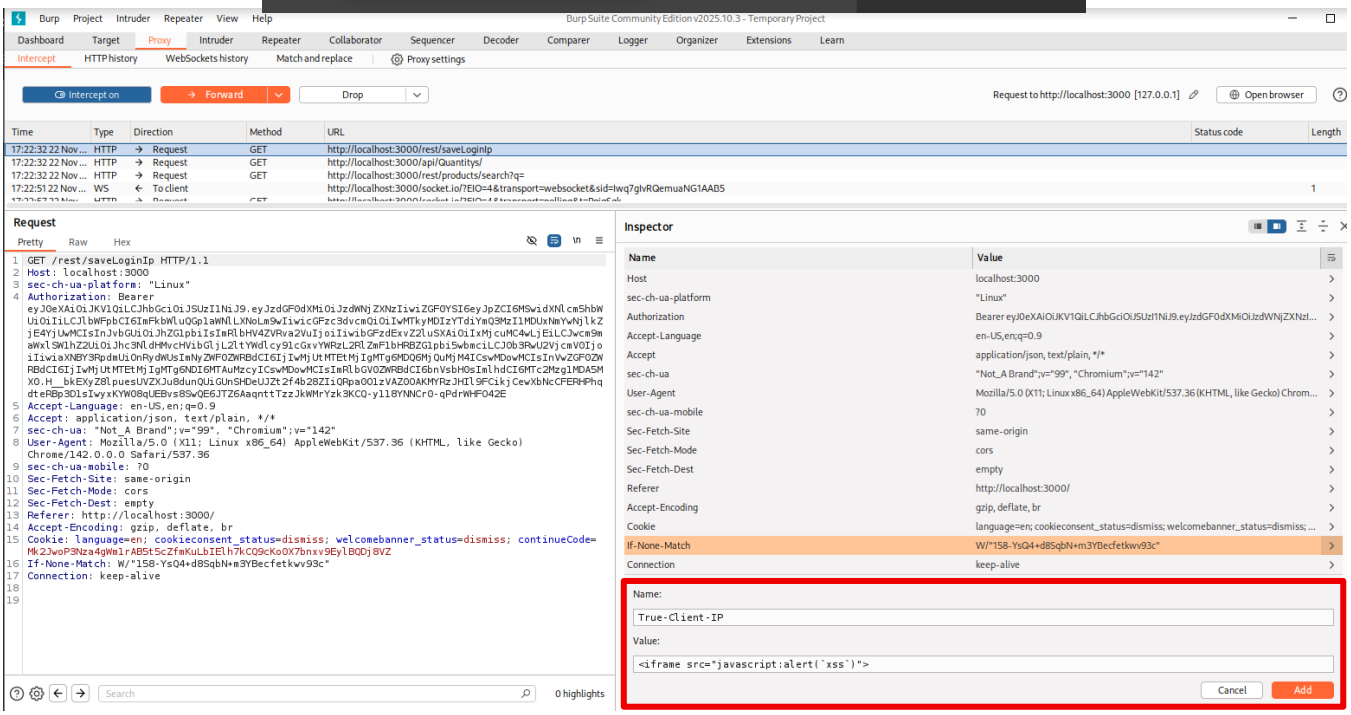
Recommendations:

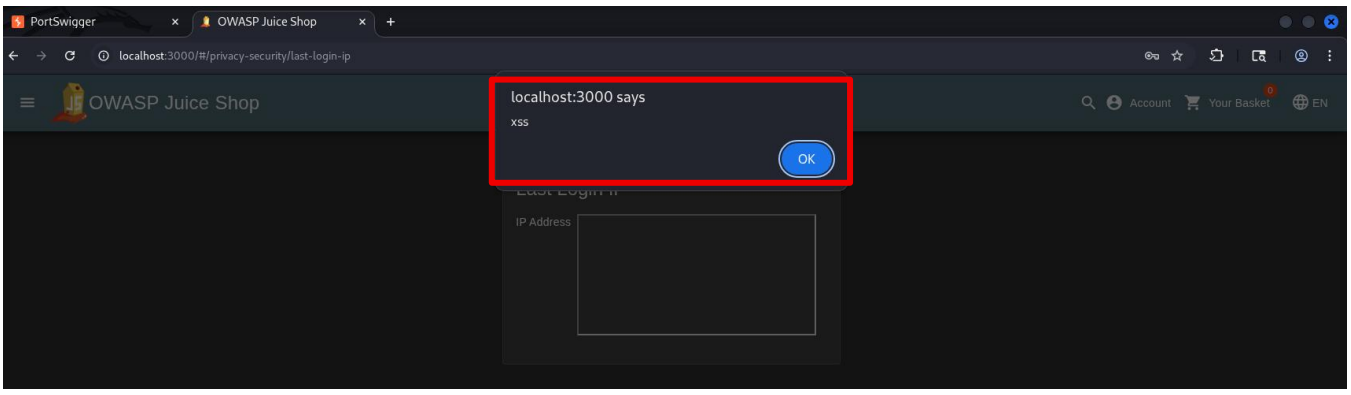
Validate and sanitize all user inputs using a whitelist approach

Encode user-supplied data before rendering it in the browser (output encoding).

Payload 3:


Account
Your Basket
0
EN
admin@juice-sh.op
Orders & Payment
Privacy Policy
Privacy & Security
Request Data Export
Logout
Request Data Erasure
anana Juice
(1000ml)
Change Password
1.99¤
+2
2FA Configuration
Last Login IP


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.10.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to http://localhost:3000 [127.0.0.1]
Open browser
Time Type Direction Method URL Status code Length
17:22:32 22 Nov... HTTP Request GET http://localhost:3000/rest/saveLoginIp
17:22:32 22 Nov... HTTP Request GET http://localhost:3000/api/Quantitys/
17:22:32 22 Nov... HTTP Request GET http://localhost:3000/rest/products/search?q=
17:22:51 22 Nov... WS To client http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=Iwq7glvRQemuaNG1AAB5 1
Request
Pretty Raw Hex
1 GET /rest/saveLoginIp HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua-platform: "Linux"
4 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcmShbW
UiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzIlMDUxNmYwNjlkZ
jE4YjUwMCIsInJvbGUiOiJhZGlpbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxMjcuMC4wLjEiLCJwcm9m
aWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy9lcGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0Ijo
iIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMTEtMjIgMTg6MDQ6MjQuMjM4ICswMDowMCIsInVwZGF0ZW
RBdCI6IjIwMjUtMTEtMjIgMTg6MTAuMzcyICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbHOsImlhdCI6MTc2Mzg1MDA5M
X0.H__bkEXyZ8lpuesUVZXJu8dunQUiGUnSHDeUJZt2f4b28ZIiQRpaO01zVAZOOAKMYRzJHIl9FCikjCewXbNcCFERHPhq
dteRBp3D1sIwyxKYW08qUEBvsBSwQE6JTZ6AaqnttTzzJkWMrYzk3KCQ-yl18YNNCrO-qPdrWHFO42E
5 Accept-Language: en-US,en;q=0.9
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "Not_A Brand";v="99", "Chromium";v="142"
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/142.0.0.0 Safari/537.36
9 sec-ch-ua-mobile: ?0
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:3000/
14 Accept-Encoding: gzip, deflate, br
15 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
Mk2JwoP3Nza4gWm1rAB5t5cZfmKuLbIElh7kCQ9cKo0X7bnxv9EylBQDj8VZ
16 If-None-Match: W/"158-YsQ4+d8SqbN+m3YBecfetkwv93c"
17 Connection: keep-alive
18
19
Search
0 highlights
Inspector
Name Value
Host localhost:3000
sec-ch-ua-platform "Linux"
Authorization Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzI...
Accept-Language en-US,en;q=0.9
Accept application/json, text/plain, */*
sec-ch-ua "Not_A Brand";v="99", "Chromium";v="142"
User-Agent Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrom...
sec-ch-ua-mobile ?0
Sec-Fetch-Site same-origin
Sec-Fetch-Mode cors
Sec-Fetch-Dest empty
Referer http://localhost:3000/
Accept-Encoding gzip, deflate, br
Cookie language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; ...
If-None-Match W/"158-YsQ4+d8SqbN+m3YBecfetkwv93c"
Connection keep-alive
Name:
True-Client-IP
Value:
<iframe src="javascript:alert(`xss`)">
Cancel
Add


PortSwigger
OWASP Juice Shop
localhost:3000/#/privacy-security/last-login-ip
OWASP Juice Shop
Account
Your Basket
EN
localhost:3000 says
xss
OK
IP Address

Cross-Site Request Forgery (CSRF)

Definition: Cross-Site Request Forgery (CSRF) is a vulnerability that allows an attacker to force a logged-in user to perform unintended actions within a web application. Because browsers automatically include cookies in outgoing requests, the server mistakenly believes the action was performed intentionally by the victim.

Risk Level: Medium

Affected Endpoint: POST http://localhost:3000/profile

Description:

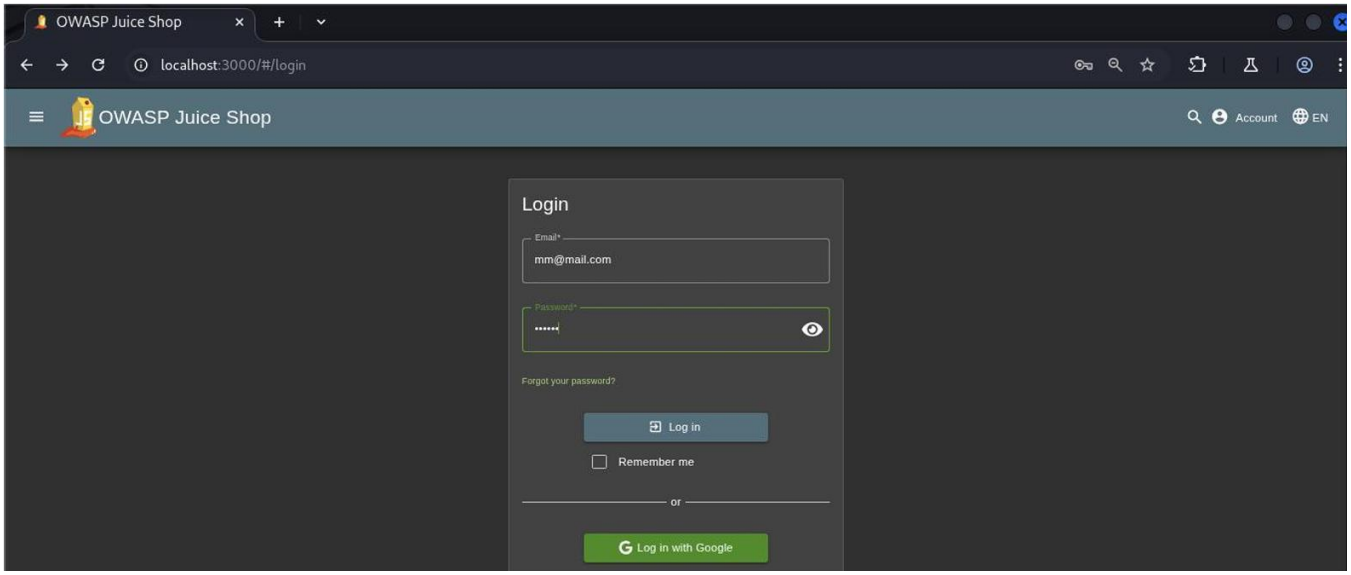
The profile update functionality in the application is vulnerable to CSRF due to the absence of essential protection mechanisms such as:

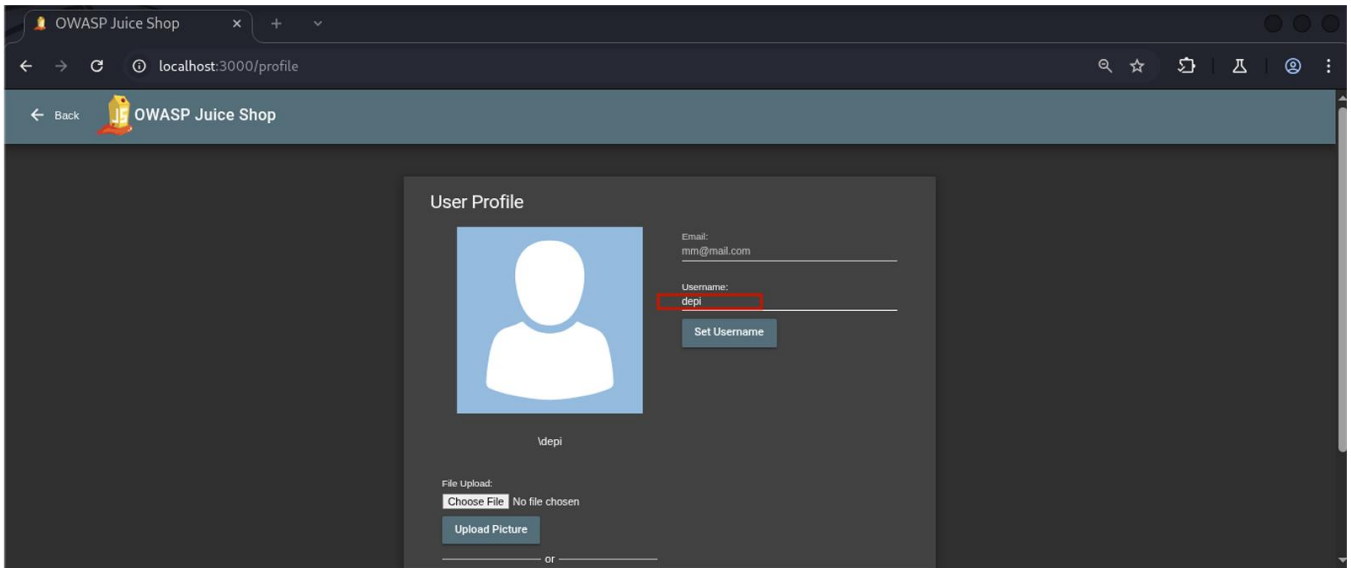
- No Anti-CSRF token validation
- No verification of request origin
- Reliance solely on session cookies for authentication

This allows an attacker to host a malicious HTML page that automatically sends a crafted POST request. When the victim-while logged into the Website visits the page, the request is executed using their session, resulting in changing the username without their consent.

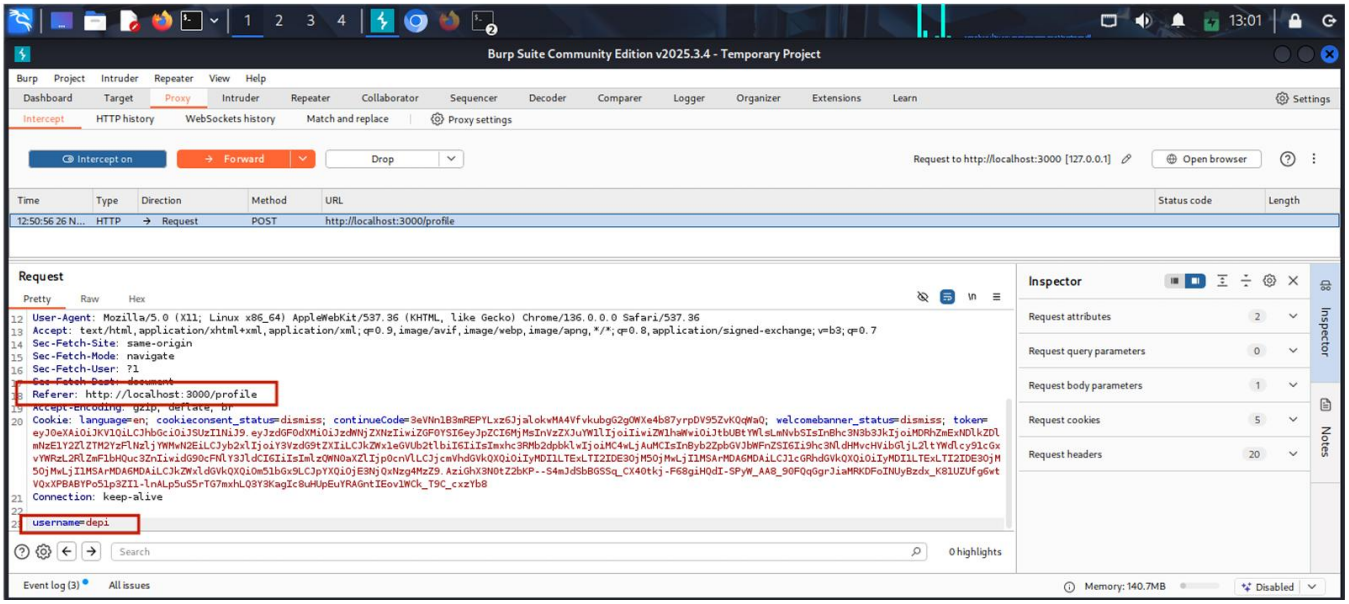
Proof of Concept:

Logging in with a normal user account

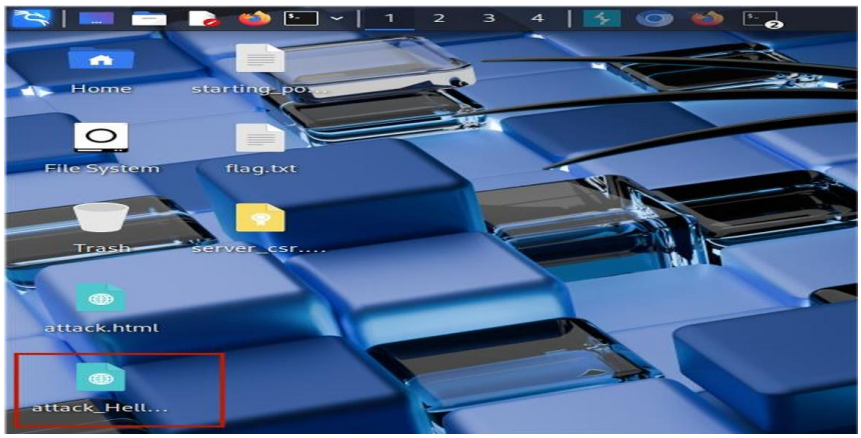


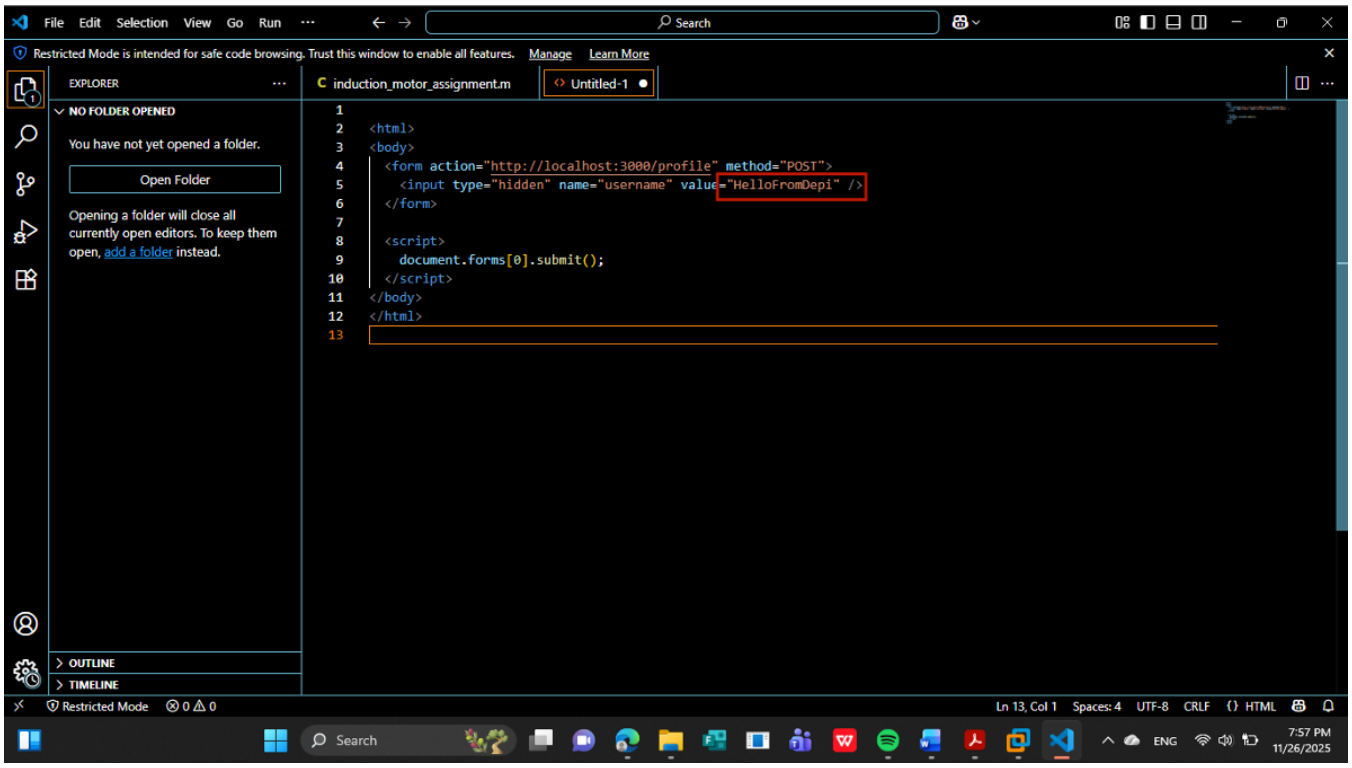


Intercepted POST Request Showing Vulnerable Parameters

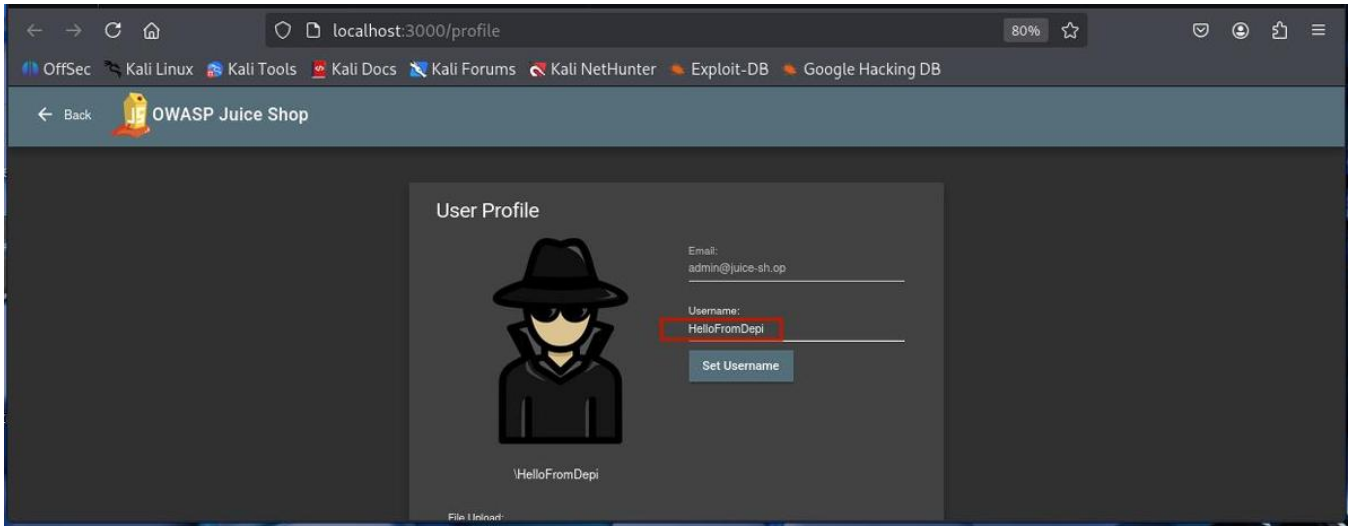


CSRF Attack HTML File Prepared on the Attacker's Machine





Username Successfully Modified via CSRF



Impact:

- Unauthorized modification of user profile data
- Ability to perform actions on behalf of the victim
- Potential escalation to account takeover if sensitive endpoints are affected
- Increased risk when combined with social engineering attacks
- Loss of integrity and trust in the application

Recommendations:

- Implement server-side Anti-CSRF tokens on all state-changing requests
- Use SameSite=Strict cookies to prevent cross-site cookie submission
- Validate Origin and Referer headers
- Restrict CORS to trusted domains only
- Implement CSRF protection middleware
- Reject requests missing CSRF tokens or valid origins