[OWASP juice shop]
Penetration Testing Report

Jihad Salah Eldeen Yahia Mostafa	**2021170146**
Hana Mohsen Sayed Goda	**2021170612**
Asmaa Salah Eldeen Mohamed	**2021170075**
Mariam salah Abdelsabour	**2021170508**
Kermina Nabil Sobhy	**2021170412**
Seham Ibrahim Mohammed	**2021170243**

Table of Contents

Contents

1 Executive Summary

This report documents the findings and results of a security assessment conducted on the OWASP Juice Shop application as part of a project to identify vulnerabilities in modern web applications. OWASP Juice Shop was chosen due to its comprehensive representation of real-world security flaws and its alignment with the OWASP Top 10 vulnerabilities.

The objective of this project was to detect, exploit, and document 15 vulnerabilities within the application and analyze their impact. The assessment focused on critical web application security concerns such as authentication flaws, injection vulnerabilities, insecure file handling, and cryptographic issues.

The testing process involved leveraging industry-standard tools such as Burp Suite, and manual techniques to identify and exploit vulnerabilities. This document provides a detailed overview of the vulnerabilities discovered, including their technical analysis, exploitation steps.

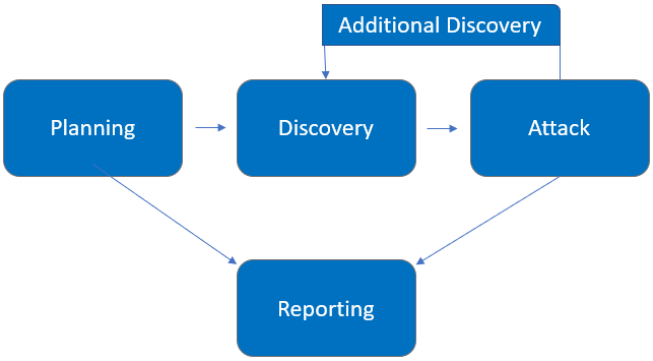
This project demonstrates a comprehensive understanding of web application security testing and highlights the importance of securing applications against potential cyber threats. The findings aim to enhance the awareness of developers and security professionals in mitigating real-world vulnerabilities effectively.

2 Assessment Overview

All testing performed is based on the NIST *SP 800-115 Technical Guide to Information Security Testing and Assessment, OWASP Testing Guide (v4), and customized testing frameworks*.

Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered and rules of engagement obtained.
- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.



2.1 Planning

Planning in penetration testing involves collaboratively defining the scope, goals, and guidelines for the test. This phase establishes the boundaries of the assessment, such as which systems are included and the desired objectives—whether it's uncovering vulnerabilities, testing response procedures, or assessing controls. Ethical and legal considerations are addressed, resources are allocated, and a communication plan is set for a successful testing engagement. This strategic groundwork ensures alignment between the testing team and the client, leading to a well-structured and effective penetration test.

2.2 Discovery

- Information Gathering: Collect relevant information about the internal infrastructure, including IP ranges, domain names, and network architecture.
- Open Source Intelligence (OSINT): Utilize publicly available sources to gather additional information about employees, technology stack, and potential weak points.
- Vulnerability Scanning: Perform automated vulnerability scans on the identified systems to uncover known security vulnerabilities.
- Network Mapping: Create a comprehensive map of the internal network, identifying live hosts, open ports, and services.

2.3 Attack

- Manual Vulnerability Assessment: Conduct an in-depth manual assessment of the identified vulnerabilities to validate their severity and potential impact.
- Exploitation: Attempt to exploit the identified vulnerabilities to determine their feasibility and potential impact on the internal systems.
- Privilege Escalation: If necessary, attempt to escalate privileges to gain deeper access into the systems and network.
- Lateral Movement: Explore the internal network for lateral movement opportunities, simulating an attacker's attempt to pivot within the environment.
- Data Exfiltration (If Agreed Upon): Simulate the extraction of sensitive data from the internal systems to demonstrate potential data breaches.

2.4 Reporting

Reporting the findings of the penetration tests is integral to the fulfilment of the previously mentioned strategic motivations and driving forces behind engaging in such a process. Hence, once the above tasks are completed, a documentation scheme is followed to report the results across different levels including technical and management levels. The report is going to be focused on the real risk behind the findings to maximize business value. Also, whenever applicable, a detailed

recommendation is included on how to fix the leveraged vulnerabilities and / or minimize their threat.

3 Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definitio n
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Risk Factors

Risk is measured by two factors: Likelihood and Impact:

Likelihood

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, attacker skill level, and client environment.

Impact

Impact measures the potential vulnerability's effect on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.

4 Scope of Engagement

The scope of this engagement includes performing Internal Penetration Testing for
*[**OWASP juice shop**]*.

4.1 Targets

Assessment	Details
Internal Penetration Test	http://localhost:3000/#

4.2 Scope Exclusions

Per client request, you will not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Phishing/Social Engineering
- XSS- reflected
- some SQL-Injection
- CSRF

All other attacks not specified above were permitted by *[**OWASP juice shop**]*.

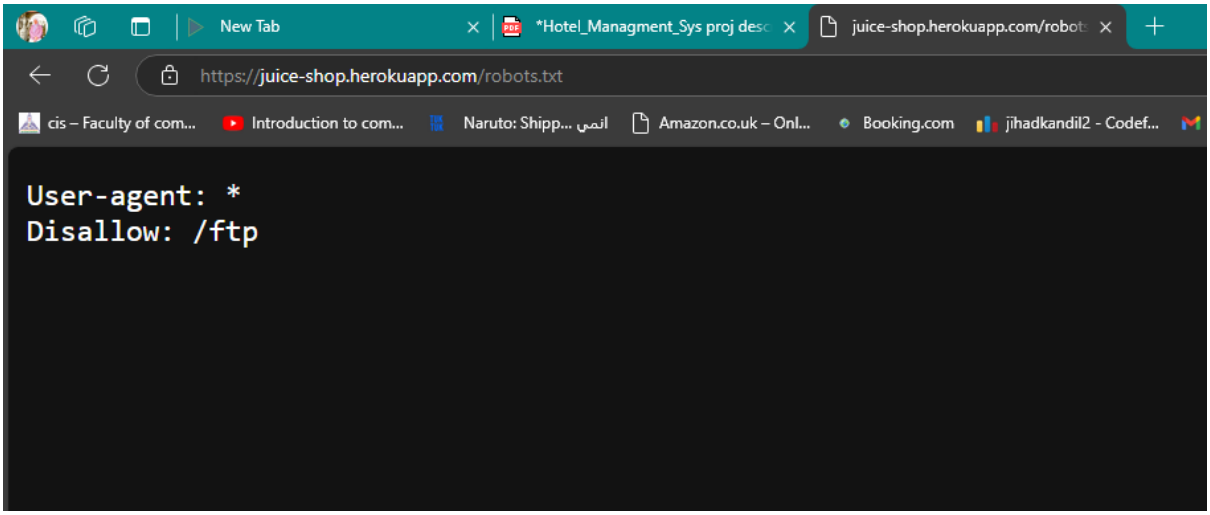
5 Technical Findings

5.1 Broken Access Control [Horizontal]:

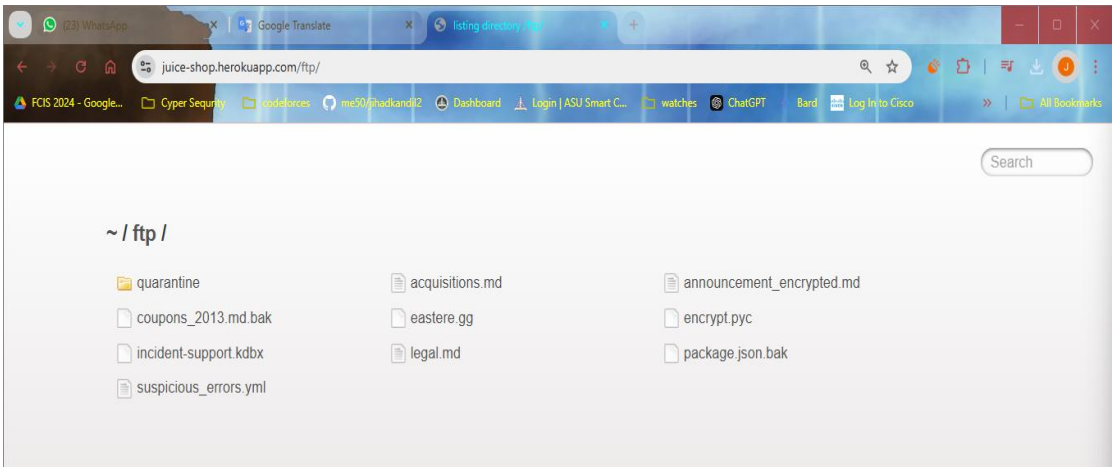
Description:	The vulnerability arises due to insufficient access control mechanisms, allowing unauthorized users to access restricted directories. By accessing the /robots.txt file, sensitive paths like /ftp are disclosed, enabling attackers to access the folder contents, which should not be publicly available.
Impact	Unauthorized access to the /ftp folder could expose sensitive files, configurations, or data stored within the directory. This could lead to information disclosure, potential exploitation, or further attack vectors if critical data (e.g., passwords or keys) is exposed.
Recommendations	1) Restrict access to sensitive directories like /ftp using proper authentication and authorization mechanisms. 2) Avoid exposing sensitive paths in the robots.txt file or restrict their visibility to authorized users only. 3) Implement least privilege principles to ensure only authorized users can access protected resources. 4) Regularly audit and monitor access logs to identify any unauthorized attempts to access restricted directories.
Affected Systems	http://localhost:3000/robots.txt
Threat Level	**Moderate**

steps to reproduce:

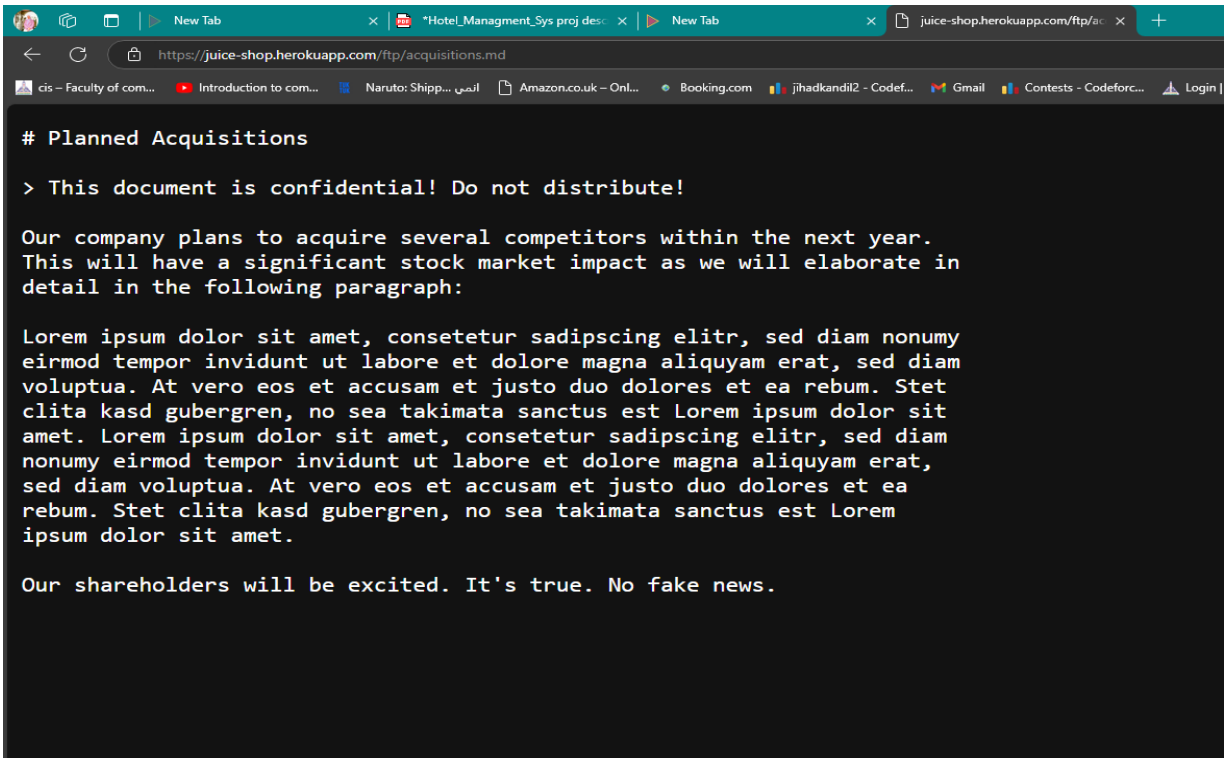
1) First I accessed : https://juice-shop.herokuapp.com/robots.txt
2) Observe the response : it shows /ftp file which search engine should not access it



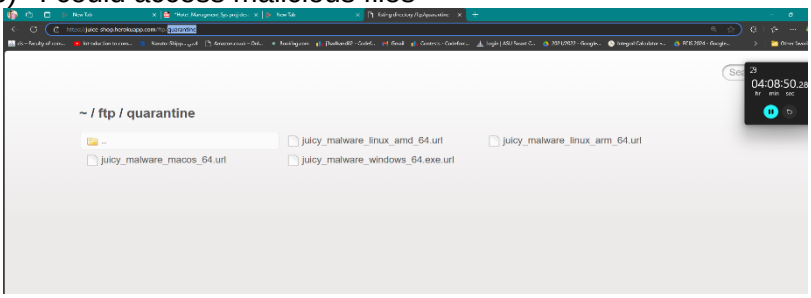
3) So I intended to access : https://juice-shop.herokuapp.com/ftp
4) Observe the response i get accessed to path folder which is not permitted:



5) I could access confidential documents like:



6) I could access malicious files

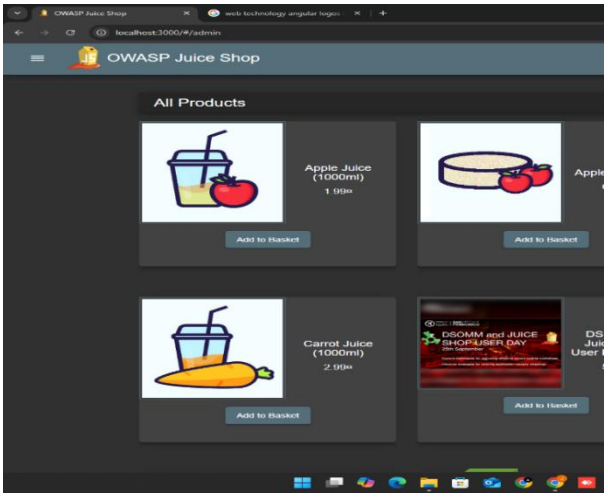


5.2 Broken Access Control [Vertical]:

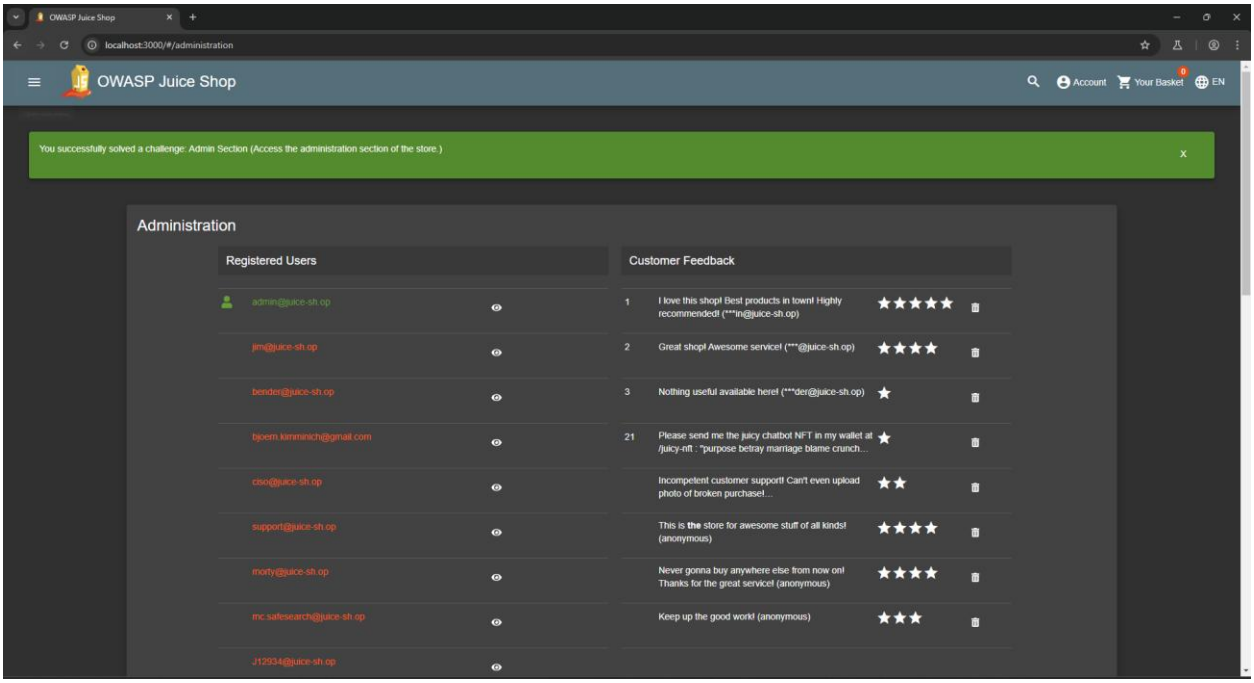
Description:	The vulnerability allows unauthorized users to escalate their privileges and access the administrative interface (`/administration`). By manipulating the URL and guessing the administrative endpoint, an attacker can bypass access controls and perform actions intended only for privileged users, such as deleting customer feedback.
Impact	Unauthorized access to the admin panel can result in: • Tampering with sensitive information or records, such as customer feedback. • Disruption of normal operations through unauthorized actions. • Increased risk of further privilege escalation or exploitation of critical functions
Recommendations	1) Implement strict role-based access controls (RBAC) to ensure only authenticated administrators can access the /administration endpoint and perform admin actions. 2) Introduce proper authentication and session management to restrict access to privileged functionalities. 3) Avoid predictable URLs for sensitive paths. Use non-guessable, unique identifiers for admin endpoints. 4) Perform rigorous security testing to ensure unauthorized users cannot access or perform actions beyond their role permissions
Affected Systems	localhost:3000/#/administration
Threat Level	**High**

steps to reproduce:

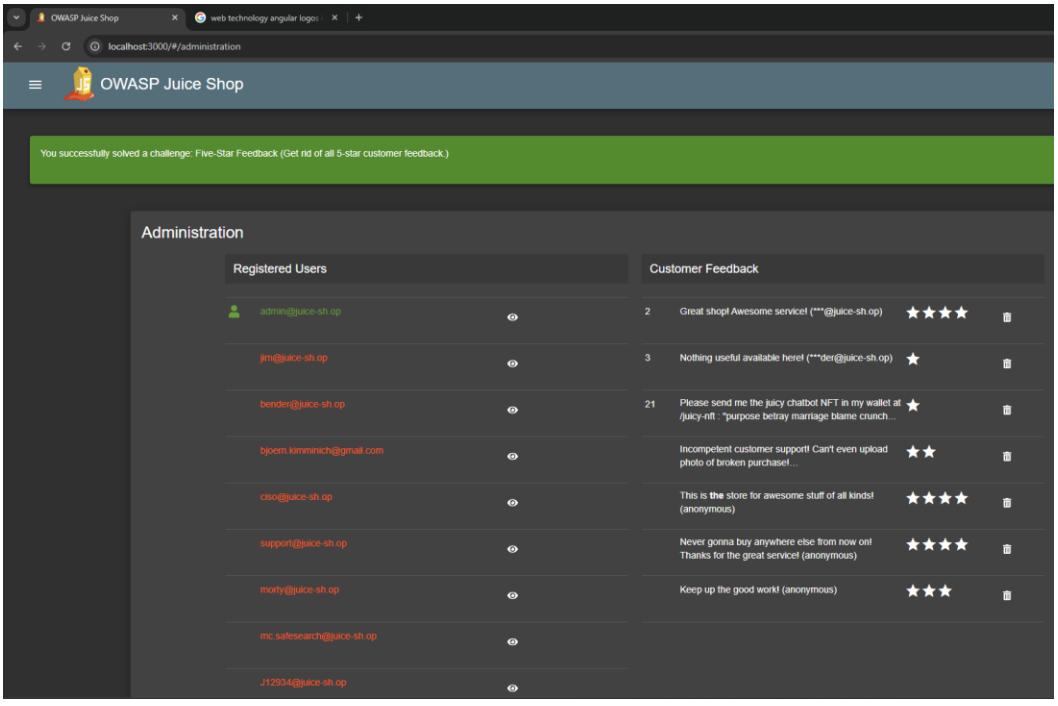
1) First, I tried to access: localhost:3000/#/admin as a predictable url but it gives me no response it is just the same home page



2) So I tried then localhost:3000/#/administration , and it works I could access that path



3) I also could do unauthorized action on the admin panel such as: delete customer feedback

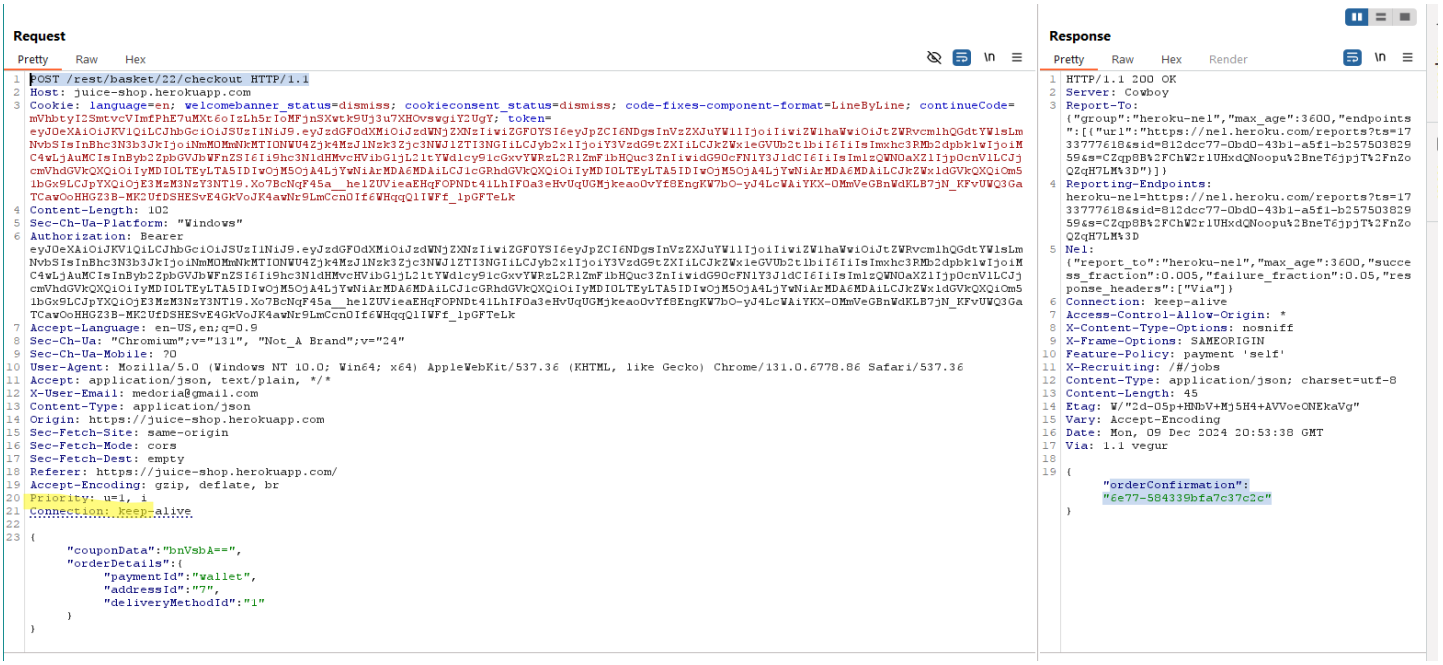


5.3 Broken Access Control (IDOR):

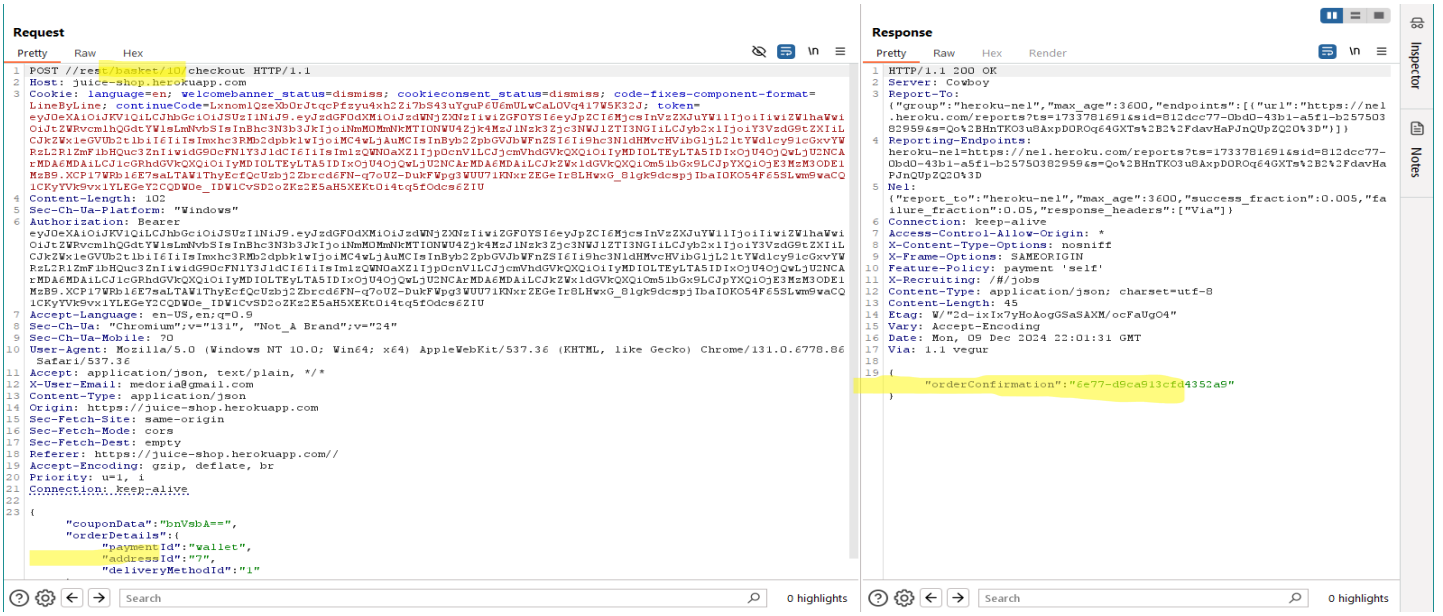
Description:	An Insecure Direct Object Reference (IDOR) vulnerability exists, allowing unauthorized users to manipulate object references, such as `basket_id`, in the request. By intercepting and modifying the request parameters, an attacker can redirect the contents of another user's basket to their address, thereby exploiting insufficient access controls in the backend.
Impact	Exploiting this vulnerability can result in: • Unauthorized access to other users' baskets or orders. • Fraudulent purchases or order manipulation. • Breach of user data confidentiality, such as basket contents and associated addresses.
Recommendations	1) Implement strict access control checks on the backend to ensure that users can only interact with their own resources, such as baskets and orders. 2) Use secure identifiers (e.g., session-based or user-scoped tokens) to validate user permissions for each request. 3) Log and monitor unusual activity, such as attempts to access unauthorized basket_id values. 4) Validate and enforce permissions server-side instead of relying solely on client-provided data.
Affected Systems	API endpoints handling basket_id and address_id parameters.
Threat Level	**High**

steps to reproduce:

1) First, I tried to make an order, before clicking on the check order Button I intercepted the request
2) I noticed that response: that request is sent with basket id [22] to address_id [7] which was me.



3) What if I changed the basket id to [10] now I will make that order in that basket into my address ? observe that the response is successful confirmation for the order.

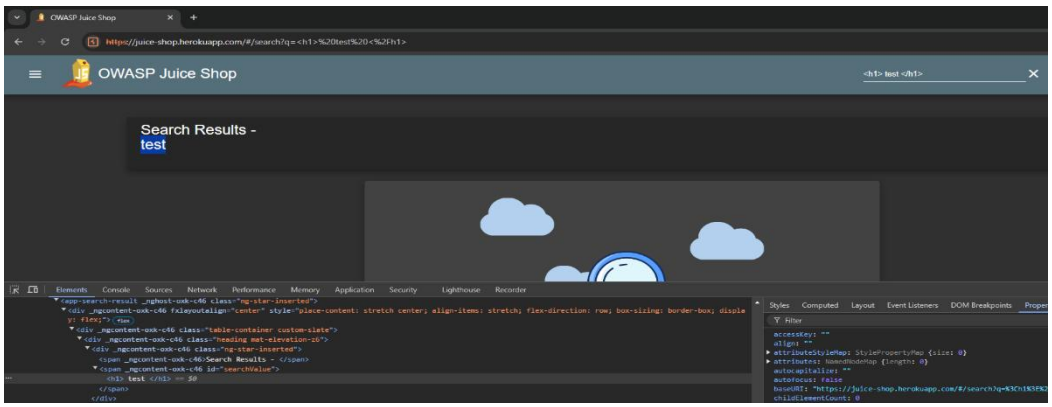


5.4 XSS(Reflected/DOM): -

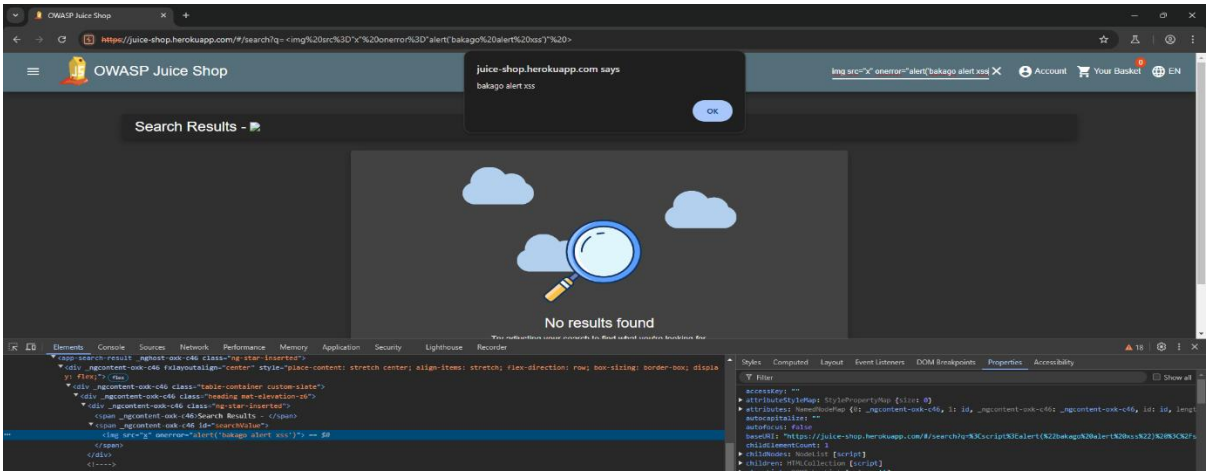
Description:	The application fails to sanitize user input in the search functionality, allowing attackers to inject malicious JavaScript or HTML payloads. This enables reflected XSS attacks, where the payload is executed immediately on the victim's browser when they interact with the manipulated link. Additionally, DOM-based XSS is exploitable by injecting payloads that modify the page's DOM directly, such as embedding an iframe for malicious content.
Impact	Exploitation of this vulnerability can result in: • Execution of arbitrary JavaScript in the victim's browser, enabling session hijacking, cookie theft, or phishing attacks. • Unauthorized manipulation of the page DOM to display malicious content. • Potential compromise of user accounts or data through social engineering or advanced attack payloads.
Recommendations	1) Properly sanitize and encode all user inputs before reflecting them in the response to prevent injection of JavaScript or HTML. 2) Use secure libraries for escaping HTML output to ensure that payloads cannot be executed. 3) Implement a Content Security Policy (CSP) to restrict the types of content that can be loaded or executed on the page. 4) Regularly test and audit user input points to identify and mitigate potential XSS vulnerabilities.
Affected Systems	• Search functionality on the OWASP Juice Shop. • DOM manipulation vulnerabilities in dynamically rendered content.
Threat Level	**High**

steps to reproduce:

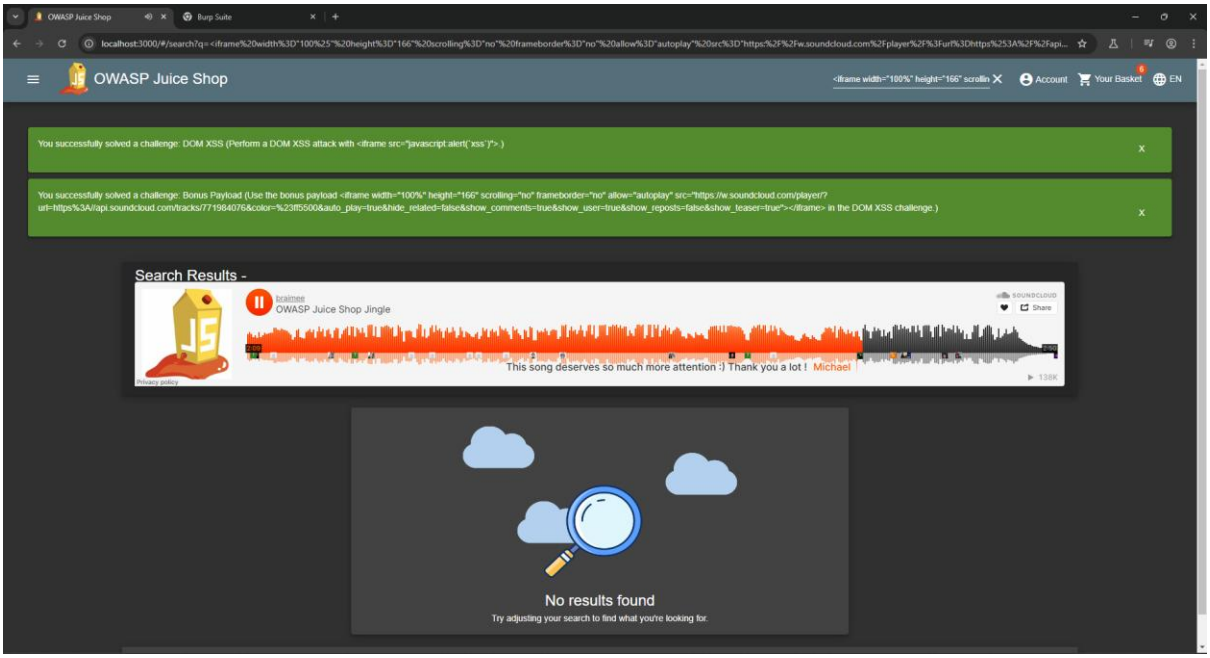
1) First, I tried in the search bar payload: <h1> test </h1>
2) I found the response reflected like this in the page:



3) Then I tried the JavaScript payload: <img src=”x” onerror=”alert(‘bakago alert xss’)” > I found the response alerted in the page like this so if I take that link and send it to the victim it will execute on him whatever the script I write :



4) I tried to change the DOM of the page on the same functionality by injecting this payload into search :<iframe width="100%" height="166" scrolling="no" frameborder="no" allow="autoplay" src="https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=true&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true"></iframe> and see the response it Works..

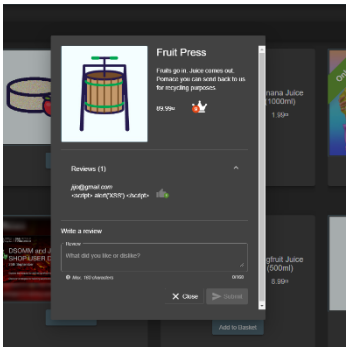


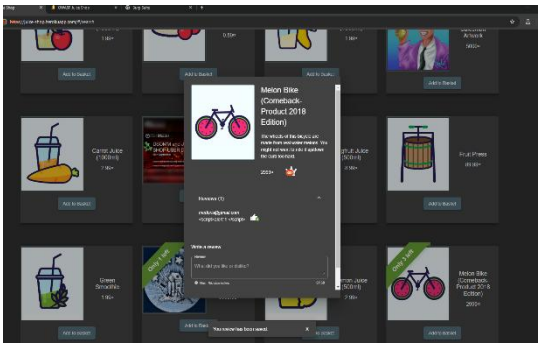
5.5 XSS (Stored): -

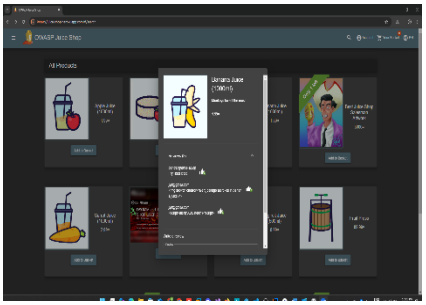
Description:	The application does not sanitize user inputs submitted in product reviews, allowing attackers to inject malicious JavaScript payloads that are stored on the server. These payloads are later executed in the browsers of other users when they view or interact with the affected reviews. This vulnerability is particularly dangerous because it can target multiple users over time.
Impact	Exploitation of this vulnerability can result in: • Persistent execution of malicious JavaScript in the victim's browser, enabling session hijacking, cookie theft, or redirection to malicious sites. • Compromise of user accounts or sensitive data for all users who access the infected reviews. • Potential exploitation for phishing or advanced social engineering attacks
Recommendations	1) Implement proper input validation and output encoding to sanitize user inputs before storing or reflecting them on the application. 2) Use secure libraries for escaping HTML and JavaScript content to prevent execution of malicious scripts. 3) Apply a Content Security Policy (CSP) to limit the execution of untrusted scripts in the browser. 4) Regularly audit user-generated content for potential malicious payloads. 5) Ensure sensitive pages, such as review downloads, implement strict access controls to reduce exploitation opportunities.
Affected Systems	Product reviews functionality
Threat Level	**Critical**

steps to reproduce:

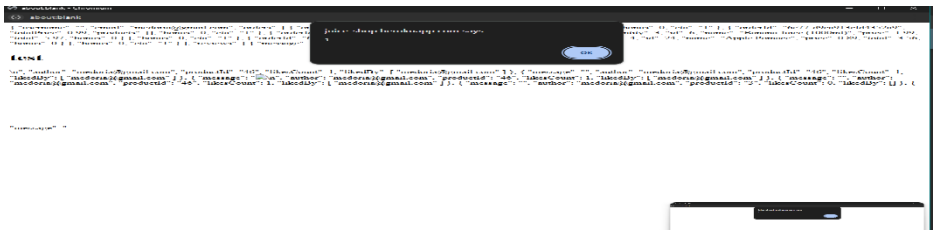
1) First, I accessed reviews page of more than one of the products and include in it XSS payloads to try and test if it will be stored in the server side:
    - <script>alert`1`</script>
    - <svg onload = alert(1)>
    - <img src=x onerror=alert(1)>
    - <script>alert(1)</script>







2) I then requested to download reviews from another account to notice that response appears which I saved in the reviews



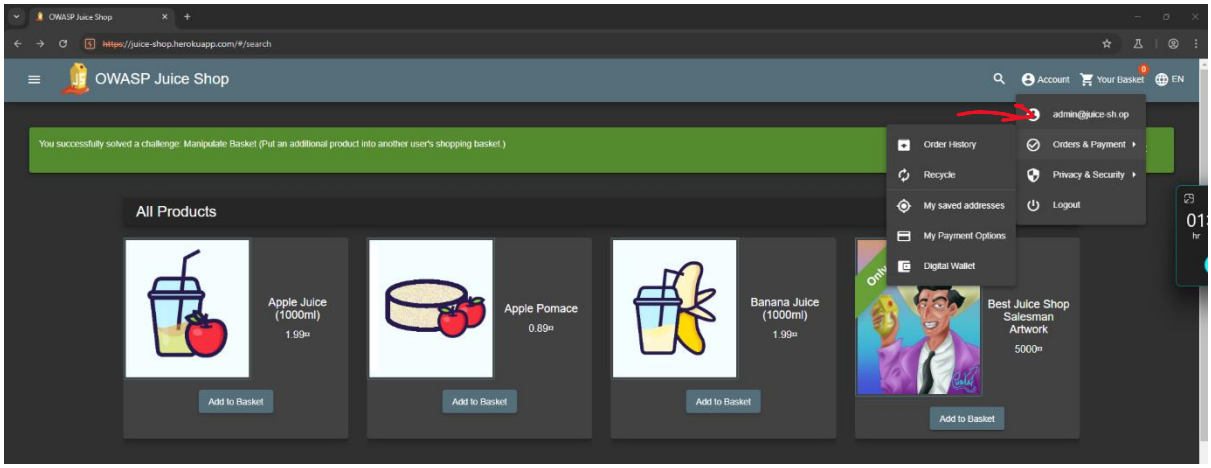
5.6 SQL-injection (Boolean Based): -

Description:	The login page of the application is vulnerable to SQL Injection. By injecting a payload such as `[' OR 1=1 --]` in the username or password field, attackers can manipulate the backend SQL query to bypass authentication. This allows unauthorized access to user accounts, including administrative accounts, by exploiting insecure query construction.
Impact	Exploitation of this vulnerability can result in: • Unauthorized access to user accounts, including administrative privileges. • Full compromise of the database, allowing attackers to read, modify, or delete sensitive data. • Execution of additional SQL commands, potentially leading to system-level attacks.
Recommendations	1) Use parameterized queries or prepared statements in all database queries to prevent SQL Injection. 2) Validate and sanitize all user inputs before processing them in SQL queries. 3) Implement error handling that avoids exposing SQL errors or sensitive information to the user. 4) Use least privilege principles for database accounts to limit the impact of a successful SQL Injection attack. 5) • Regularly audit and test the application for injection vulnerabilities using tools such as SQLMap.
Affected Systems	Login functionality of the OWASP Juice Shop.
Threat Level	**Critical**

steps to reproduce:

1) First: I tried in login page payload [' OR 1=1 --]

2) Observe the response: (I accessed another user's account and my luck it was an admin page)

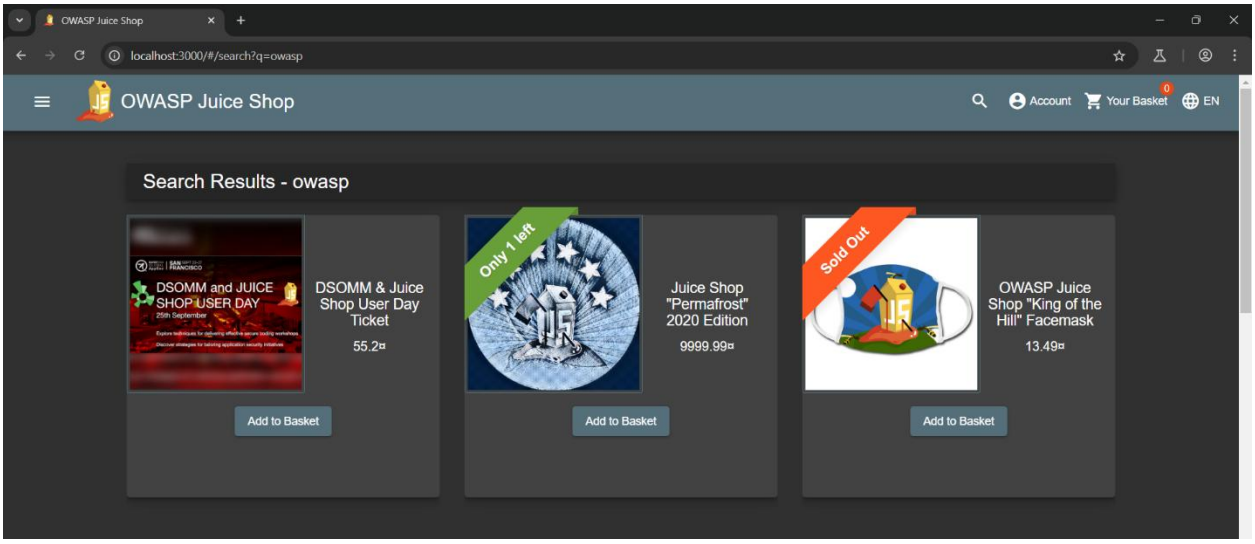


5.7 SQL-injection (UNION Based):

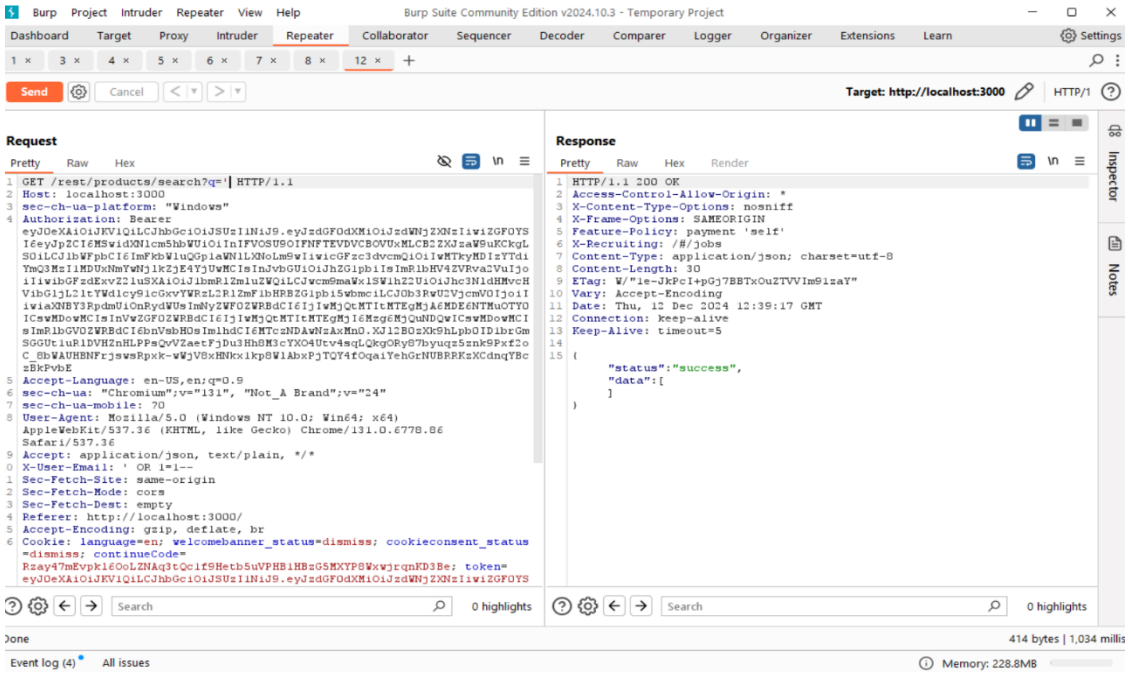
Description:	The search functionality of the application is vulnerable to UNION-based SQL Injection. By injecting malicious payloads into the search parameter, attackers can exploit insecure query construction to retrieve all data from the database. The `UNION SELECT` statement was used to concatenate attacker-controlled queries with legitimate queries, eventually identifying the correct number of columns needed to extract sensitive information from the database.
Impact	Exploitation of this vulnerability can result in: • Retrieval of sensitive database information, such as user credentials, product data, and configuration details. • Exposure of internal database schema and table structures. • Further exploitation of database contents, potentially leading to complete compromise of application data.
Recommendations	• Use parameterized queries or prepared statements to prevent SQL Injection in all database interactions. • Validate and sanitize all user inputs, including query parameters, to block injection payloads. • Disable database error messages from being exposed to users, as they provide critical information for attackers. • Implement database access controls to restrict access to sensitive tables or fields. • Conduct regular security testing, including automated scans and manual testing, to detect and mitigate injection vulnerabilities.
Affected Systems	Search functionality in OWASP Juice Shop.
Threat Level	**Critical**

steps to reproduce:

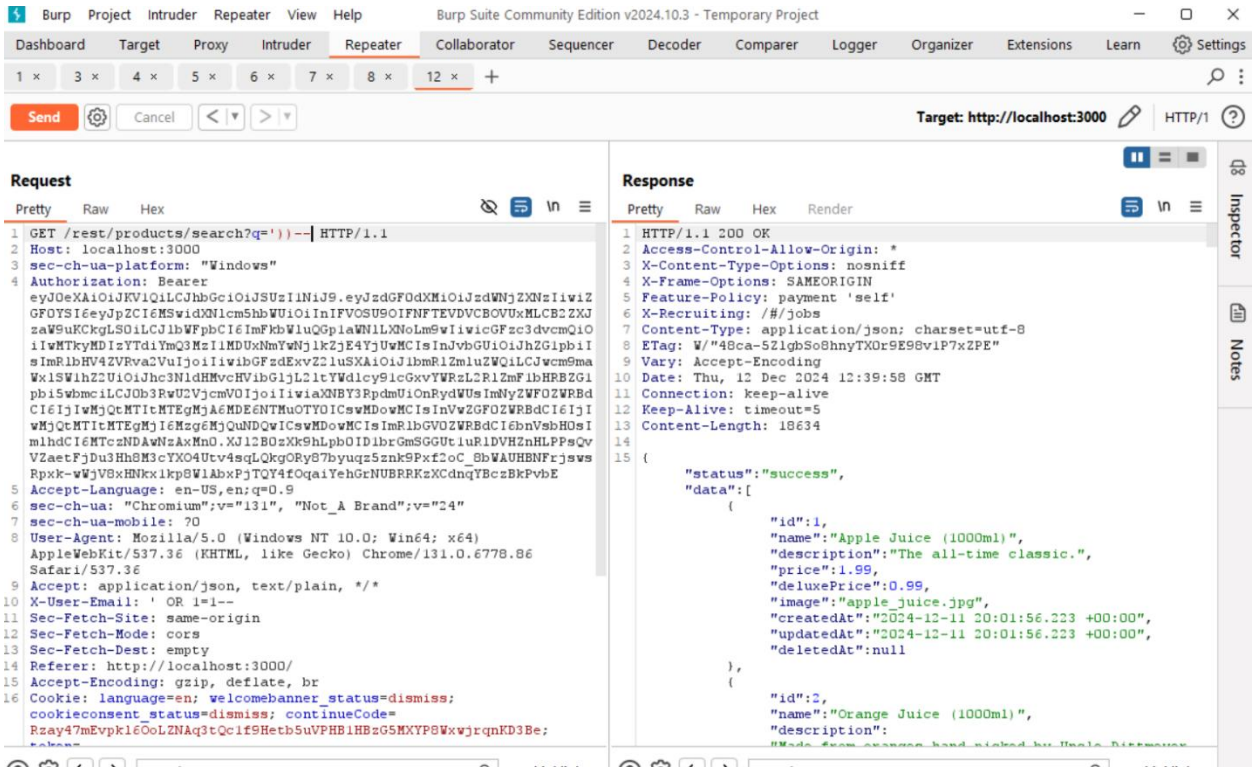
1) In order to test search function for sql vuln I intended to make any search to see if this executed in data base (searched for OWASP) the result is the products contain that word as provided:



2) So I intercepted that request in burb to test it with query param = ‘ I observed the response is 200 ok as provided then there is a vulnerability



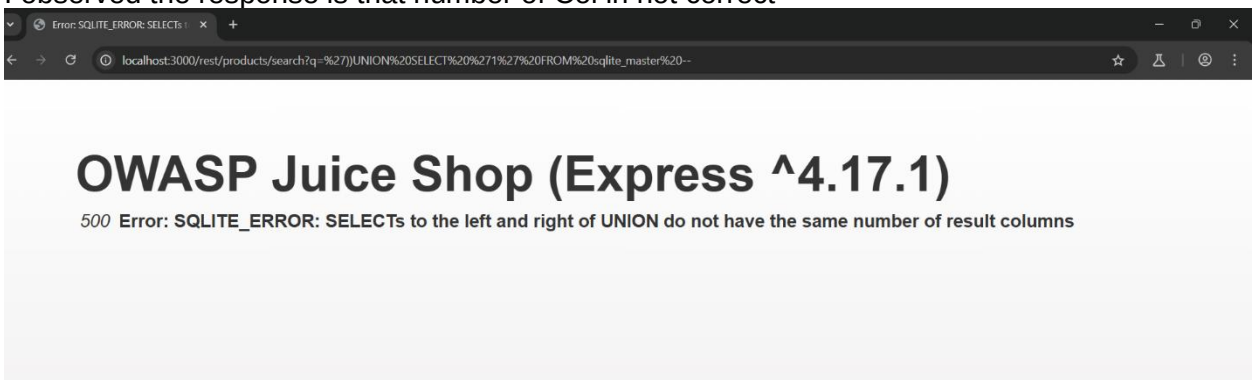
3) So now I try this payload in search query = ‘))-- and observe that response retrieve all the data in database as provided:



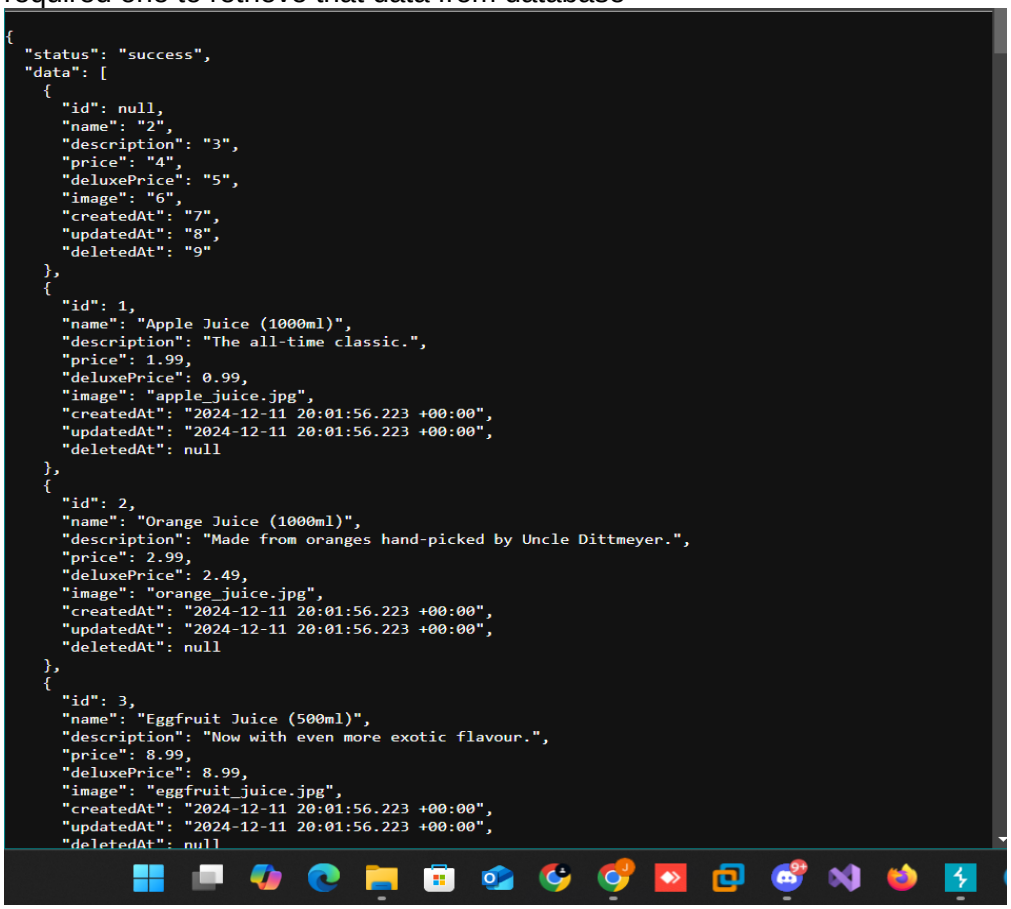
4) Then the target is to see if UNION will work , so I tried this payload in the URL search param:

UNION SELECT ‘1’ FROM sqlite.master –

I observed the response is that number of Col in not correct



5) Here I catched the vulnerability and tried to try diff number of col I found that 9 is the required one to retrieve that data from database

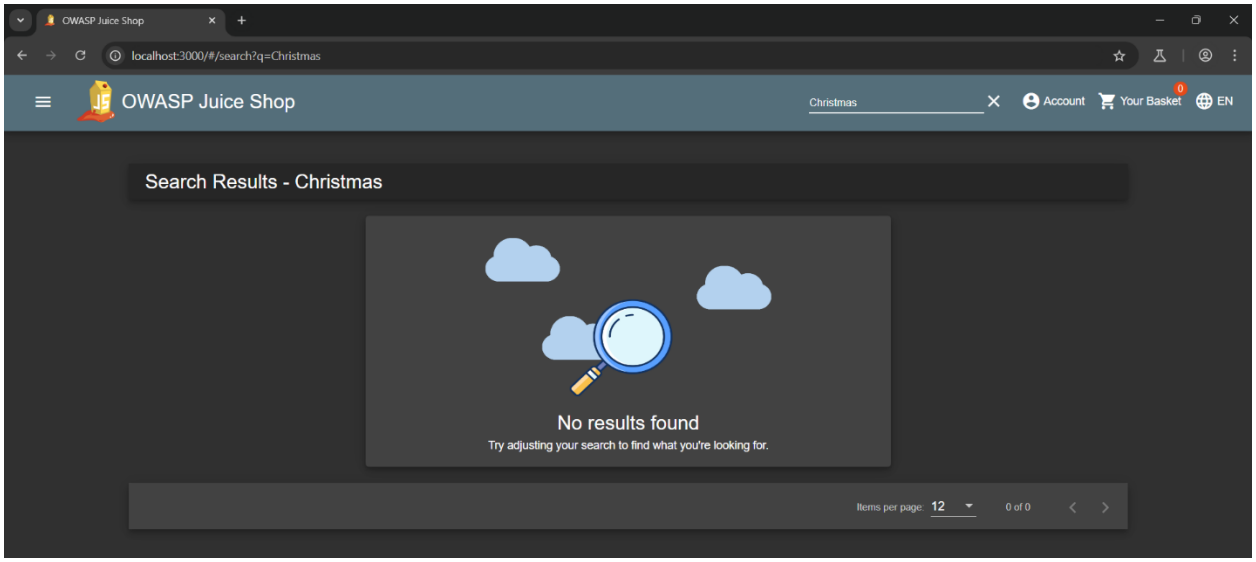


5.8 SQL-injection (Error Based):

Description:	The search functionality in the application is vulnerable to error-based SQL Injection. By injecting payloads that deliberately generate SQL errors, the attacker can gain insights into the database's structure and behavior. This vulnerability was further exploited by manipulating API requests, such as altering product IDs in the basket, to successfully add unauthorized items to the basket and complete an order that should not be permitted.
Impact	Exploitation of this vulnerability can result in: • Unauthorized access to sensitive database information by leveraging error messages to map database structures. • Manipulation of database queries to perform unauthorized actions, such as adding restricted products to a basket. • Violation of business logic, enabling attackers to bypass restrictions on products or user actions.
Recommendations	• Use parameterized queries or prepared statements in all database interactions to eliminate injection vulnerabilities. • Sanitize and validate all user inputs, including query parameters, to prevent SQL errors or manipulation. • Suppress detailed database error messages to ensure sensitive information is not exposed to users. • Implement robust authorization checks on sensitive API endpoints to ensure users can only interact with authorized data. • Perform regular security testing and code reviews to detect and fix SQL Injection vulnerabilities.
Affected Systems	Backend API endpoints handling basket operations. Search functionality in OWASP Juice Shop.
Threat Level	**Critical**

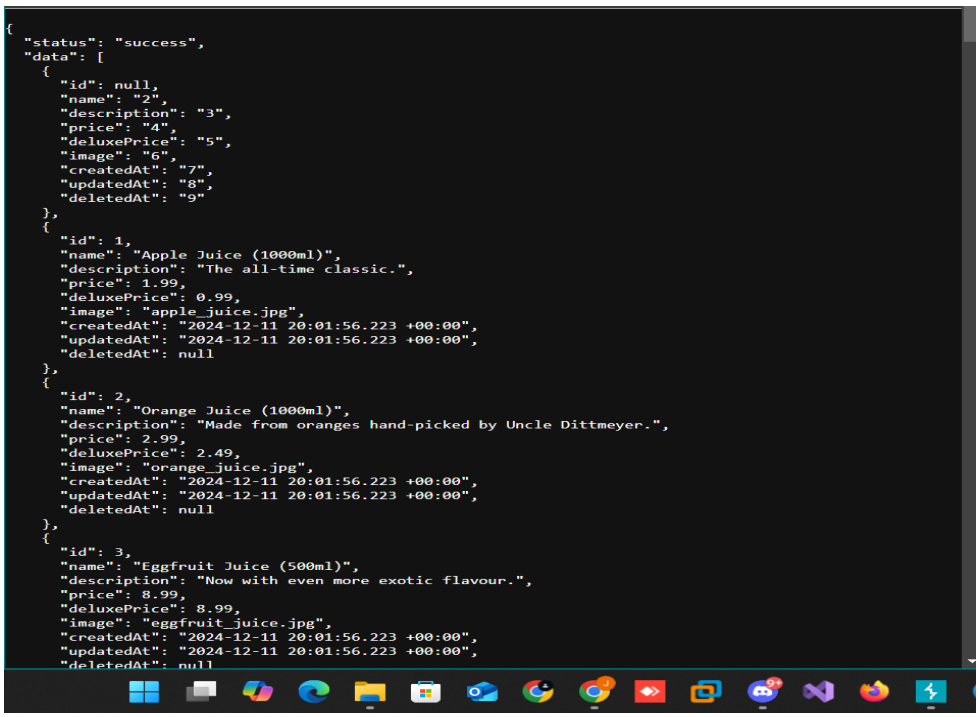
steps to reproduce:

1) In order to test search function for sql vuln I intended to make any search to see if this executed in database (searched for Christmas) the result is the products contain that word as provided:

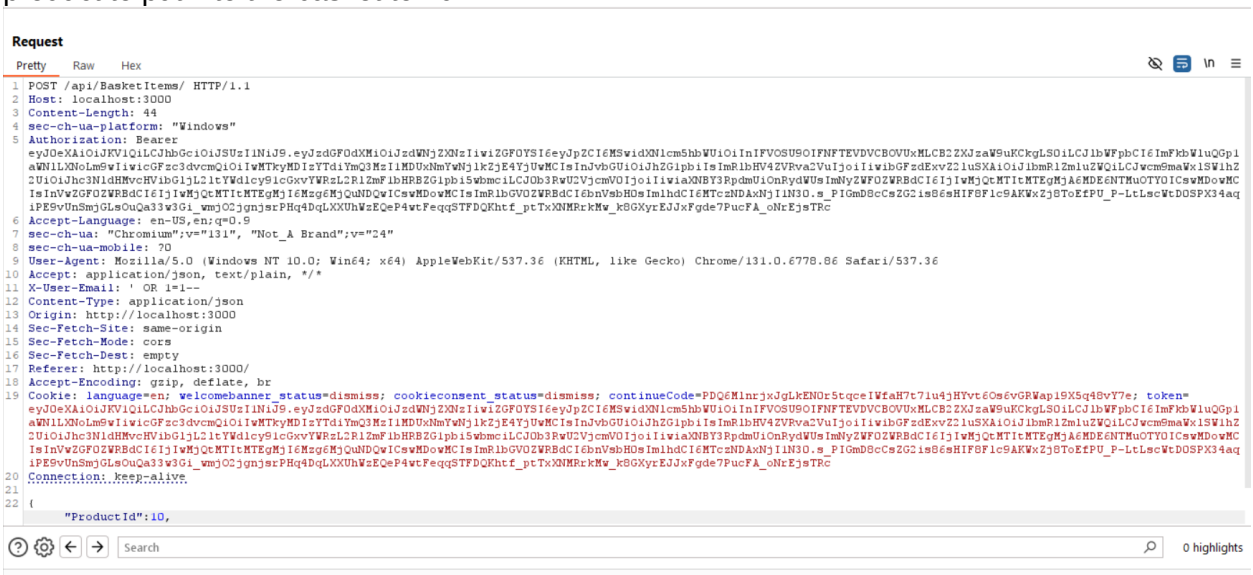


2) So I intercepted that request in burb to test it with query param = ')) I observed the response includes error message so I noticed that the request is sent to database and get executed

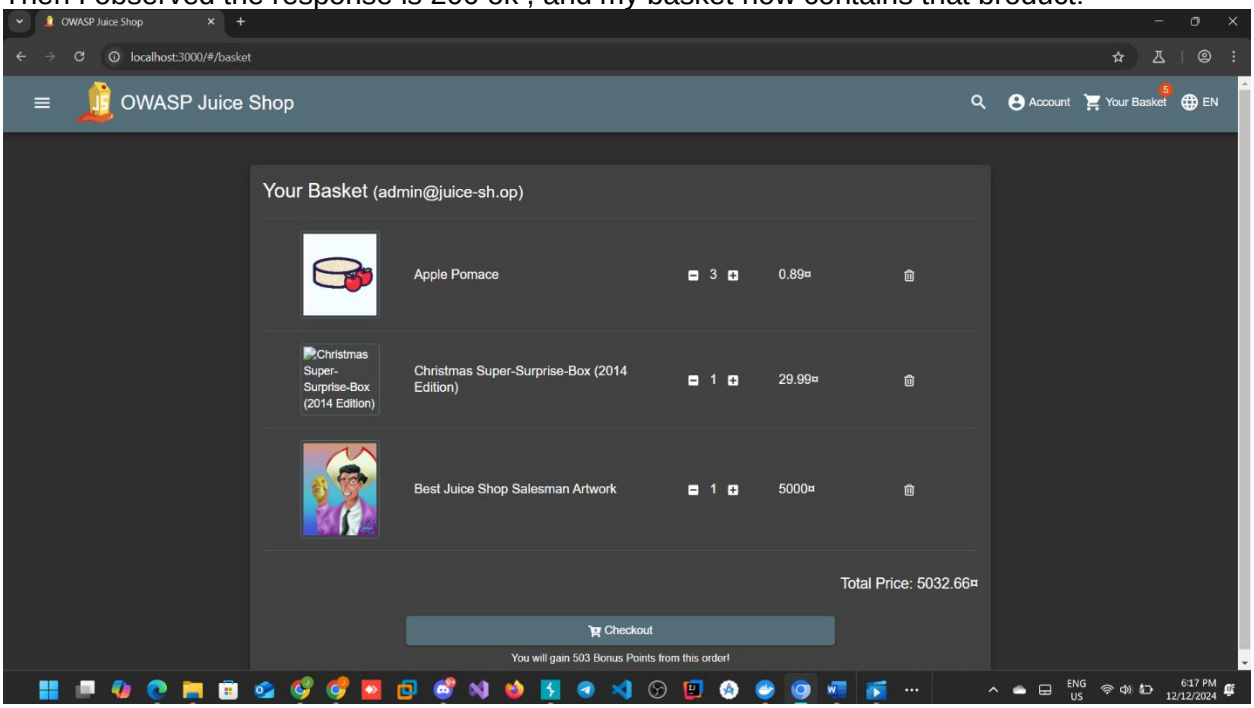
3) Try now put two dashes to comment every thing else in the query param in the url q=)) — I observed the response all items is retrieved from data base :



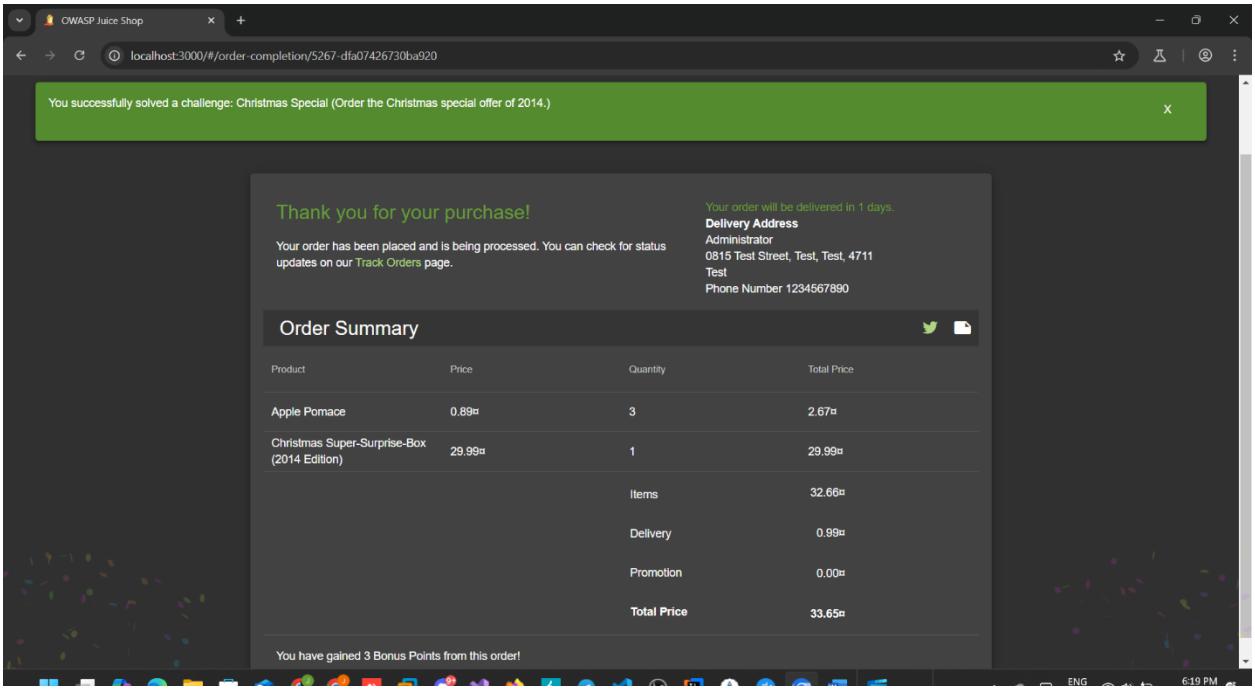
4) I observed that the item of id = 10 not showed in the home page so I would try to add it to my basket and check on it to make order
5) So I intercepted the request of adding to basket /api/BasketItem/ and change the id of the product to put into the basket to 10



6) Then I observed the response is 200 ok , and my basket now contains that product:



7) Then finally I manged to make the order successfully which should not happen

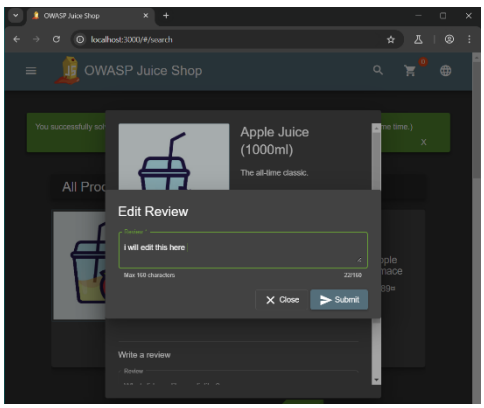

OWASP Juice Shop
localhost:3000/#/order-completion/5267-dfa07426730ba920
You successfully solved a challenge: Christmas Special (Order the Christmas special offer of 2014.)
x
Thank you for your purchase!
Your order has been placed and is being processed. You can check for status updates on our Track Orders page.
Your order will be delivered in 1 days.
Delivery Address
Administrator
0815 Test Street, Test, Test, 4711
Test
Phone Number 1234567890
Order Summary
Product
Price
Quantity
Total Price
Apple Pomace
0.89¤
3
2.67¤
Christmas Super-Surprise-Box (2014 Edition)
29.99¤
1
29.99¤
Items
32.66¤
Delivery
0.99¤
Promotion
0.00¤
Total Price
33.65¤
You have gained 3 Bonus Points from this order!
ENG
6:19 PM

5.9 NO SQL:

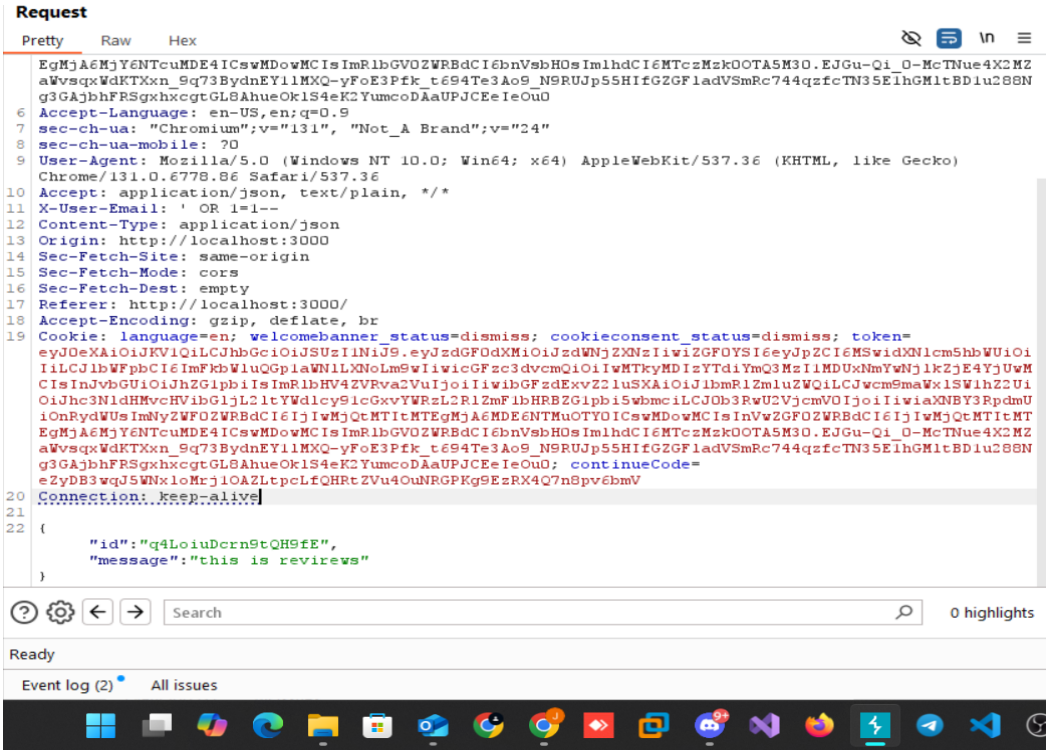
Description:	NoSQL Injection occurs when an attacker manipulates NoSQL queries to perform unauthorized actions in the database. In this instance, the review editing functionality of the OWASP Juice Shop website was tested. A request to edit a review contained the product ID and a message in the request body. By modifying the body to include NoSQL operators such as `$set` and `$in`, it was possible to edit multiple reviews simultaneously, indicating the database improperly processes NoSQL queries.
Impact	- Unauthorized modification of multiple database records. - Exposure of sensitive data through injection techniques. - Circumvention of access controls, enabling mass updates or deletions. - Possibility of database corruption or unauthorized actions affecting application integrity.
Recommendations	1) Validate and sanitize all inputs to block NoSQL operators and special characters. 2) Use Object-Document Mapping (ODM) libraries that enforce strict schema validation. 3) Implement role-based access controls to restrict actions on the database. 4) Perform regular penetration tests to detect and remediate NoSQL vulnerabilities. 5) Monitor logs for suspicious activity and implement rate limiting to reduce the impact of automated attacks.
Affected Systems	Review Editing Endpoint
Threat Level	**High**

steps to reproduce:

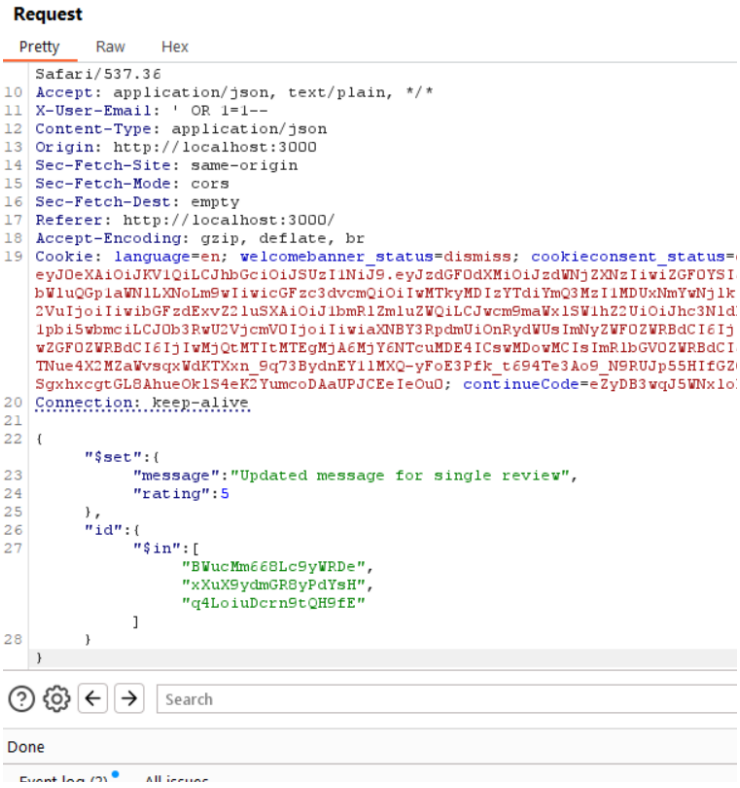
1) I intended to test if using a nosql query will affect the database so I opened review page of one of the products to get the request of edit review endpoint to check if I can edit more than one review in the database at a time.



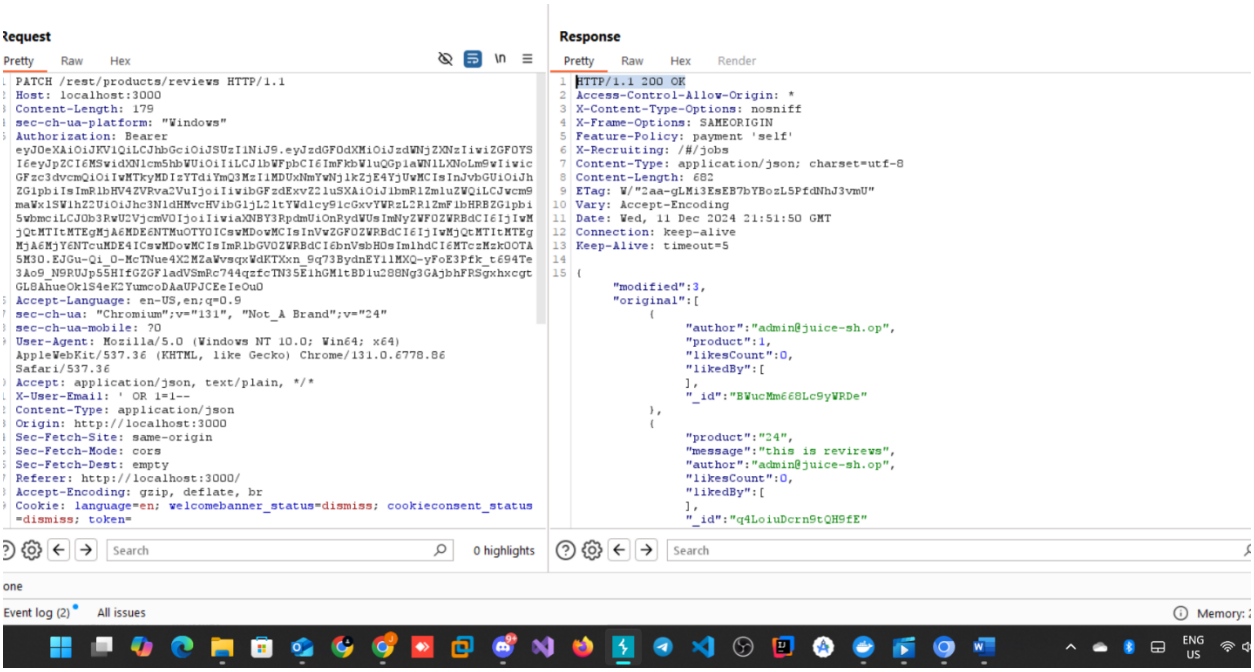
2) So the request has this form in the burb the body contain {id for product and the mssge}



3) now I will try to send another body include id for more than one product using $set ,$in to execute NOSQL query in the database



4) we found that the response is 200 ok and I made that action so the database has NOSQL database and is vulnerable to that attack

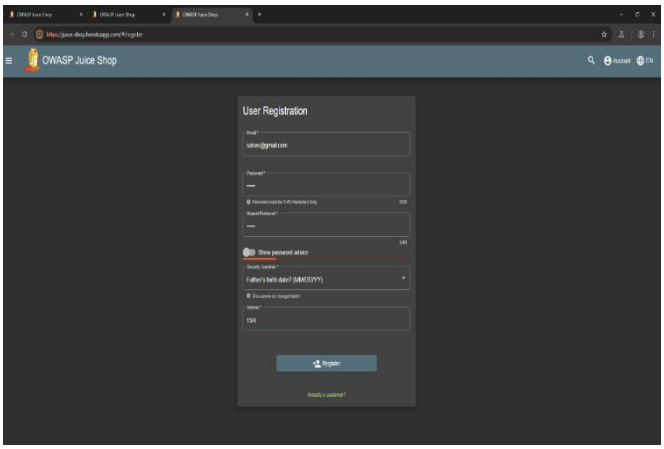


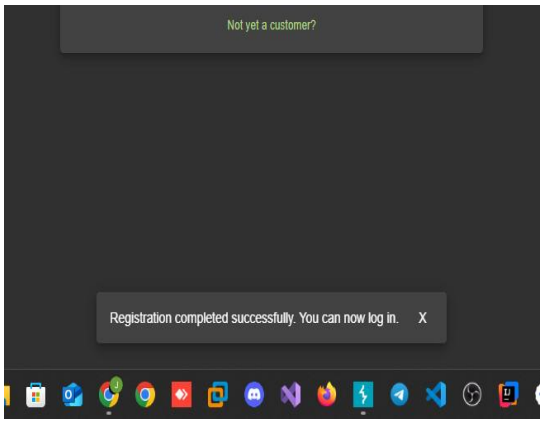
5.10 Authentication:

Description:	The system accepts weak passwords, such as "12345," during registration, which exposes it to brute-force attacks and unauthorized access to user accounts. This indicates a lack of password strength enforcement, allowing attackers to easily guess weak credentials.
Impact	Exploitation of this vulnerability can result in: • Unauthorized access to user accounts, enabling attackers to steal sensitive information. • Increased risk of brute-force attacks that can compromise multiple user accounts. • Potential for account takeover, leading to data breaches or malicious activities within the system.
Recommendations	1. Implement password strength policies requiring a mix of uppercase, lowercase, digits, and special characters. 2. Enforce minimum password length, such as at least 8 characters. 3. Introduce account lockout mechanisms after a set number of failed login attempts to mitigate brute-force attacks. 4. Utilize CAPTCHA on login and registration forms to prevent automated attack scripts. 5. Educate users to create strong, unique passwords for their accounts.
Affected Systems	Registration and login endpoints of the application.
Threat Level	**High**

steps to reproduce:

1) First, I tried to test if the system accept week password or not , putting the password 12345 in the register field and the site accepted it as a valid password which exposed it to Brute force attack and unauthorized access to users account .



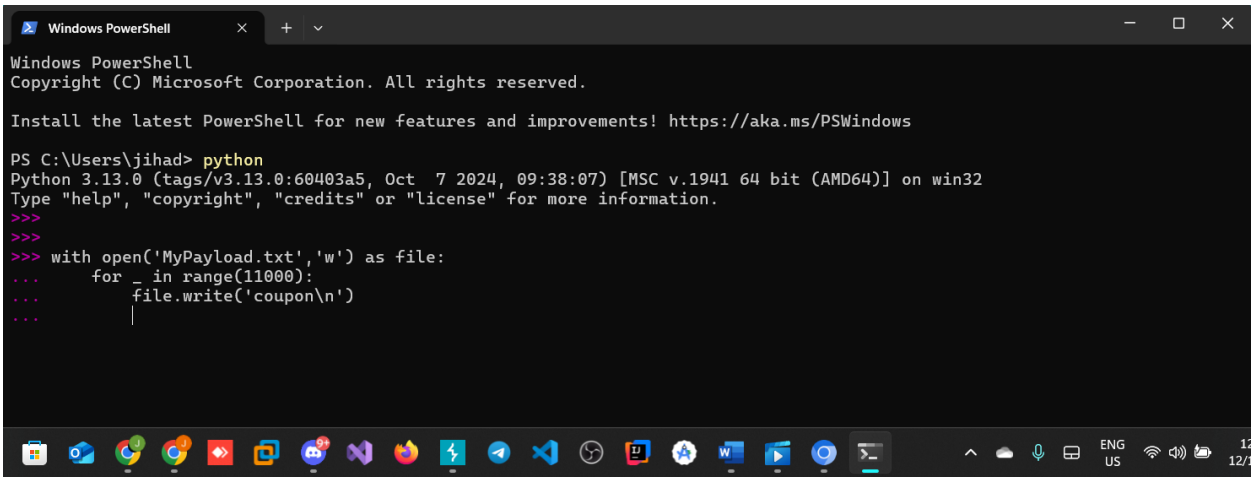


5.11 IDOR-Broken Authentication (Brute Force Attack): -

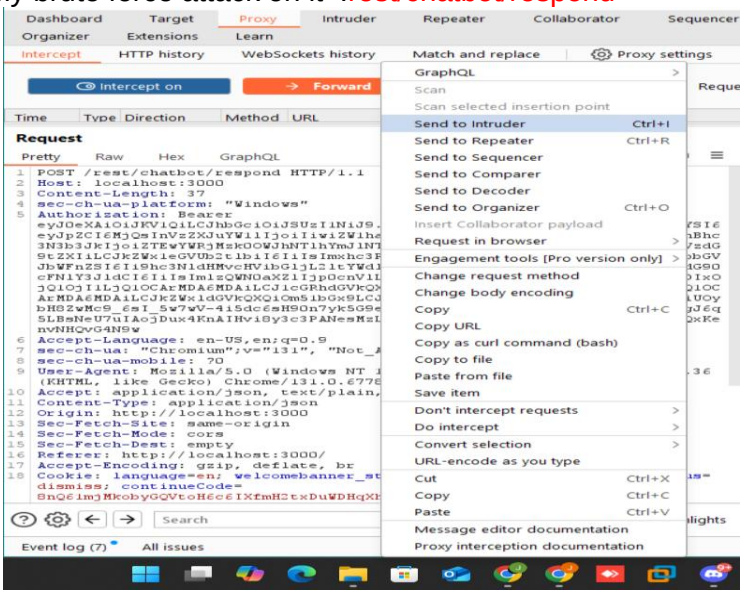
Description:	The chat bot feature of the application is vulnerable to a brute force attack due to insufficient rate limiting and lack of proper authentication mechanisms. By sending multiple requests to the `/rest/chatbot/respond` endpoint with the word "coupon" repeated 11,000 times, the attacker was able to exploit the vulnerability and retrieve a valid 10% discount coupon.
Impact	Exploitation of this vulnerability can result in: • Abuse of application features, such as generating unlimited discount coupons, causing financial losses. • Increased server load or potential denial-of-service (DoS) due to brute force attacks. • Violation of the application's integrity and trust, as users can exploit the system for unauthorized benefits.
Recommendations	• Implement rate limiting on the `/rest/chatbot/respond` endpoint to prevent brute force attacks. • Add CAPTCHA verification to ensure that requests are coming from legitimate users and not automated scripts. • Enforce authentication and authorization checks to restrict access to coupon generation functionality. • Monitor and log unusual activity on endpoints to detect and mitigate brute force attempts. • Regularly test and audit application endpoints to ensure compliance with security best practices.
Affected Systems	Backend API endpoint: /rest/chatbot/respond.
Threat Level	**High**

Steps to reproduce:

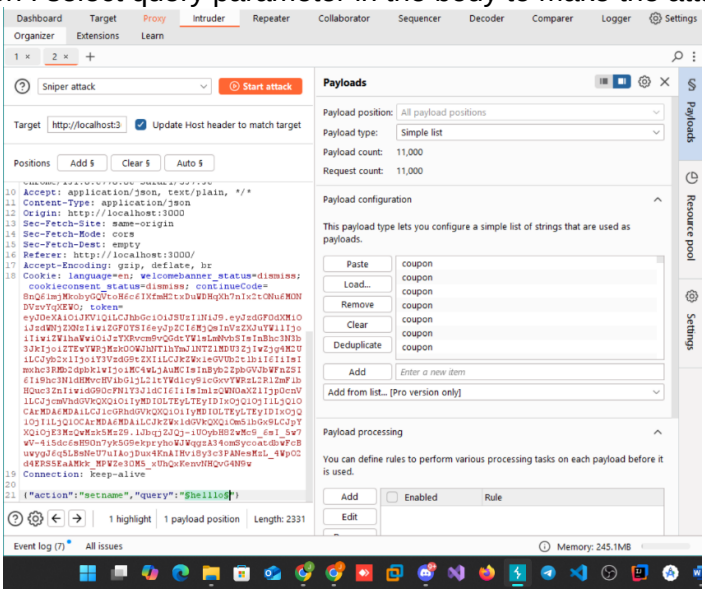
1) I intended to test if there will be a vulnerability in the area of the chat bot if I send to him multiple messages requesting an coupon could it respond with one to me
so In a text file I write 11000 coupon word by python code



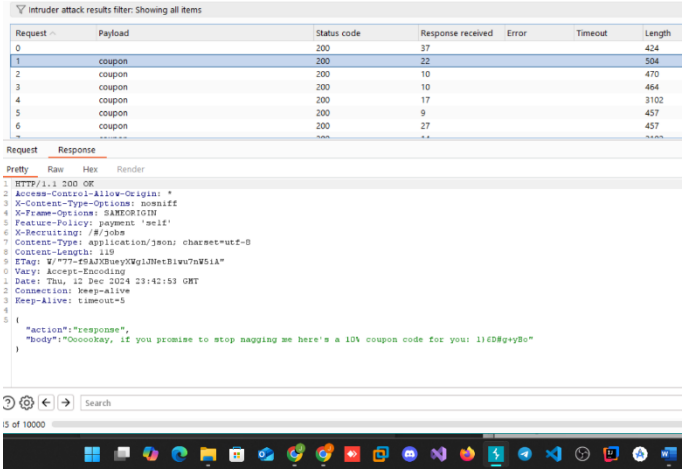
2) I then intercepted the request in the chat bot sending message and send it to intruder to apply brute force attack on it :rest/chatbot/respond



3) Then I select query parameter in the body to make the attack on it



4) I checked out the response body to see that my attack successed to get a 10%copoun from the bot

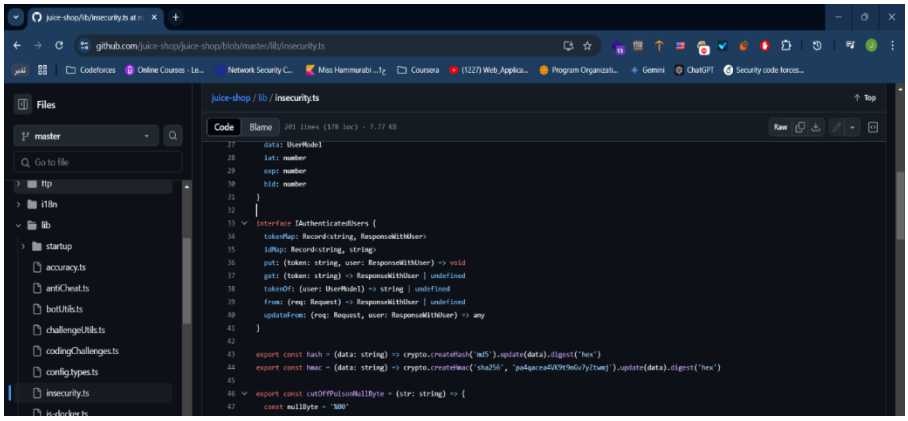


5.12 Cryptographic Failure:

Description:	The application uses the MD5 hashing algorithm, as identified in the source code under the `insecurity.ts` file on GitHub. MD5 is widely considered insecure due to its vulnerability to collision and pre-image attacks. This cryptographic failure could allow attackers to crack hashed passwords or other sensitive data, leading to potential compromise of user accounts or system integrity.
Impact	Exploitation of this vulnerability can result in: • Easy cracking of hashed passwords or sensitive data using modern computational resources or precomputed rainbow tables. • Unauthorized access to user accounts or system resources. • Failure to comply with modern security standards and best practices, exposing the application to regulatory penalties
Recommendations	• Replace MD5 with a secure hashing algorithm, such as SHA-256 or bcrypt, to ensure the integrity and security of hashed data. • Use a strong salting mechanism with the hashing process to defend against rainbow table attacks. • Perform regular code reviews and security audits to identify and replace insecure cryptographic implementations. • Educate developers on modern cryptographic standards to avoid insecure practices in future code.
Affected Systems	Any functionality relying on MD5 for hashing, such as password storage or token generation.
Threat Level	**Critical**

steps to reproduce:

1) I intended to see what the hashing algorithm is this site use so I accessed the src code on the git hub of the site at: github.com/juice-shop/juice-shop/blob/master/lib/insecurity.ts
2) I found that they are using md5 as hashing algorithm which is very week and vulnerable

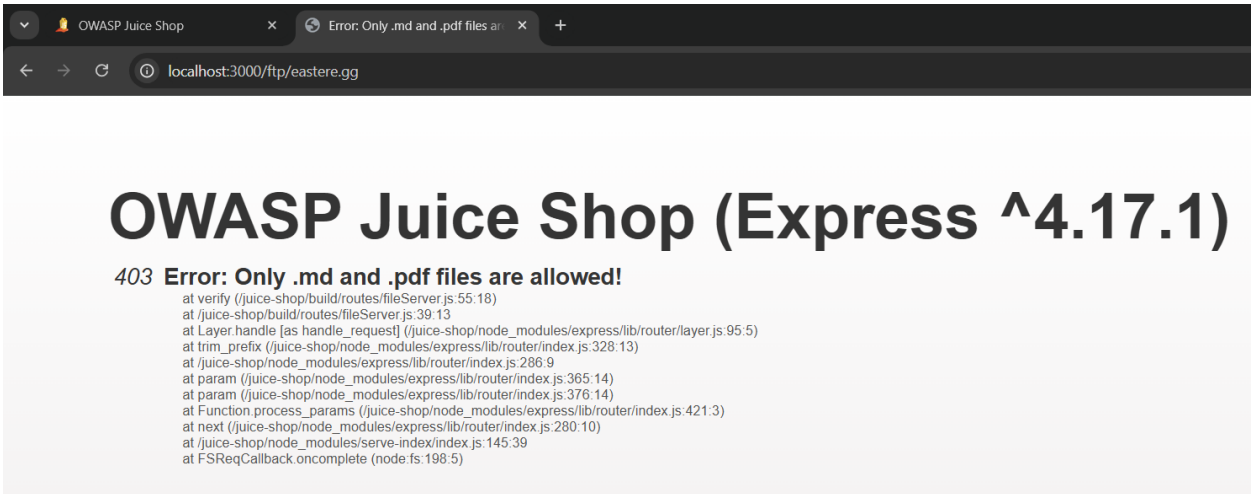


5.13 Local file Read by file extension bypass:

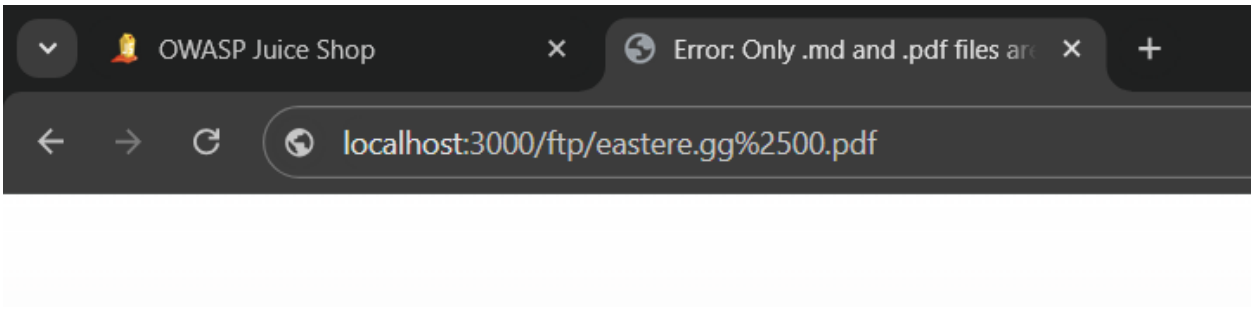
Description:	The application improperly enforces file type restrictions, allowing attackers to bypass these restrictions using techniques like appending a null byte (`%00`) to the filename. By accessing the `/ftp` directory and appending `%2500` to the filename (e.g., `eastere.gg%2500`), the attacker was able to bypass the restriction and download an otherwise inaccessible file.
Impact	Exploitation of this vulnerability can result in: • Unauthorized access to sensitive files stored on the server. • Disclosure of internal system files, configurations, or application logic. • Further exploitation if the downloaded files contain credentials, sensitive data, or exploitable information.
Recommendations	• Properly validate and sanitize user-supplied file paths on the server side to prevent null byte or type-based bypasses. • Enforce strict file access policies to ensure only authorized files can be downloaded. • Avoid exposing sensitive directories like `/ftp` to public users. • Implement logging and monitoring to detect unusual file access patterns. • Regularly audit and test file upload and download functionality for bypass vulnerabilities.
Affected Systems	/ftp directory and related file download functionality
Threat Level	**High**

steps to reproduce:

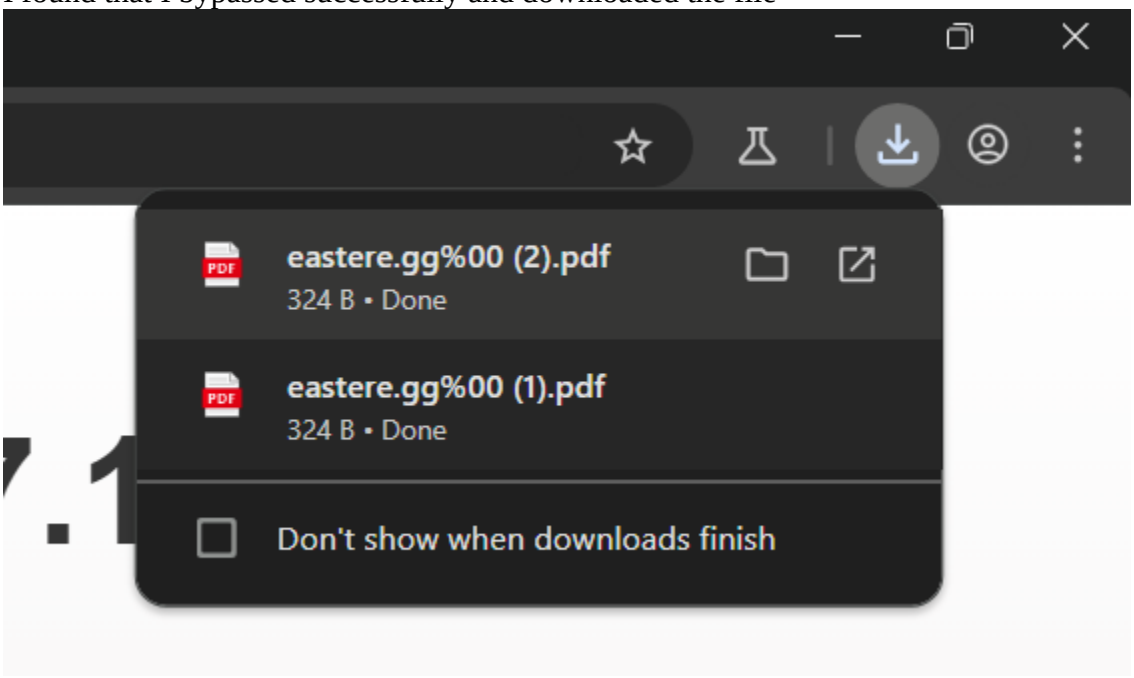
1) I tried to test if I can download file un downloadable so I accessed /ftp I found that eastere.gg file is not allowed to be downloaded.



2) I then tried to bypass the type restriction on the file to download it by providing the Null Byte %2500 at the end of the file as this:



3) I found that I bypassed successfully and downloaded the file

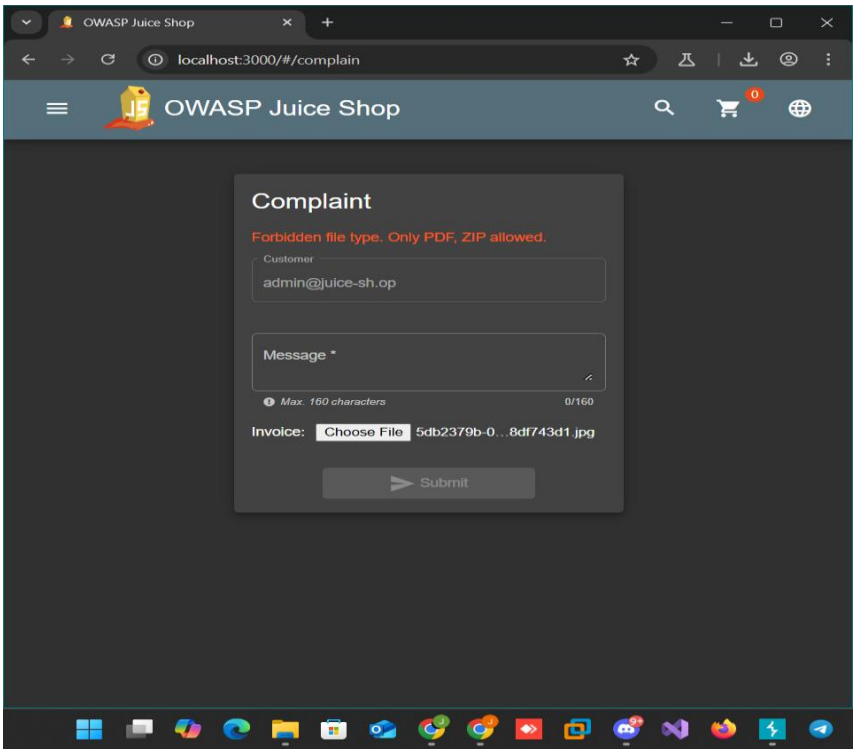


5.14 Security misconfiguration

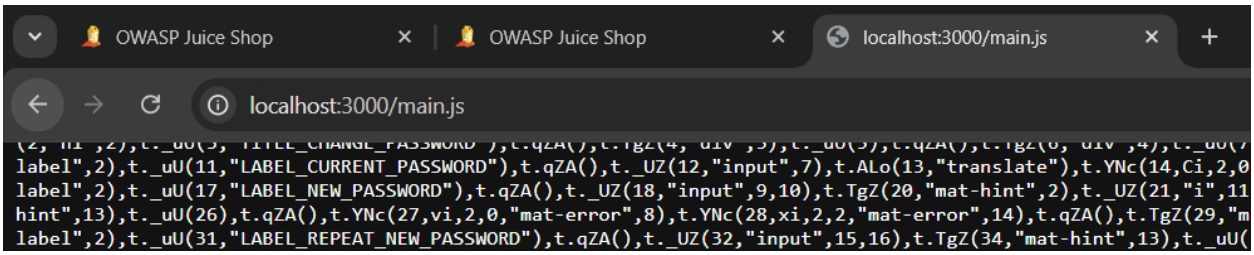
Description:	The application is improperly configured to enforce file type restrictions on the complaint page. While the UI indicates that only `.pdf` and `.zip` files are allowed, inspection of the source code in `main.js` revealed that `application/xml` is also an accepted MIME type. This allowed the attacker to bypass the visible restriction and upload a potentially harmful `.xml` file.
Impact	Exploitation of this vulnerability can result in: • Uploading unauthorized file types that may contain harmful content. • Potential exploitation if the uploaded `.xml` file is processed insecurely (e.g., XXE attacks). • Inconsistent security policies leading to confusion among users and administrators.
Recommendations	• Exploitation of this vulnerability can result in: • Uploading unauthorized file types that may contain harmful content. • Potential exploitation if the uploaded .xml file is processed insecurely (e.g., XXE attacks). • Inconsistent security policies leading to confusion among users and administrators.
Affected Systems	Complaint page file upload functionality
Threat Level	**Medium**

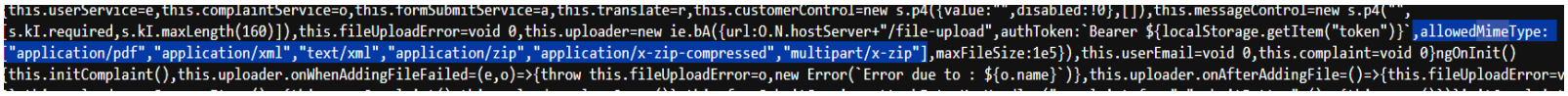
steps to reproduce:

1) In the complaint page I tried to upload file of any type I noticed file restriction in the site to just pdf and zip

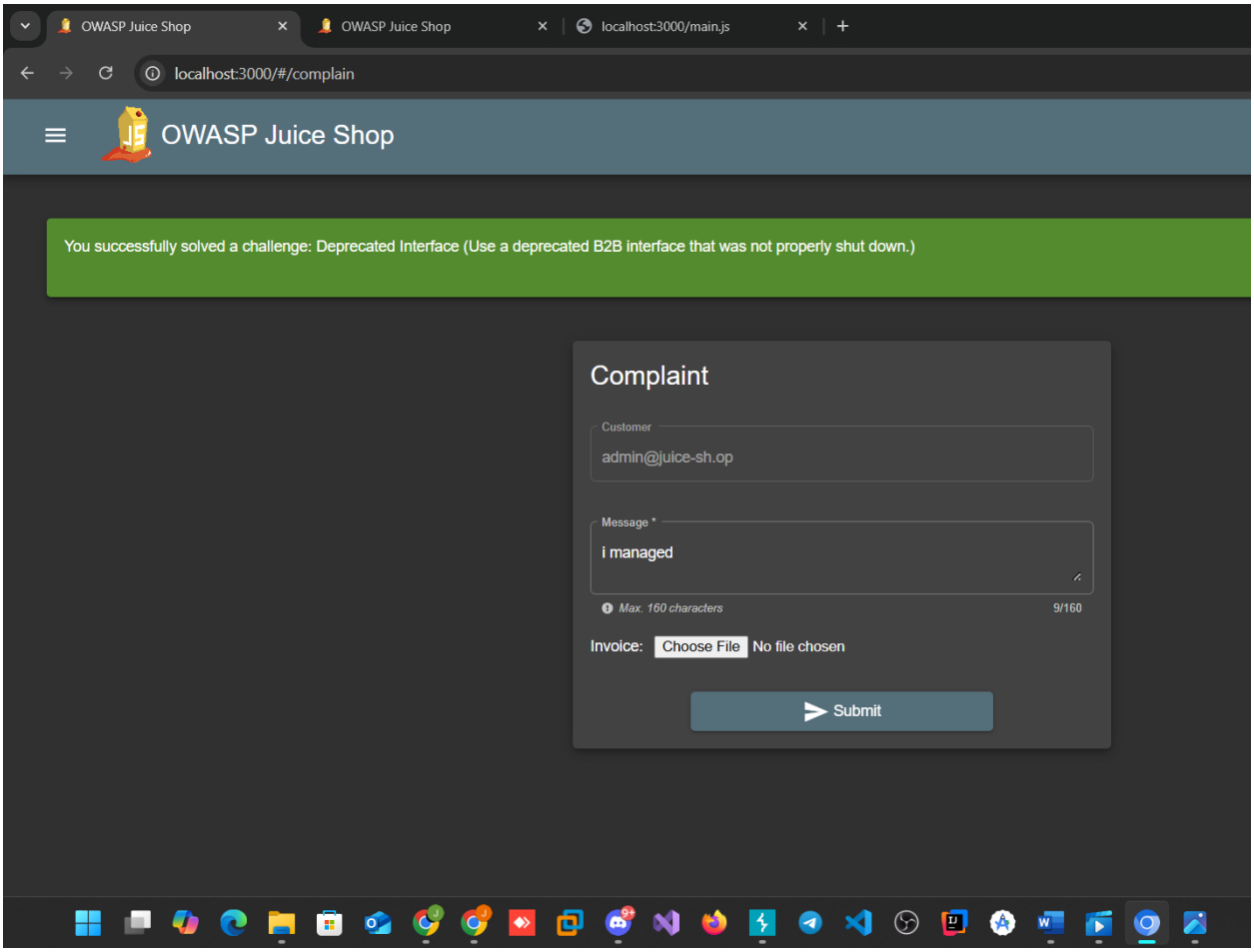


2) so I Intended to test to upload a harmful file ,from inspection tool I find that there is a main.js file that I can see some src code in it , I noticed inside that file that allowedMimeType:"application/xml" in addition to another types





3) so, I tried to upload that kind of file .xml and I successed in this.

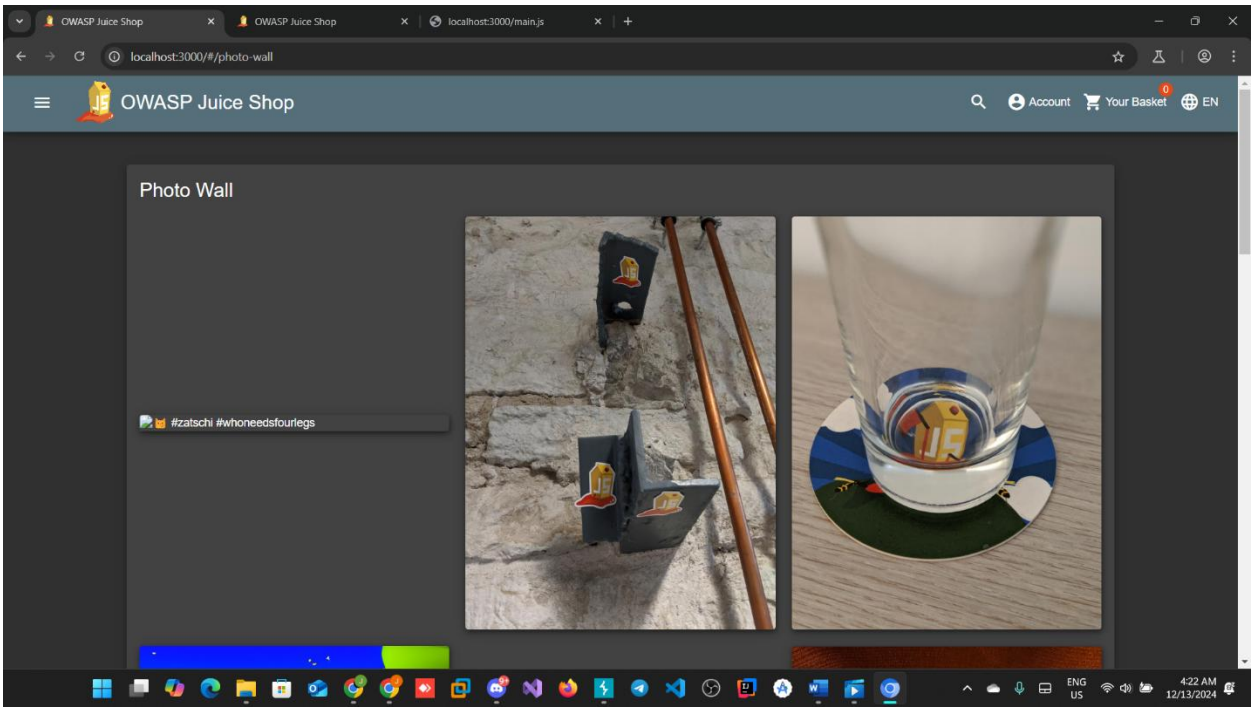

OWASP Juice Shop
OWASP Juice Shop
localhost:3000/main.js
localhost:3000/#/complain
OWASP Juice Shop
You successfully solved a challenge: Deprecated Interface (Use a deprecated B2B interface that was not properly shut down.)
Complaint
Customer
admin@juice-sh.op
Message *
i managed
Max. 160 characters
9/160
Invoice:
Choose File
No file chosen
Submit

5.15 improper input validation (Missencoding)

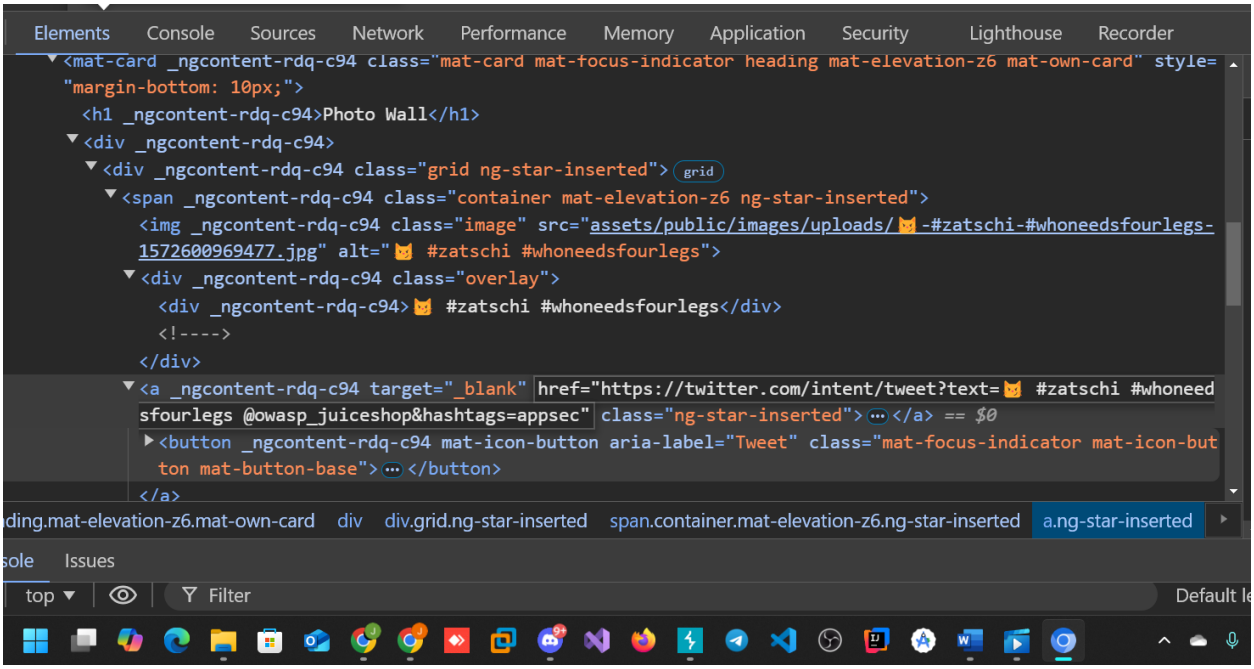
Description:	The application fails to properly handle and validate encoded characters in URLs. On the photo wall page, a hidden photo was inaccessible due to a `#` character placed before the `src` attribute. By replacing the `#` with its encoded equivalent (`%23`), the attacker was able to bypass the restriction and access the hidden photo.
Impact	Exploitation of this vulnerability can result in: • Unauthorized access to restricted content, such as hidden photos or sensitive resources. • Potential exposure of private or sensitive application assets. • A broader attack surface for exploitation, such as manipulation of encoded values in other parts of the application.
Recommendations	• Properly validate and sanitize all user-supplied inputs, including encoded characters, to ensure they align with expected values. • Avoid relying on frontend obfuscation, such as `#` in the `src` attribute, to enforce access restrictions. • Implement strict server-side access controls to ensure unauthorized users cannot access restricted content. • Perform regular audits of encoding and decoding logic in application URLs and parameters.
Affected Systems	• Photo wall page in OWASP Juice Shop. • Any feature relying on encoded or decoded URLs for access control.
Threat Level	**Medium**

steps to reproduce:

1) In the photo wall page I see that there is a hidden photo and I am not allowed to see it

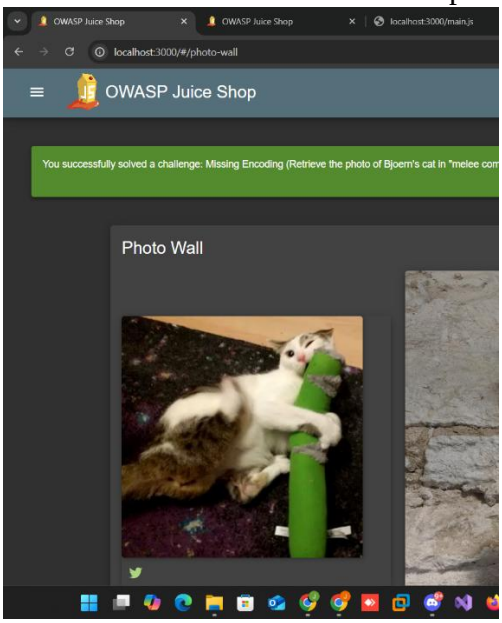


2) so I tried to inspect the page I found that # is putted before src of the image



3) So I tried to put the encoded value of # which is %23 to see what happens

4) I found that I could see the hidden photo

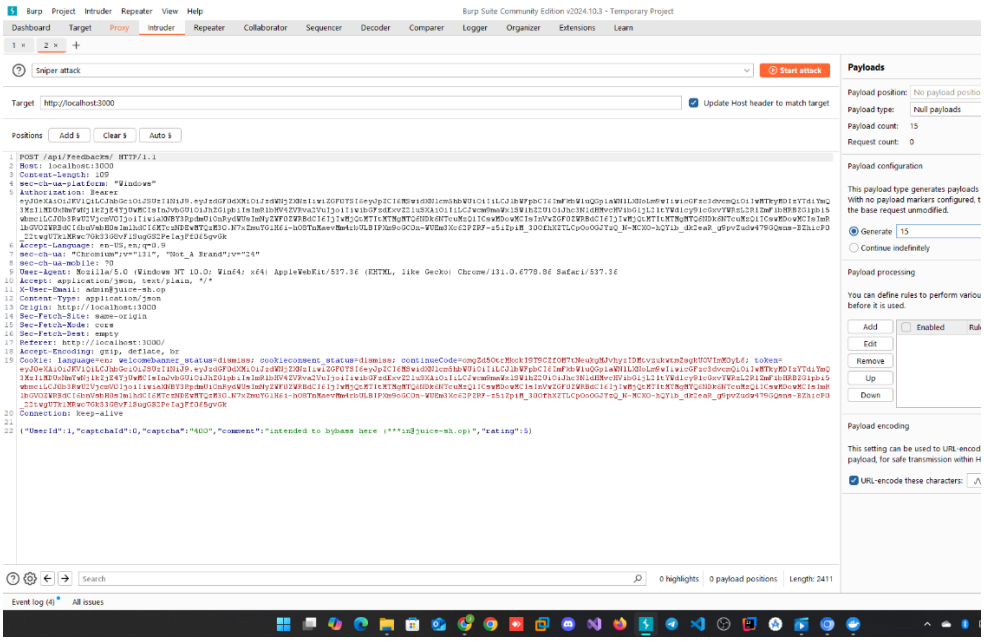


5.16 Broken Anti Automation (Captcha) [BONUS]

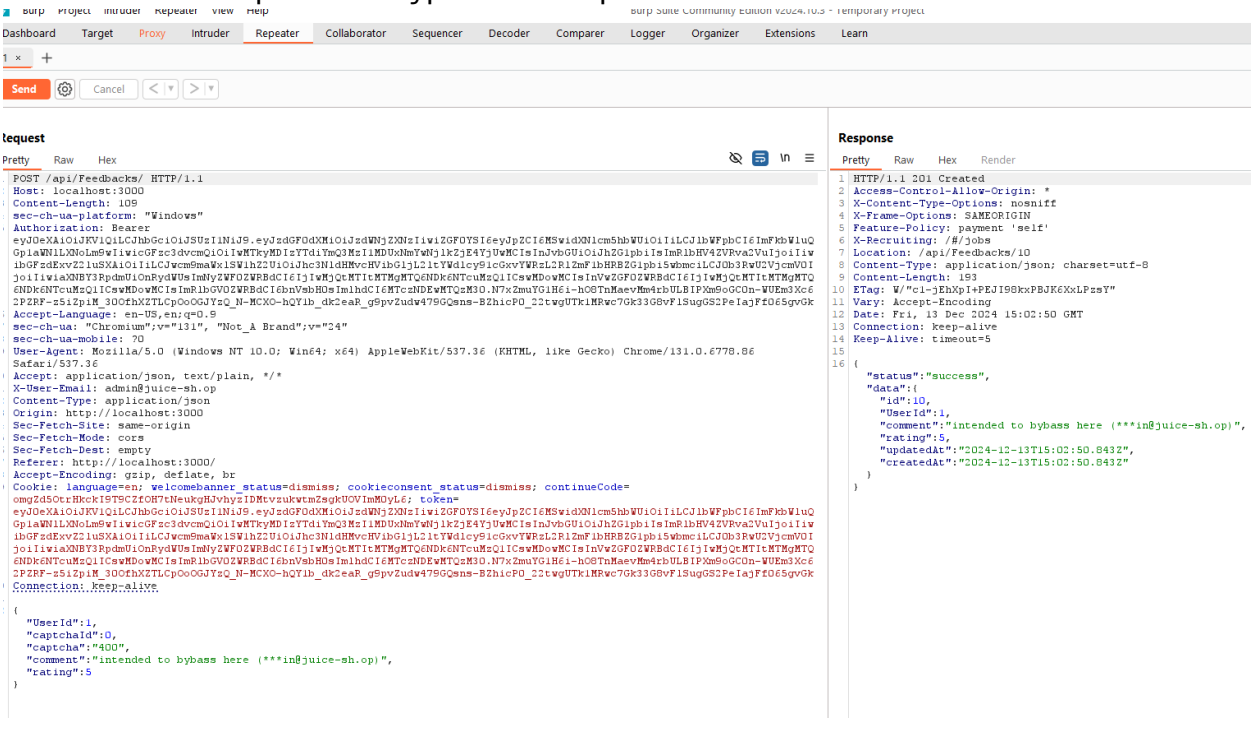
Description:	The application implements a captcha mechanism on the customer feedback page but fails to validate captcha responses properly. By intercepting and analyzing the requests to the `/api/feedbacks/` endpoint, it was discovered that the captcha ID can be bypassed by repeatedly sending the same response. This vulnerability allows attackers to automate requests and submit multiple feedback entries without solving the captcha.
Impact	Exploitation of this vulnerability can result in: • Automated spam or abusive content submissions. • Decreased application reliability due to high server load from bot traffic. • Potential reputational damage from an unregulated feedback system.
Recommendations	• Implement a robust captcha mechanism with proper server-side validation to ensure each response is unique and correct. • Introduce rate-limiting to restrict the number of feedback submissions from a single user or IP address within a given time. • Use dynamic captcha IDs and answers that cannot be easily guessed or reused. • Log and monitor feedback submissions to detect and block automated attacks.
Affected Systems	Backend API endpoint: /api/feedbacks/.
Threat Level	**High**

steps to reproduce:

1) I am trying to bypass captcha in the customer feedback page
2) I write a comment and made a rate to intercept the request /api/feedbacks/



3) I send this request to repeater to check some things which I found that the captcha id= 0 has answer 400 success response , I send it to intruder to make a quick attack 10 requests to bypass this captcha



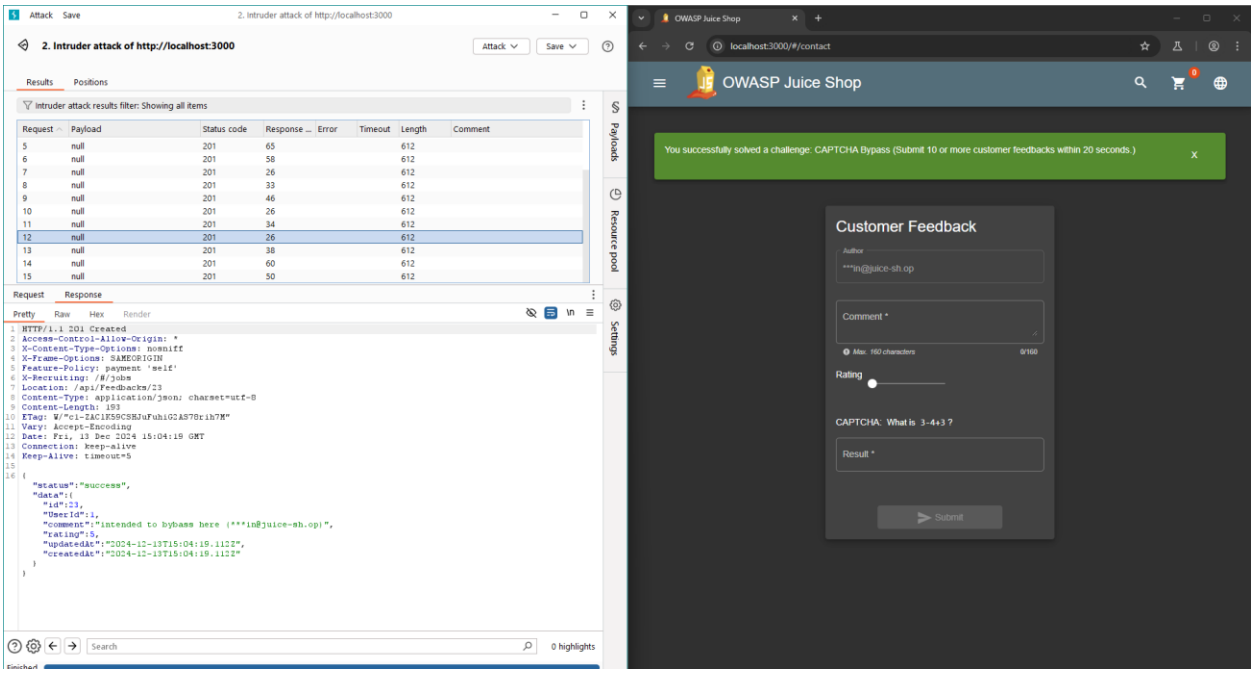
```
POST /api/Feedbacks/ HTTP/1.1
Host: localhost:3000
Content-Length: 109
sec-ch-ua-platform: "Windows"
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMTItMTMgMTQ6NDk6NTcuMzQ1ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMTItMTMgMTQ6NDk6NTcuMzQ1ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTczNDEwMTQzM30.N7xZmuYGiHéi-hOSTnMaevMm4rbULBIPXm9oGCOn-WUEm3Xc62PZRF-z5iZpiM_3OOfhXZTLCpOoOGJYzQ_N-MCXO-hQYib_dk2eaR_g9pvZudv479GQsns-BZhicPO_22twgUTkiMRwc7Gk33GBvFlSugGS2PeIajFfO65gvGk
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Chromium";v="131", "Not_A Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86
Safari/537.36
Accept: application/json, text/plain, */*
X-User-Email: admin@juice-sh.op
Content-Type: application/json
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
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=
omgZd5OtrHkckI9TSCZfOH7tNeukgHJvhyzIDMtvzukwtmZsgkUOVImMOyL6; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMTItMTMgMTQ6NDk6NTcuMzQxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMTItMTMgMTQ6NDk6NTcuMzQxICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTczNDEwMTQzM30.N7xZmuYGiHéi-hOSTnMaevMm4rbULBIPXm9oGCOn-WUEm3Xc62PZRF-z5iZpiM_3OOfhXZTLCpOoOGJYzQ_N-MCXO-hQYib_dk2eaR_g9pvZudv479GQsns-BZhicPO_22twgUTkiMRwc7Gk33GBvFlSugGS2PeIajFfO65gvGk
Connection: keep-alive

{
  "UserId":1,
  "captchaId":0,
  "captcha":"400",
  "comment":"intended to bybass here (***in@juice-sh.op)",
  "rating":5
}
```

```
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/10
Content-Type: application/json; charset=utf-8
Content-Length: 193
ETag: W/"c1-jEhXpI+PEJI98kxPBJK6XxLPzsY"
Vary: Accept-Encoding
Date: Fri, 13 Dec 2024 15:02:50 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "status":"success",
  "data":{
    "id":10,
    "UserId":1,
    "comment":"intended to bybass here (***in@juice-sh.op)",
    "rating":5,
    "updatedAt":"2024-12-13T15:02:50.843Z",
    "createdAt":"2024-12-13T15:02:50.843Z"
  }
}
```

4) I succussed to make that attack for different captcha id I send the same answer all of them is success one .

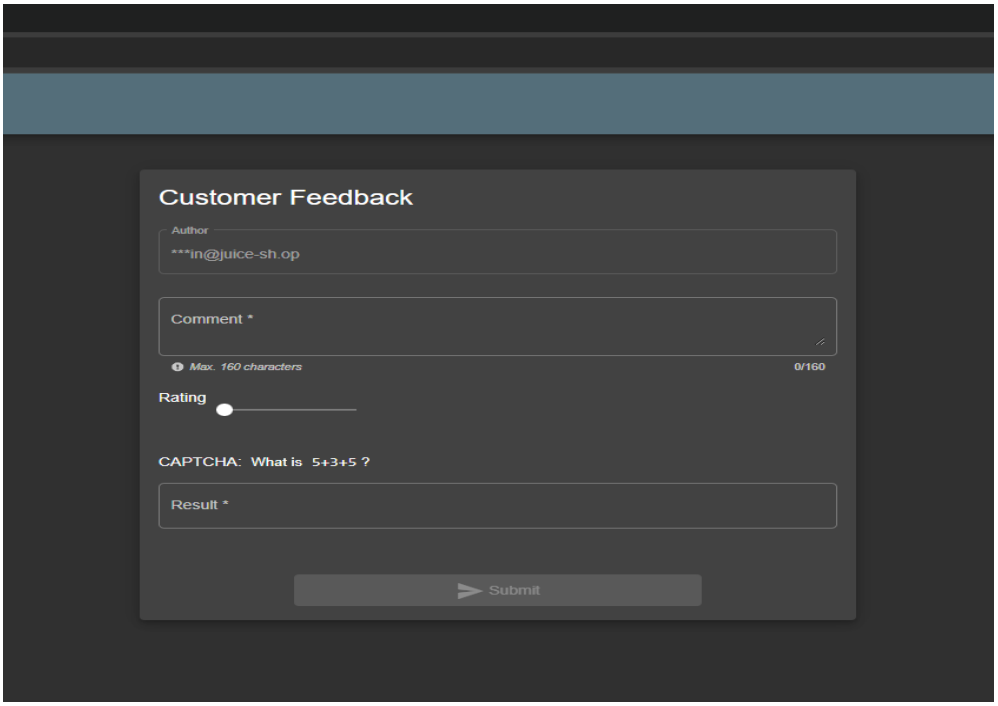

Attack Save
2. Intruder attack of http://localhost:3000
2. Intruder attack of http://localhost:3000
Attack
Save
Results
Positions
Intruder attack results filter: Showing all items
Request
Payload
Status code
Response ...
Error
Timeout
Length
Comment
5 null 201 65 612
6 null 201 58 612
7 null 201 26 612
8 null 201 33 612
9 null 201 46 612
10 null 201 26 612
11 null 201 34 612
12 null 201 26 612
13 null 201 38 612
14 null 201 60 612
15 null 201 50 612
Request
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/23
Content-Type: application/json; charset=utf-8
Content-Length: 193
ETag: W/"c1-ZAC1K59CSEJuFuhiG2AS7Srih7M"
Vary: Accept-Encoding
Date: Fri, 13 Dec 2024 15:04:19 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{"status":"success","data":{"id":23,"UserId":1,"comment":"intended to bybass here (***in@juice-sh.op)","rating":5,"updatedAt":"2024-12-13T15:04:19.112Z","createdAt":"2024-12-13T15:04:19.112Z"}}
Search
0 highlights
Finished
Payloads
Resource pool
Settings
OWASP Juice Shop
localhost:3000/#/contact
OWASP Juice Shop
You successfully solved a challenge: CAPTCHA Bypass (Submit 10 or more customer feedbacks within 20 seconds.)
x
Customer Feedback
Author
***in@juice-sh.op
Comment *
Max. 160 characters
0/160
Rating
CAPTCHA: What is 3-4+3 ?
Result *
Submit

5.17 Broken Access Control (Forged feedback) [BONUS]

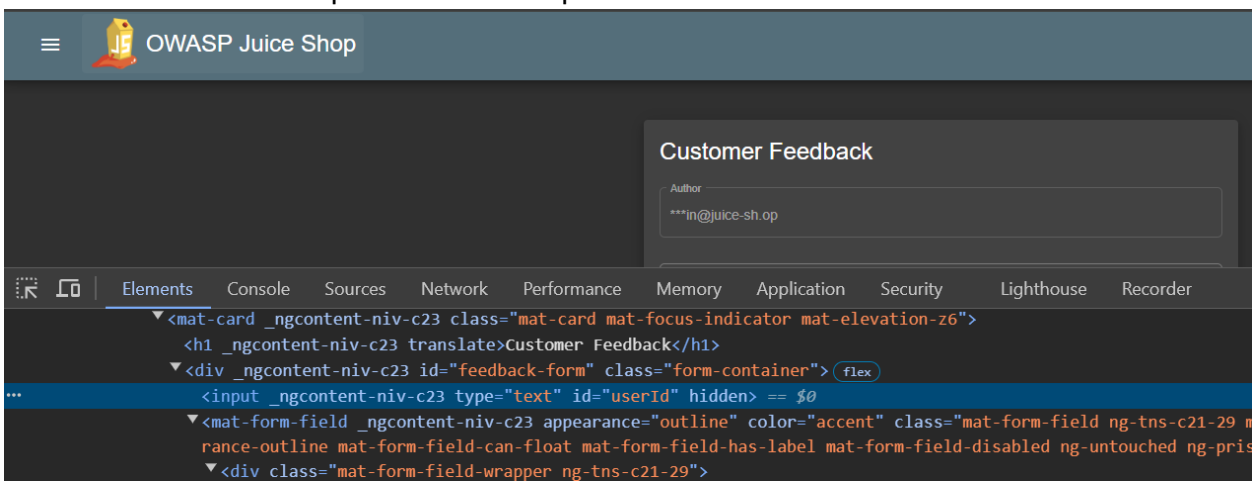
Description:	The application improperly restricts access to the `author` field in the customer feedback functionality. By manipulating the hidden input field in the feedback form, an attacker can modify the `author` ID using developer tools. Further exploitation via Burp Suite allowed changing the `author` ID in the API request, enabling the attacker to forge feedback on behalf of other users.
Impact	Exploitation of this vulnerability can result in: • Forged or fraudulent feedback submissions, damaging the integrity of the feedback system. • Impersonation of other users, leading to potential reputational harm. • Undermining trust in the application by enabling attackers to manipulate user-generated content.
Recommendations	• Ensure sensitive fields like `author` are not editable by the user, even in hidden inputs, and validate all values server-side. • Implement proper access controls to ensure users can only submit feedback on their own behalf. • Use unique, unguessable identifiers for sensitive fields like `author` to prevent easy manipulation. • Regularly audit and test input forms for hidden field manipulation vulnerabilities.
Affected Systems	Backend API endpoints handling feedback submissions
Threat Level	**Medium**

steps to reproduce:

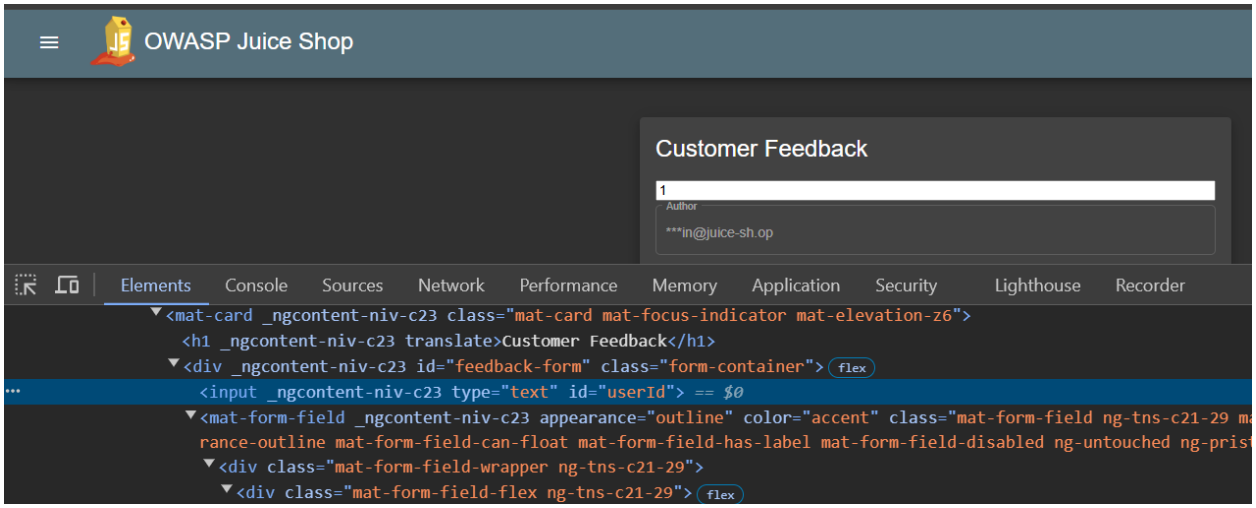
1) In the customer feedback page I found that author is not allowed to write their name instead it is automatically setted to the author email



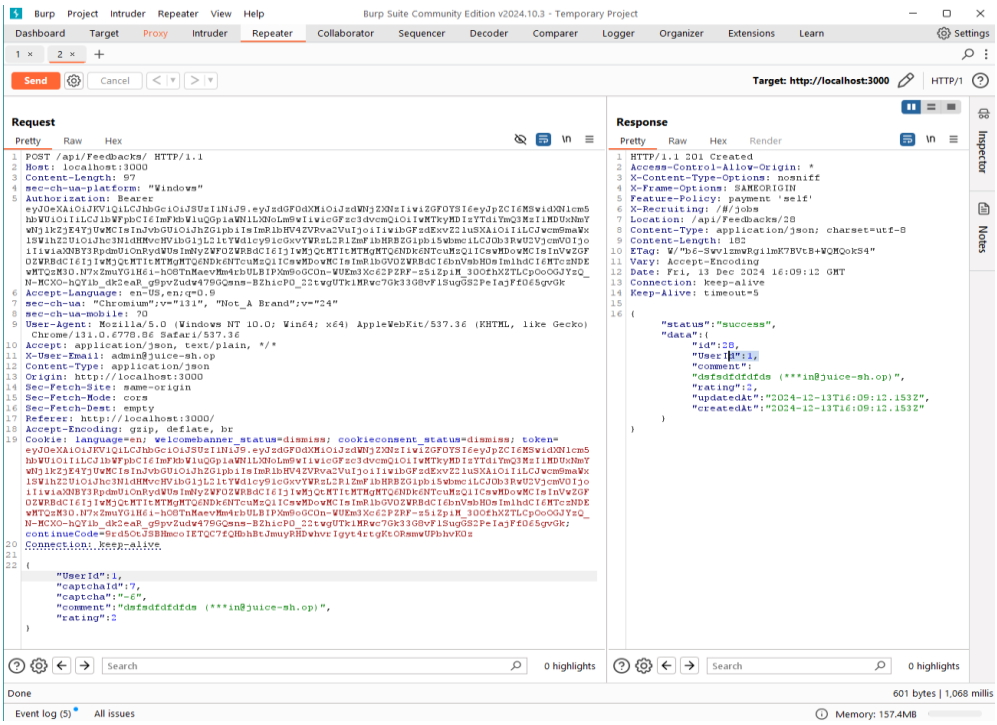
2) I used inspection tool to see that the input field of this section is hidden to disallow me to send input in it with request



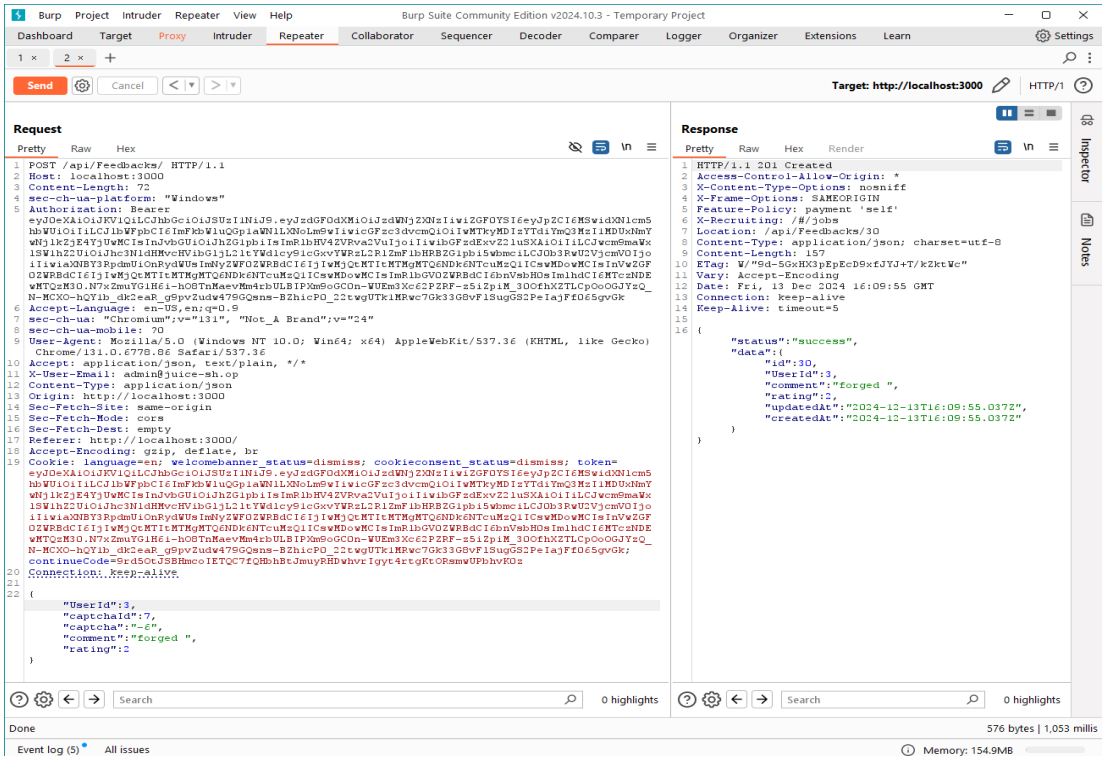
3) Then I removed hidden word to see that it appears in the page with id =1 which is mine



4) I take it in burbsuite to use repeater to test this part first it was successful submission on id = 1



5) Then by changing the id in the request I manipulated to forge a feedback

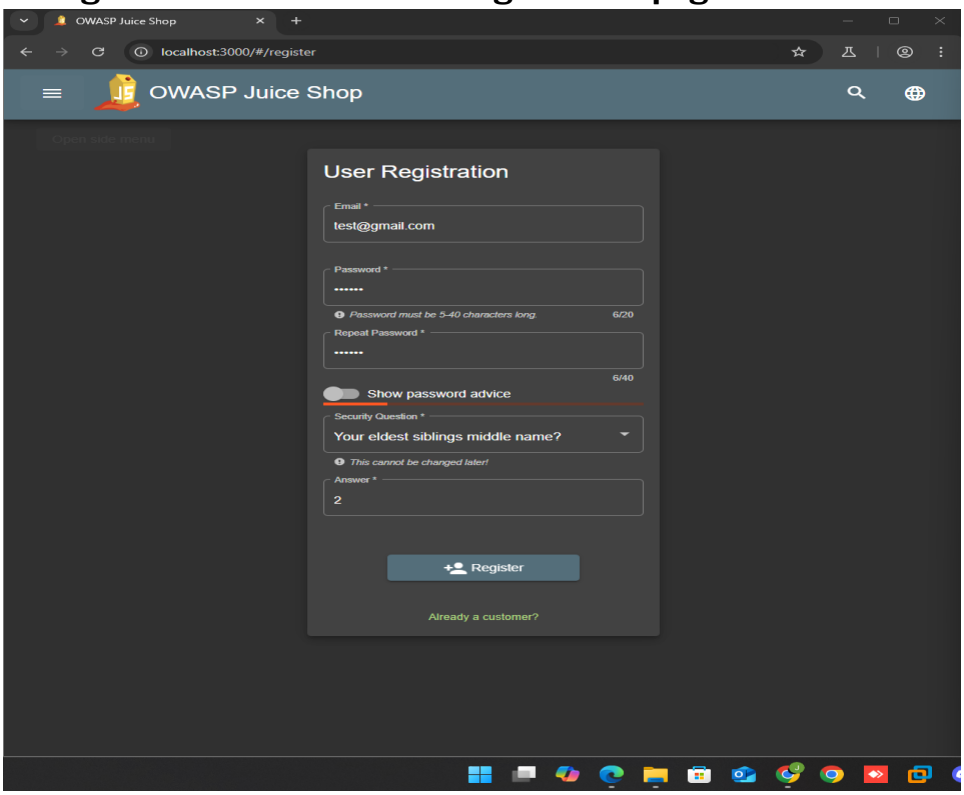

Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2024.10.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Send
Cancel
Target: http://localhost:3000
HTTP/1
Request
Pretty Raw Hex
POST /api/Feedbacks/ HTTP/1.1
Host: localhost:3000
Content-Length: 72
sec-ch-ua-platform: "Windows"
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGppaWNlLXNoLm5wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmY
wNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWx
lSWihZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy9cGxvYWRzL2R1ZmFlbHRBZGipbi5wbmciLCJOb3RwU2VjcmVOIjo
iIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMTItMTMgMTQ6NDk6NTcuMzQiICswMDowMCIsInVwZGF
0ZWRBdCI6IjIwMjQtMTItMTMgMTQ6NDk6NTcuMzQiICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTczNDE
wMTQzM30.N7xZmuYGiH6i-h08TnMaevMm4ibULBIPXm8oGC0n-WUEm3Xc62PZRF-z5i2piM_3OOfbXZTLCpOoGJYzQ_
N-MCXO-hQYib_dk2eaR_g9pvZudw479GQsns-BZhicPO_22twgUTk1MRwc7Gk33G8vF1SugGS2PeIajFfO65gvGk
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Chromium";v="131", "Not_A Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.6778.86 Safari/537.36
Accept: application/json, text/plain, */*
X-User-Email: admin@juice-sh.op
Content-Type: application/json
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
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGppaWNlLXNoLm5wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmY
wNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWx
lSWihZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy9cGxvYWRzL2R1ZmFlbHRBZGipbi5wbmciLCJOb3RwU2VjcmVOIjo
iIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMTItMTMgMTQ6NDk6NTcuMzQiICswMDowMCIsInVwZGF
0ZWRBdCI6IjIwMjQtMTItMTMgMTQ6NDk6NTcuMzQiICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTczNDE
wMTQzM30.N7xZmuYGiH6i-h08TnMaevMm4ibULBIPXm8oGC0n-WUEm3Xc62PZRF-z5i2piM_3OOfbXZTLCpOoGJYzQ_
N-MCXO-hQYib_dk2eaR_g9pvZudw479GQsns-BZhicPO_22twgUTk1MRwc7Gk33G8vF1SugGS2PeIajFfO65gvGk;
continueCode=9rd5OtJSBHmcoIETQC7fQHbhBtJmuyPHDwhvrIgyt4rtgKtORsmwUPbhvKOz
Connection: keep-alive
{
"UserId":3,
"captchaId":7,
"captcha":"-6",
"comment":"forged ",
"rating":2
}
Search
0 highlights
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/30
Content-Type: application/json; charset=utf-8
Content-Length: 157
ETag: W/"9d-5GxHX3pEpEcD9xfJYJ+T/kZktWc"
Vary: Accept-Encoding
Date: Fri, 13 Dec 2024 16:09:55 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
"status":"success",
"data":{
"id":30,
"UserId":3,
"comment":"forged ",
"rating":2,
"updatedAt":"2024-12-13T16:09:55.037Z",
"createdAt":"2024-12-13T16:09:55.037Z"
}
}
Search
0 highlights
Inspector
Notes
Done
576 bytes | 1,053 millis
Event log (5)
All issues
Memory: 154.9MB

5.18 Parameter Based Access control [BONUS]

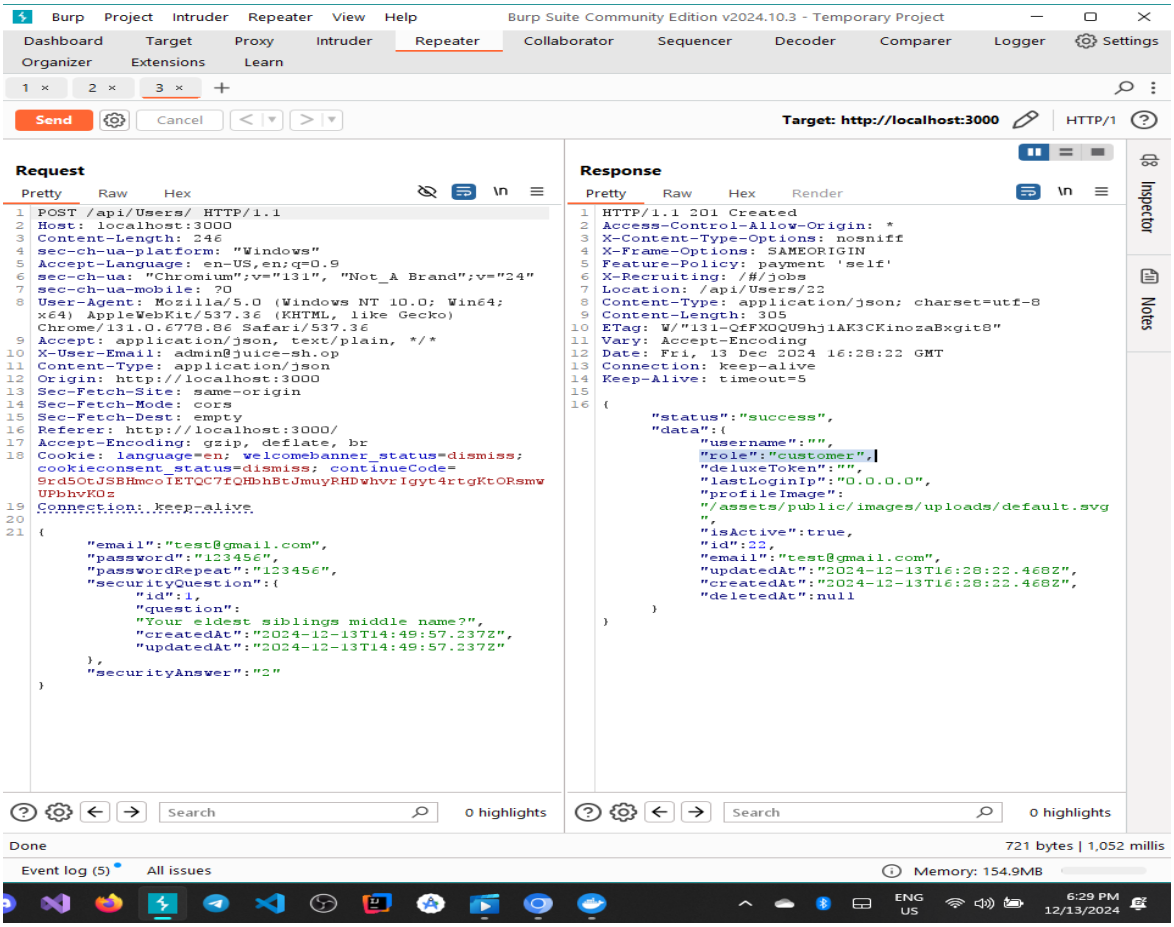
Description:	The system is vulnerable to parameter-based access control issues. A normal user's role is defined by a parameter (e.g., `role='customer'`) in the response. By intercepting and modifying this parameter (for example, changing "customer" to "admin"), an attacker can escalate their privileges and access restricted areas, such as the administration dashboard, without proper authorization.
Impact	• Unauthorized privilege escalation, allowing regular users to access administrative functions. • Potential data manipulation or deletion by unauthorized users with administrative access. • Compromise of sensitive information and system settings due to improper access control enforcement
Recommendations	• Implement proper server-side access control checks for each sensitive operation, ensuring that users cannot modify their roles or access permissions via parameters. • Use session-based or token-based authorization to track user roles and permissions, rather than relying solely on parameters in requests or responses. • Apply the principle of least privilege, ensuring users only have the permissions necessary for their role. • Regularly audit and test for privilege escalation vulnerabilities by attempting unauthorized access to restricted endpoints. • Implement role-based access control (RBAC) at both the server-side and client-side to validate user roles during each request.
Affected Systems	User management and dashboard functionality that are not properly protected by access control check
Threat Level	**High**

steps to reproduce:

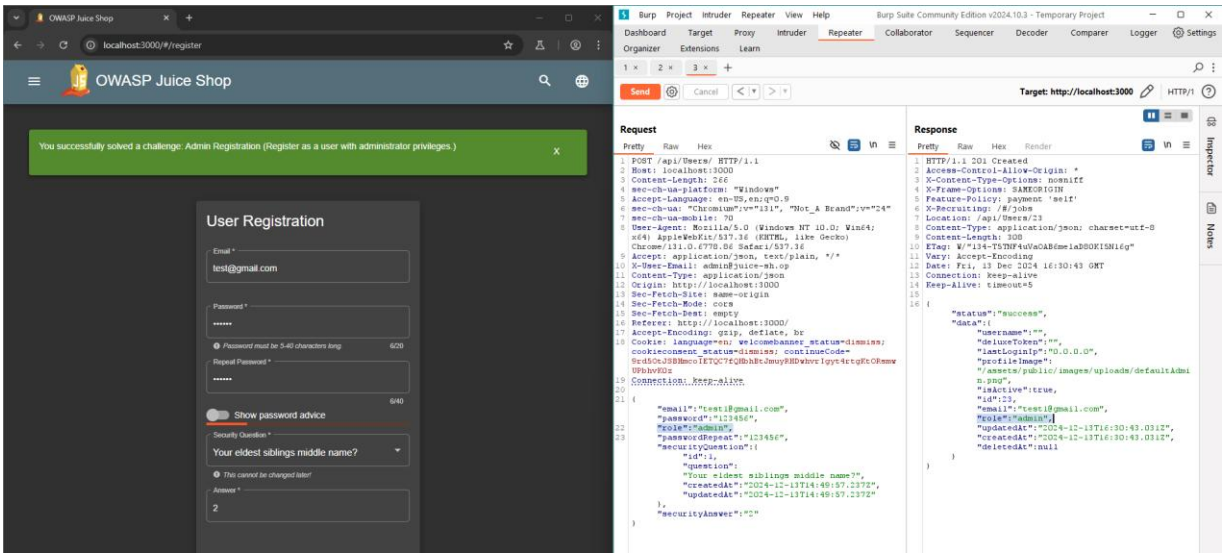
1) **First register normal user in the registration page**



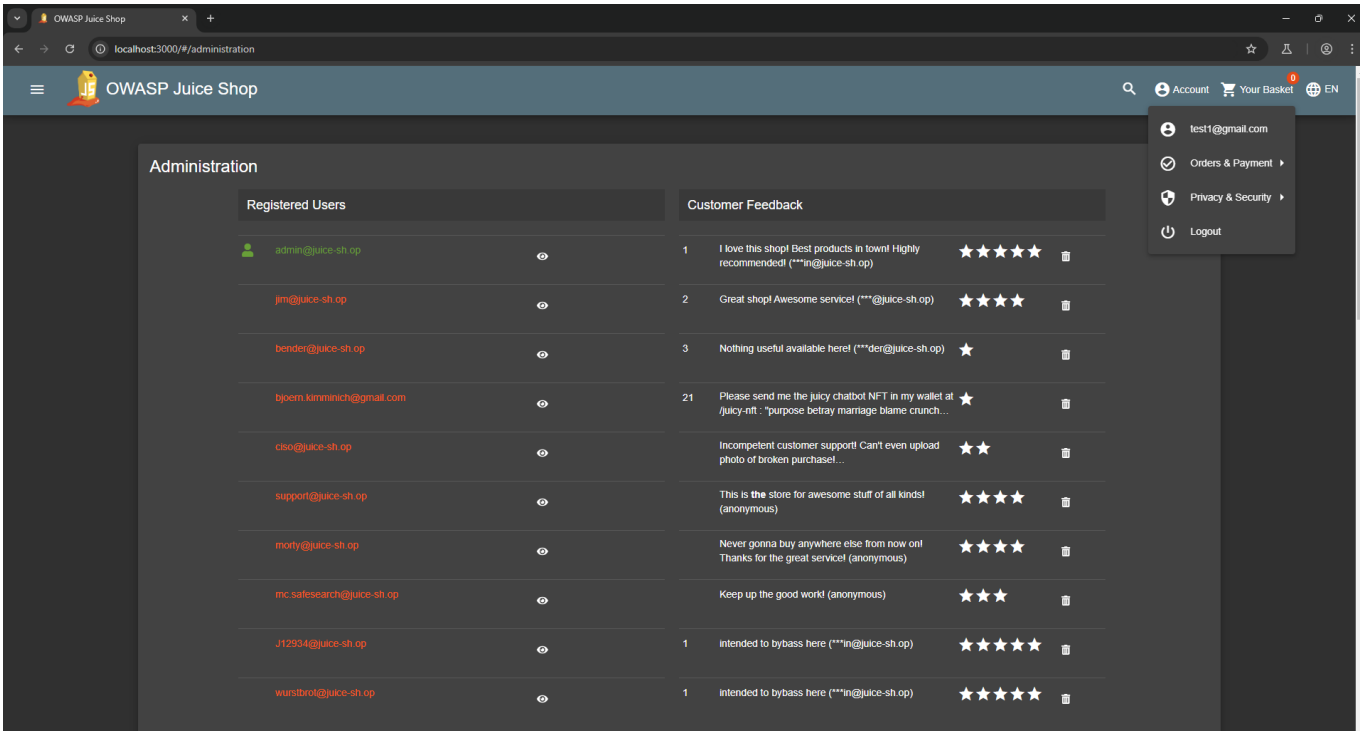
2) **Intercept request and send it to the repeater**
3) **Send it and i noticed that in the response it has attribute “role =’customer’**

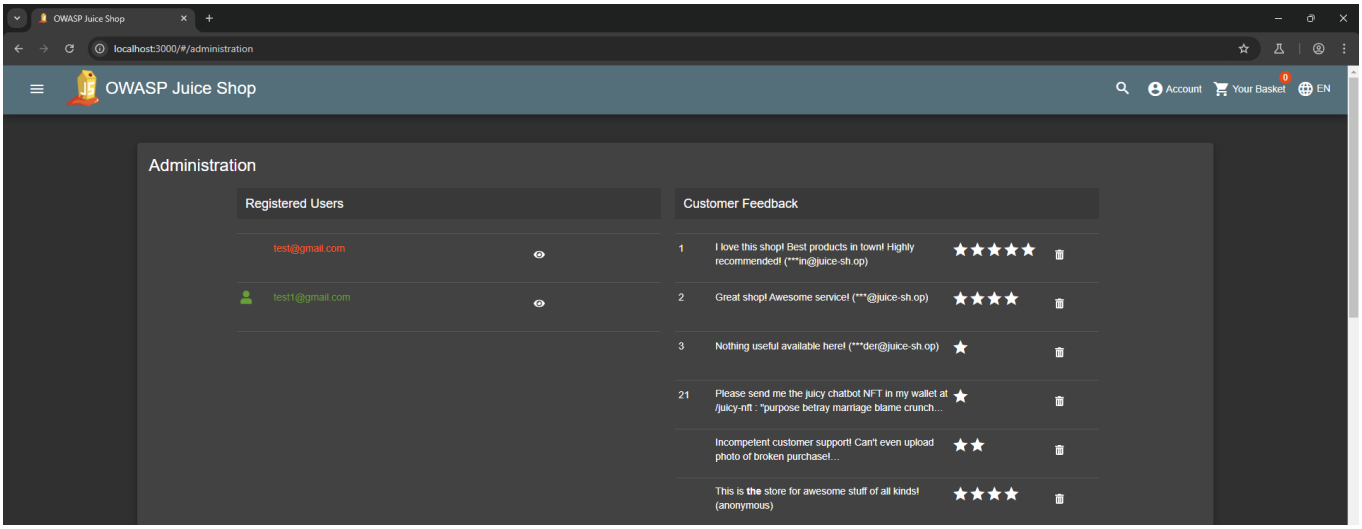


4) Then try send it with request and observe that the response 200ok



5) In the account I logged in with it getting to administration dashboard to check that it is really admin user now test1@gmail.com


OWASP Juice Shop
localhost:3000/#/administration
OWASP Juice Shop
Account
Your Basket
EN
test1@gmail.com
Orders & Payment
Privacy & Security
Logout
Administration
Registered Users
admin@juice-sh.op
jim@juice-sh.op
bender@juice-sh.op
bjoern.kimminich@gmail.com
ciso@juice-sh.op
support@juice-sh.op
morty@juice-sh.op
mc.safesearch@juice-sh.op
J12934@juice-sh.op
wurstbrot@juice-sh.op
Customer Feedback
1 I love this shop! Best products in town! Highly recommended! (***in@juice-sh.op)
2 Great shop! Awesome service! (***@juice-sh.op)
3 Nothing useful available here! (***der@juice-sh.op)
21 Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriage blame crunch...
Incompetent customer support! Can't even upload photo of broken purchase!...
This is the store for awesome stuff of all kinds! (anonymous)
Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous)
Keep up the good work! (anonymous)
1 intended to bybass here (***in@juice-sh.op)
1 intended to bybass here (***in@juice-sh.op)

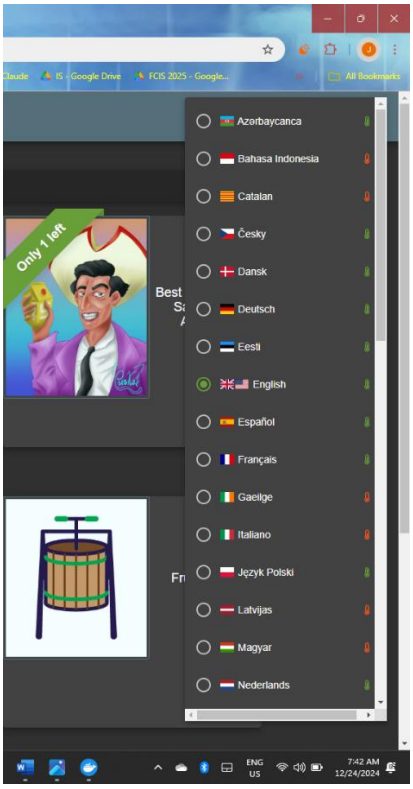

OWASP Juice Shop
localhost:3000/#/administration
OWASP Juice Shop
Account
Your Basket
EN
Administration
Registered Users
test@gmail.com
test1@gmail.com
Customer Feedback
1 I love this shop! Best products in town! Highly recommended! (***in@juice-sh.op)
2 Great shop! Awesome service! (***@juice-sh.op)
3 Nothing useful available here! (***der@juice-sh.op)
21 Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriage blame crunch...
Incompetent customer support! Can't even upload photo of broken purchase!...
This is the store for awesome stuff of all kinds! (anonymous)

5.19 Broken Anti Automation (add Extra lang) [BONUS]

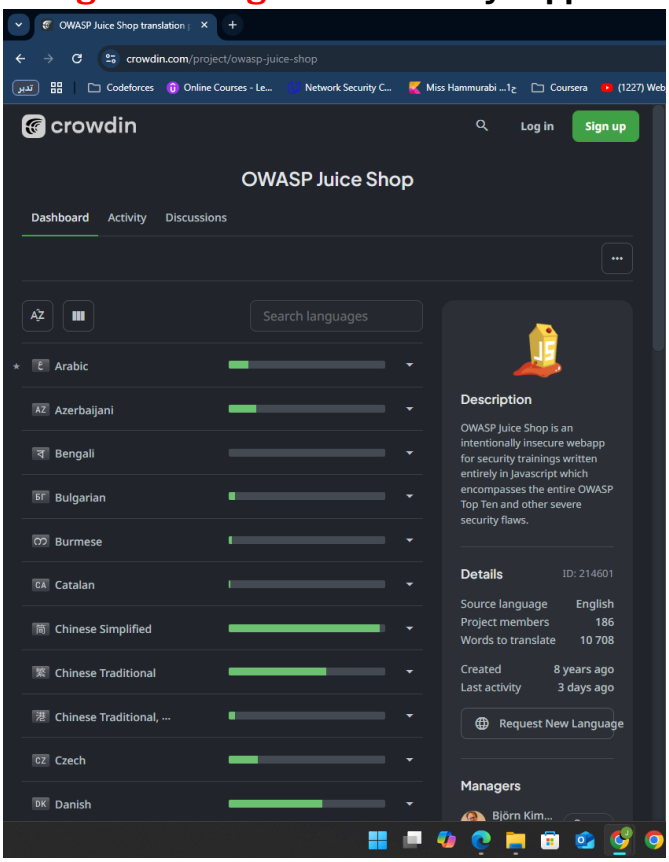
Description:	The system allows users to change the language of the application by modifying a language parameter in the request. By using an inspection tool (e.g., Burp Suite), the attacker can intercept the language change request and modify the parameter value to unsupported languages like Bengali, Hungarian, or Klingon. Despite these languages not being fully supported, the system accepts the changes, and the response is successful (200 OK), allowing the attacker to manipulate the language setting. While the vulnerability does not result in immediate security risks, it could lead to user confusion or exposure of incomplete/insecure language settings, impacting user experience and potentially revealing sensitive data.
Impact	Exploitation of this vulnerability can result in: • Display of unsupported or incomplete language translations, leading to confusion for legitimate users. • Potential exposure of application functionality or data in a language not intended for the user, affecting the application's security posture. • Unauthorized access to areas of the application that may be presented differently depending on the language selected.
Recommendations	• Validate language selection parameters to ensure only supported languages are accepted. • Provide fallback mechanisms to default to a safe, fully supported language if an unsupported language is requested. • Ensure that the language setting is tied to user authentication and preferences, preventing unauthorized users from altering it. • Regularly audit language-related features to ensure proper validation of inputs and to prevent manipulation by unauthorized users. • Test the application for proper handling of language parameters during security assessments.
Affected Systems	Any system with a language selection mechanism that relies on user-controlled parameters without proper validation.
Threat Level	**Medium**

steps to reproduce:

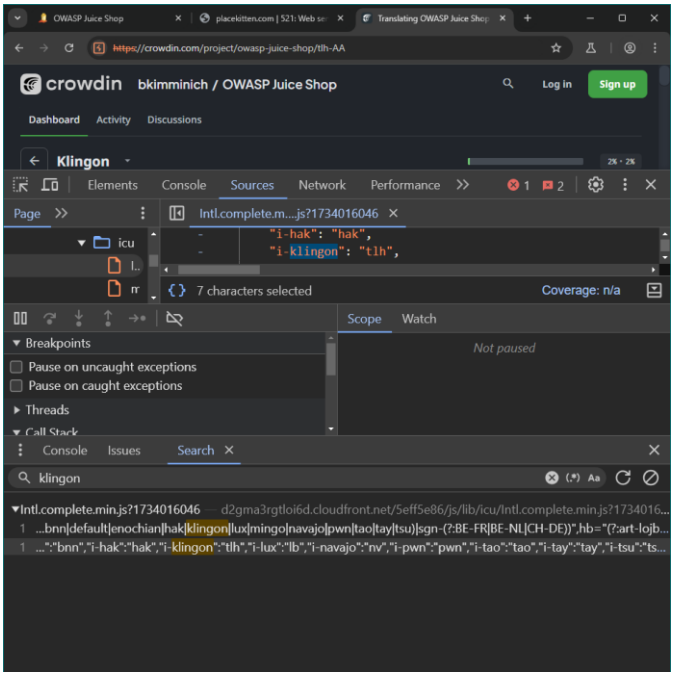
1) **First I found that owasp supports some languages like:**



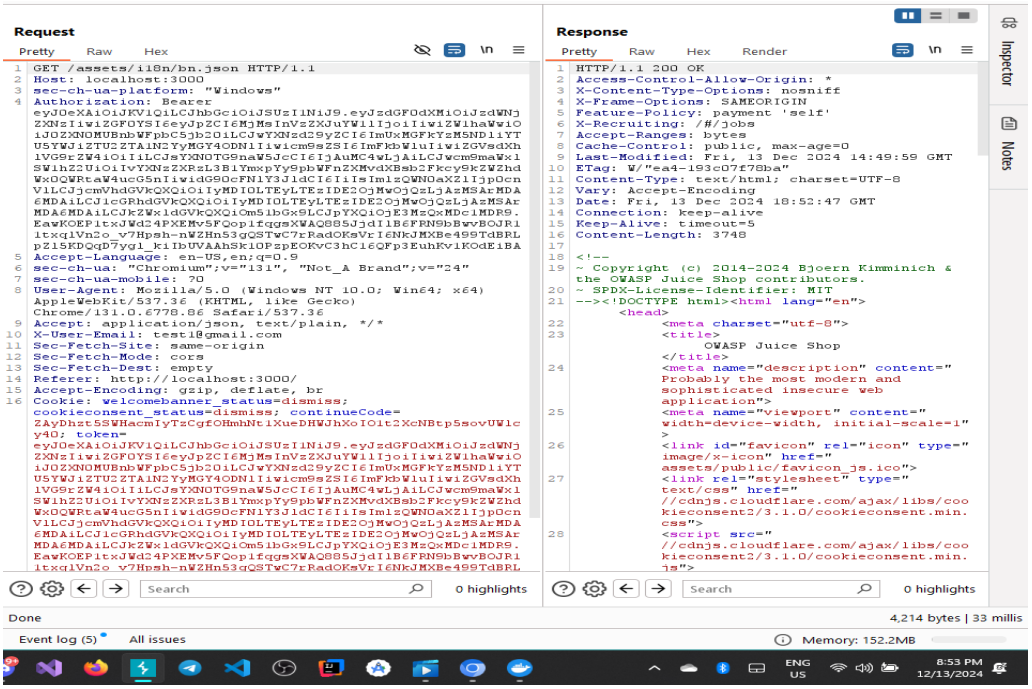
2) **So I intended to search on owasp juice shop languages I saw that Bengali /Hungarion/Klingon is not totally supported yet**



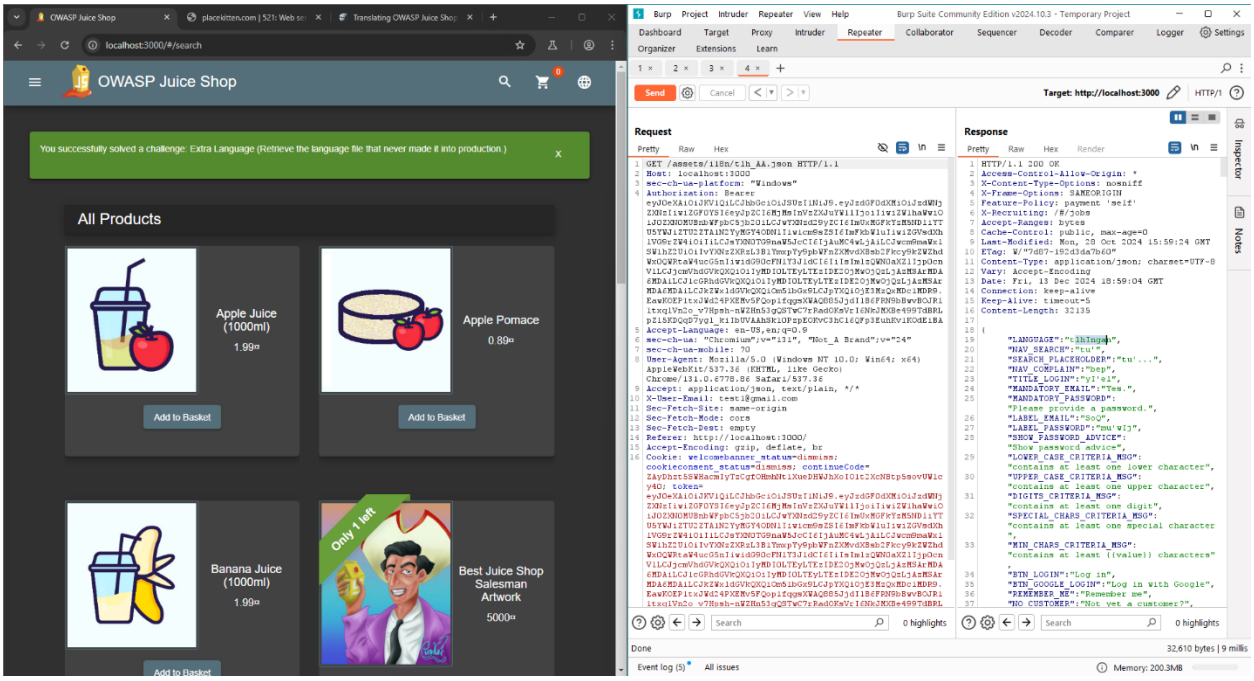
3) **I choose to use inspection tool on these languages and get its code**



4) **I take the request of the change language in my burb and in the repeater I changed the code in the request before send it**



5) **I managed to change to it as the response is 200ok**

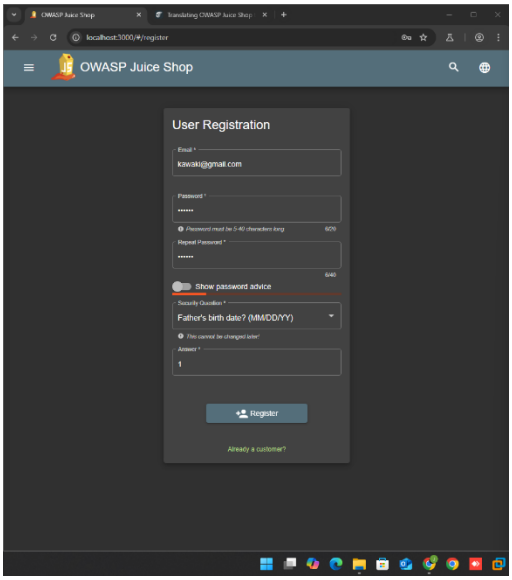

OWASP Juice Shop
localhost:3000/#/search
You successfully solved a challenge: Extra Language (Retrieve the language file that never made it into production)
All Products
Apple Juice (1000ml)
1.99¤
Add to Basket
Apple Pomace
0.89¤
Add to Basket
Banana Juice (1000ml)
1.99¤
Only 1 left
Best Juice Shop Salesman Artwork
5000¤
Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2024.10.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Settings
Organizer Extensions Learn
Send Cancel
Target: http://localhost:3000
HTTP/1
Request
Pretty Raw Hex
GET /assets/i18n/tlh_AA.json HTTP/1.1
Host: localhost:3000
sec-ch-ua-platform: "Windows"
Authorization: Bearer
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Chromium";v="131", "Not_A Brand";v="24"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.6778.86 Safari/537.36
Accept: application/json, text/plain, */*
X-User-Email: test1@gmail.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=
token=
Response
Pretty Raw Hex Render
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
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Mon, 28 Oct 2024 15:56:24 GMT
ETag: W/"7d87-192d3da7b60"
Content-Type: application/json; charset=UTF-8
Vary: Accept-Encoding
Date: Fri, 13 Dec 2024 18:59:04 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 32135
{
"LANGUAGE":"
"NAV_SEARCH":"
"SEARCH_PLACEHOLDER":"
"NAV_COMPLAIN":"
"TITLE_LOGIN":"
"MANDATORY_EMAIL":"
"MANDATORY_PASSWORD":
"Please provide a password.",
"LABEL_EMAIL":"
"LABEL_PASSWORD":"
"SHOW_PASSWORD_ADVICE":
"Show password advice",
"LOWER_CASE_CRITERIA_MSG":
"contains at least one lower character",
"UPPER_CASE_CRITERIA_MSG":
"contains at least one upper character",
"DIGITS_CRITERIA_MSG":
"contains at least one digit",
"SPECIAL_CHARS_CRITERIA_MSG":
"contains at least one special character
",
"MIN_CHARS_CRITERIA_MSG":
"contains at least {{value}} characters"
,
"BTN_LOGIN":"Log in",
"BTN_GOOGLE_LOGIN":"Log in with Google",
"REMEMBER_ME":"Remember me",
"NO_CUSTOMER":"Not yet a customer?",
Inspector
Notes
Done
32,610 bytes | 9 millis
Event log (5)
All issues
Memory: 200.3MB

5.20 Improper input validation (Empty-user-reg) [BONUS]

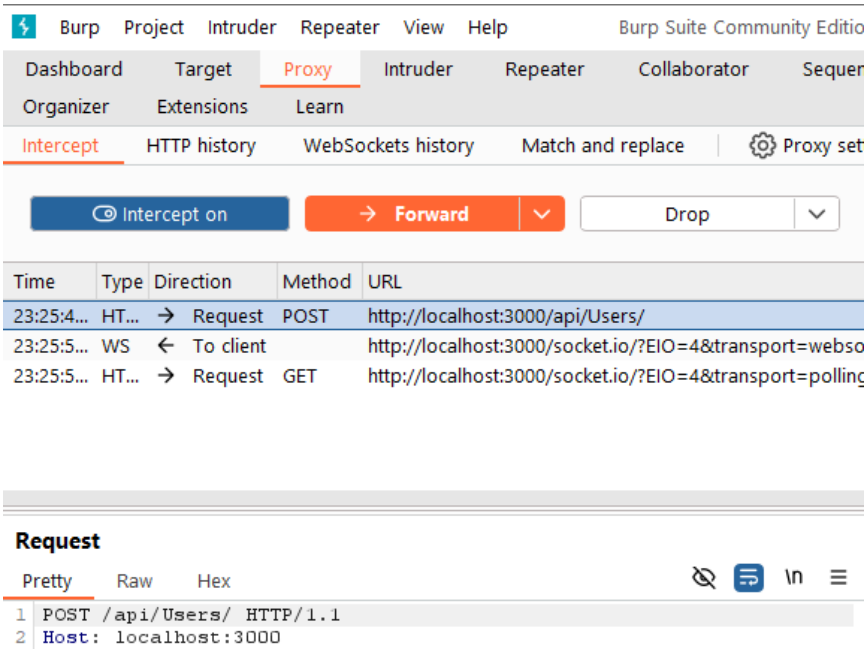
Description:	The system fails to properly validate input during user registration. By intercepting the registration request and sending an empty payload, an attacker can bypass validation and register an empty or incomplete user account. Despite the server returning a `400 Bad Request` response, the user is still successfully registered in the system. This indicates improper input validation or failure to sanitize input correctly. The vulnerability allows unauthorized or malformed accounts to be created in the system, potentially leading to security breaches, data manipulation, or abuse of system functionality.
Impact	Exploitation of this vulnerability can result in: • The ability to register empty or incomplete user accounts, potentially creating orphaned accounts that could be exploited later. • Unauthorized access to certain areas of the system if the empty user account is somehow treated as valid. • Compromise of data integrity by allowing incomplete or invalid accounts to exist in the system without proper checks.
Recommendations	• Implement server-side validation to ensure that all required fields are populated with valid data before processing the registration request. • Reject empty or malformed user registration data with clear error messages explaining the validation issues. • Ensure that user registration functionality is not bypassed by modifying requests directly (e.g., using Burp Suite). • Audit and test the registration process regularly to ensure that all input fields are correctly validated and sanitized. • Implement logging and monitoring to detect suspicious account creation attempts, such as registration of empty or malformed accounts.
Affected Systems	User registration endpoints (/api/users/) that lack proper input validation.
Threat Level	**High**

steps to reproduce:

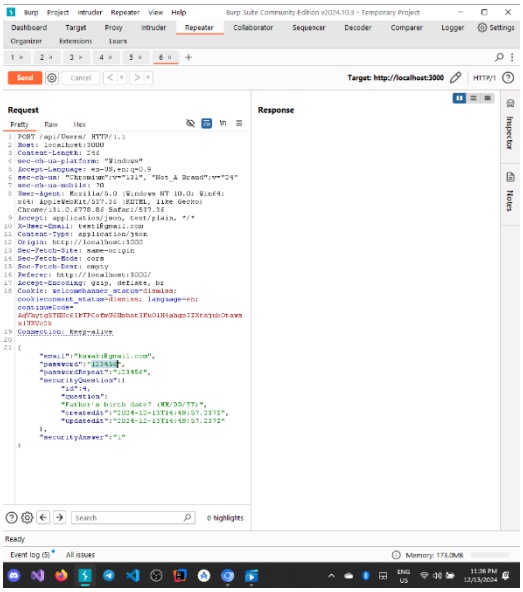
1) **I intended to test if I could make a fake account and test validation on registration**



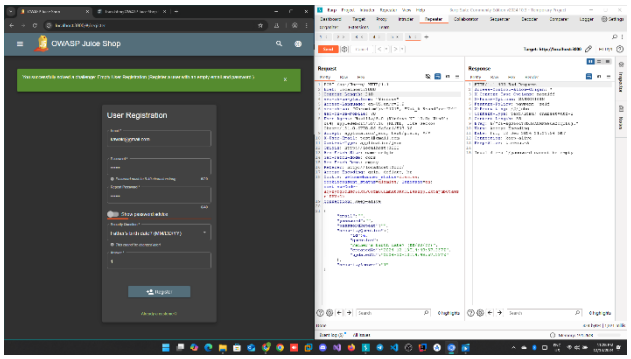
2) **So I intercepted the request /api/users/ and send to repeater**



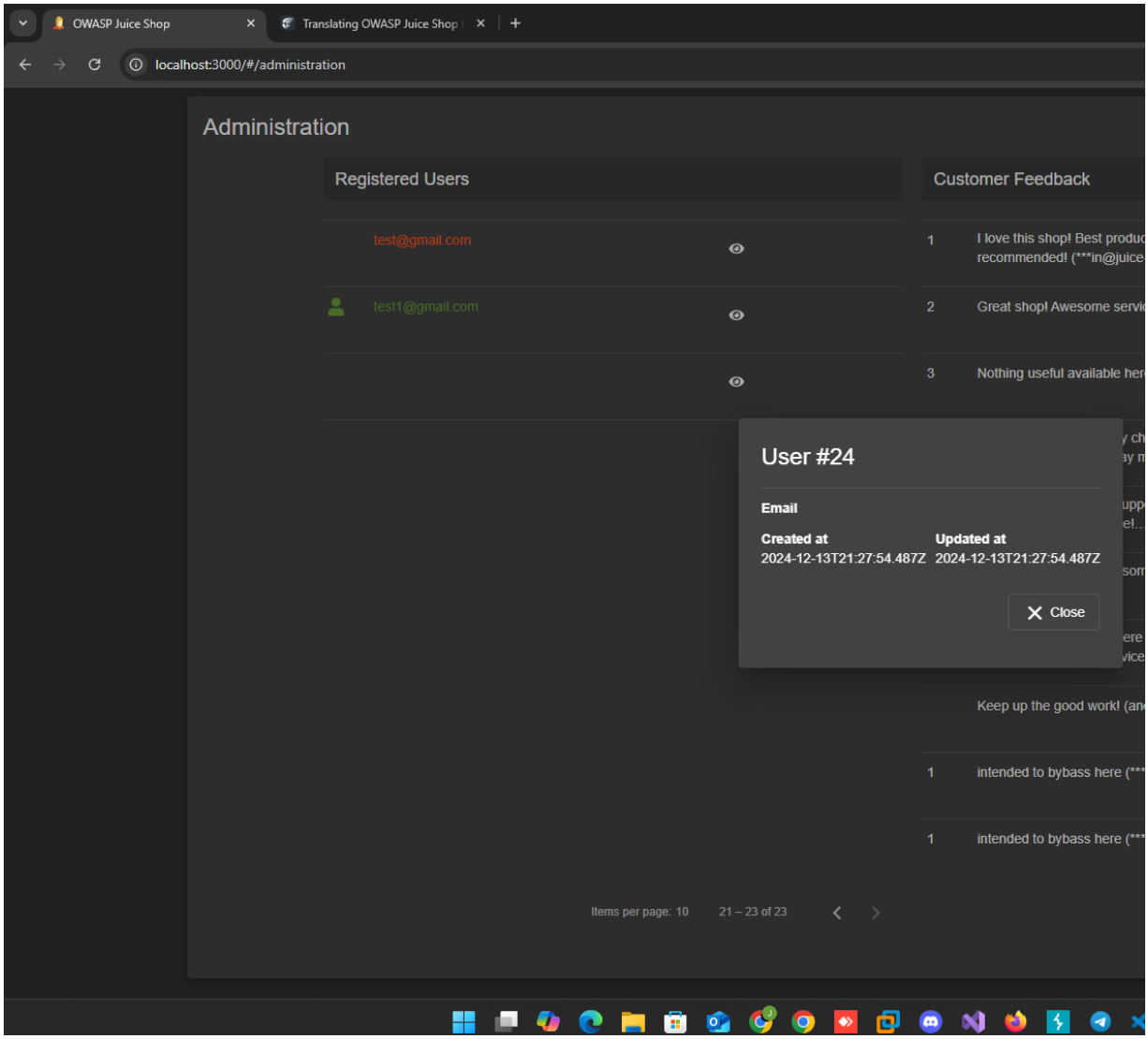
3) **I intended to delete data in the body of the registration send empty one.**



4) **It responds with 400 bad request response**



5) **But I have already previous access to the admin page when I test the users on the system I saw that the empty user is really registered successfully on the system**



5.21 [BONUS]

steps to reproduce:

Description:	
Impact	
Recommendations	
Affected Systems	
Threat Level	

5.22 [BONUS]

Description:	
Impact	
Recommendations	
Affected Systems	
Threat Level	

steps to reproduce:

5.23 [BONUS]

Description:	
Impact	
Recommendations	
Affected Systems	
Threat Level	

steps to reproduce:

5.24 [BONUS]

Description:	
Impact	
Recommendations	
Affected Systems	
Threat Level	

steps to reproduce:

5.25 [BONUS]

Description:	
Impact	
Recommendations	
Affected Systems	
Threat Level	

steps to reproduce: