SECURITY ASSESSMENT

<< OWASP Juice Shop>>

Submitted to: << Sprints >>
Security Analyst: << Team4 >>

Date of Testing: << 10/10/2024 >>
Date of Report Delivery: << 14/10/2024 >>

Table of Contents

Contents

Security Engagement Summary

Engagement Overview

<<

The purpose of this vulnerability assessment was to perform a comprehensive security evaluation of the Juice Shop web application. Commissioned by the Dojuicer (Sprints) Development Team, the goal of this engagement was to identify vulnerabilities that could be exploited by attackers to compromise the application's data integrity, confidentiality, or availability.

The report outlines the vulnerabilities found, their severity, potential business impacts, and detailed recommendations for remediation. It aims to guide the development team in implementing necessary controls to mitigate risks and strengthen the overall security posture of the Juice Shop application.

Engagement Objectives:-

- Identify and prioritize security vulnerabilities within the Juice Shop application.
- Evaluate the impact and likelihood of some vulnerability on the system's security.
- Develop actionable remediation strategies for the identified issues.

>>

Scope

<<

The assessment was scoped to focus exclusively on the Juice Shop web application components, ensuring that the analysis was both thorough and relevant to the specific environment in question.

In-Scope Components:

- User Authentication & Authorization: Focus on validating credentials, password policies, and session management.
- Data Input Fields: Evaluation of all user input fields for injection vulnerabilities, including search bars and form entries.
- Administrative Interface: Analysis of access control and privilege management for admin-level functions.
- Client-Side and Server-Side Validation: Review of the application's data handling on both the client and server ends.

Out-of-Scope Components:

- Underlying Infrastructure: Server configurations, network topology, and third-party service integrations were excluded.
- Internal Corporate Network Security: This assessment was limited to the public-facing aspects of the web application only.

>>

Executive Risk Analysis

<<

Several critical vulnerabilities were discovered during the assessment, each representing a significant security risk. The most concerning issues involve SQL Injection that can result in full admin account compromise, DOM XSS attacks, and improper file exposure that reveals confidential information.

Vulnerabilities Identified:

1. **Data Breach Risk: The presence of SQL Injection and IDOR vulnerabilities could lead to unauthorized access to sensitive customer data, including Personally Identifiable Information (PII) and confidential business information. This may result in legal repercussions, compliance violations, and a loss of customer trust.**

2. **Reputational Damage: Exploitation of Cross-Site Scripting (XSS) could allow attackers to compromise user sessions or deface the website, severely damaging the company's reputation and user confidence in the platform.**

3. **Operational Disruption: Vulnerabilities like Confidential File Exposure can provide attackers with access to critical files, including configuration data, leading to potential disruptions in service, unauthorized system modifications, and exposure of intellectual property.**

4. **Compliance Violations: Unauthorized data access may result in non-compliance with data protection regulations such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA), potentially incurring substantial financial penalties.4. Confidential File Exposure (High): Sensitive files, such as configuration files and backup data, are exposed through an unprotected FTP directory.**

Overall Risk Level: High

These vulnerabilities could result in unauthorized access, data theft, and full control over the application if left unpatched, leading to severe reputational and financial damage.

>>

Executive Recommendation

<<

To effectively mitigate the vulnerabilities identified, the following remediation steps are recommended:

1. **SQL Injection Mitigation:**
   - Parameterized Queries: Implement prepared statements and parameterized queries in the login functionality to prevent SQL injection.
   - Sanitize Inputs: Enforce strict input validation and sanitization across all fields, particularly those interacting with the database.

2. **XSS Mitigation (DOM-based, Persistent and Reflected):**
   - Encode Output: Use secure encoding libraries to sanitize all data before rendering in the browser.
   - Content Security Policy (CSP): Deploy a robust CSP to limit the execution of scripts and control data sources.
   - Continuous Code Review: Implement a formal process for reviewing JavaScript and client-side code for vulnerabilities.

3. **IDOR Mitigation:**
   - Access Control Validation: Ensure all sensitive actions are tied to proper session management and authenticated users.
   - Role-Based Access Control (RBAC): Implement RBAC to prevent unauthorized users from accessing or modifying resources.

4. **File Exposure Mitigation:**
   - Secure File Access: Restrict access to sensitive files, such as configuration files and backups, by placing them in non-public directories.
   - Audit and Encrypt Files: Regularly audit exposed directories and ensure sensitive data is encrypted.

Long-Term Security Enhancements:

- Deploy Web Application Firewalls (WAF): To filter and monitor incoming traffic for signs of SQLi, XSS, and other injection attacks.
- Regular Penetration Testing: Schedule quarterly vulnerability assessments to stay ahead of emerging threats.
- Security Awareness Training: Educate developers on secure coding practices to minimize future vulnerabilities.

>>

Significant Vulnerability Summary

<<

The vulnerabilities discovered are listed below in descending order of risk, highlighting their severity and potential impact on the Juice Shop application.

>>

Vulnerability Name	Severity	CVSS Score	Description
SQL Injection in Login page	Critical	9.8	Allows attackers to bypass authentication.
Insecure Direct Object Reference (IDOR)	High	7.5	Attacker can modify the UserId and submitting feedback to another user
Exposure in Confidential Files.	High	7.2	Sensitive files such as package.json.bak and acquisition.md are accessible through unprotected FTP path.
DOM Cross-Site Scripting (XSS) in Search Bar	Medium	6.1	Untrusted data is written directly into the HTML document by the client-side JavaScript without proper sanitization, leading to the execution of malicious scripts within the user's session

Significant Vulnerability Detail

>>

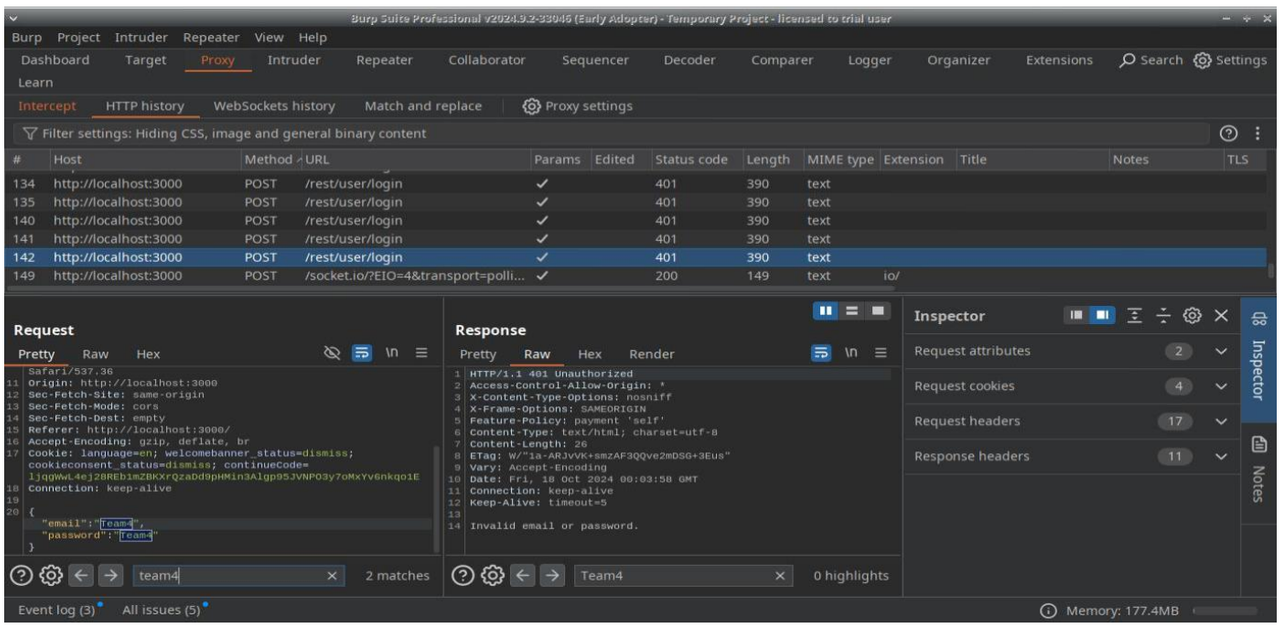
SQL Injection in Login Page – Admin Account Compromise

- CWE Reference: CWE-89
- CVSS Score: 9.8 (Critical)
- Severity: Critical

  This vulnerability's critical rating is due to its ease of exploitation and potential to cause severe damage to the data integrity and overall security of the application.

- Description:

  The login form allows for SQL Injection through the email field, where attackers can manipulate the SQL query to bypass authentication and gain admin privileges.



- Proof-of-Concept (PoC):

  Intercept Request: Using Burp Suite, intercept the login request and modify the email field with the following payload:

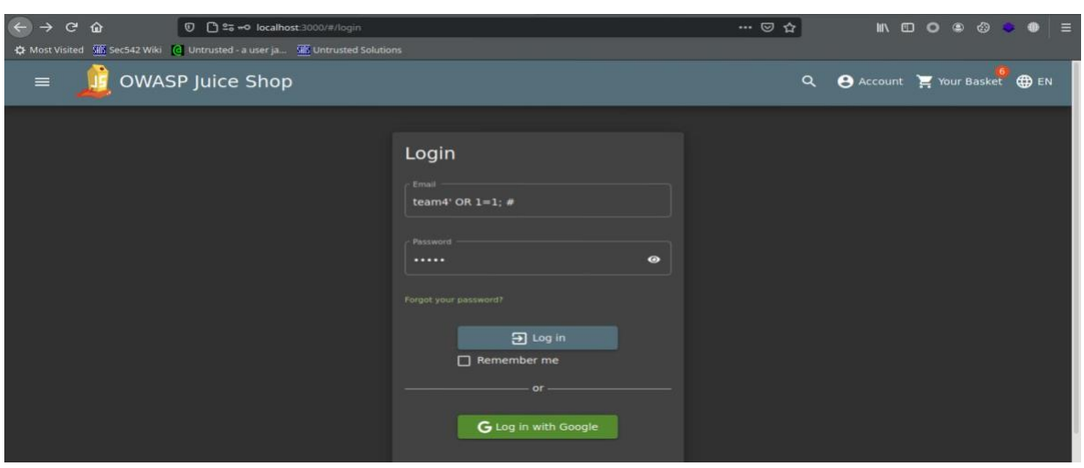
  - team4' OR 1=1; --

  Full SQL query:

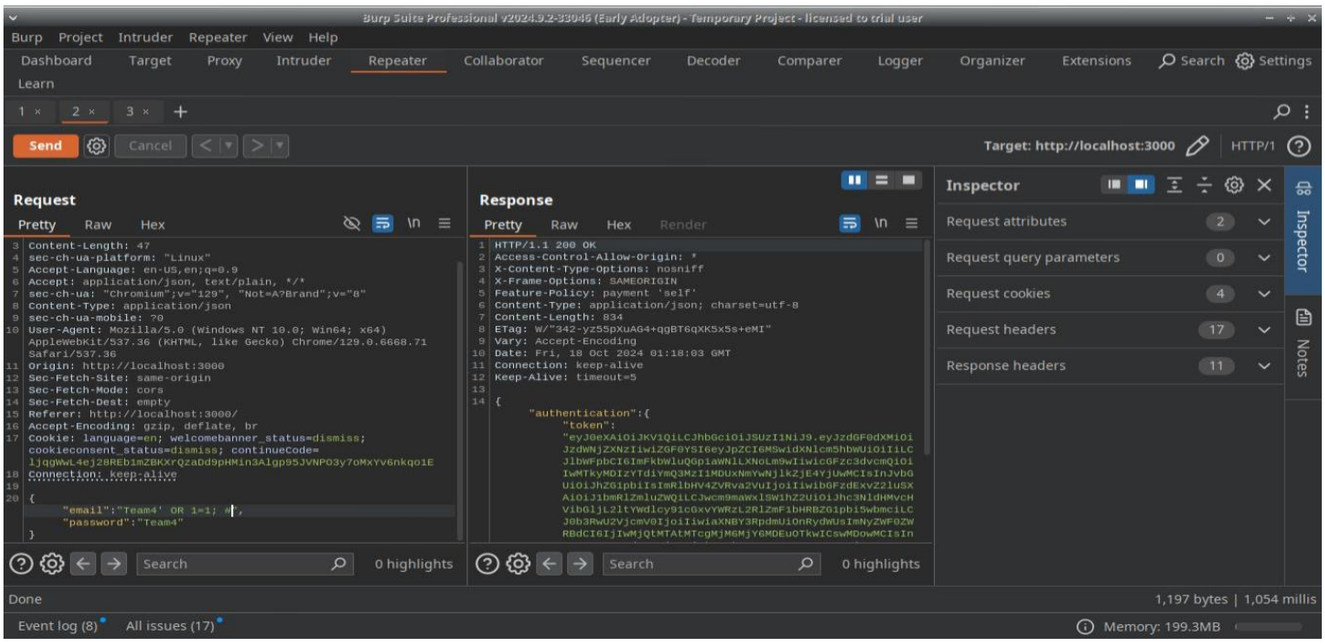
  - SELECT admin FROM users WHERE username = 'team4' AND password = 'te4am';

  After adding The Payload:

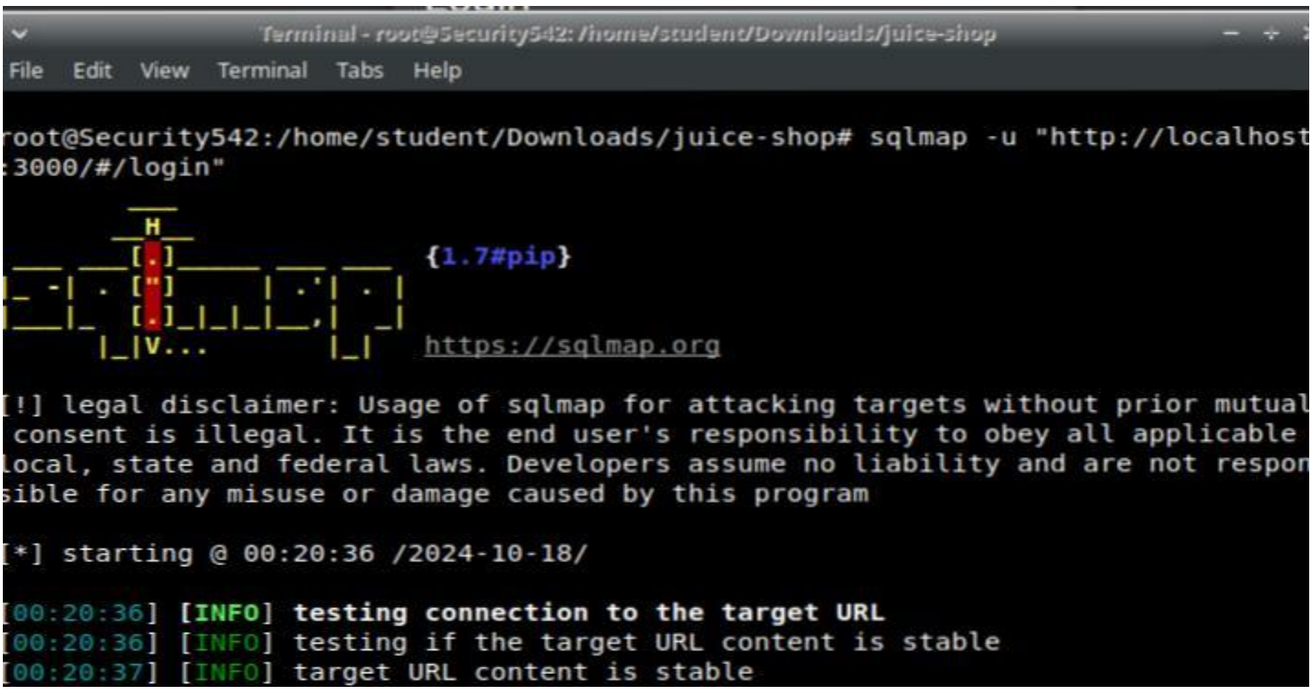
  - SELECT admin FROM users WHERE username = 'team4' OR 1=1; --' AND password = 'te4am';



2. Exploit SQL Injection: Submit the modified request to the server, resulting in bypassed authentication and admin access.



3. Automate with SQLMap: Execute SQLMap to automate SQL Injection and retrieve admin credentials.



Likelihood of Exploit

- Attack Vector: Network-based
- Privileges Required: None (attacker acts as an unauthenticated user)
- Likelihood: Very High – SQL Injection attacks are well-documented and commonly exploited using automated tools available to low-skilled attackers.

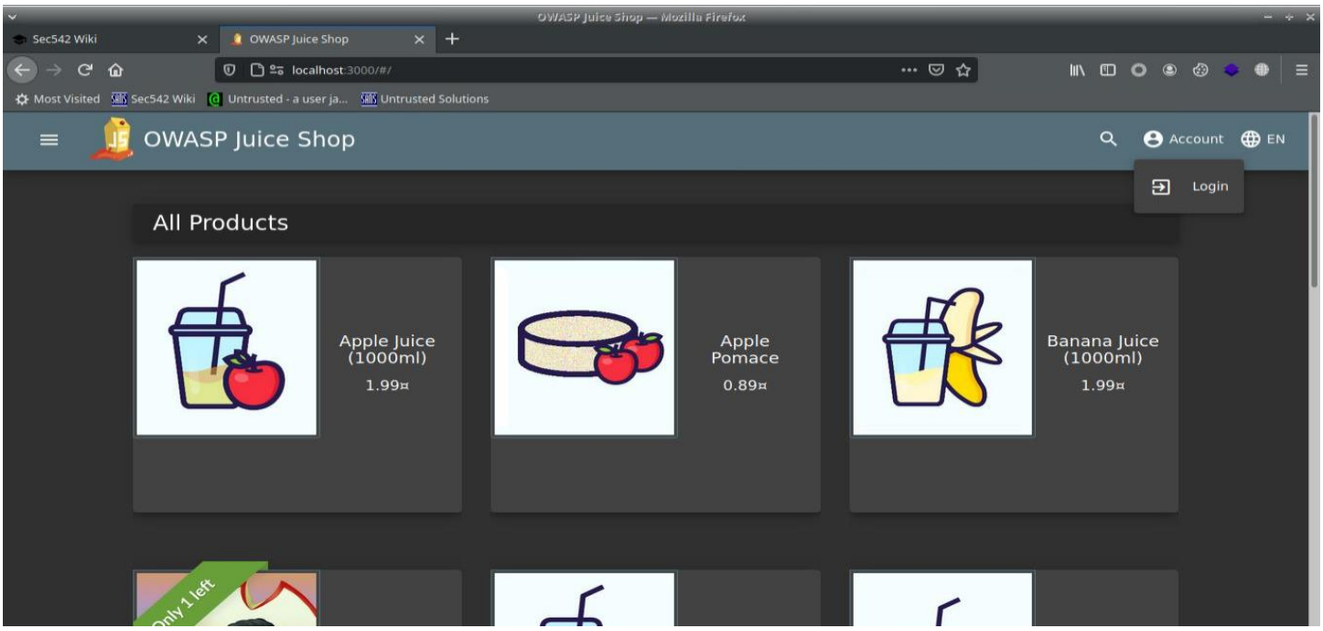
Impact

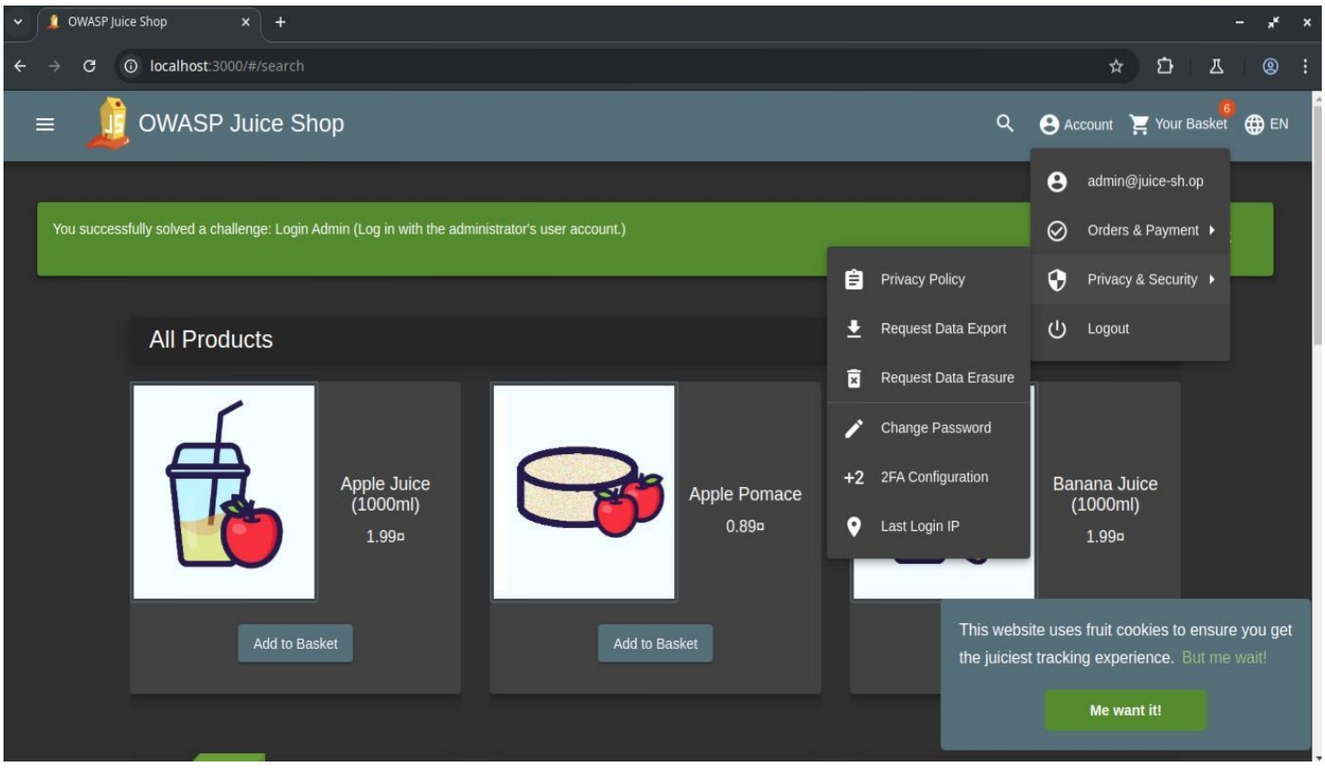
- Confidentiality: High (Unauthorized data retrieval)
- Integrity: High (Data manipulation possible)
- Availability: High – SQL Injection can disrupt the application's operations by corrupting or deleting critical data.

Remediation

- Implement Parameterized Queries: Use secure APIs or prepared statements that handle SQL inputs properly.
- Input Validation: Enforce strict input sanitation on all fields to reject illegal characters.
- Database Security: Regularly monitor and audit database access logs to detect unusual activity.

Admin Account Access:





6.2 DOM-BASED Cross-Site Scripting (XSS) in Search Bar

- CWE Reference: CWE-79
- CVSS Score: 6.1 (Moderate)
- Severity: Medium

  DOM-based XSS vulnerabilities are moderate in severity because they require user interaction to execute, yet they can still pose significant risks to user data confidentiality.

- Description:

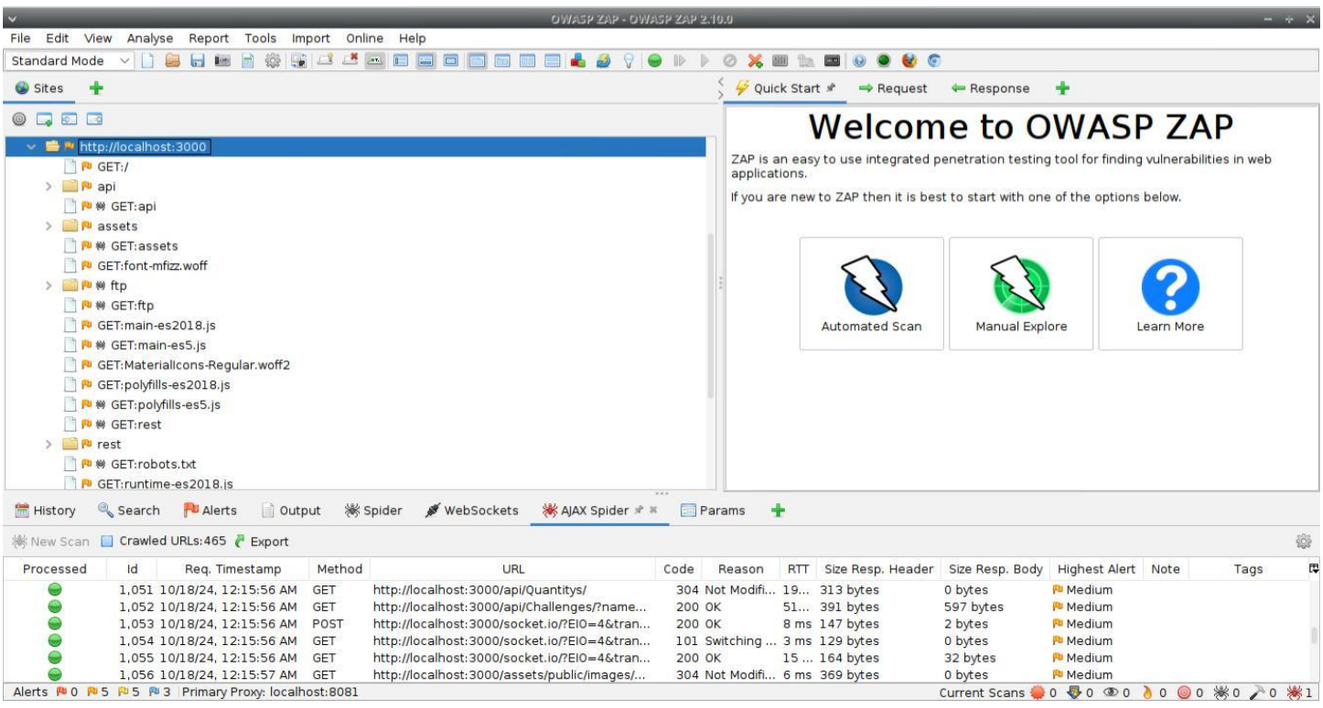
  - DOM-based XSS (Cross-Site Scripting) is a client-side vulnerability that arises when the web application's JavaScript code processes user inputs directly within the Document Object Model (DOM) without adequate sanitization. In OWASP Juice Shop, this vulnerability was identified in the search bar. When user inputs are directly injected into the DOM, it allows the attacker to execute arbitrary JavaScript code in the context of other users' sessions.

  PoC:

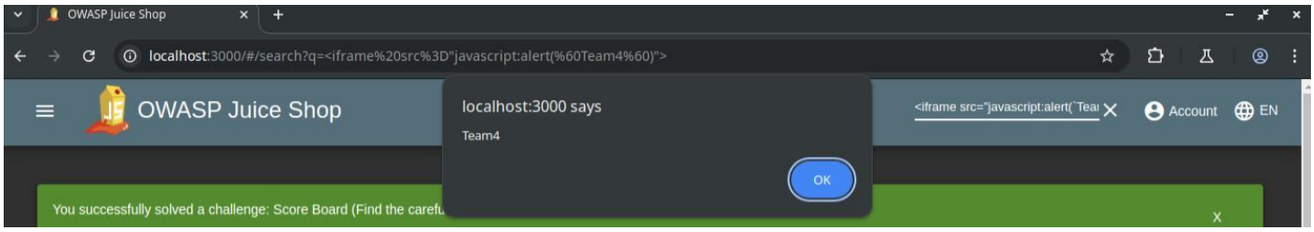
  Payload Used:

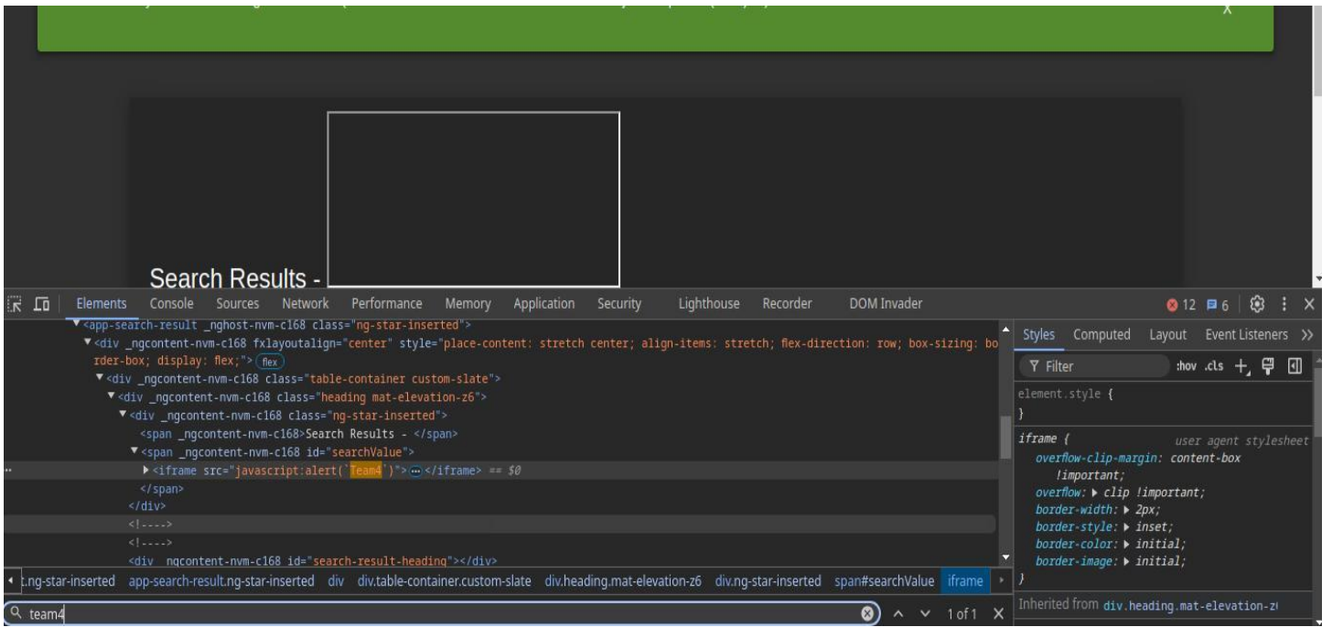
  - <iframe src="javascript:alert('Team4')"></iframe>

- Tool Used: OWASP ZAP was used to spider and identify the input fields vulnerable to XSS.



- Manual Testing: Observed the execution of the payload in the browser and recorded it using the Developer Tools console.





Likelihood of Exploit

- Attack Vector: Client-side, through user interaction
- Privileges Required: None (open to all users)
- Likelihood: High – XSS attacks are easy to execute using crafted payloads in accessible input fields.

Impact

- Confidentiality: Low to Medium – This vulnerability can lead to the theft of session tokens or user credentials if successfully exploited.
- Integrity: Low to Medium – Attacker-controlled scripts can manipulate the page's content, altering how information is displayed to users.
- Availability: Low – DOM-based XSS typically affects the client's browser, with minimal impact on server-side operations.

Remediation

- Sanitize and Encode Inputs: Ensure all user inputs are properly sanitized to strip harmful characters.
- Implement CSP: Enforce a strong Content Security Policy to block unauthorized script execution.
- Continuous Code Review: Regularly inspect code for unsafe handling of user-generated content.

6.3 Insecure Direct Object Reference (IDOR) in Feedback Submission

- CWE Reference: CWE-639
- CVSS Score: 7.5 (High)
- Severity: High

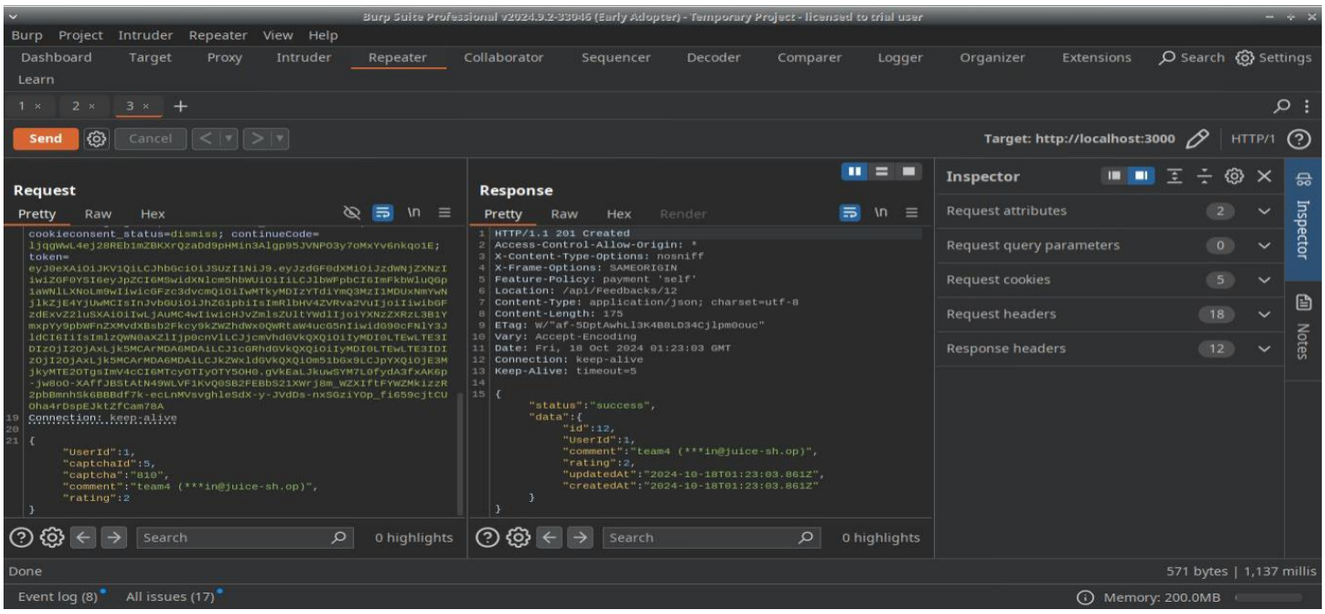
  IDOR vulnerabilities have a high impact due to their ability to expose sensitive information and alter other users' data without proper access controls.

Description:

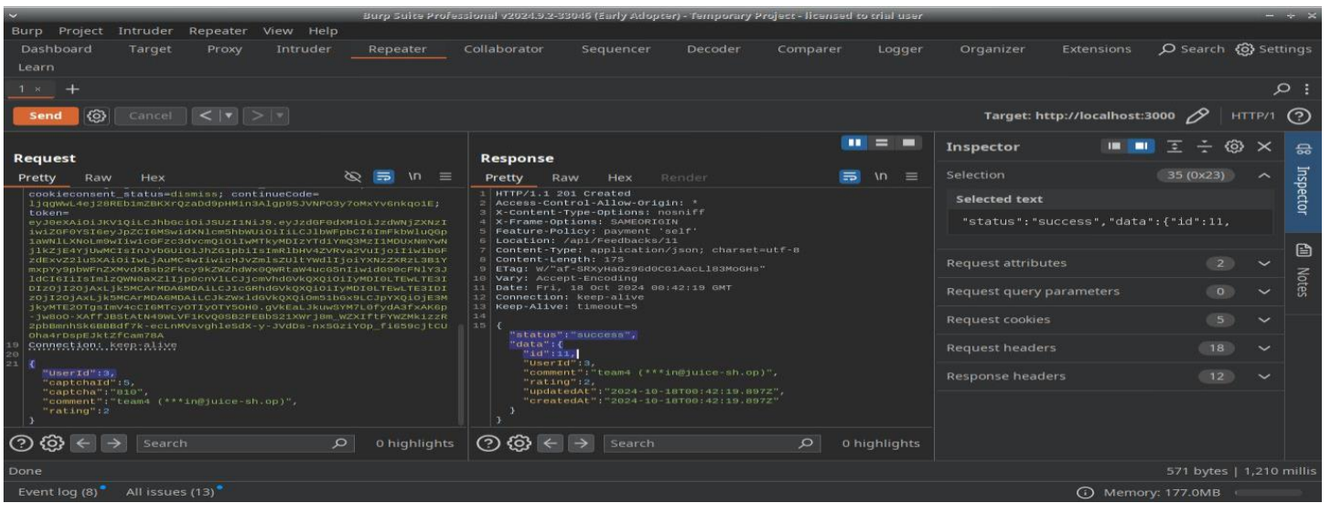
Attackers can manipulate the UserId parameter in the feedback submission form to post feedback under another user's account, without being authorized to do so.

PoC:

- Intercept Request: The feedback submission request was intercepted using Burp Suite.



- Modify UserId: The UserId in the request was changed to impersonate another user.



Likelihood of Exploit

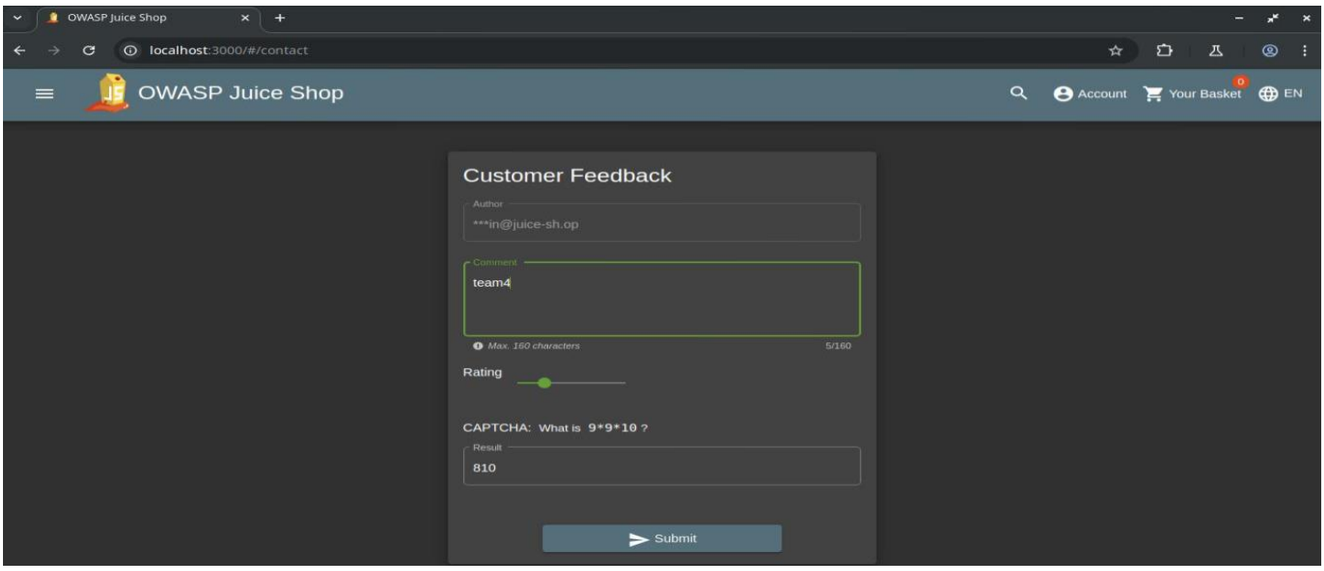
- Attack Vector: Web-based with direct URL manipulation
- Privileges Required: Normal user privileges required initially
- Likelihood: High – Low technical skills are needed to exploit IDOR by altering user IDs directly.

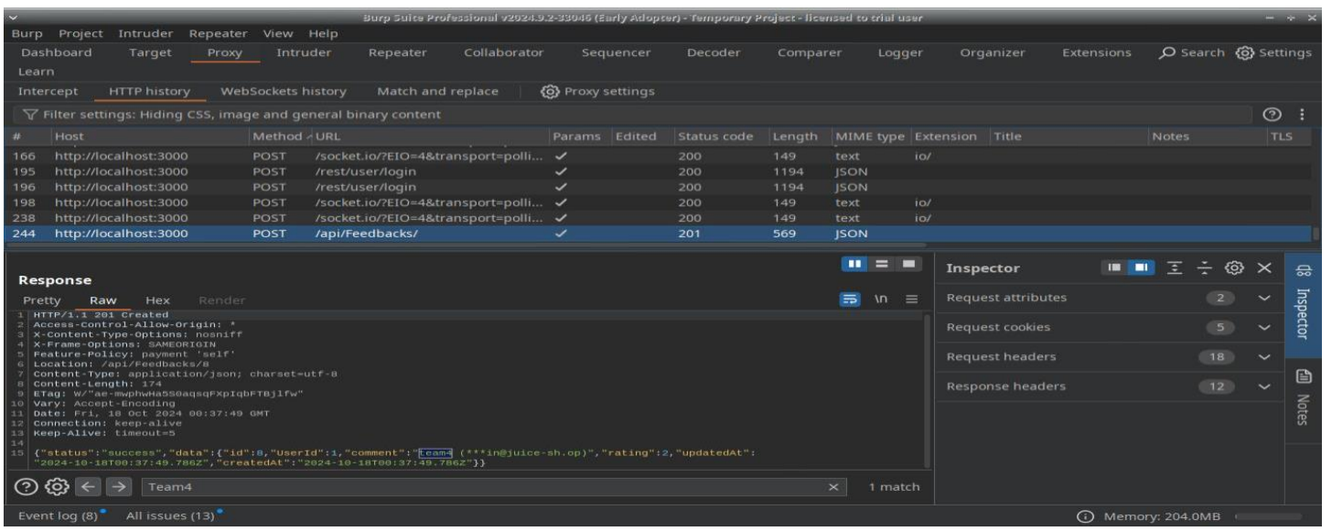
Impact

- Confidentiality: High (Access to other users' sensitive information)
- Integrity: High – An attacker can modify data, affecting its accuracy and trustworthiness.
- Availability: Low (No direct impact on system uptime)

Remediation

- Implement Access Controls: Ensure that user permissions are properly validated server-side.
- Use Indirect References: Replace direct object references with mapped IDs or tokens.
- Audit Sensitive Operations: Regularly review access logs to detect unauthorized access patterns.





6.4 Exposure of Confidential Files in “About Us” Section

- CWE Reference: CWE-552
- CVSS Score: 7.2 (High)
- Severity: High

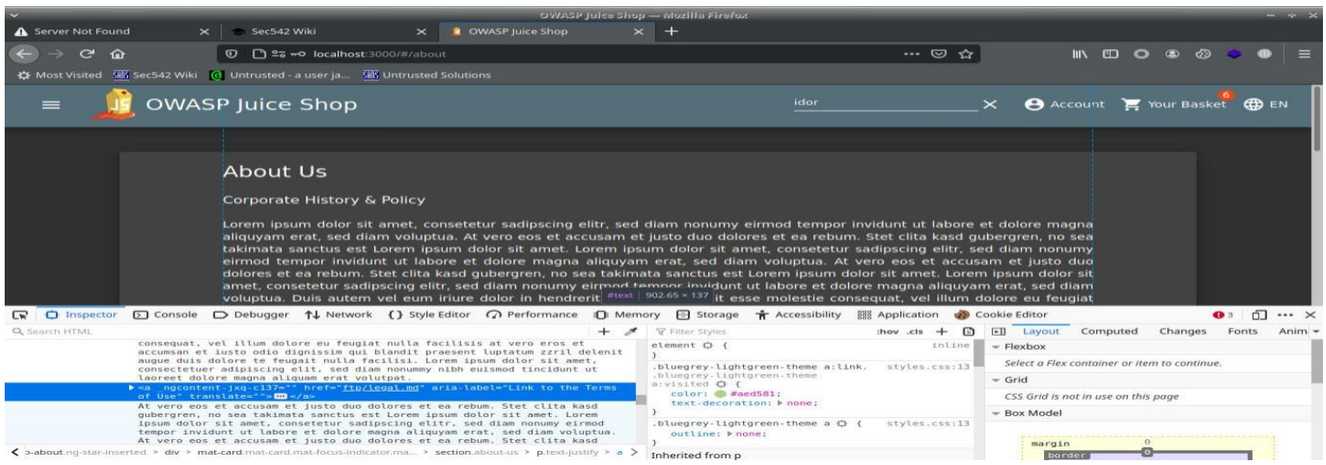
  The high severity is justified by the risk of unauthorized access to sensitive data that could compromise business operations or reveal critical system configurations.

- Description:

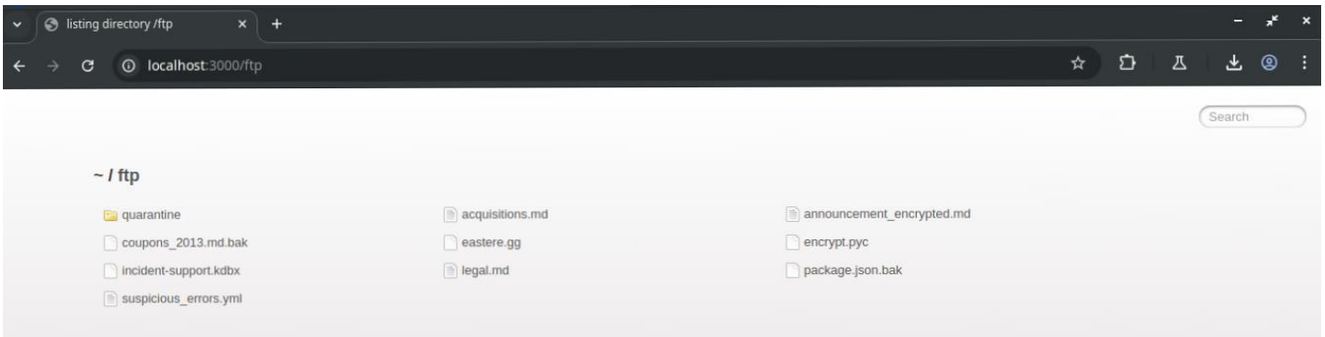
  Several sensitive files, including package.json.bak and acquisitions.md, are exposed through an unprotected FTP link in the "About Us" section. These files contain internal configurations and sensitive data.

Proof-of-Concept (PoC):

- Discover FTP Link: Use Developer Tools to inspect the "About Us" section and find a link to an exposed FTP directory.



- Download Exposed Files: Open the FTP link and download the exposed files, confirming the presence of sensitive information.



Likelihood of Exploit

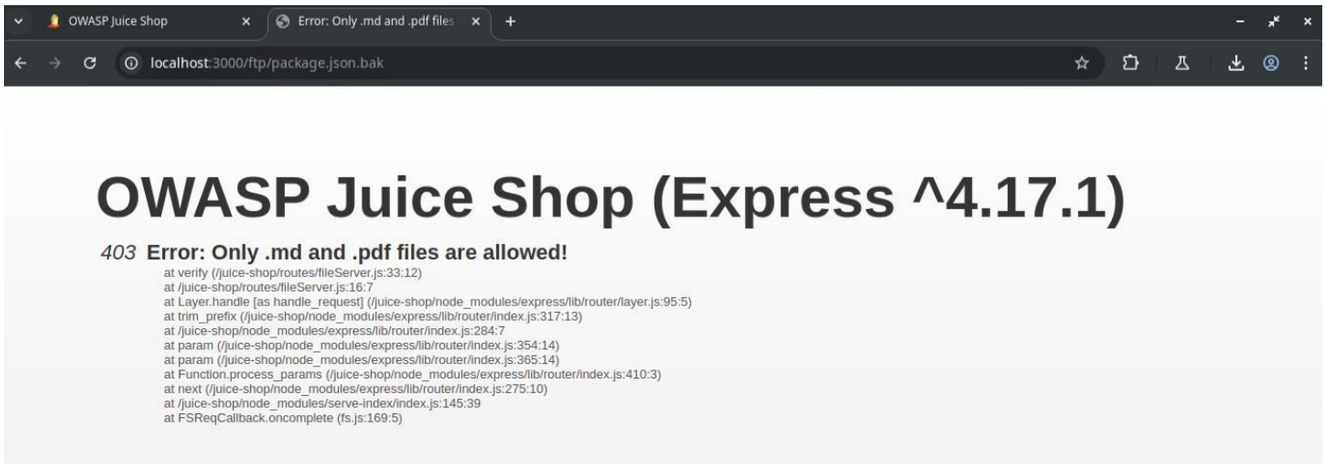
- Attack Vector: Network-based, leveraging file paths in the web directory
- Privileges Required: None (open access via directory browsing)
- Likelihood: High – Easily exploitable by anyone with access to the web application.

Impact

- Confidentiality: High (Leakage of sensitive business and configuration data)
- Integrity: None – File exposure typically does not involve modifying or corrupting files.
- Availability: Low (No direct impact on system operations)

Remediation

- Restrict Directory Access: Limit access to FTP directories to authorized users only.
- Secure Sensitive Files: Encrypt or relocate sensitive files to secure storage locations.
- Regular Security Audits: Periodically check file permissions and access logs.



Methodology

<<

Initial Access:

The web application was accessed through its publicly available interface as a standard user. No specific credentials were used initially, as the focus was to test the application's security controls from an unauthenticated standpoint.

Step-by-Step Approach:

1. Detailed Setup: Detailed Docker installation commands and environment setup for testing Juice Shop.
2. Reconnaissance: Identified potential vulnerabilities using ZAP Spider and manual exploration.
3. Exploitation Techniques: Used Burp Suite for interception and SQLMap for automated attacks.
4. Validation: Systematic documentation with screenshots to confirm findings.
5. Reporting: Structured the report to align with industry standards, using screenshots for clear visuals.

>>

Assessment Toolset

<<

Tools Used:

- SQLMap: For detecting and exploiting SQL Injection vulnerabilities.
- Burp Suite: Used to intercept and modify requests, especially for testing command injection and XSS.
- OWASP ZAP: Employed for scanning XSS vulnerabilities.
- Manual Testing: Additional manual validation using browser developer tools and crafted HTTP requests.

>>

Assessment Methodology Detail

<<

1. Setup Phase

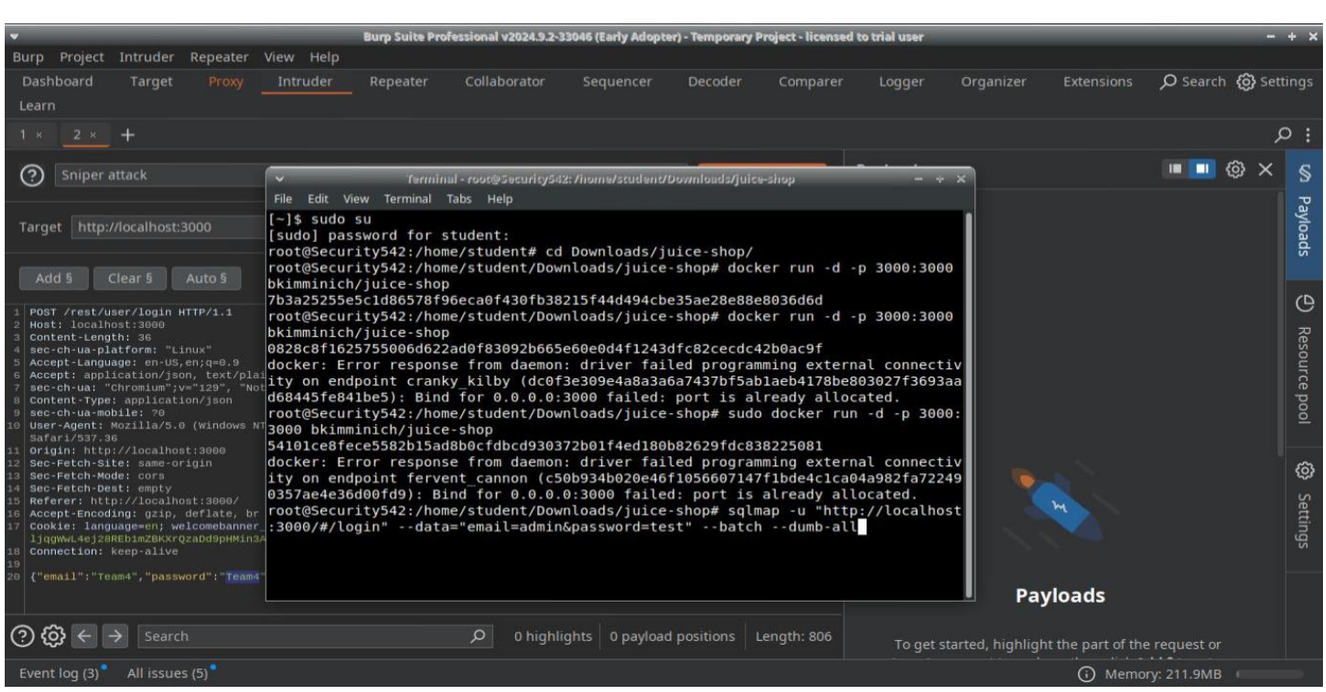
Objective: Establish the testing environment for a secure and controlled assessment of the Juice Shop application.

Tools Used:

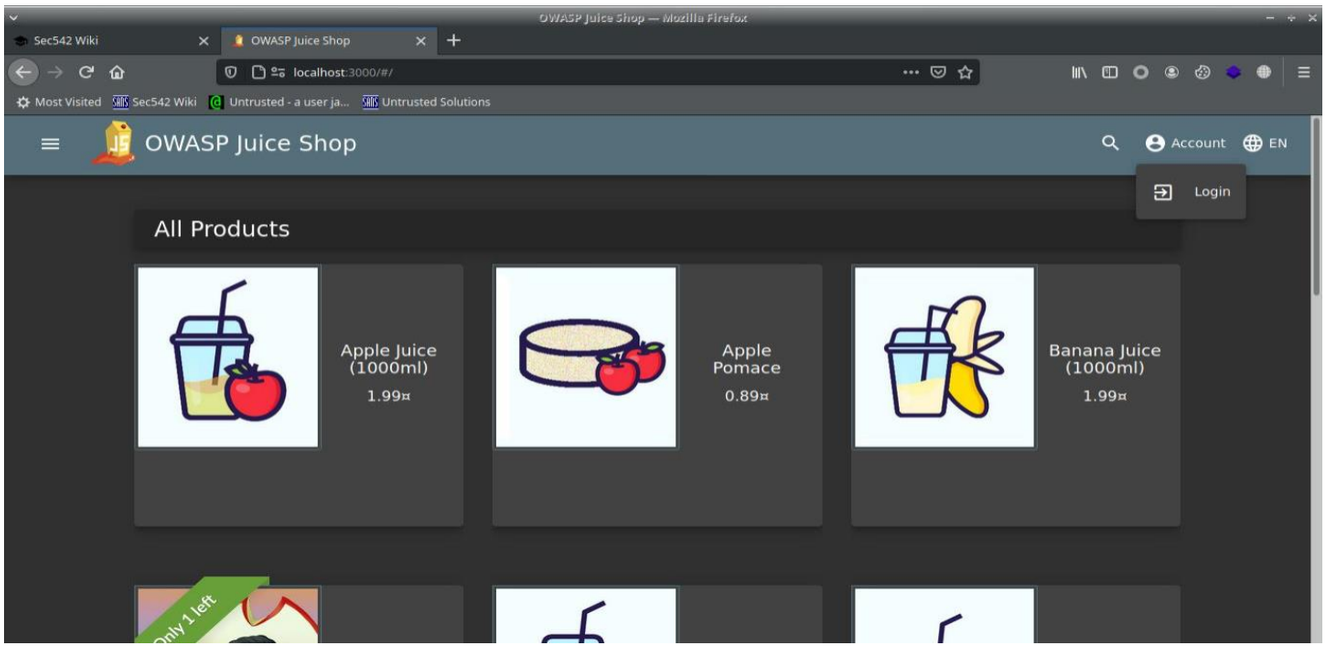
- Docker: Used to install and run the Juice Shop instance on a local machine.
- System Configuration: Configured the network settings to allow uninterrupted access to the web application.

Steps:

1. Installation: The Juice Shop was installed using Docker with the command: docker run -d -p 3000:3000 bkimminich/juice-shop.



2. Environment Check: Verified that the application was running properly by accessing it via http://localhost:3000.

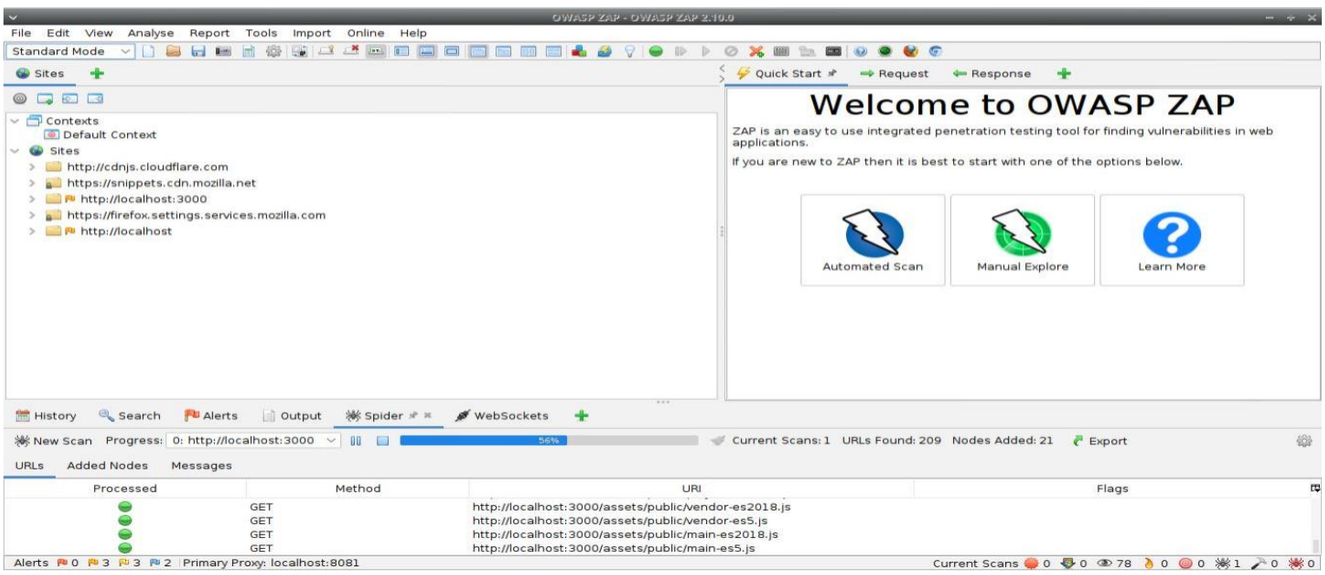


2. Reconnaissance Phase

Objective: Gather information about the application's structure, identify potential attack vectors, and map out areas of interest.

Tools Used:

- Browser Developer Tools: For inspecting page elements, finding hidden endpoints, and analyzing JavaScript code.
- OWASP ZAP Spider: For automated discovery of pages and parameters.

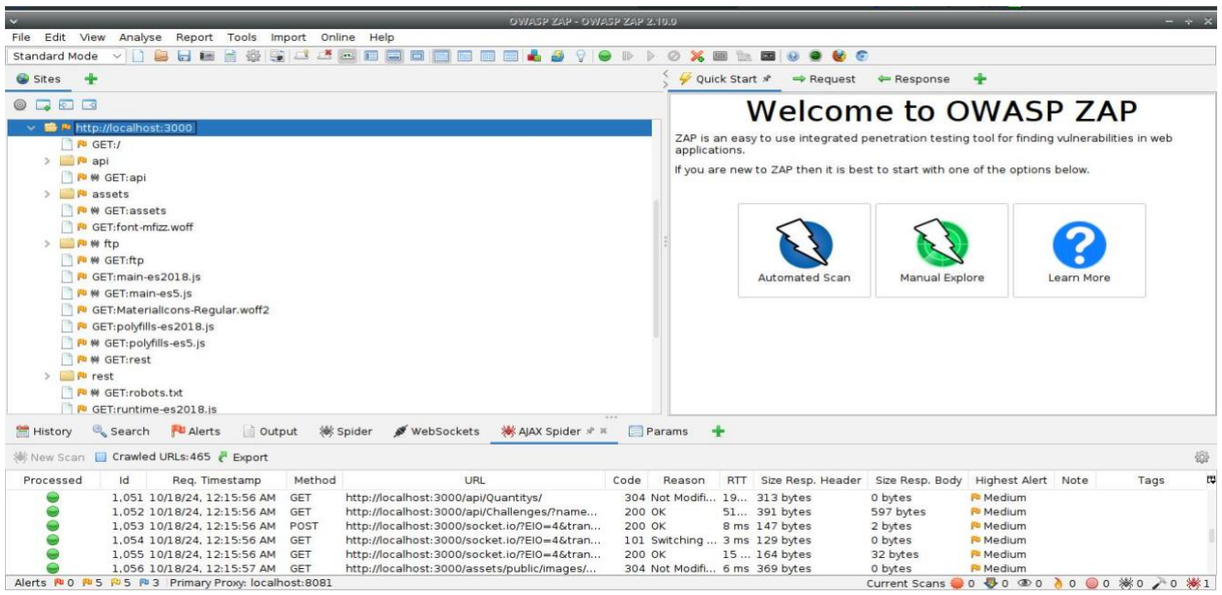


Steps:

1. Browsing: Explored each page of the Juice Shop to identify input fields and functions.

2. Spidering: Used the ZAP Spider to automatically map out the application's structure and extract hidden URLs and parameters.

3. Parameter Analysis: Documented all parameters identified by the spider for future exploitation attempts.



3. Exploitation Phase

Objective: Actively test identified vulnerabilities to confirm their existence and evaluate their impact.

Tools Used:

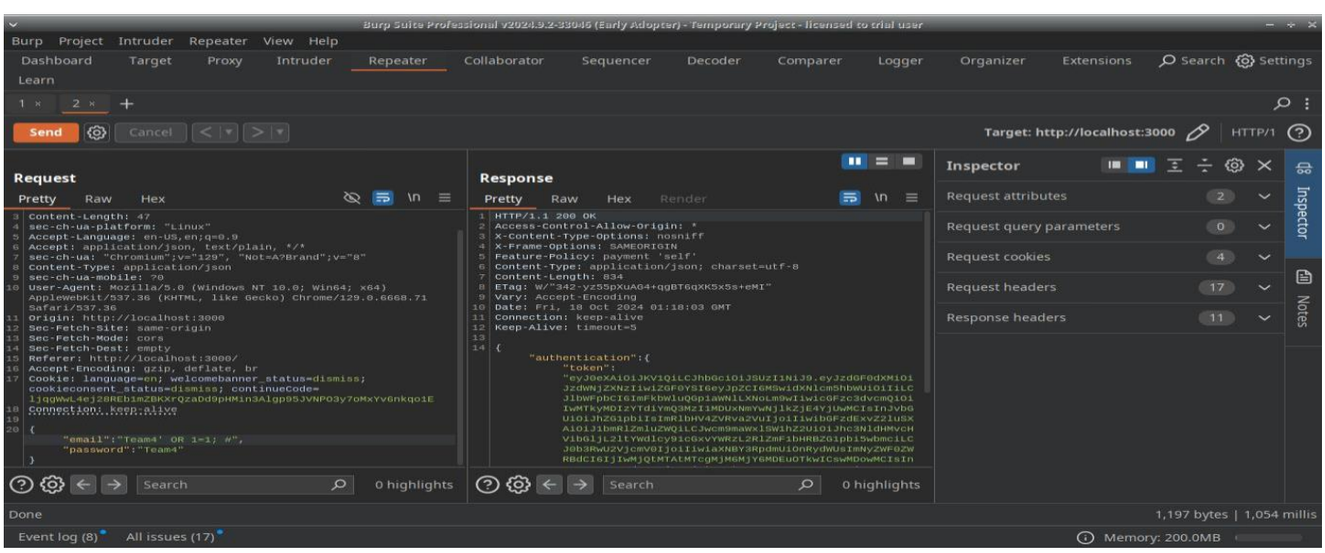
- Burp Suite: For intercepting HTTP requests and manipulating data in real-time.
- SQLMap: For automated SQL Injection exploitation.
- Custom Scripts: For testing Cross-Site Scripting (XSS) payloads.

Exploitation Scenarios:

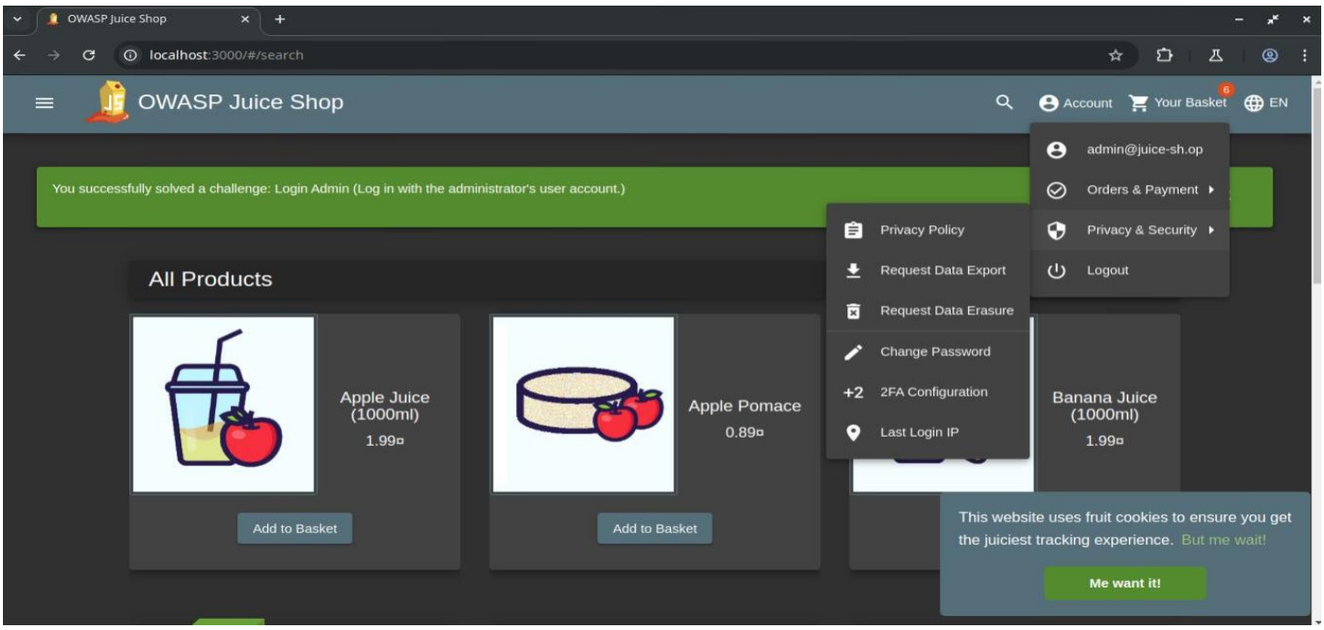
Scenario 1: SQL Injection

1. Intercept Login Request: Intercepted the login request using Burp Suite.

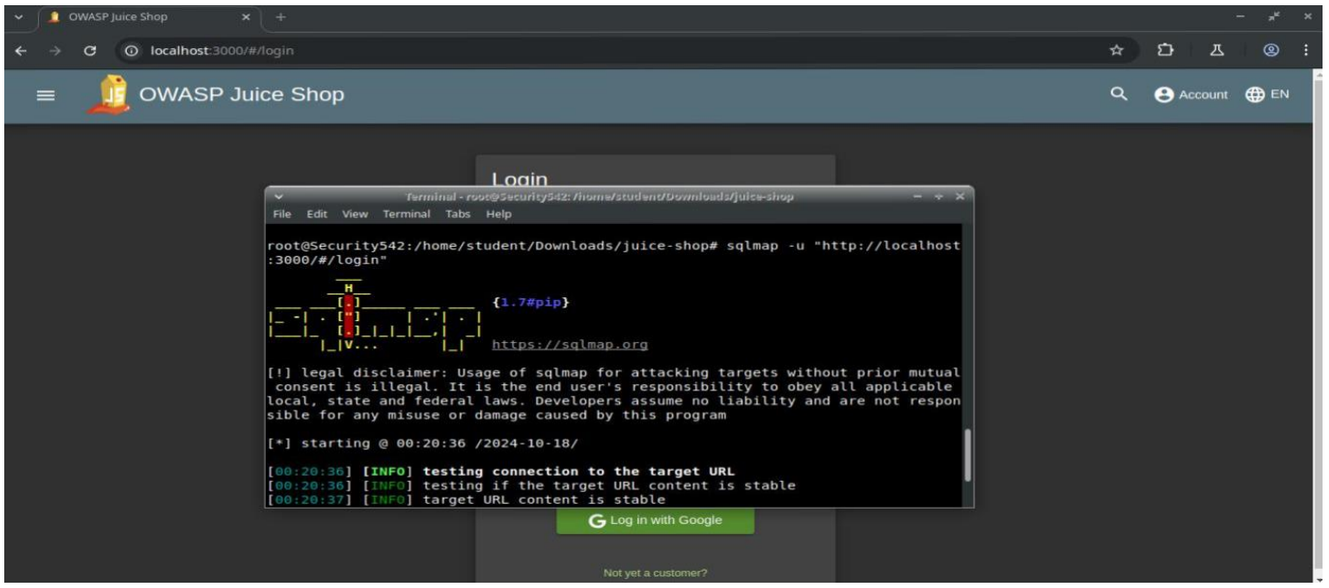
    - Payload: Modified the email field to: team4' OR 1=1; --



2. Verification: Successfully bypassed authentication to access the admin account.

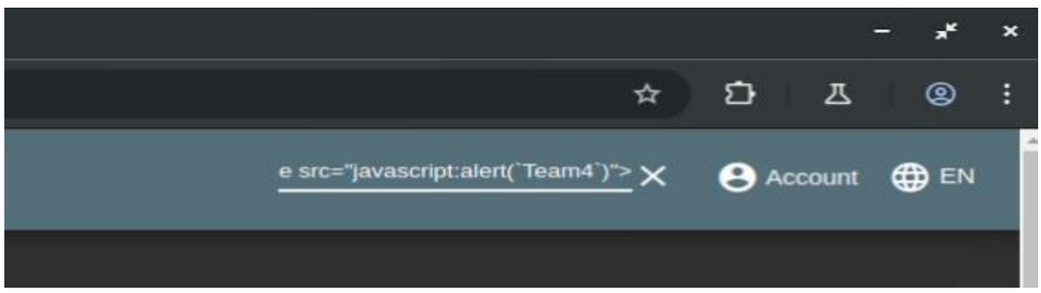


3. Automated Exploitation: Used SQLMap to automate the injection attack, confirming the vulnerability and retrieving the database schema.

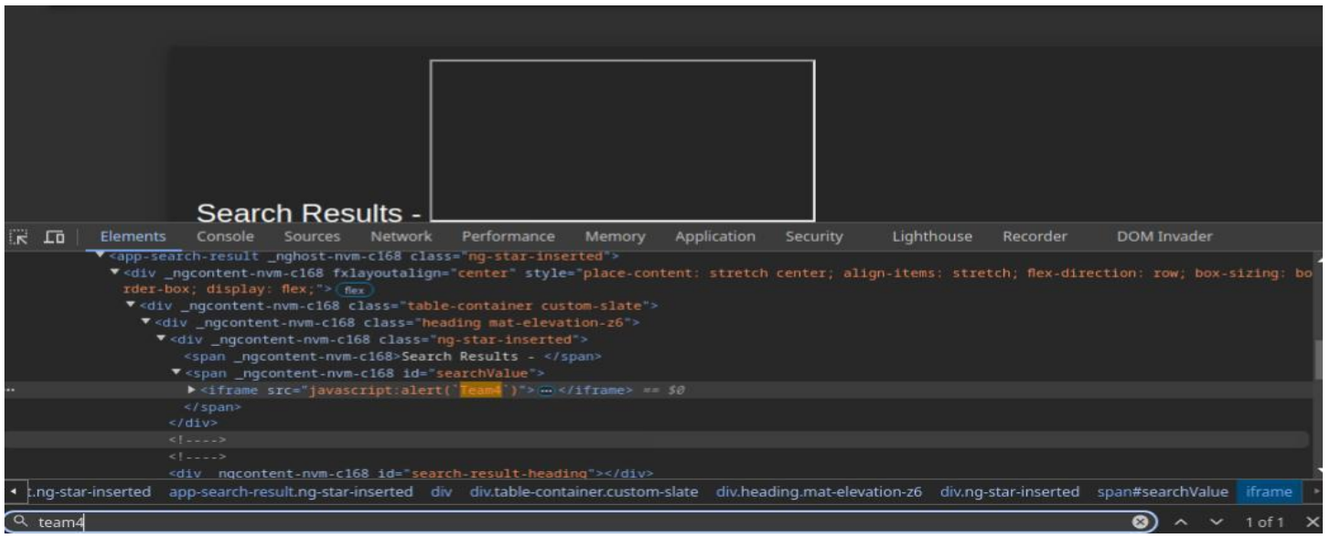


Scenario 2: Cross-Site Scripting (XSS)

1. Injection in Search Bar:
    - HTML Tag with our payload <iframe src="javascript:alert('Team4')"></iframe> into the search bar.

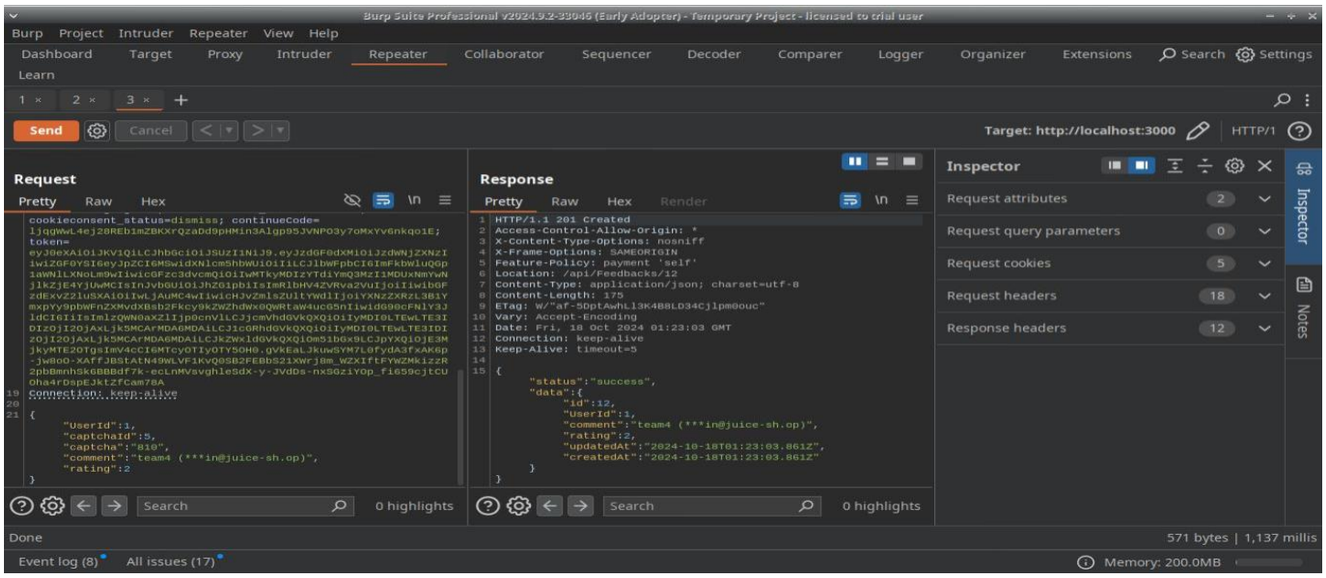


2. Validation: Observed the execution of the payload in the browser and recorded it using the Developer Tools console.

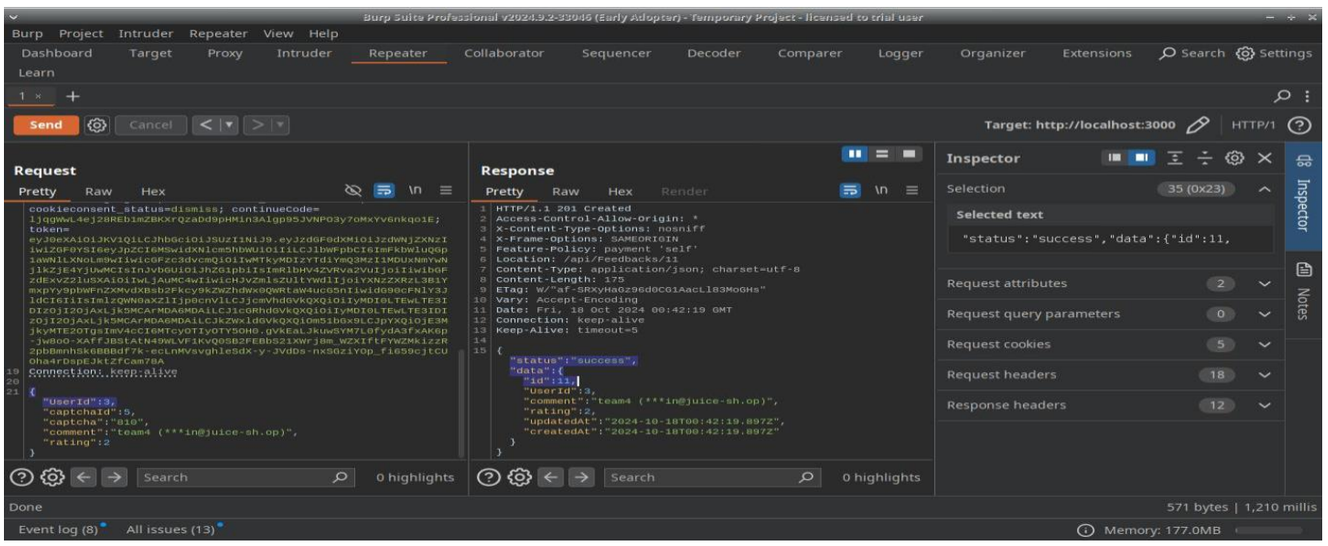


Scenario 3: IDOR (Insecure Direct Object Reference)

1. Manipulating UserId Parameter: Used Burp Suite to modify the UserId parameter in the feedback functionality.

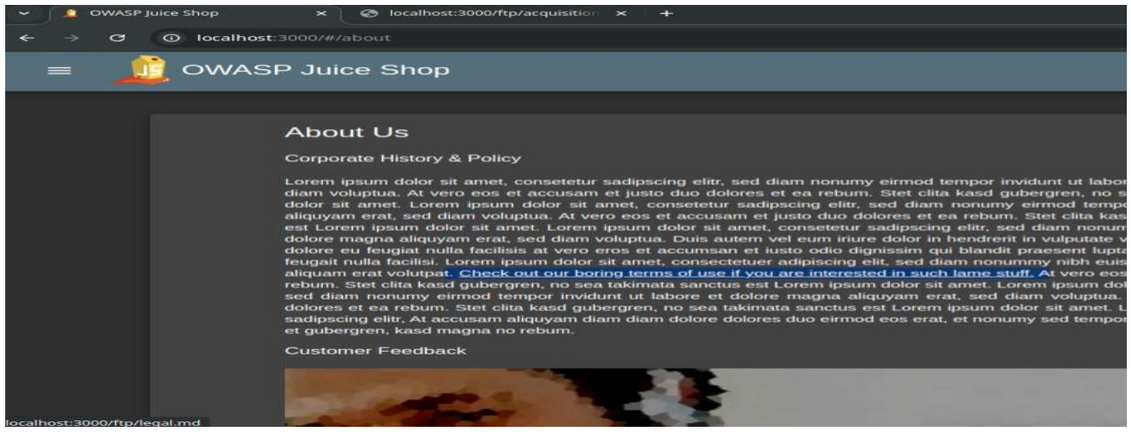


Result: Gained access to another user's feedback by changing the UserId value, indicating a lack of proper access control

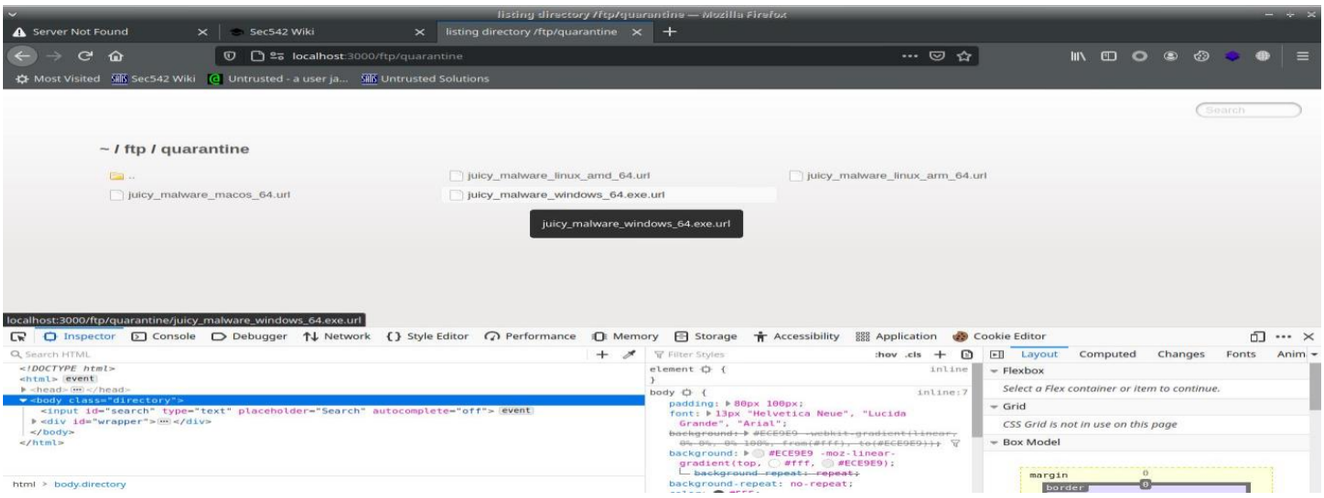


Scenario 4: Confidential File Exposure

1. File Discovery: Browsed to the ‘About Us’ section and used the Developer Tools to discover a hidden FTP directory.



2. Accessing Sensitive Files: Identified and opened files like acquisitions.md and package.json.bak, confirming their exposure.



4. Validation Phase

Objective: Validate each discovered vulnerability to ensure reproducibility and gather conclusive evidence for reporting.

Tools Used:

- Logs and Screenshots: Captured all evidence of successful exploits to document the vulnerabilities.
- Exploit Analysis: Evaluated the impact of each vulnerability to classify its severity accurately.

Steps:

1. Screenshot Documentation: Captured screenshots of successful attacks, including before-and-after states in Burp Suite and browser alerts.

2. Impact Assessment: Analyzed the potential impact on the application's integrity, confidentiality, and availability.

5. Reporting Phase

Objective: Compile all findings into a detailed report, including technical analysis, PoCs, impact assessments, and remediation strategies.

Tools Used:

- Document Editor (MS Word): For creating the structured vulnerability report.

- Visual Enhancements: Added screenshots to ensure clarity and completeness.

Steps:

1. Report Structuring: Organized the report into sections, covering all aspects of the vulnerability assessment.

2. Executive Summary: Summarized key findings, risk levels, and immediate remediation recommendations for quick review.

Conclusion

The Juice Shop web application has multiple critical vulnerabilities that pose a severe risk to its integrity and security. Immediate remediation efforts are necessary to protect the system from potential exploitation. The organization should adopt a proactive approach by implementing advanced security measures, continuous monitoring, and regular security assessments to stay ahead of emerging threats.