WEB APPLICATION SECURITY TESTING

NAME: SMRIDHI GERA

TASK 1: WEB APPLICATION SECURITY TESTING

PROGRAM: FUTURE INTERNS- CYBERSECURITY INTERNSHIP

DATE: OCTOBER 2025

TARGET:APPLICATION :OWASP JUICE SHOP

Task Summary

This assessment focused on identifying various loopholes and vulnerabilities in the **OWASP Juice Shop** application through **Burp Suite** and manual evaluation techniques. Several vulnerabilities were successfully detected, including **SQL Injection** and **User Information Leakage**, which could lead to potential attacks and account compromises. The findings were mapped to the **OWASP Top 10 (2021)** categories based on associated risk levels.

Tools Used

- Docker
- OWASP Juice Shop
- Burp Suite

Steps & Procedure

1. The **OWASP Juice Shop** application was deployed locally using **Docker**, ensuring a lightweight and isolated testing environment with all dependencies pre-configured.
2. **Burp Suite** was utilized for intercepting, scanning, and analyzing web requests and responses to identify vulnerabilities within the application.
3. All discovered issues were manually validated and mapped to the **OWASP Top 10 (2021)** categories based on their severity and potential impact.
4. Corresponding **mitigation strategies** were proposed for each identified vulnerability to enhance overall application security.

Identified Vulnerabilities

1. SQL Injection

Description:
The login form was vulnerable to SQL Injection, as it accepted unsanitized input that was directly concatenated into SQL queries. Attackers could inject payloads like ' OR '1'='1 to bypass authentication and access restricted areas.

Security Impact:

- Complete bypass of authentication.
- Unauthorized access to database records.
- Potential full compromise of database contents.

Risk Level: High

Evidence:

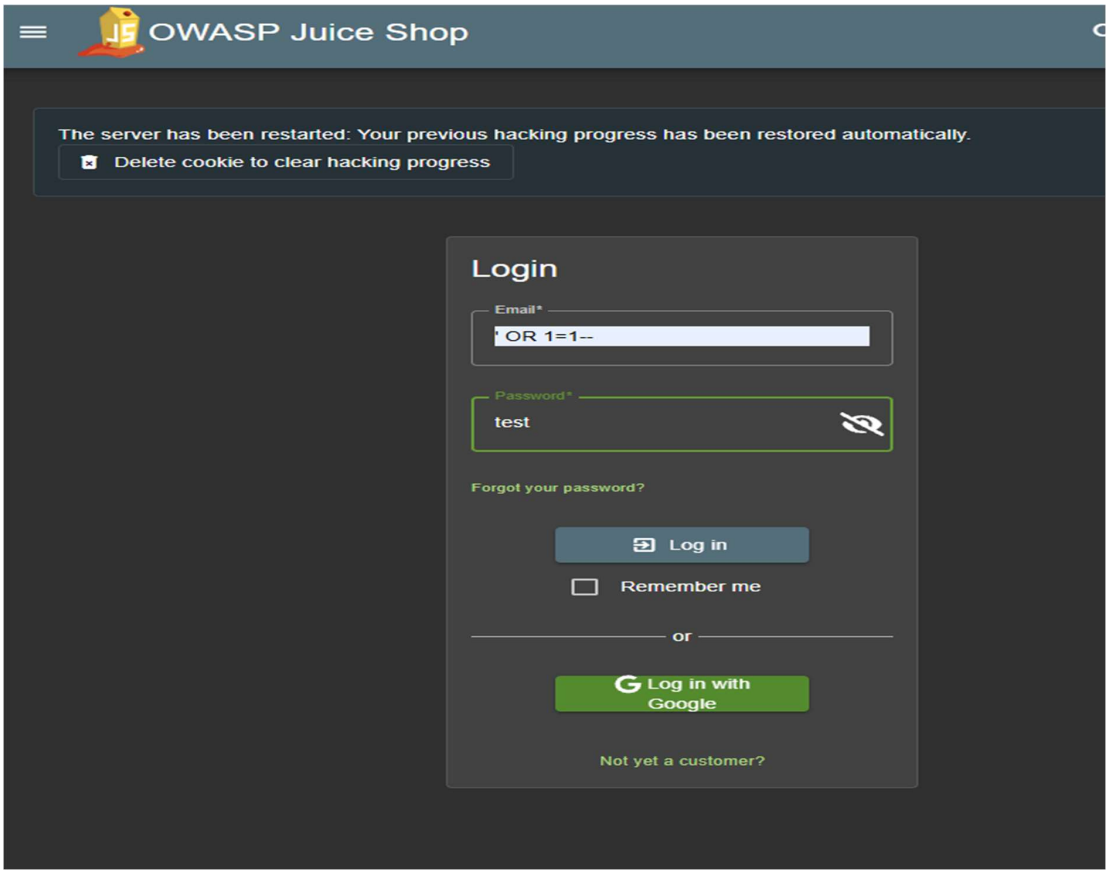


Figure 1: SQLi Payload in Login Form

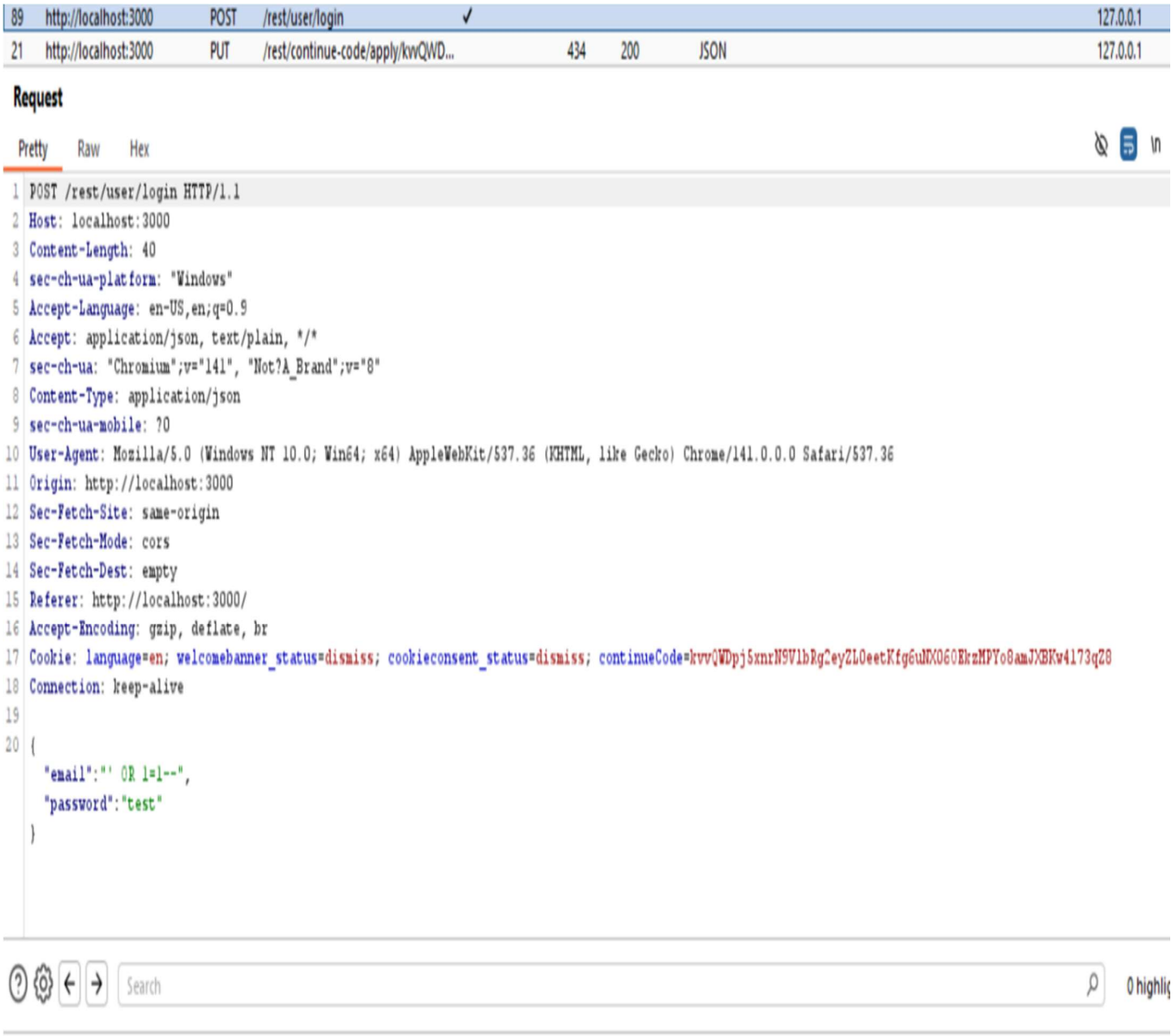


Figure 2: burp suite intercepted request containing malicious payload

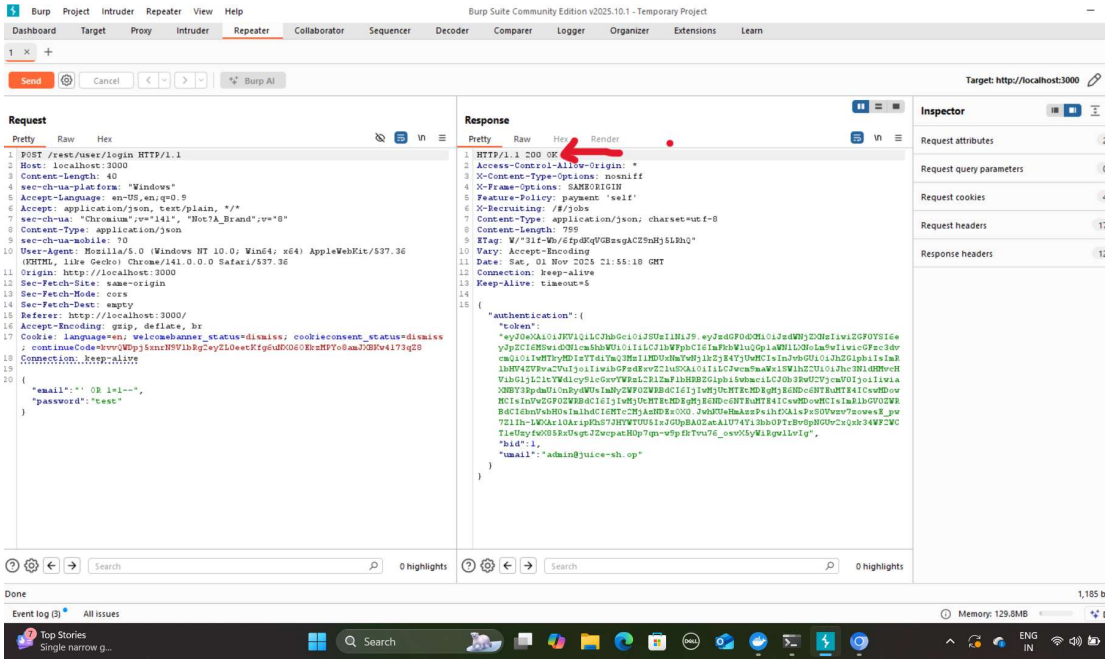


Figure 3: burp suite response showing http status 200 ok ,comforming successful login.

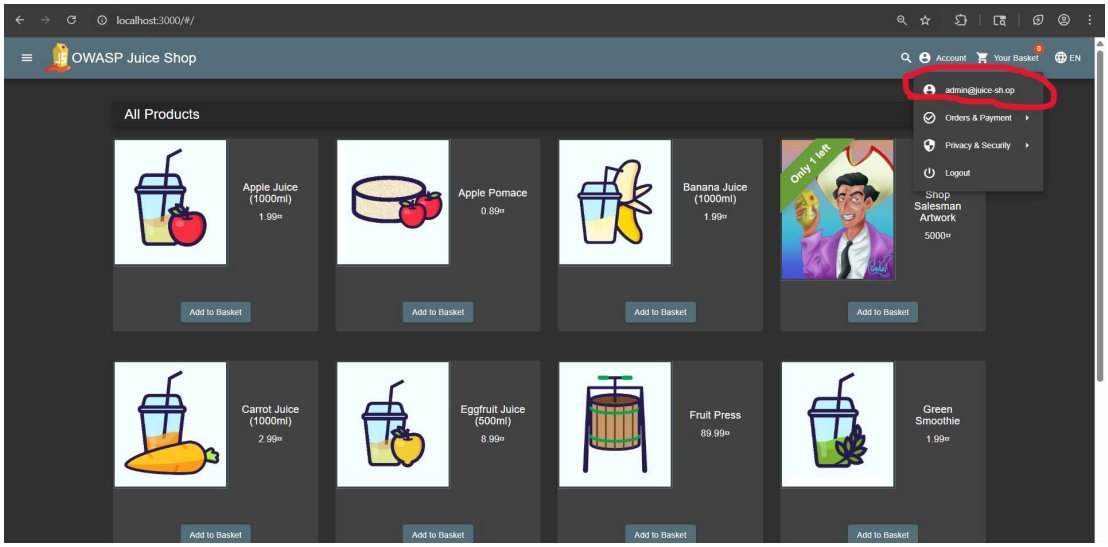


Figure 4: post login view showing access to admin account.

Mitigation Strategies:

- Use parameterized queries or prepared statements.
- Sanitize user inputs with strict whitelisting.
- Disable detailed SQL error messages in production.
- Implement least privilege access for the database user.

2. Cross-Site Scripting (XSS)

Description:
Certain input fields (such as feedback or product search) failed to properly sanitize user input, allowing JavaScript injection that executes in the victim's browser.

Security Impact:

- Theft of session cookies or tokens.
- Execution of unauthorized actions in user accounts.
- Phishing or defacement attacks on legitimate users.

Risk Level: High

EVIDENCE:

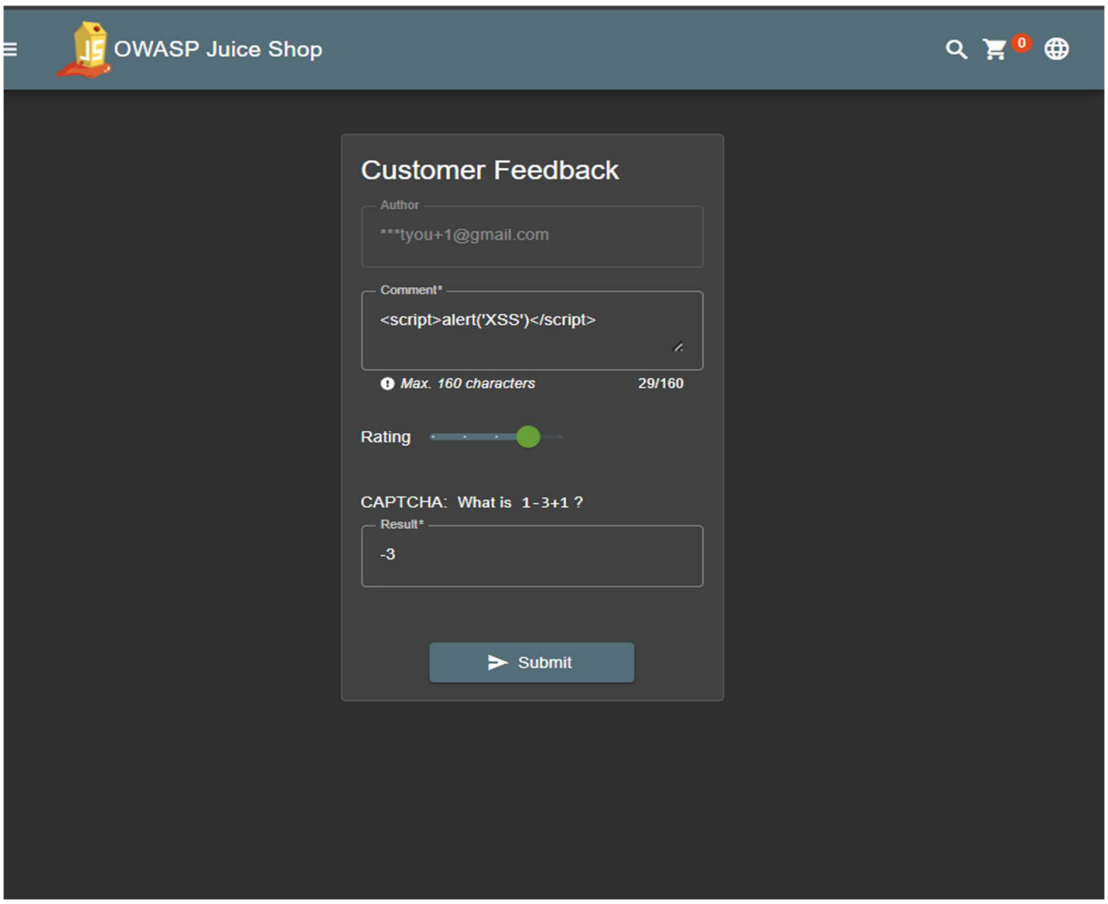


Figure1: feedback form with payload

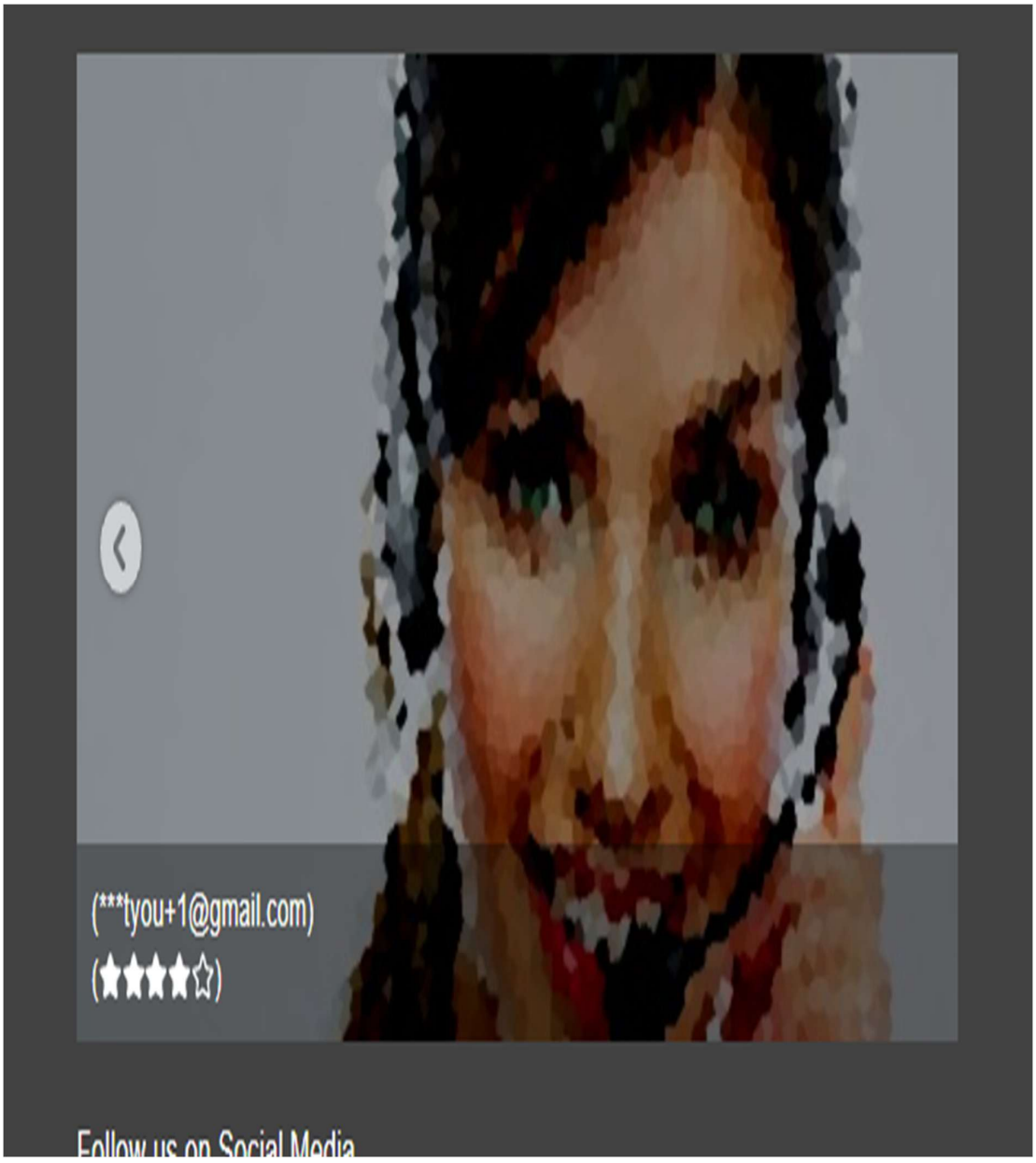


Figure2: alert box triggered after visiting abou

Mitigation Strategies:

- Implement input validation and output encoding.
- Use frameworks or libraries with built-in XSS protection.
- Apply a strong Content Security Policy (CSP).

3. Replay and Impersonation

Description:
Session tokens and authenticated requests could be replayed without verification, meaning previously captured requests (like feedback submission) could be resent to perform repeated actions.

Security Impact:

- Impersonation of legitimate users.
- Replay of actions such as duplicate submissions or transactions.
- Potential exploitation of long-lived tokens.

Risk Level: Medium

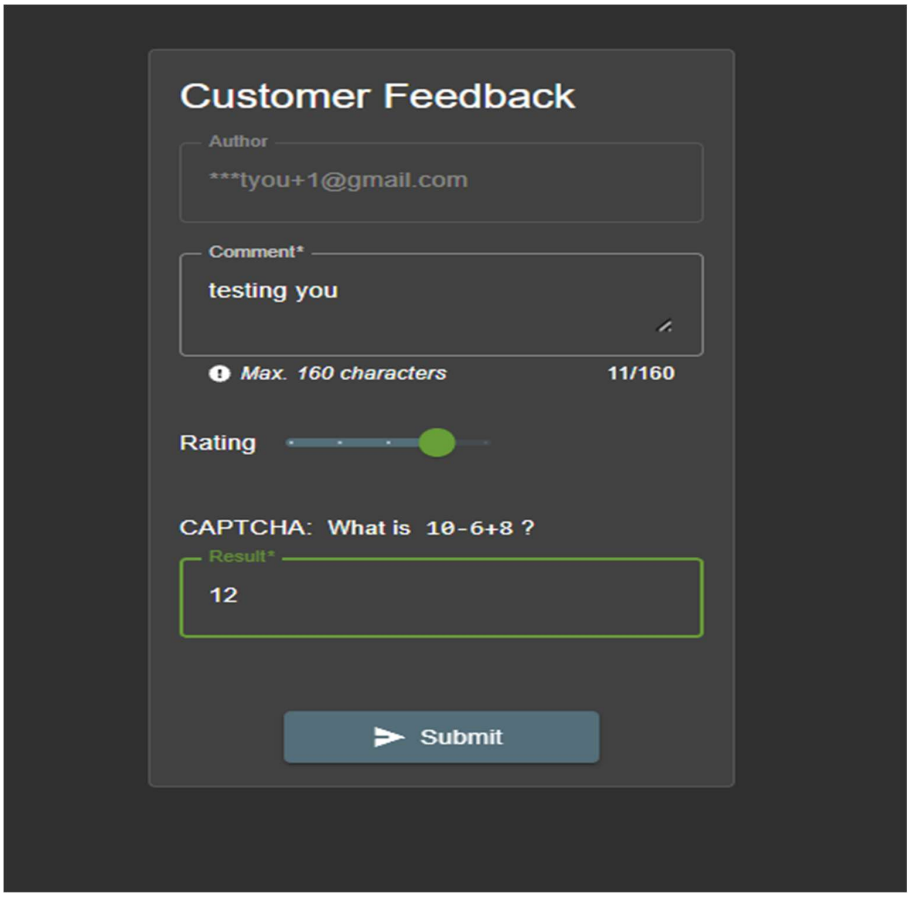


Figure 1: original feedback request submitted through the customer feedback form.

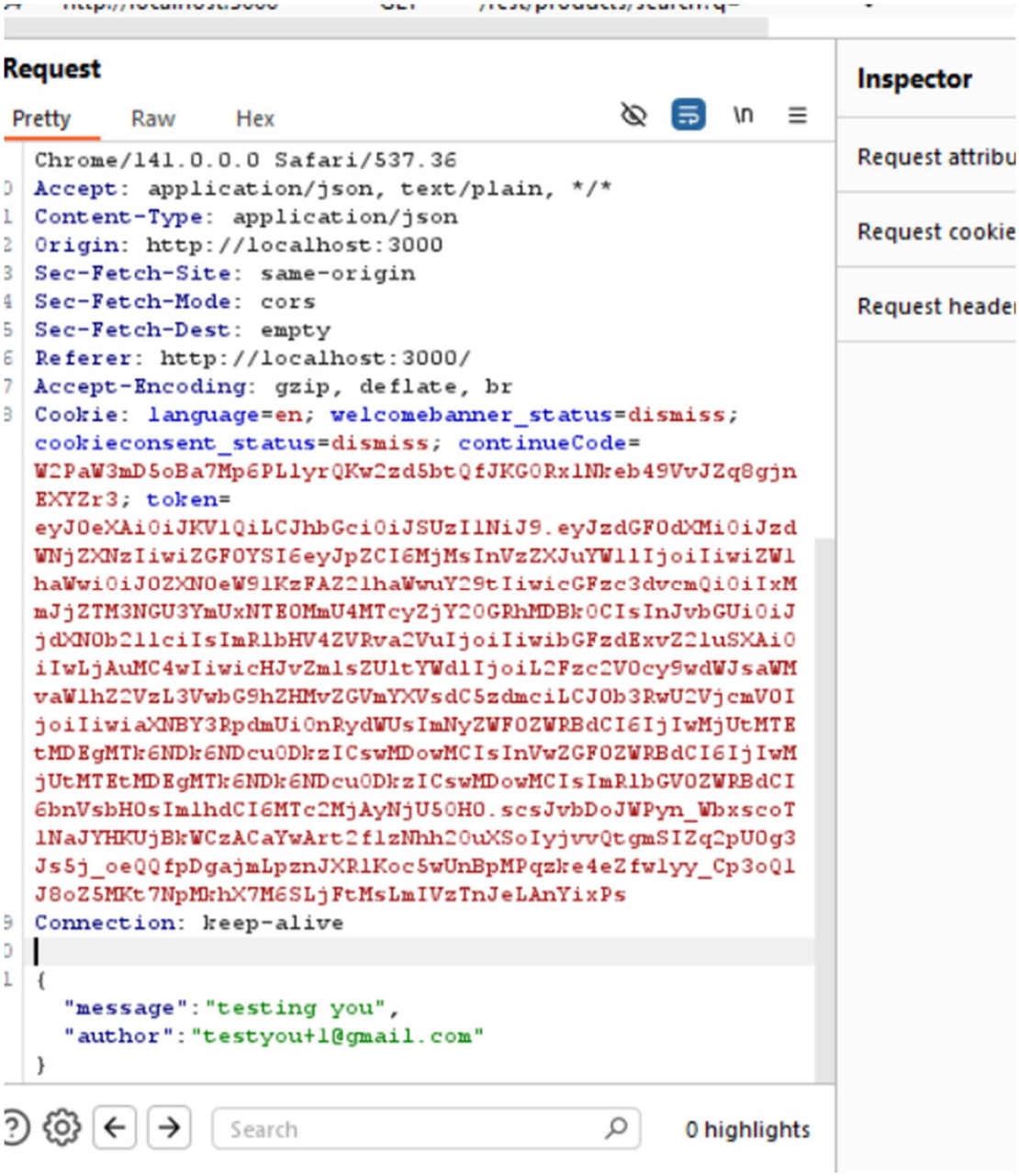


Figure 2: replayed request sent via burp suite repeater and accepted multiple times.

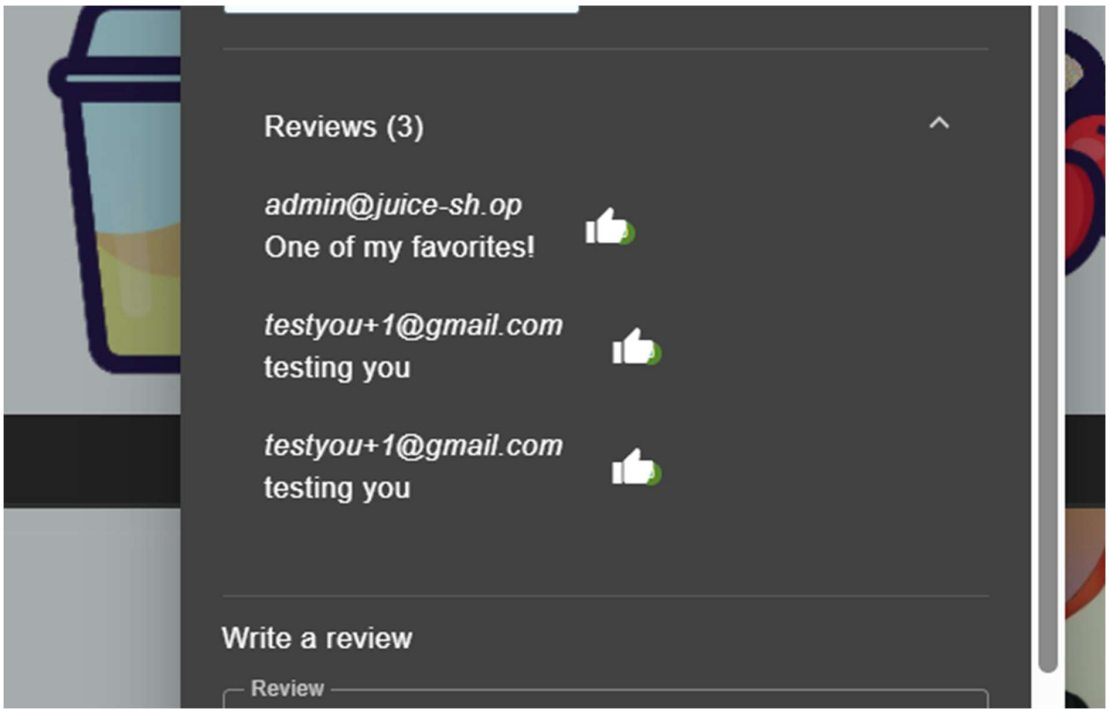


Figure 3: execution of replayed request multiple times

Mitigation Strategies:

- Implement nonce or timestamp validation to prevent replay.
- Use short-lived session tokens and refresh them periodically.
- Invalidate tokens immediately after logout or session timeout.

4. Insecure Token Design

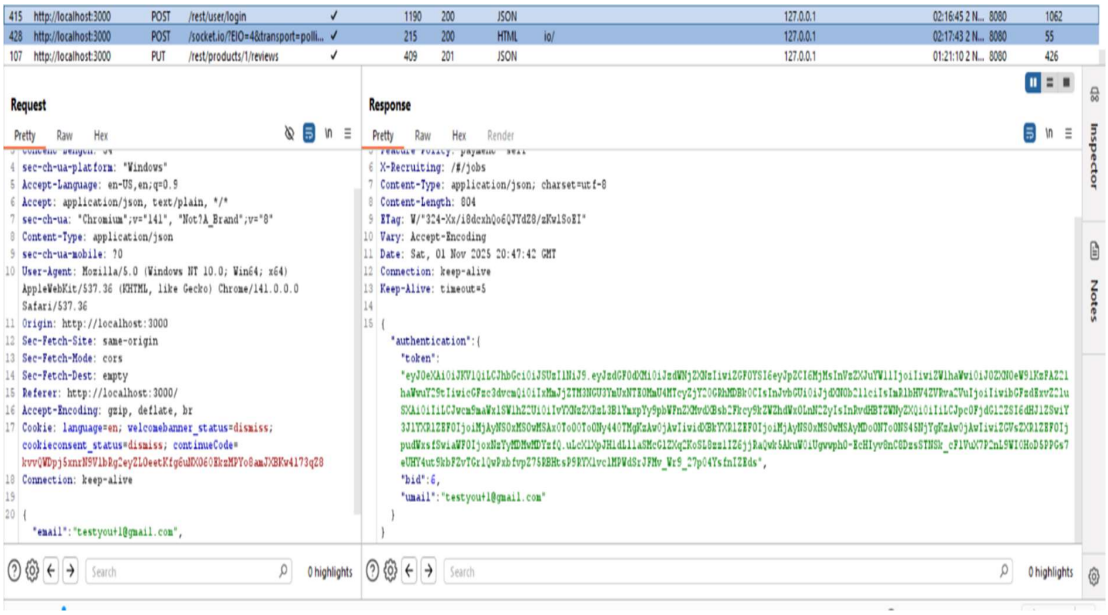
Description:
The application's JSON Web Tokens (JWTs) used weak algorithms such as MD5 for signing, which are susceptible to collision attacks. Tokens also contained hashed user credentials that can be cracked.

Security Impact:

- Forged or modified tokens may be accepted by the server.
- Attackers can gain unauthorized access or persist sessions indefinitely.
- High risk of account takeover and privilege escalation.

Risk Level: High

Evidence:



Mitigation Strategies:

- Use strong signing algorithms (e.g., HS256 or RS256).
- Include token expiration and implement key rotation.
- Avoid storing sensitive information inside JWTs.

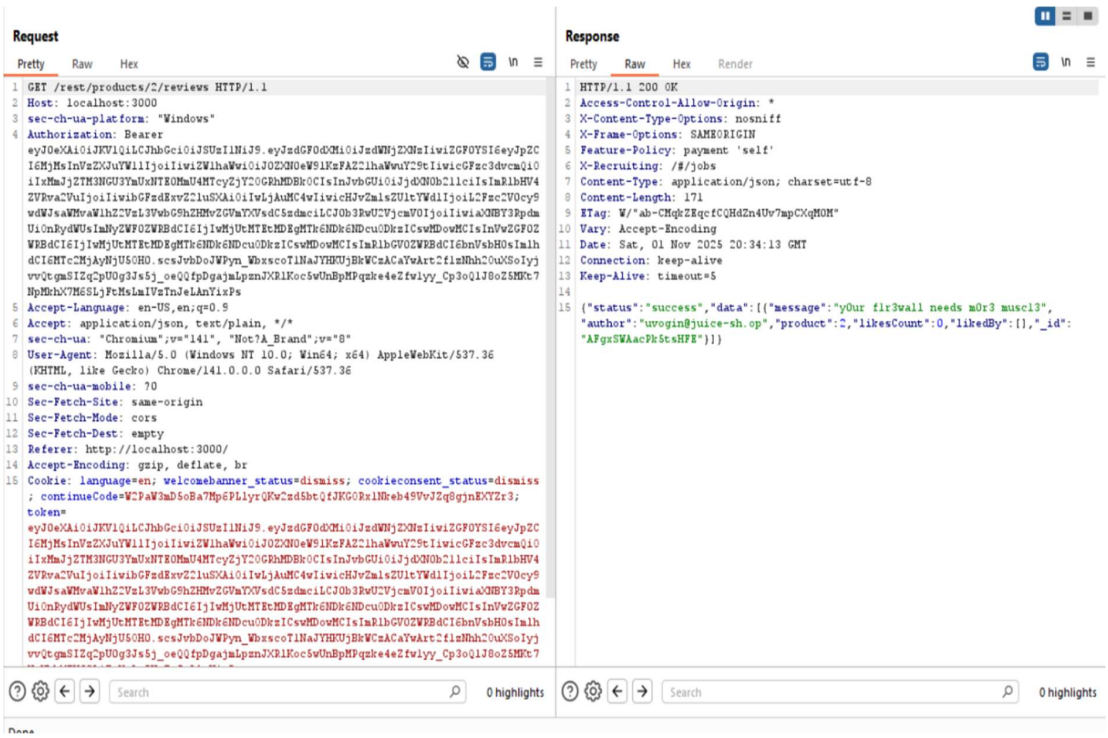
5. Insecure Direct Object Reference (IDOR)

Description:
Internal references like user IDs were exposed in URLs and API endpoints without verifying ownership. Modifying these IDs (e.g., userId=101 → userId=102) allowed access to other users' data.

Security Impact:

- Unauthorized access to other users' personal data.
- Data manipulation and privacy violations.
- Breach of confidentiality and data integrity.

Risk Level: High



Mitigation Strategies:

- Validate resource ownership server-side before processing requests.
- Apply "deny by default" access policies.
- Enforce authorization checks on every endpoint.

OWASP Top 10 (2021) Mapping

Detected Vulnerability	OWASP Category	Risk Rating
SQL Injection	A03 - Injection	High
Cross-Site Scripting (XSS)	A03 - Injection	High
Replay & Impersonation	A07 - Identification & Authentication Failures	Medium
Insecure Token Design	A02 - Cryptographic Failures	High
Insecure Direct Object Reference (IDOR)	A01 - Broken Access Control	High

Conclusion

The vulnerability assessment of OWASP Juice Shop using Docker and Burp Suite revealed multiple security weaknesses such as **SQL Injection**, **Cross-Site Scripting**, **Insecure Token Design**, **Replay & Impersonation**, and **Broken Access Control (IDOR)**.
These vulnerabilities can lead to **data breaches, account hijacking, and unauthorized access** if left unresolved.