



Web Application Security Assessment Report

Name: Odita Chukwudumebi Tobeckukwu

Task 1: Web Application Security Testing

Program: Future Interns Cybersecurity Internship

Date: September 2025

Target Application: OWASP Juice shop (Intentionally Vulnerable App for penetration testing)

Task Summary

This assessment was focused on detecting various loopholes and vulnerabilities in the OWASP Juice Shop application using Burp suite and manual evaluation. A good amount of vulnerabilities were detected, including SQL injection and user information leakage, which could lead to specific attacks and account compromise. These findings were mapped to the OWASP Top 10(2021) categories based on risk.

Tools Used:

- Kali Linux
- OWASP Juice Shop
- Burp Suite

Steps & Procedure

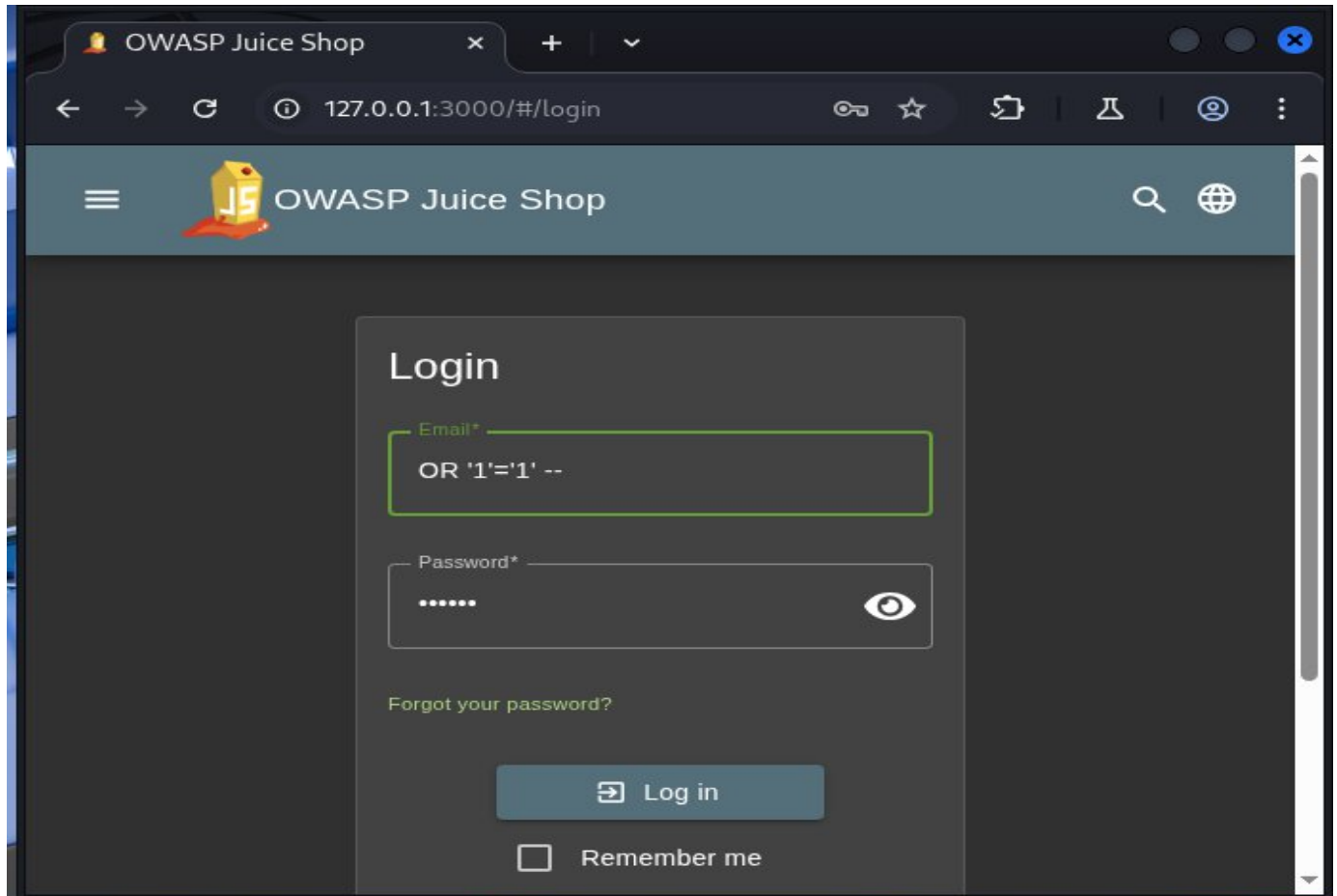
- The OWASP Juice Shop was deployed locally on a virtual machine (Kali Linux) with all required dependencies. Providing enough facilities for hands-on experience and testing allowing realistic penetration testing
- Burp Suite was used to perform manual scanning and testing, enabling the identification of web requests and response, user interactions and potential loopholes within the site
- The findings were carefully examined and aligned accurately with the OWASP top 10 framework. Subsequently, corrective measure were suggested to address the identify the security vulnerabilities

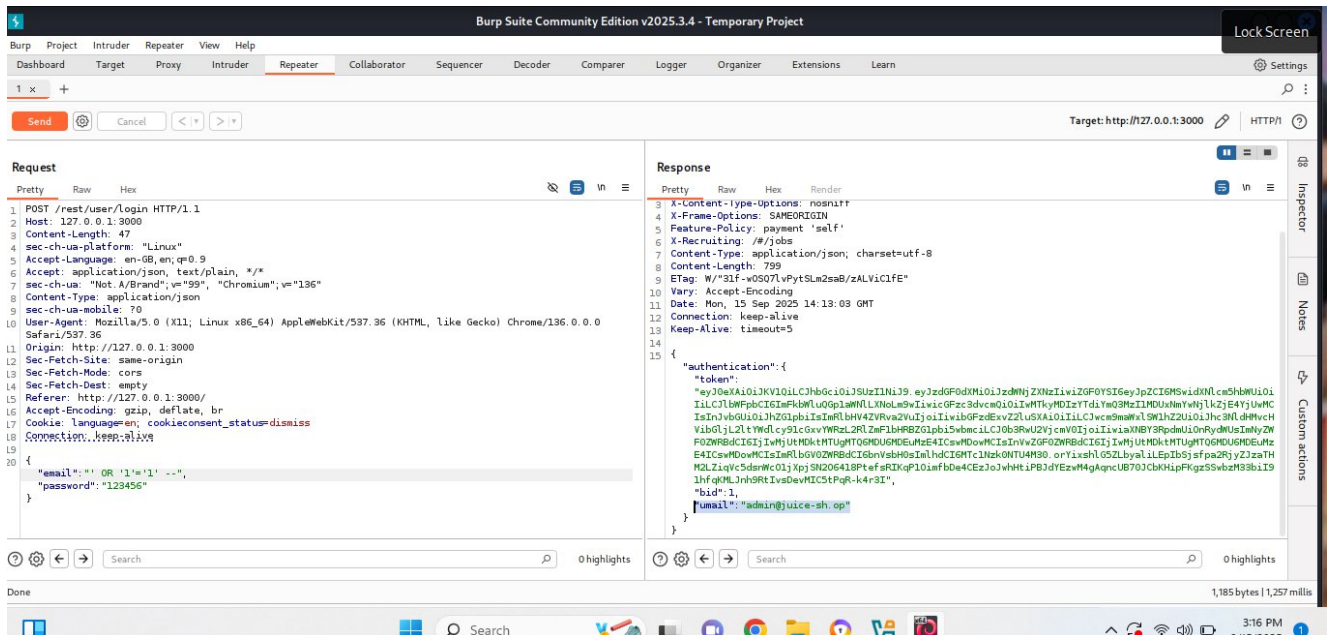
Identified Vulnerabilities

1. SQL Injection

The application accepts unsanitized user input in endpoints that construct SQL queries (e.g., search, product lookup, or authentication-related parameters) but in this case the login form was the test subject. Because input is concatenated into database queries instead of being parameterized, specially crafted inputs (for example ' OR '1'='1') alter the SQL logic. This allows attackers to manipulate queries to bypass authentication, retrieve unintended records, or cause the database to execute arbitrary SQL, potentially exposing sensitive data or allowing full database compromise. This is a very high-risked vulnerability.

Evidence:





Mitigation strategies:

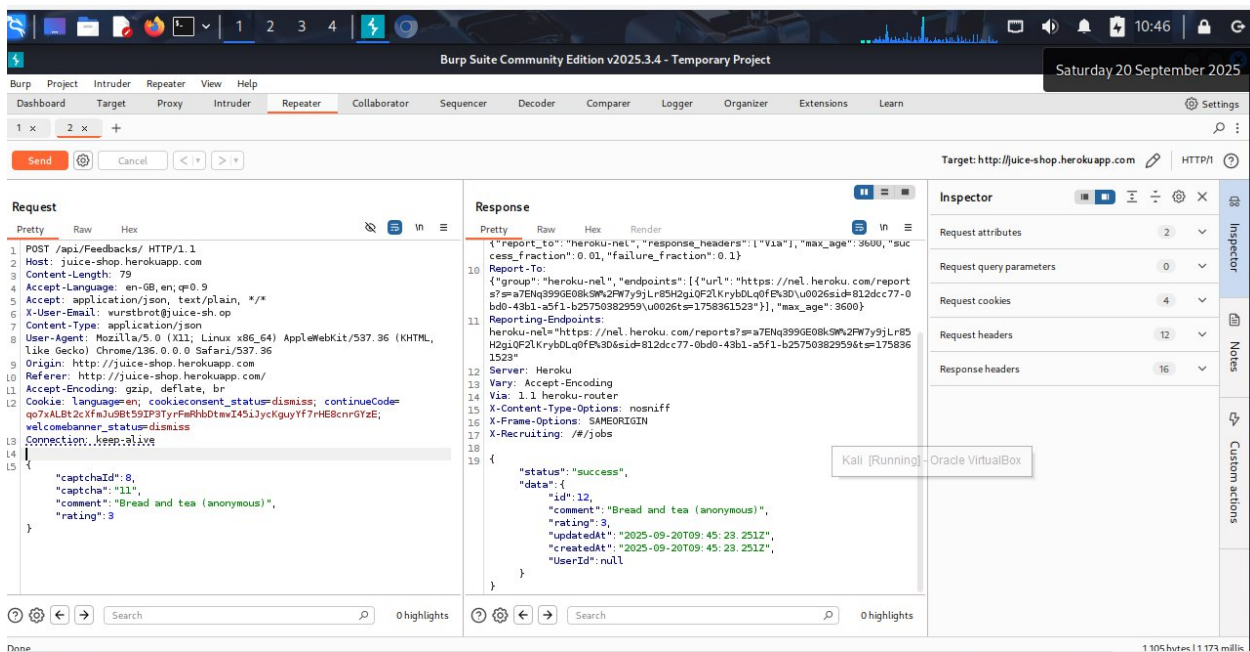
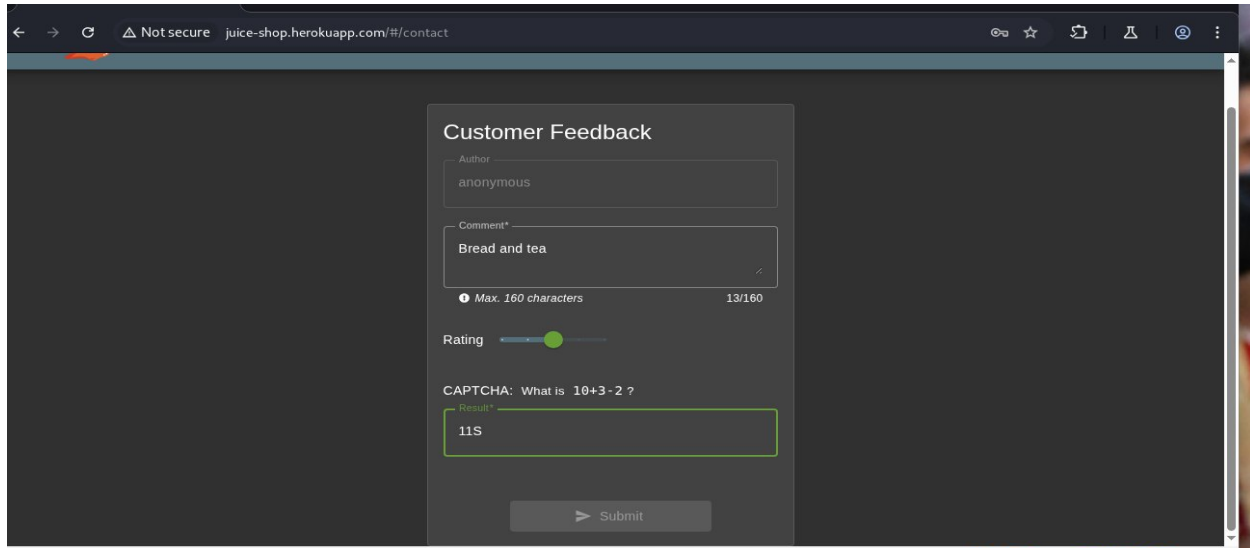
- Use parameterized queries or prepared statements across all database calls.
- Validate and sanitize user inputs against strict whitelists.
- Remove verbose database error messages from user responses.
- Apply the principle of least privilege to the database account used by the application.

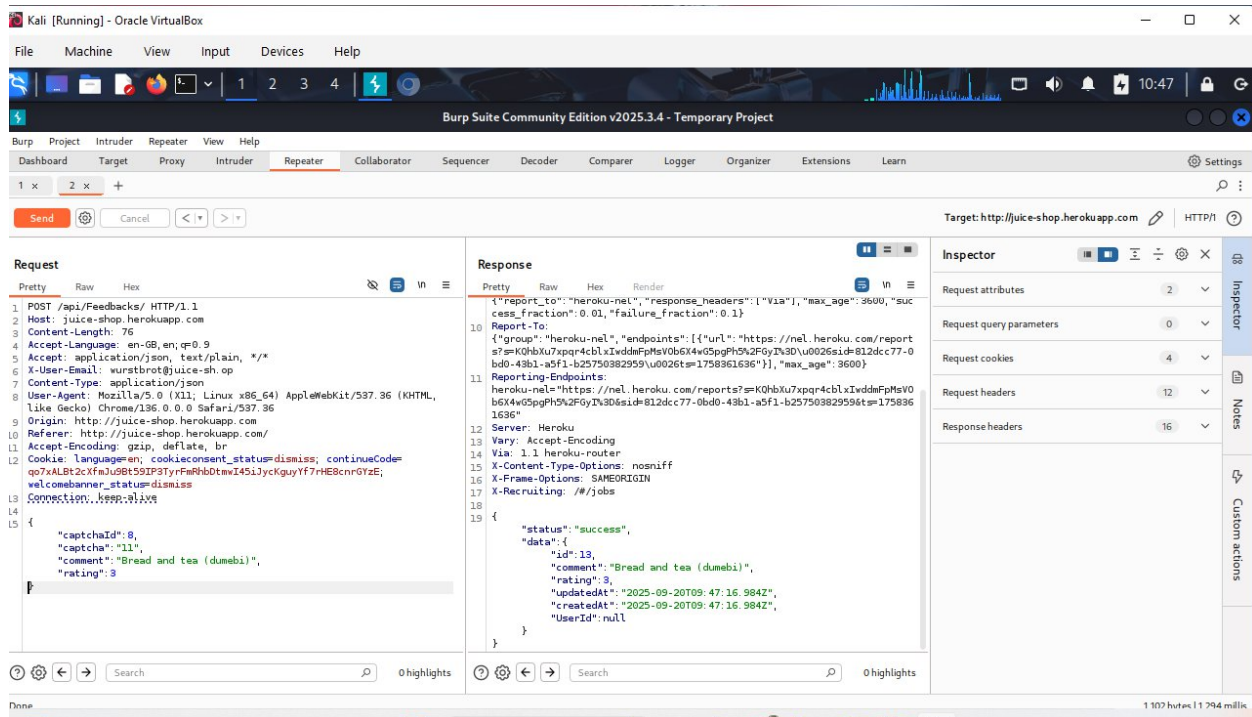
2. Replay and Impersonation

OWASP Juice Shop issues session tokens and request able actions without sufficient freshness checks. Captured authenticated requests (including session cookies or bearer tokens) can be replayed unchanged and still accepted by the server because tokens remain valid for extended periods and critical actions lack additional verification. This enables an attacker who obtains a token (via network capture, XSS, or local compromise) to impersonate the original user or replay prior requests to repeat privileged actions. In this case the feedback feature is vulnerable to replay attacks, as previously captured requests can be resent allowing duplicate

submissions without validation, enabling attackers submit another request under another user's account. This is a medium risk vulnerability.

Evidence:





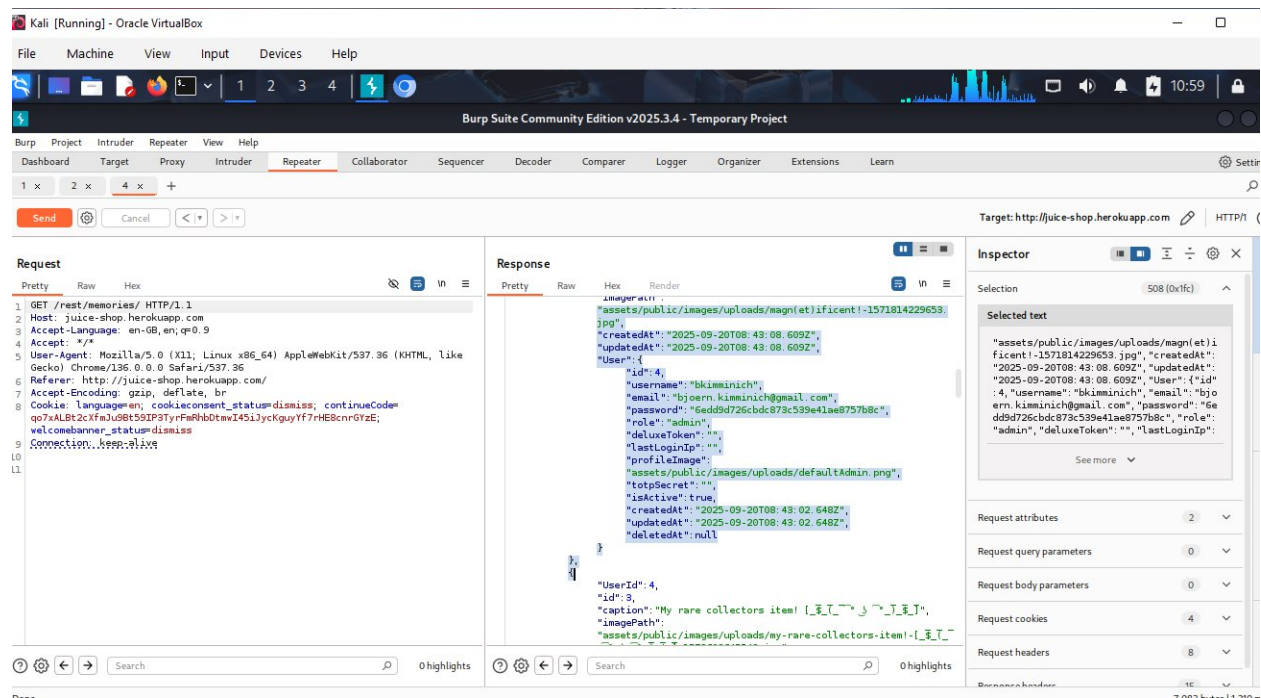
Mitigation Strategies:

- Implement one-time nonces or request IDs for sensitive actions.
- Enforce access control on feedback actions
- Deploy strict identity validation measures

3. User Enum and Information Leakage

Authentication and account management endpoints disclose whether a username/email exists through varying error messages, HTTP status codes, or response timing. This allows attackers to enumerate valid accounts by observing

system responses to crafted requests. In this case, the application photo wall section reveals user accounts through inconsistent responses and further exposes sensitive system details giving attackers both a target list and insights to the underlying environment. Bruce force attempts can be encouraged, credential compromise and potential exploitation of known vulnerabilities, leading to complete account takeover.



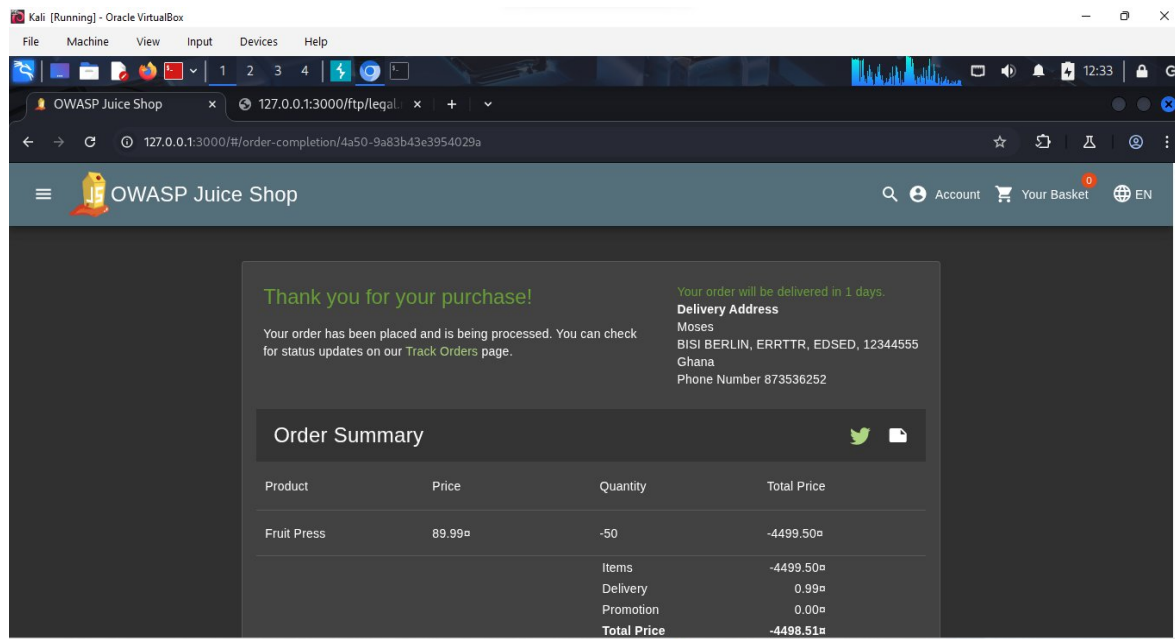
Mitigation Strategies

- Use generic error messages that do not disclose user existence (e.g., “Invalid credentials” for all login failures).
- Monitor and log suspicious login enumeration patterns.
- Use CAPTCHA or other anti-automation measures for high-volume attempts.

4. Business Logic Exploitation

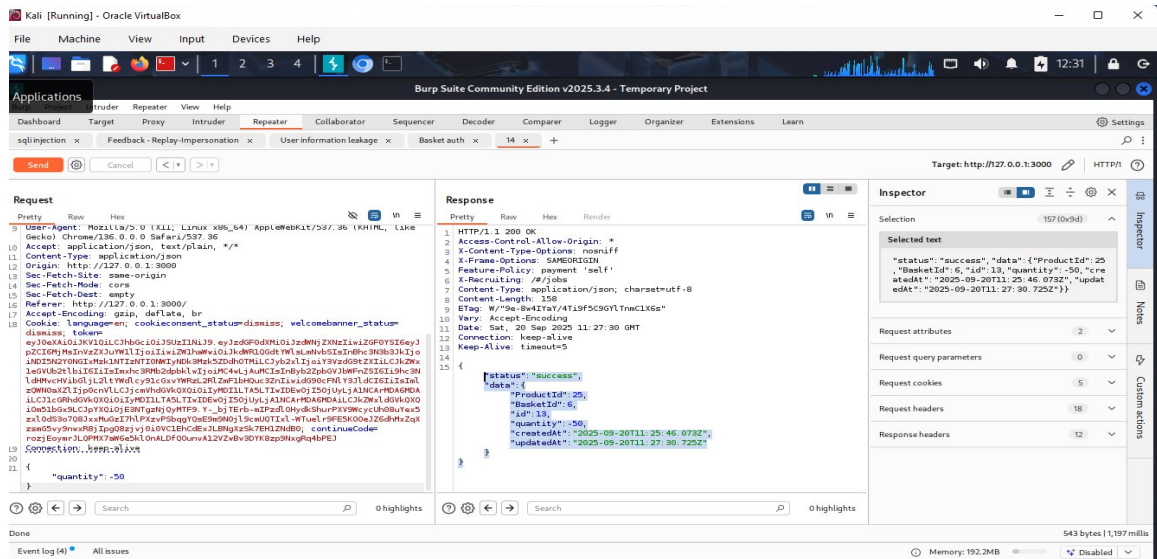
The payback/refund workflow relies on client-supplied parameters and lacks comprehensive server-side validation of transaction ownership, state, and idempotency. Requests that alter amounts, transaction IDs, or user references are processed without cross-checking the original payment record or enforcing single-use controls. This allows attackers to request refunds for other users, trigger duplicate paybacks by replaying requests, or manipulate amounts to fraudulently obtain funds. In this case the payment system in OWASP Juice Shop contains a business logical flaw that can be exploited to generate a negative and illegitimate credit. This vulnerability enables unauthorized credit manipulation, leading to financial loss and abuse of the payment system.

Evidence:



The screenshot shows a web browser window displaying the OWASP Juice Shop interface. The page title is "Thank you for your purchase!" and it includes a confirmation message: "Your order has been placed and is being processed. You can check for status updates on our [Track Orders](#) page." The delivery address is listed as "Moses, BISI BERLIN, ERRTR, EDESD, 12344555, Ghana, Phone Number 873536252". Below this, there is an "Order Summary" table.

Product	Price	Quantity	Total Price
Fruit Press	89.99	-50	-4499.50
Items			-4499.50
Delivery			0.99
Promotion			0.00
Total Price			-4498.51



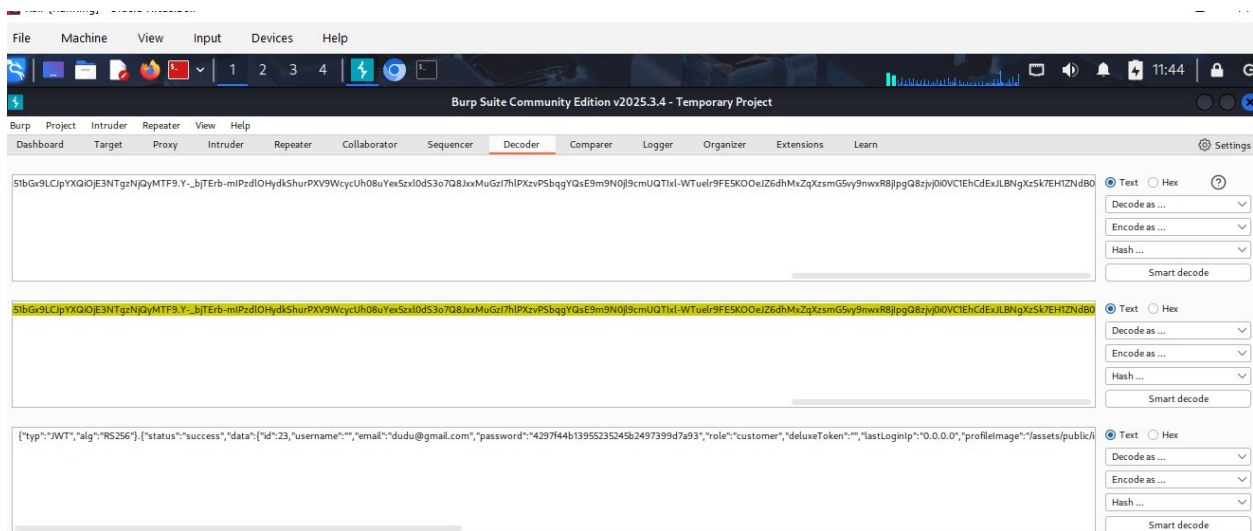
Mitigation Strategies:

- Apply role-based access controls and require additional authorization for high-risk transactions.
- Enforce idempotency: ensure the same request cannot be processed more than once.
- Perform all payback/refund validation server-side, checking transaction ownership and state.

5. Insecure Token Design

Tokens observed in the application (session identifiers, API keys, or JWTs) exhibit weak design traits: insufficient entropy, overly long validity, or improper signature handling. Tokens issued by the application (JWTs or session identifiers) show predictable patterns, weak signing algorithms, or lack critical attributes like expiration. Some tokens remain valid indefinitely, and modified tokens may still be accepted, exposing the application to forgery or token reuse. In this case OWASP Juice Shop's JSON Web Token (JWT) contains user credentials, hashed within MD5 which happens to be a weak and reversible algorithm. This design flaw exposes sensitive information within the token, allowing attackers crack user passwords and compromise accounts, Allowing account compromise. This is a very high risk vulnerability

Evidence:



Mitigation Strategies:

- Use strong cryptographic algorithms for token signing (e.g., HS256 with long secrets, RS256 with key pairs).
- Ensure tokens have sufficient entropy to resist guessing.
- Rotate signing keys periodically and revoke compromised tokens immediately.

6. Insecure Direct Object Reference (IDOR)

The application exposes internal object references (e.g., user IDs, basket IDs) directly in URLs or parameters without verifying ownership or access rights. Attackers can modify these identifiers in requests (e.g., changing `userId=101` to `userId=102`) to access or manipulate other users' data. This vulnerability grants unauthorized access to sensitive user data.

[illegible]

- Validate ownership of requested resources before returning data.
- Enforce strict server-side authorization checks for every request.
- Apply “deny by default” principles for all sensitive resources.

OWASP Top 10 Mapping (2021)

Detected Vulnerability	OWASP TOP 10	Risk Rating
SQL injection in Login form	A03 - Injection	High
Insecure Token Design(JWT with MD5 credentials)	A02 – Cryptographic Failures	High
Insecure Direct Object Reference	A01 – Broken Access Control	High
User Enum & Information Leakage	A05 – Security Misconfiguration/ A02 – Cryptographic Failures	Medium
Replay & Impersonation in feedback feature	A01 – Broken Access Control	Medium
Business Logical Exploitation	A04 – Insecure Design	Medium

CONCLUSION

The assessment of OWASP Juice Shop revealed critical vulnerabilities including SQL Injection, Replay and Impersonation, User Enumeration, Insecure Token Design, IDOR, and Business Logic Exploitation. These flaws exist due to poor input validation, weak access controls, insecure session management, and insufficient server-side checks. If left unresolved, they expose the application to data breaches, account compromise, and financial fraud. Implementing the recommended mitigations and aligning with OWASP best practices will greatly enhance the application's security posture.