WEB APPLICATION SECURITY ASSESSMENT REPORT

NAME: AFOLABI ADESOYE THEOPHILUS

Task 01: Web Application Security Assessment

Target: OWASP Juice Shop (local demo)

Program: Future Interns Cybersecurity Internship

DATE: 10/20/2025

Task Summary

This report documents the findings of a manual security assessment conducted on **OWASP Juice Shop**. The testing focused on identifying common web application vulnerabilities through manual analysis using Burp Suite. The vulnerabilities discovered include **SQL Injection**, **Insecure Direct Object Reference (IDOR)**, **Cross-Site Scripting (XSS)**, and a **Replay Vulnerability** in the feedback submission feature. Each identified issue has been mapped to the OWASP Top 10 (2021) categories and is accompanied by supporting technical evidence, detailed impact analysis, and recommended remediation strategies to help strengthen the application's overall security posture.

Tools Utilized

- Kali Linux – Penetration testing environment
- OWASP Juice Shop – Intentionally vulnerable web application
- Burp Suite – Web vulnerability scanner and proxy tool

Procedure Overview

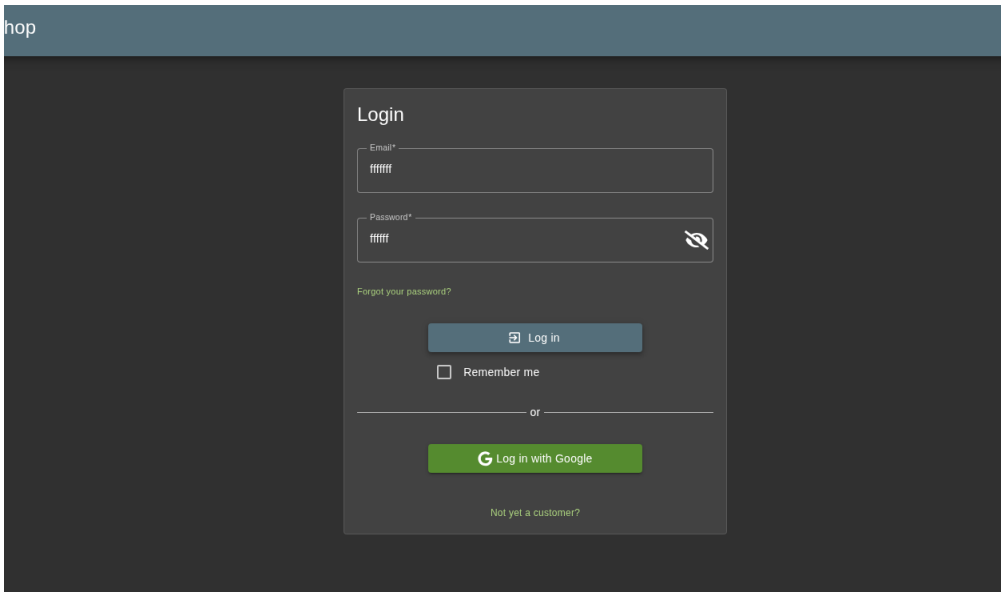
- The OWASP Juice Shop application was set up on a Linux virtual machine, with all required dependencies (Node.js and npm) installed through the command line.
- This local deployment served as the target environment for both manual and automated security testing activities.
- Burp Suite was utilized to intercept HTTP traffic and conduct active scans to detect possible input points and vulnerabilities.
- Each identified issue was manually verified and classified based on the OWASP Top 10 (2021) standards.
- Appropriate remediation measures were recommended for every vulnerability to strengthen the overall security of the application.

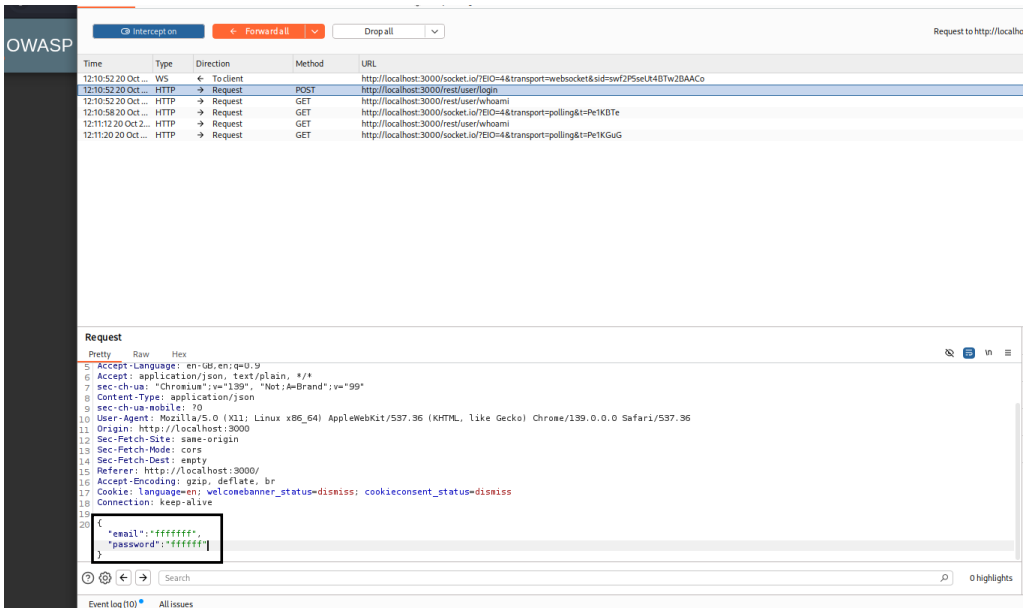
- **IDENTIFIED VULNERABILITIES IN OWASP JUICE SHOP**

Finding 1 – SQL Injection (boolean-based SQL injection used for login bypass)

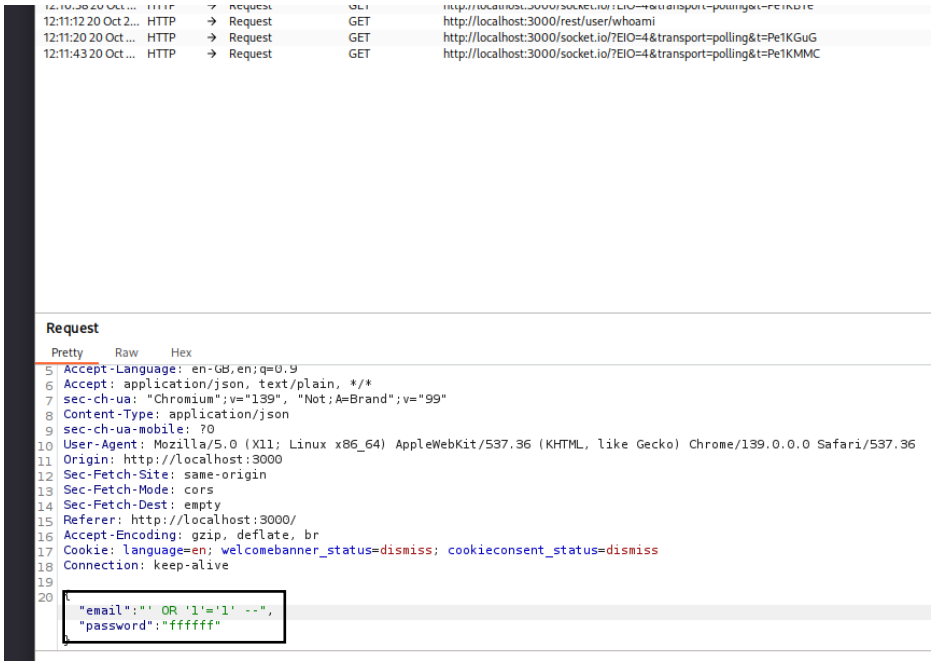
During the assessment of the login feature on OWASP Juice Shop, it was discovered that the application is vulnerable to SQL injection:

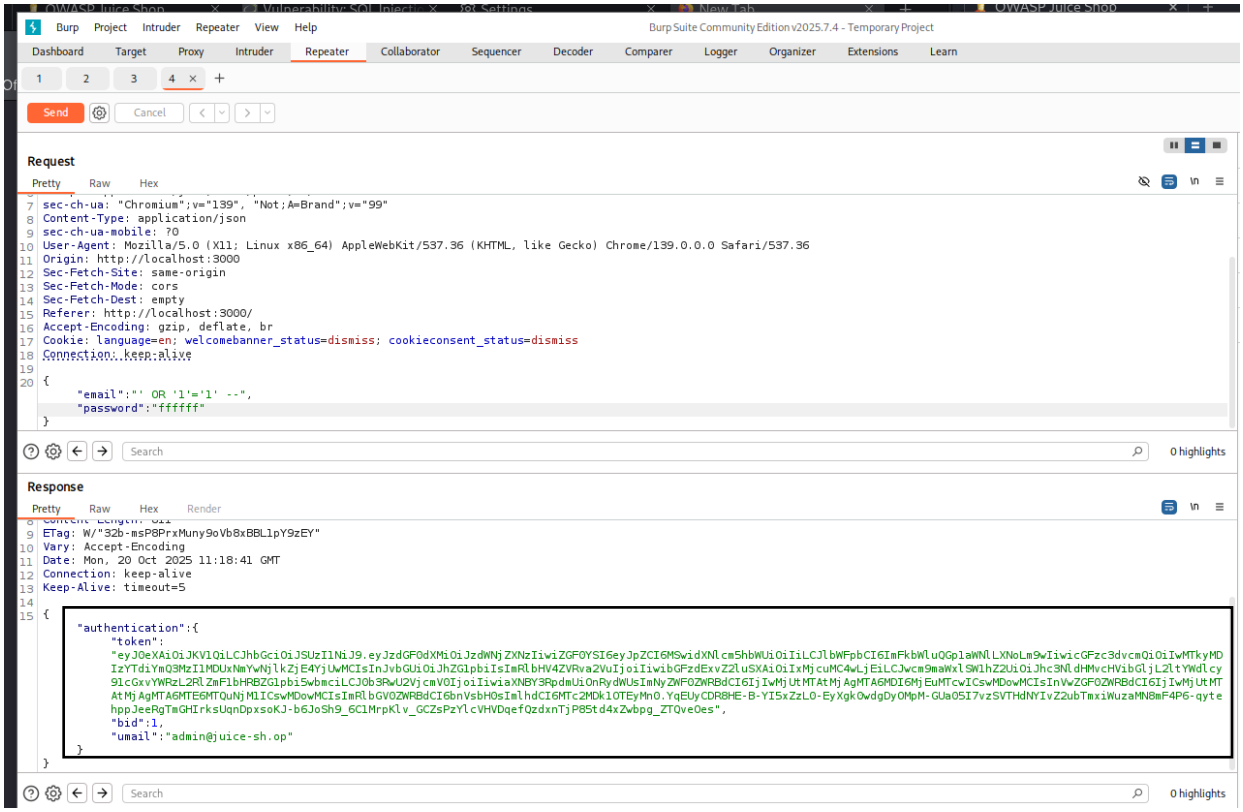
I first submitted wrong credentials to the login page so the POST request appeared in **Burp Suite's Proxy** (Intercept ON),



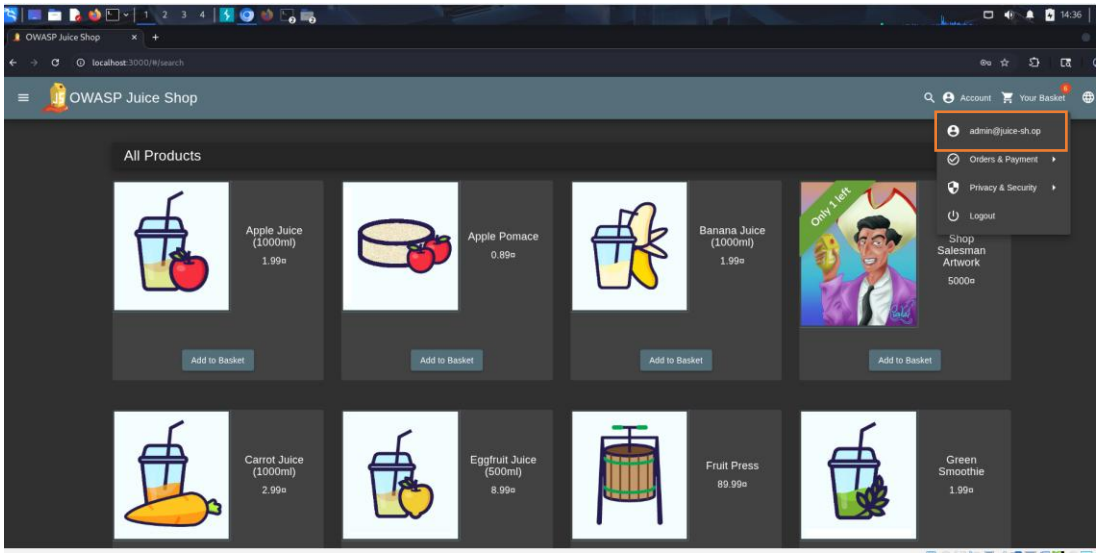


then I edited the intercepted password value to the payload (**' OR '1'='1' –)** and sent the modified request to Repeater for testing;





the payload makes the SQL WHERE clause always true ('1'='1') and -- comments out the rest of the query, allowing authentication bypass and resulting in an authenticated response (redirect/Set-Cookie). This is a tautology / boolean-based SQL injection used for login bypass.



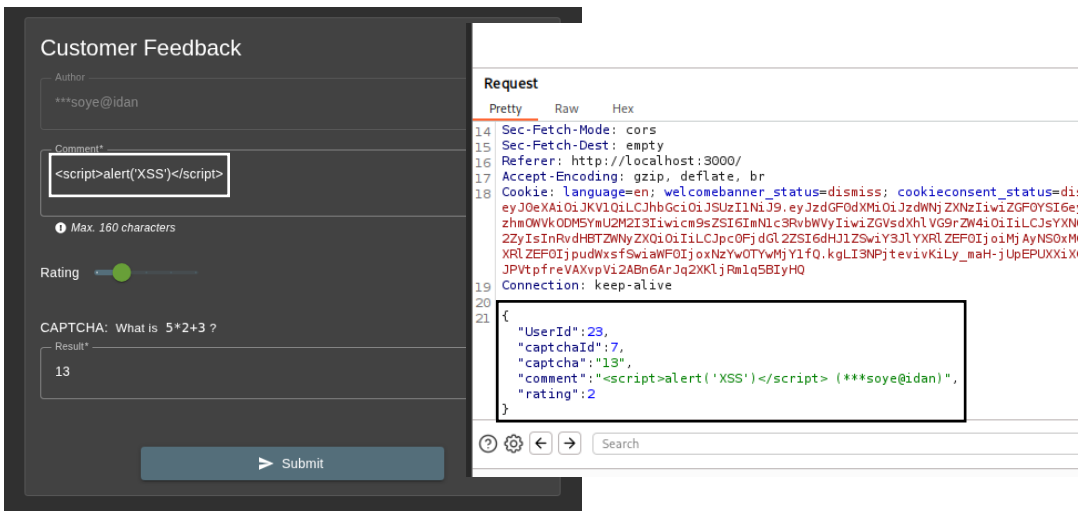
Severity: High (often Critical for authentication-bypass SQLi)

Impact: an attacker can log in without a password, take over user accounts (possibly admins), view or steal private data (emails, orders, etc.), change or delete information, and bypass any checks that rely on normal login, all of which can lead to service disruption, data breaches, and real-world consequences for users and the business.

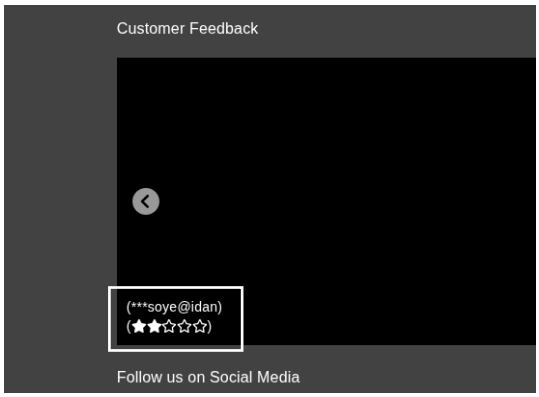
Remediation: use parameterized queries or prepared statements, avoid concatenating user input into SQL, hide DB error details, and enforce server-side authentication checks.

Finding 2 – XSS (Cross-Site Scripting)

The feedback form reads user-supplied input and inserts it into the page DOM without proper sanitization or output encoding, so attacker-controlled text is interpreted by the browser and executed as script, a classic DOM-based XSS where the payload never needs to be returned by the server.



During testing I submitted the payload (**<script>alert('XSS')</script>**) in the Juice Shop feedback form; the application accepted and executed it in the browser (an alert appeared), confirming a DOM-based XSS vulnerability, the payload was handled client-side and ran without being returned by the server.



Alert box triggered after visiting the About Us page, confirming script execution

Severity: High

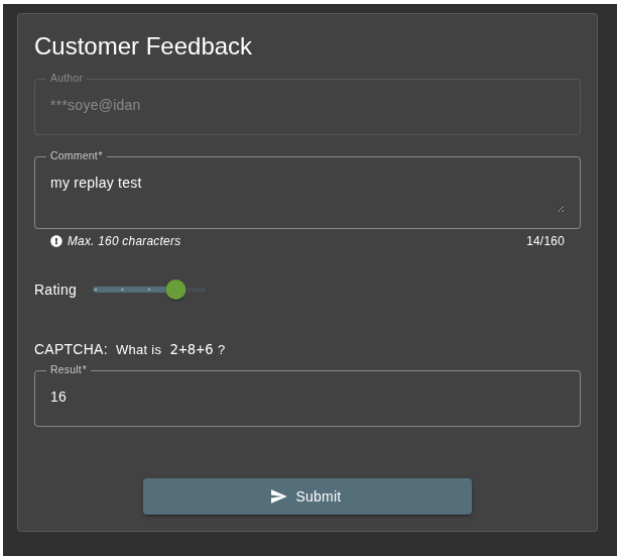
Impact: An attacker can run arbitrary JavaScript in any victim's browser who views the affected page or clicks a crafted link. That can lead to session cookie/token theft, account takeover, performing actions as the user (CSRF-like), stealing or modifying displayed data, installing malicious scripts, and phishing-style UI spoofing, all of which can cause user harm, data breaches, and reputational/legal fallout for the service.

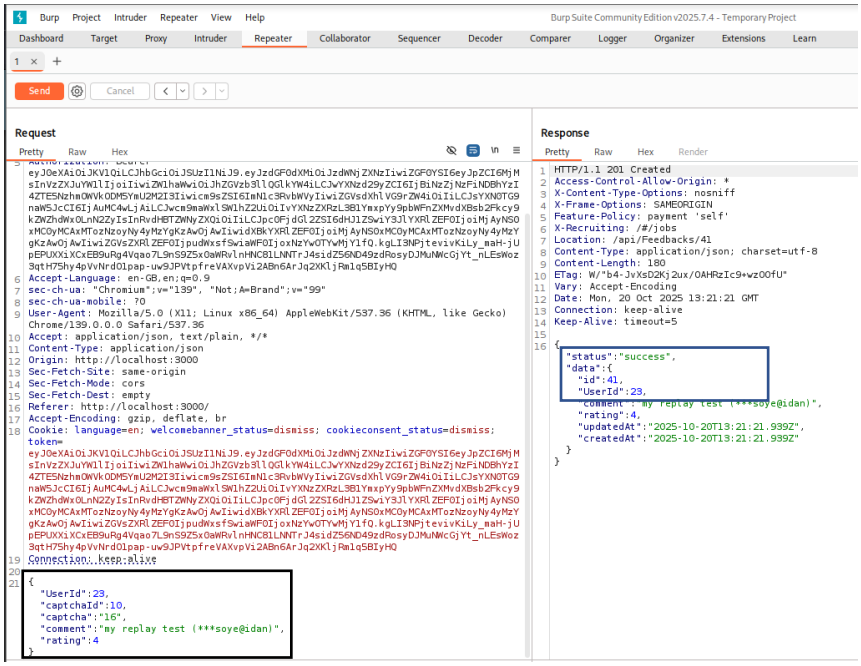
Remediation:

- **Replace unsafe sinks**: stop using innerHTML, document.write, eval, and similar APIs with untrusted data. Use textContent, innerText, or setAttribute for inserting plain text.
- **Sanitize if HTML is required**: use a vetted sanitizer like DOMPurify to clean any HTML before insertion.
- **Validate & canonicalize inputs**: never trust URL fragments or client-side inputs; validate format and strip dangerous characters where possible.

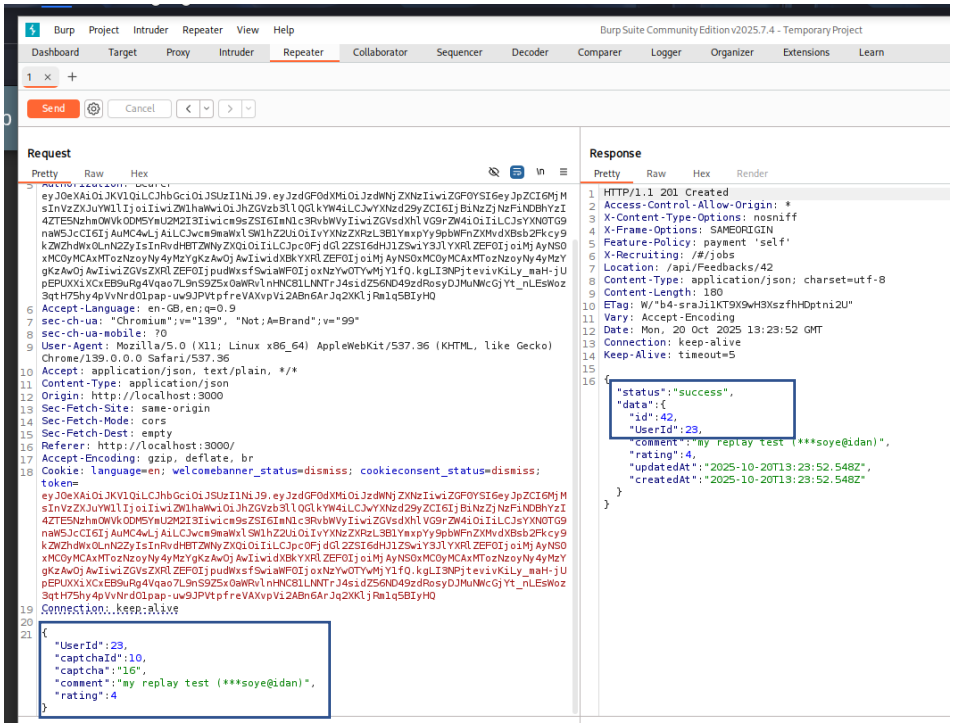
Finding 3 – Replay Vulnerability in Feedback Submission

During testing, OWASP Juice Shop was found to be vulnerable to a replay vulnerability: the server accepts the same feedback POST (or a slightly modified copy) multiple times without detecting duplicates or validating freshness/uniqueness, allowing an attacker who captures a request to replay it and perform the same action repeatedly (spam, duplicate records, or other undesired writes).





Original submission: POST /api/Feedbacks returned 201 Created and stored the feedback as **id: 41**.



Replayed submission: the **same request** replayed returned **201 Created** and stored a new record **id: 42**, proving the server accepted a duplicate submission (no idempotency/duplicate check)

The feedback endpoint accepted repeated identical requests: I replayed a captured POST in Burp Repeater and received 201 Created responses for each replay, with the server creating new feedback entries (different id values). This confirms a replay vulnerability.

Severity: Medium

Impact: Attackers can repeat actions (spam submissions, duplicate records, unwanted state changes), amplify impact if the endpoint triggers side effects (emails, workflows, transactions)

Remediation:

- **Idempotency keys / unique request IDs:** require clients to send a unique X-Request-ID and reject duplicates server-side.
- **One-time nonces / CSRF tokens:** ensure tokens are single-use and tied to a session.
- **Rate limiting / throttling:** per IP/user to reduce automated replay attempts.

Finding 4 – Insecure Direct Object Reference (IDOR)

During testing an IDOR was discovered: by changing the object ID in requests (for example GET /api/BasketItems/{id} I was able to access other users' baskets and orders (different userId/owner data) without proper server-side authorization, demonstrating missing ownership checks.

Severity: High

Impact: This vulnerability allows an attacker to view or manipulate other users' baskets and orders by simply changing an ID value in the request. As a result, attackers could access sensitive information, modify or delete other users' data, place or cancel orders on their behalf, and potentially escalate privileges or disrupt business operations. This leads to loss of data integrity, privacy violations, and reputational or financial damage.

Remediation:

- Implement server-side ownership checks to verify that the authenticated user is authorized to access or modify the requested resource
- Enforce role-based access control (RBAC) and centralized authorization logic.
- Validate and sanitize all client-supplied identifiers server-side.

Vulnerability	OWASP Category (2021)	Impact	Severity
SQL Injection	A03:2021 – Injection	Authentication bypass and unauthorized access	High
Cross-Site Scripting (XSS)	A07:2021 – Identification & Authentication Failures	Arbitrary script execution in browser	High
Replay Vulnerability	A01:2021 – Broken Access Control	Duplicate feedback accepted without validation	High
Insecure Direct Object Reference (IDOR)	A02:2021 – Cryptographic Failures	Unauthorized access to other users' data	Medium

Vulnerability Summary Table Mapping to OWASP Top 10