SECURITY ASSESSMENT REPORT — TASK 1

Cyber Security Internship at Future Interns

Intern: Shahana Parveen

CIN ID: FIT/SEP25/CS4100

Date: 02 October 2025

Table of Contents

1. Introduction

This report documents the results of Task 1 of my Cyber Security Internship at Future Interns. The goal was to perform a Web Application Security Assessment of a deliberately vulnerable application (OWASP Juice Shop), identify common security issues, and document them according to OWASP standards.

The assessment was performed using industry tools such as OWASP ZAP and Burp Suite to simulate real-world penetration testing scenarios.

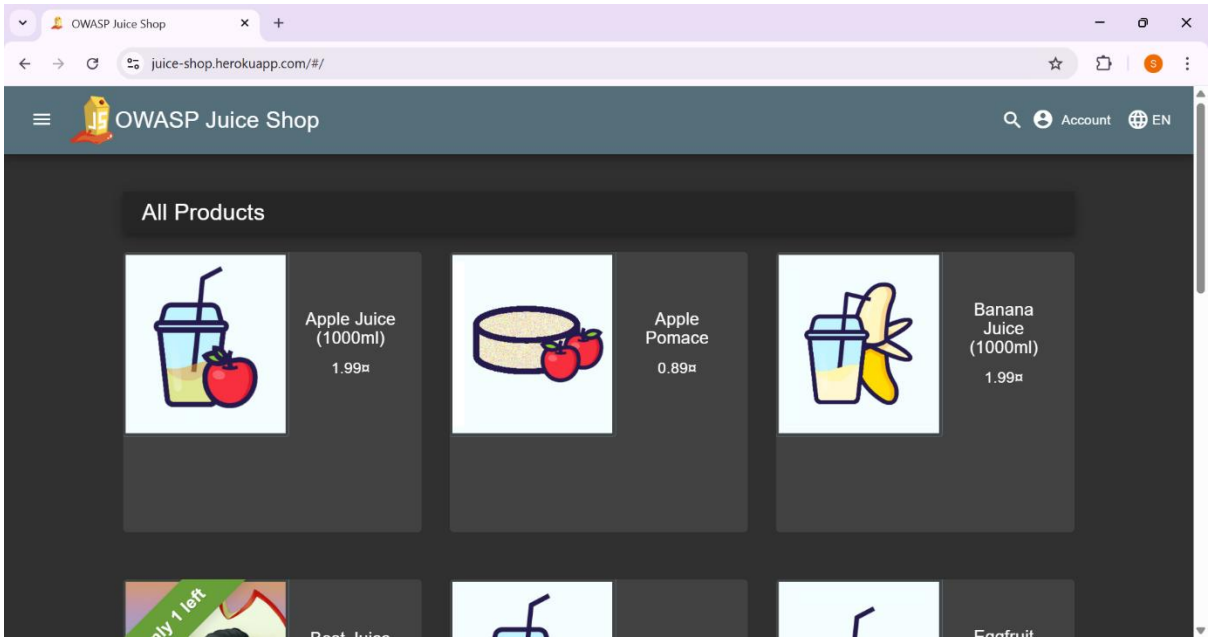

Figure - 01_app_home.png — OWASP Juice Shop home page used as the test target.

2. Methodology

The following methodology was followed during the assessment:

- **Setup:** Deployed OWASP Juice Shop (deliberately vulnerable web app).
- **Scanning:** Performed an automated vulnerability scan using OWASP ZAP.
- **Manual Testing:** Intercepted traffic with Burp Suite and tested parameters manually.
- **Vulnerabilities Tested:** SQL Injection, Cross-Site Scripting (XSS), and other OWASP Top 10 flaws.
- **Documentation:** Collected logs, screenshots, and wrote findings in this report.

3. Tools Used

- OWASP ZAP (automated vulnerability scanner)
- Burp Suite (manual web testing toolkit)
- SQLMap (optional SQLi testing)
- OWASP Juice Shop (test environment)
- Windows 11 + Firefox (testing environment)

4. Findings

4.1 SQL Injection (High) — /complete/search

Description:

The `term` parameter in `/complete/search` is not properly sanitized. A crafted input (`' OR '1'='1`) altered the server response, indicating a SQL injection vulnerability.

Evidence:

- Request (Burp): GET /complete/search?client=firefox&term=%27%20OR%20%271%27%3D%271

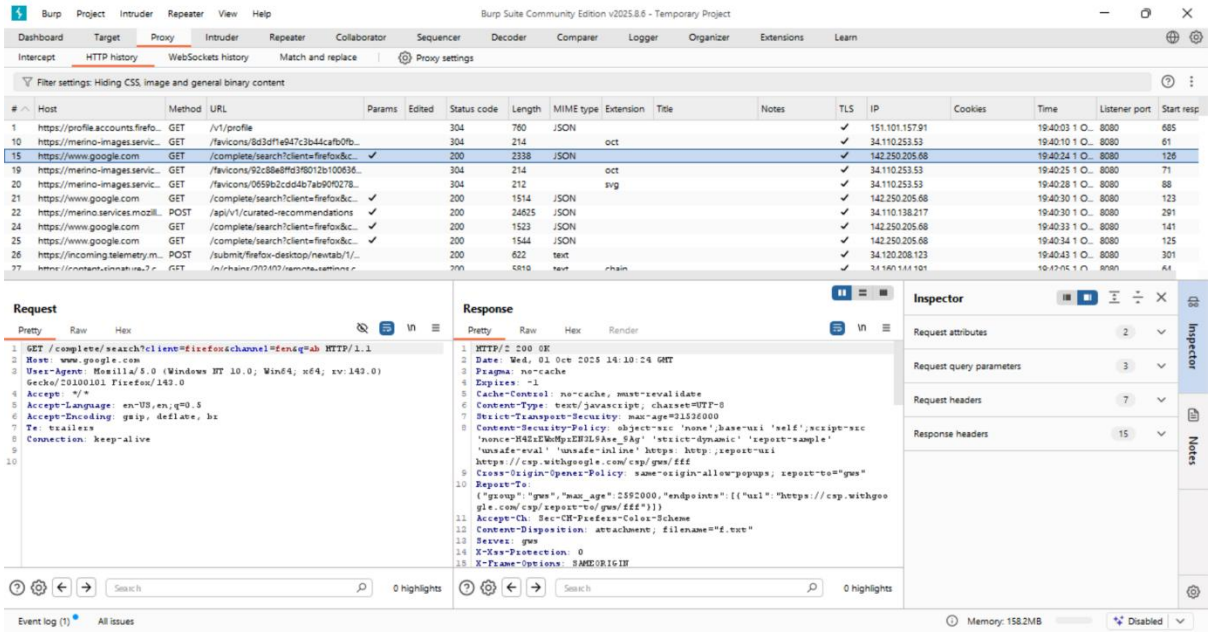

Figure — 03_burp_request.png — Burp Suite HTTP history showing the SQL injection payload in the request

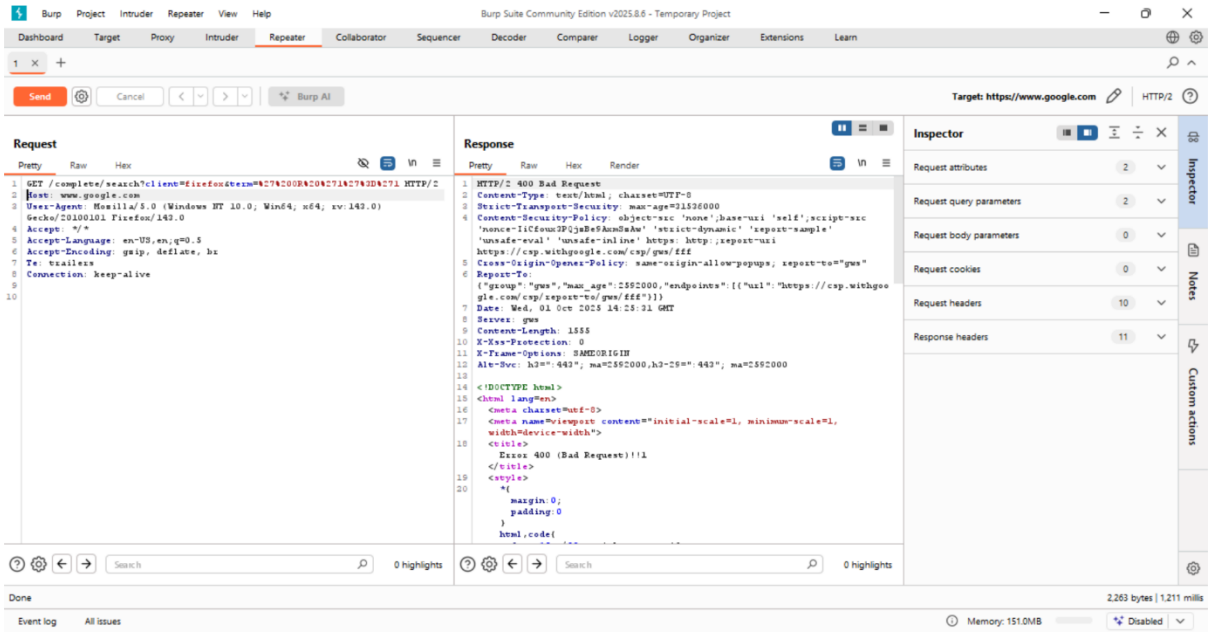

Figure — 04_sql_injection.png — Burp response showing altered output confirming SQL injection behavior

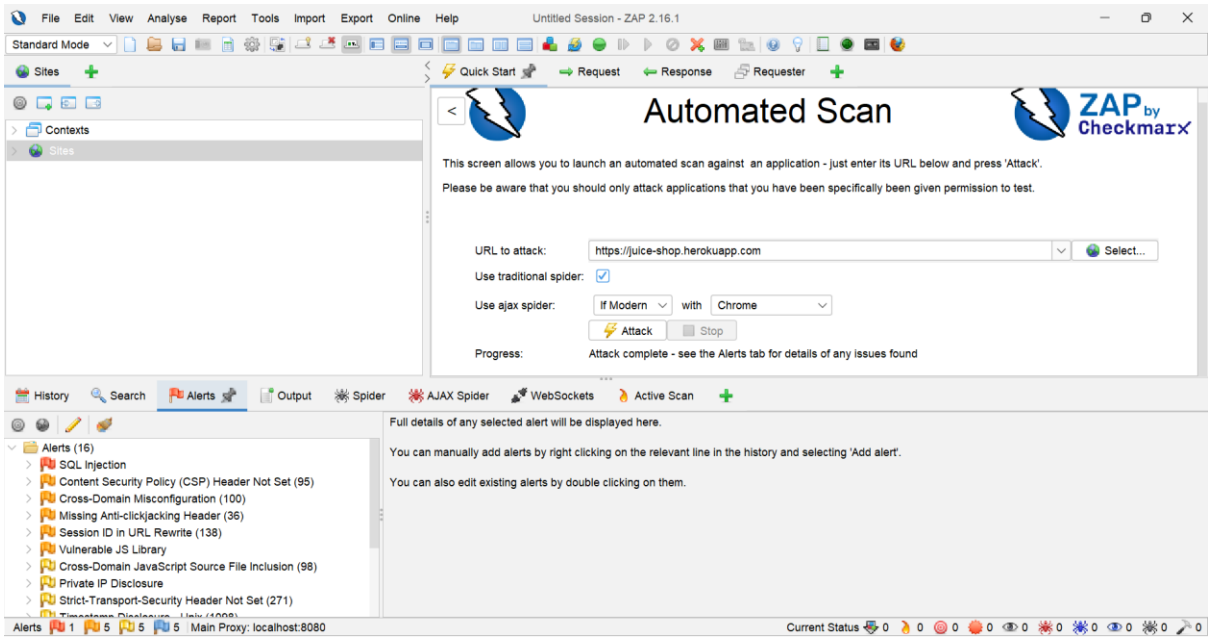

Figure — 02_zap_findings.png — OWASP ZAP automated alert summary highlighting SQL Injection (supporting manual proof)

Impact:

An attacker can read or modify database data, bypass authentication, or escalate privileges.

Risk Rating: HIGH

Recommendation:

1. Use parameterized queries / prepared statements.

2. Validate and sanitize inputs server-side (whitelist).

3. Apply least-privilege to DB accounts.

4. Add WAF rules and re-scan after fixing.

Verification:

Re-run SQLMap and ZAP; confirm no altered results and ZAP alert cleared.

4.2 Broken Authentication (High) — Login Functionality

Description:

The application allows login with weak/default credentials (example: admin:admin), indicating insufficient authentication controls.

Evidence:

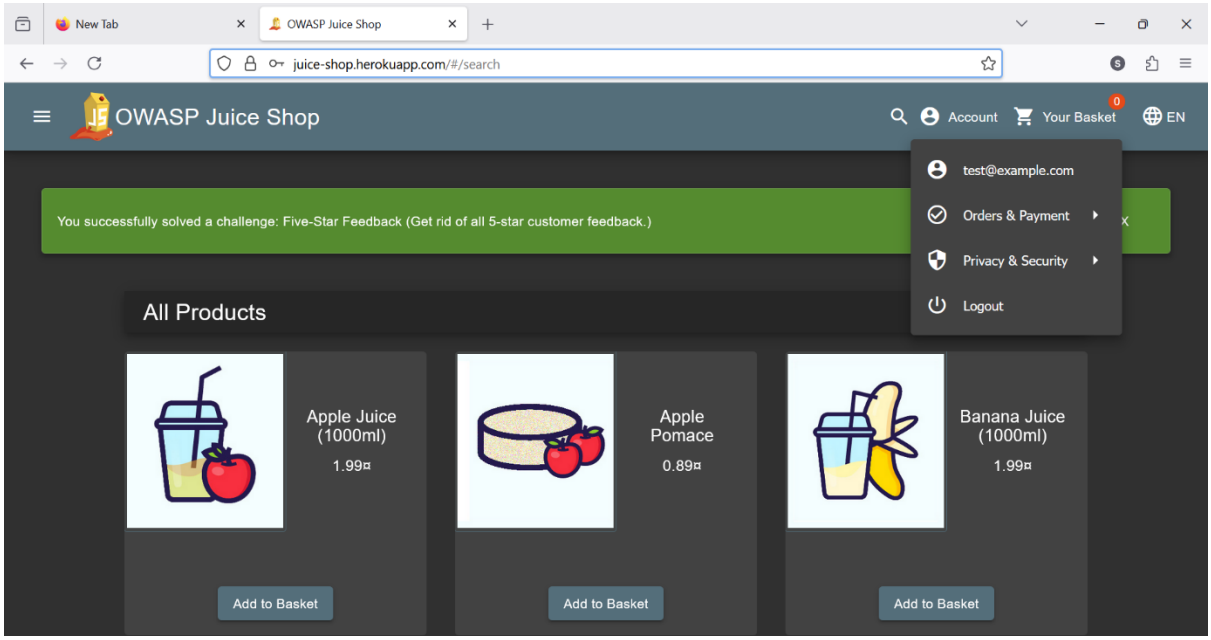

Figure — 05_auth_success.png — Application screen showing successful login using weak/default credentials (proof of broken authentication)

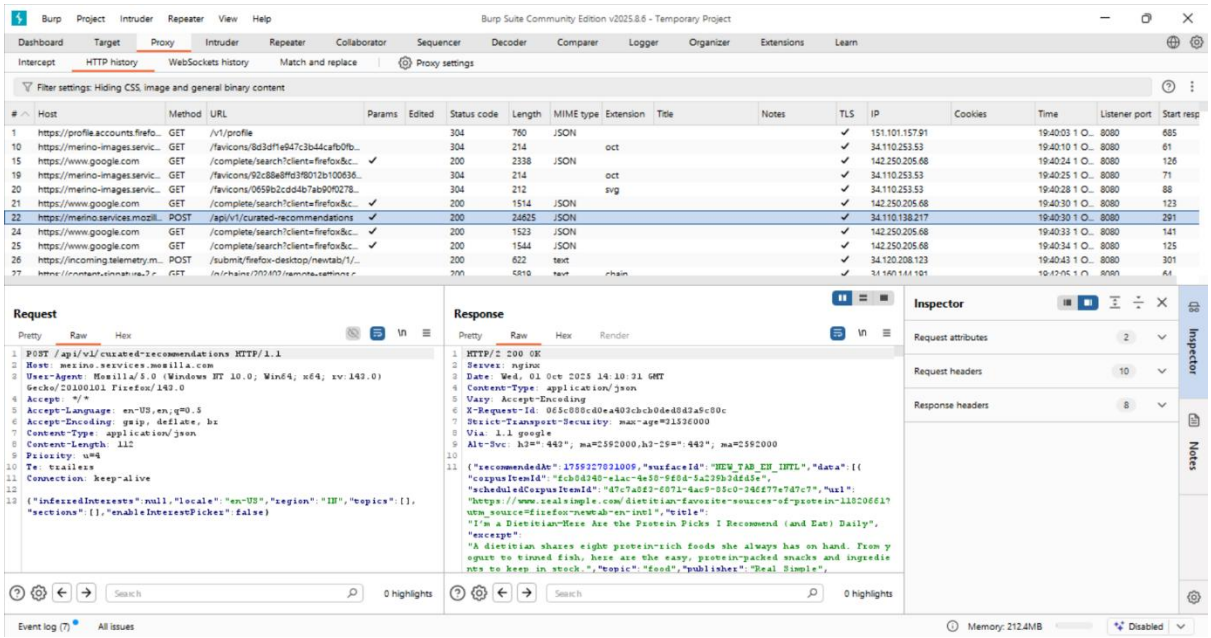

Figure — 06_auth_request.png — Burp request/response showing credentials sent and successful authentication response

Impact:

An attacker with default/weak credentials may gain administrative access and compromise the application.

Risk Rating: HIGH

Recommendation:

1. Enforce strong password policy and disallow default credentials.

2. Implement account lockout or exponential backoff after multiple failed attempts.

3. Offer or require multi-factor authentication (MFA) for privileged accounts.

4. Use secure password storage (bcrypt, Argon2).

Verification:

Confirm default credentials are rejected and lockout/MFA works.

4.3 Insecure Cookies (Medium) — Session Cookie

Description:

Session cookies are issued without HttpOnly and Secure flags, making them accessible to JavaScript and potentially exposed over insecure transport.

Evidence:

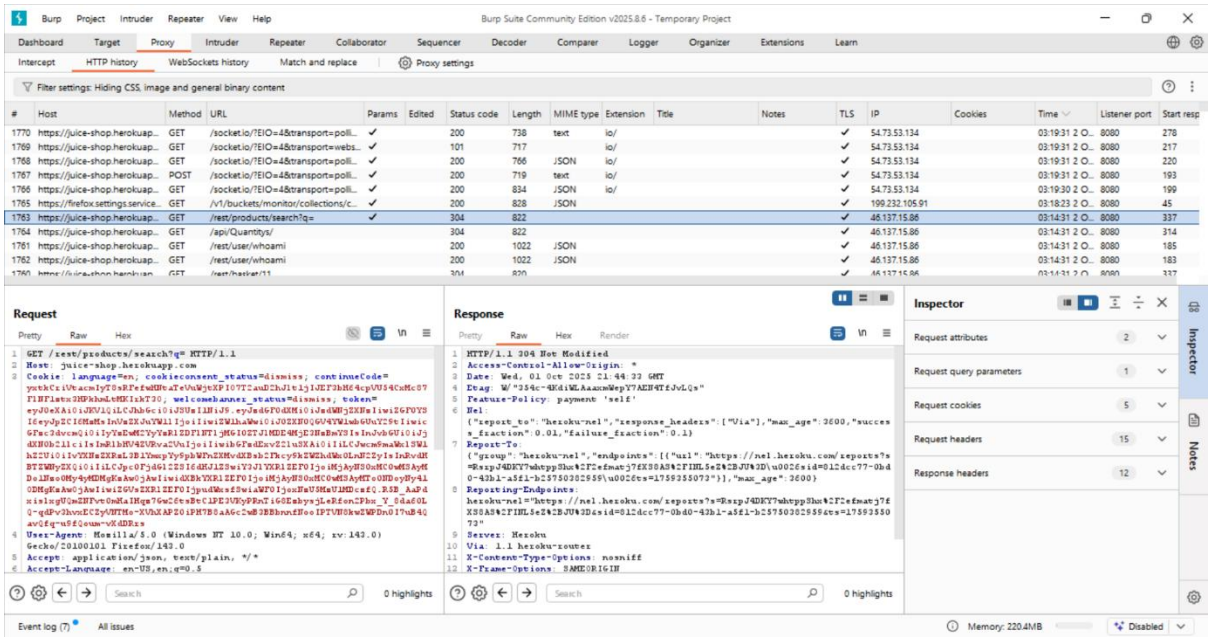

Figure — 07_insecure_cookie.png — Burp Suite Request showing cookies transmitted without Secure and HttpOnly flags (evidence of insecure cookie handling)

Impact:

Cookies may be stolen through XSS or intercepted over HTTP, leading to session hijacking.

Risk Rating: MEDIUM

Recommendation:

1. Set HttpOnly and Secure on session cookies.

2. Add SameSite=Strict or Lax.

3. Rotate session IDs on authentication and privilege changes.

Verification:

Confirm response headers: Set-Cookie: token=...; Path=/; HttpOnly; Secure; SameSite=Strict.

4.4 Missing Security Headers — /rest/products/search

Description:

The application's HTTP response (see Response → Raw) contains some security headers (e.g., X-Content-Type-Options and X-Frame-Options) but lacks several recommended headers such as Content-Security-Policy (CSP), Strict-Transport-Security (HSTS), and Referrer-Policy. The absence of these headers increases risk of XSS, clickjacking, and other content-injection attacks.

Evidence:

- Request: GET /rest/products/search?q=

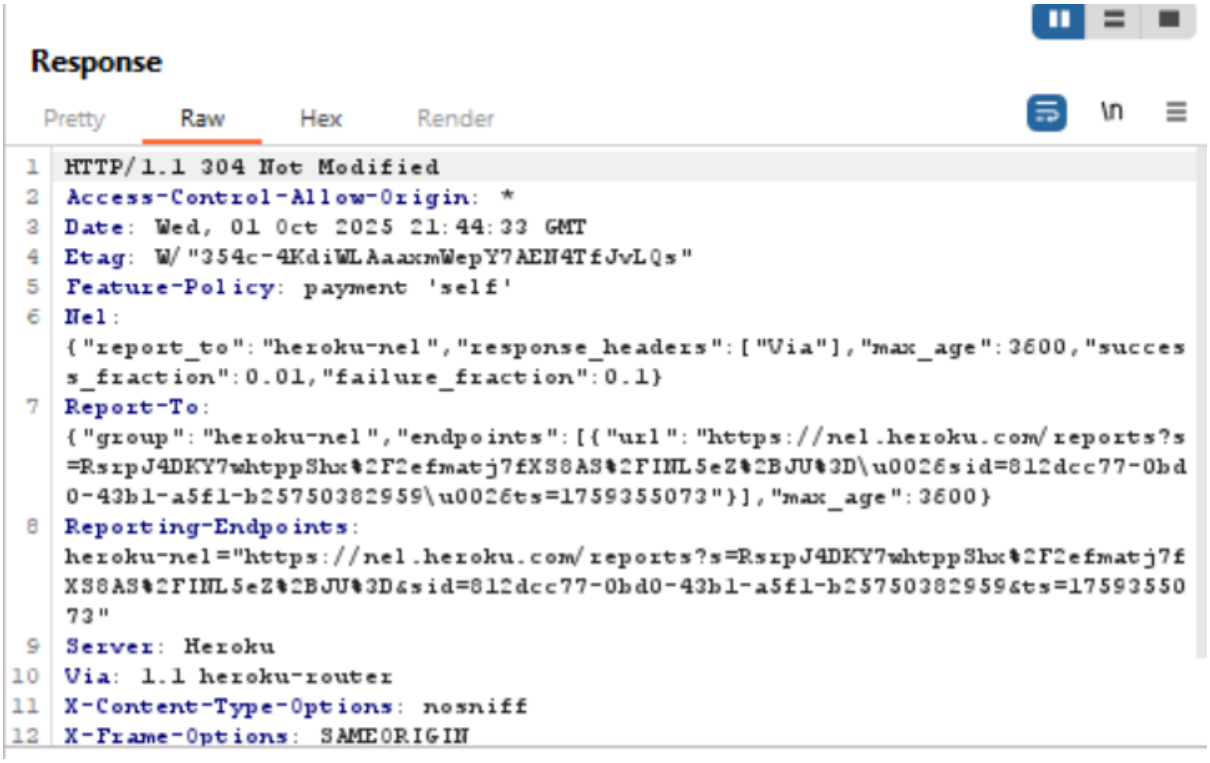

Figure 8: Burp Suite Response showing missing security headers (Content-Security-Policy, Strict-Transport-Security, Referrer-Policy) while X-Frame-Options and X-Content-Type-Options are present

Impact:

Without CSP and HSTS, the application is more exposed to cross-site scripting, content injection, and downgrade attacks.

Risk Rating: MEDIUM

Recommendation:

1. Add a Content-Security-Policy (CSP), e.g.:

Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'; object-src 'none'; frame-ancestors 'none';

2. Enable Strict-Transport-Security (HSTS) if serving only over HTTPS:

   Strict-Transport-Security: max-age=31536000; includeSubDomains; preload

3. Add Referrer-Policy and consider X-XSS-Protection if desired:

   Referrer-Policy: no-referrer

4. Verify these headers are applied site-wide (all responses).

Verification:

After applying headers, reload the page and capture Response → Raw to confirm CSP, HSTS and Referrer-Policy are present.

5. OWASP Top 10 Mapping

- SQL Injection — A03:2021 (Injection) — High
- Broken Authentication — A07:2021 (Identification & Authentication Failures) — High
- Insecure Cookies — A02/A07 (Session Management / Authentication Failures) — Medium
- Missing Security Headers — A05:2021 (Security Misconfiguration) — Medium

6. Conclusion

The assessment identified multiple issues requiring remediation:

- HIGH: SQL Injection (4.1), Broken Authentication (4.2)

- MEDIUM: Insecure Cookies (4.3), Missing Security Headers (4.4)

These findings demonstrate practical vulnerabilities in the tested Juice Shop instance. Apply recommendations and re-scan to validate fixes.

7. Appendix — Evidence & Files

Screenshots (Screenshots/ folder):

- 12_report_cover.png — Report cover

- 01_app_home.png — Application home

- 02_zap_findings.png — ZAP scan summary

- 03_burp_request.png — Burp request with SQLi payload

- 04_sql_injection.png — Burp response confirming SQLi

- 05_auth_success.png — Successful login with weak/default credentials

- 06_auth_request.png — Burp login request/response capture

- 07_insecure_cookie.png — Response showing missing cookie flags

- 08_missing_headers.png — Response showing missing security headers

Logs (Logs/ folder):

- zap_report.pdf — OWASP ZAP full scan report