SECURITY ASSESSMENT REPORT

1. Executive Summary

1.1 Overview

On 29th December 2025, a comprehensive web application security assessment was conducted on the OWASP Juice Shop web application. The primary objective of this assessment was to identify security vulnerabilities and evaluate the application's alignment with the OWASP Top 10 2025 framework.

1.2 Risk Profile

Based on the result of this assessment, the overall security posture of the application is currently rated as high. During the evaluation, a total of 8 vulnerabilities were identified, categorised by severity below:

Medium	Low	Informational
3	2	3

1.3 Key Findings

The assessment identified several vulnerabilities that could compromise the application's security posture. The following findings represent the most significant areas of concern:

- Insecure Design (A06:2025): The application lacks a Content Security Policy (CSP) header, which poses an increased risk of Cross-site scripting (XSS) attacks. and leaves the application vulnerable to data injection and click-jacking.
- Broken Access Control (A01:2025): A Cross-Domain misconfiguration was identified, and this could allow malicious third-party sites to interact with the application and potentially access sensitive user data.
- Software Supply Chain Failures (A03:2025): The application utilises a JS library which appears vulnerable. Using unpatched libraries provides a direct pathway for attackers to exploit publicly documented flaws to compromise the client-side environment.

1.4 Strategic Recommendation

It is recommended that the development team prioritise the remediation of the medium risks to prevent them from escalating. Specifically, the team should focus on implementing a robust centralised authentication filter to address systemic issues found across the application.

Following the remediation phase, a re-test should be conducted to ascertain that all identified vulnerabilities have been effectively resolved

.

2. Scope of Work

Assessment target: OWASP Juice Shop Web Application

Target URL: http://localhost:3000/

Methodology: Black-box testing (no source code or server access provided)

Testing Window: 29 December 2025

Exclusions: Testing was limited strictly to the web application layer. Network-level infrastructure, Denial of Service (DoS testing) and social engineering of users were strictly out of scope.

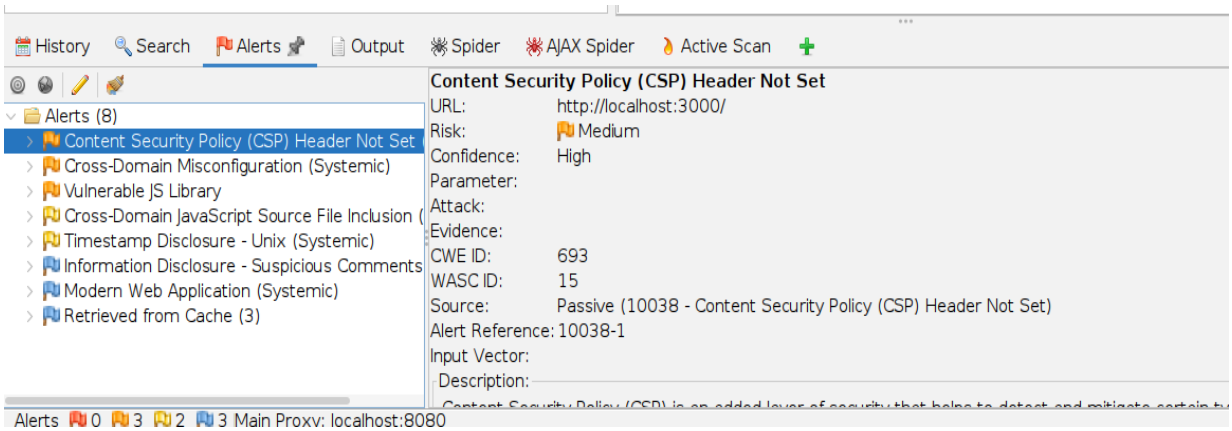
3. Detailed Findings

3.1 Finding #1: Insecure Design (A06:2025)

Severity: Medium

Description: The scanner identified that the Content Security Policy (CSP) Header was not set. This can leave the application vulnerable to attacks such as Cross-site scripting, data injection and clickjacking.

Evidence:



The ZAP scan result above shows the related vulnerability finding

Remediation:

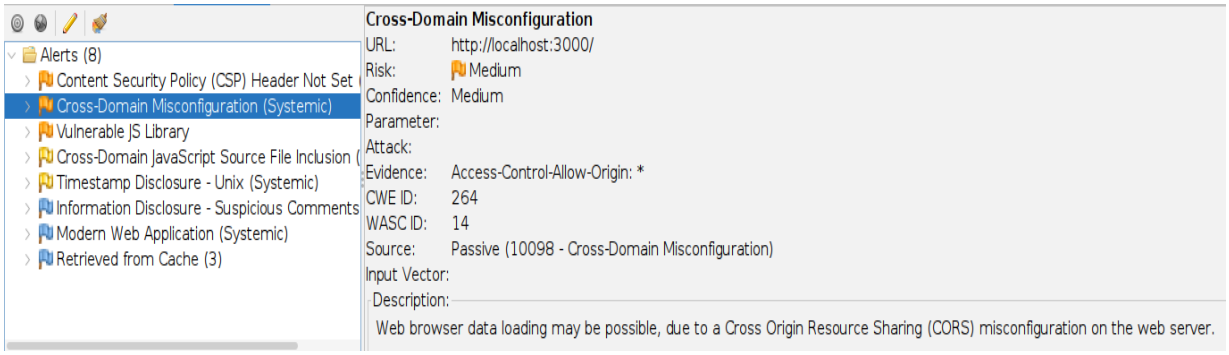
- Implement a strict Content Security Policy (CSP) header across all application responses to restrict the sources from which scripts can be loaded.

3.2 Finding #2: Broken Access control (A01:2025)

Severity: Medium

Description: A Cross-Domain Misconfiguration was identified, arising from a misconfiguration in the Cross-Origin Resource Sharing (CORS) settings on the web server. The Access-Control-Allow-Origin is set to a wildcard (*), thus allowing access from any origin, which can expose sensitive data to any site that can send credentials.

Evidence:



The image above shows the Access-Control-Allow-Origin set to a wildcard (*).

Remediation:

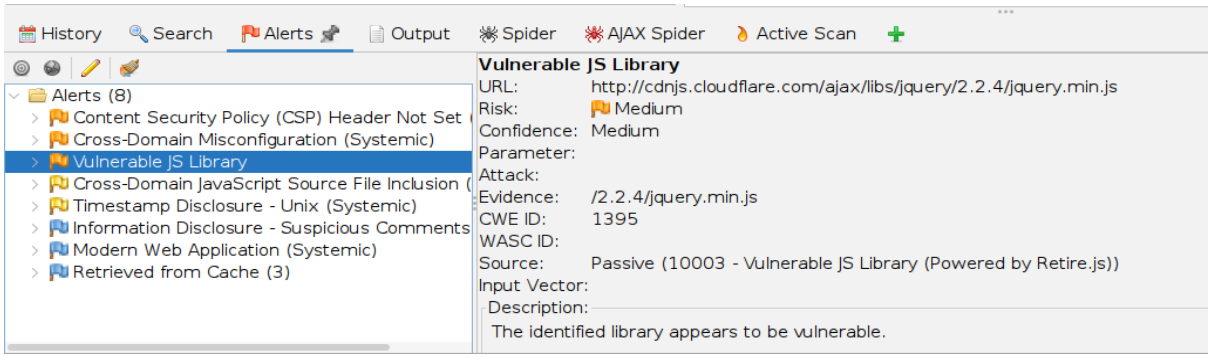
- Ensure proper configuration of CORS.
- Restrict the allow-origin to only domains that require access, to prevent unauthorised cross-domain requests.
- Replace the wildcard (*) with a specific whitelist of trusted domains.
- Implement authentication and authorisation for sensitive resources, to further control access to cross-domain requests.

3.3 Finding #3: Software Supply Chain Failures (A03:2025)

Severity: Medium

Description: The scanner revealed that a JavaScript library appears to be vulnerable.

Evidence:



The image above shows the evidence of the vulnerable JS library as /2.2.3/query.min.js

Remediation:

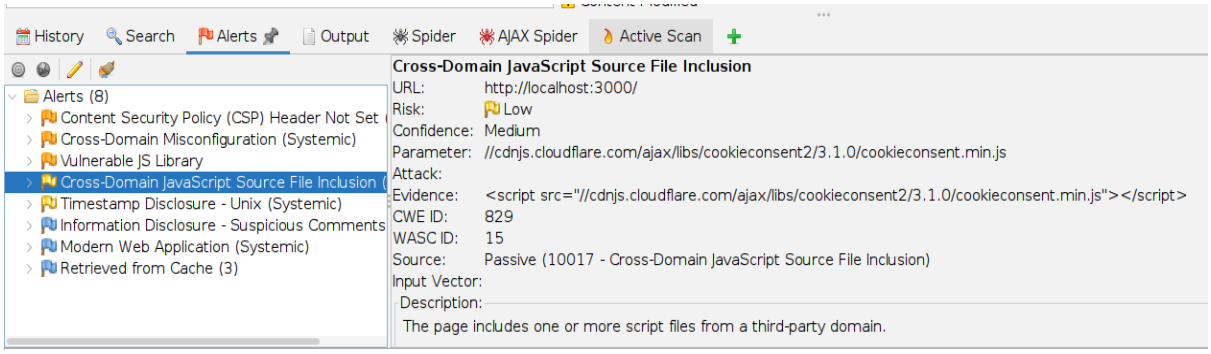
- Perform a Software Composition Analysis (SCA) to identify all outdated libraries. Specifically, update the JS Library to the latest version to remediate known CVEs.

3.4 Finding #4: Software/Data Integrity (A08:2025)

Severity: Low

Description: A Cross-Domain JavaScript Source File Inclusion was identified, where the page was found to have included scripts from an external domain. If not properly managed, it can lead to code execution, data leakage and supply chain attacks.

Evidence:



Remediation:

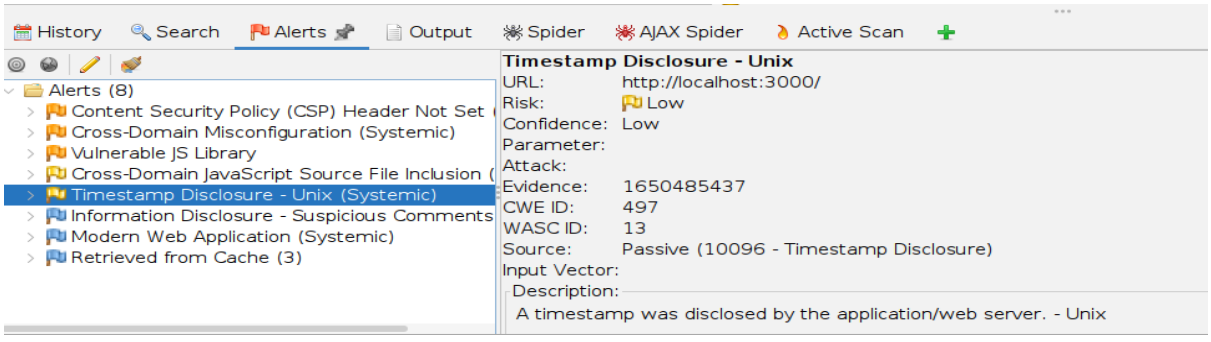
- Where possible, host and serve all scripts from your domain to mitigate the risk of including malicious scripts from third-party domains.

3.5 Finding #5: Broken Access Control (A01:2025)

Severity: Low

Description: A timestamp was disclosed by the application/web server - Unix.

Evidence:



The image above shows the evidence of the timestamp as 1650485437, which evaluates to 2022-04-20 15:10:37.

Remediation:

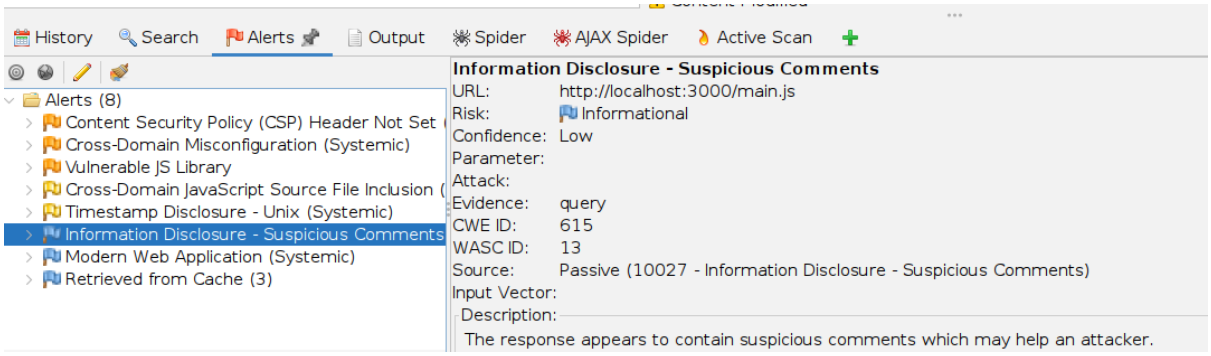
- Review alerts manually to confirm that any timestamp disclosure alerts are actual server timestamp leaks and not used to generate sensitive information on the server side and that the data cannot be used to disclose exploitable patterns.
- Ensure appropriate access controls are in place to restrict access to sensitive information, including timestamps.

3.6 Finding #6: Broken Access Control (A01:2025)

Severity: Informational

Description: An Information Disclosure vulnerability was identified, with a query response containing suspicious comments/information that could be useful to attackers.

Evidence:



The image above shows the query response from the URL, appearing to contain suspicious comments which an attacker may leverage to exploit the web application.

Remediation:

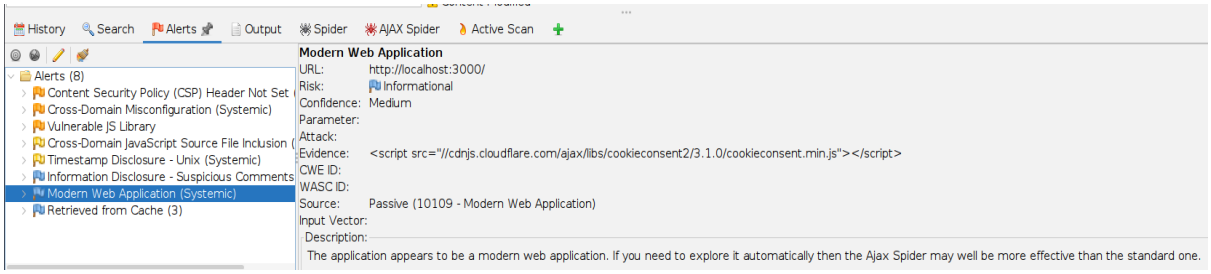
- Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

3.7 Finding #7: Modern Web Application, not mapped to OWASP Top 10.

Severity: Informational

Description: The application is seen to be a modern web application.

Evidence:



The image above shows the ZAP scan result alert on the new web application.

Remediation:

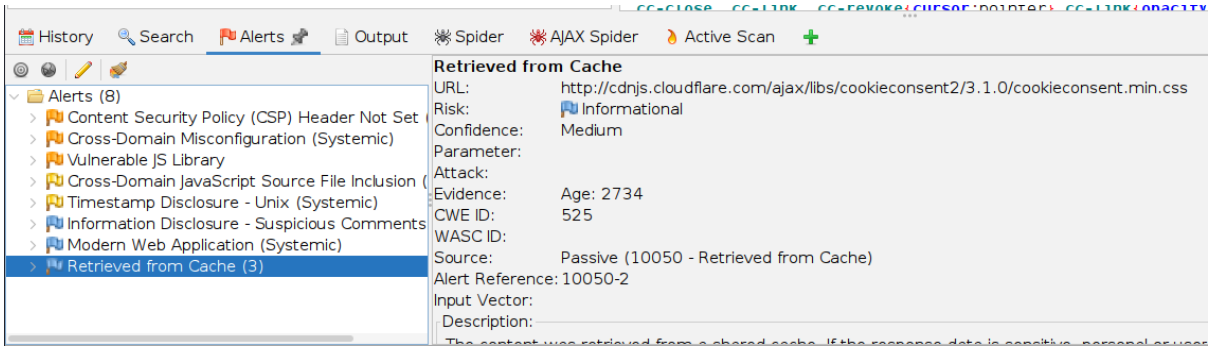
No changes required

3.8 Finding #8: Insecure Design (A06:2025)

Severity: Informational

Description: Retrieved from Cache

Evidence:



The image above shows the content was retrieved from a shared cache. If the response data is sensitive, it may result in sensitive information being leaked.

Remediation:

- Review and audit cached content to ensure that all content served from shared caches does not include sensitive, personal or user-specific information.
- Implement cache validation mechanisms to ensure that only appropriate content is being served.
- Ensure that user-specific or sensitive content is never cached in shared cache systems.
- Test and monitor cache behaviour regularly to prevent information leakage.

4. Conclusion

The security assessment of the OWASP Juice Shop application identified eight (8) vulnerabilities ranging from Medium to Informational severity. It is worth noting that finding #7 does not constitute a direct vulnerability but may introduce risk if an underlying application flaw exists. Although no Critical or High-severity vulnerabilities were detected, the Medium-severity findings, particularly those related to Broken Access Control (A01:2025), Software Supply Chain Failures (A03:2025), and Software and Data Integrity (A06:2025), indicate that the application's defence-in-depth controls require further strengthening.

The most significant concerns relate to the use of outdated JavaScript libraries, a cross-domain configuration, and the absence of a Content Security Policy (CSP) header. If left unaddressed, these vulnerabilities could serve as an entry point for more sophisticated attacks, including Cross-Site Scripting (XSS) or data exfiltration.

It is recommended that the development team prioritise the remediation of the Medium-severity findings immediately. Resolving the Low and Informational issues will further harden the application against future threats. Following the implementation of the suggested improvements, a follow-up validation scan is recommended to ensure all vulnerabilities have been successfully remediated.