WEB APPLICATION SECURITY ASSESMENT REPORT

NAME: NNONAH VICTOR C.

TASK 1: WEB APPLICATION SECURITY TESTING

PROGRAM: FUTURE INTERNS CYBERSECURITY INTERNSHIP

DATE: AUGUST 2025

TARGET APPLICATION: OWASP JUICE SHOP (INTENTIONALLY VULNERABLE APP)

Task Summary

This assessment involved a comprehensive security evaluation of the OWASP Juice Shop application, utilizing both OWASP ZAP for automated scans and manual testing. The primary objective was to identify and analyze critical vulnerabilities including SQL Injection, Cross-Site Scripting (XSS), and security misconfigurations which could lead to unauthorized access and data compromise.

The testing was conducted on the application's publicly available deployment on Heroku, providing a realistic and pre-configured environment for simulating web application attacks. All findings were meticulously examined, mapped to the corresponding categories in the OWASP Top 10 (2021) framework, and prioritized based on risk. Remediation measures were then recommended to effectively address and mitigate the identified security vulnerabilities.

Tools Used:

- ✓ Kali Linux
- ✓ OWASP Juice Shop
- ✓ OWASP ZAP

Procedure

- ✓ The OWASP Juice Shop application was accessed via its publicly available deployment on Heroku (https://juice-shop.herokuapp.com). This provided a realistic and pre-configured environment for simulating web application attacks without requiring a local setup.
- ✓ The hosted Heroku deployment served as the designated target for all automated and manual security testing conducted with OWASP ZAP.
- ✓ We utilized the tool to execute active scanning processes, systematically identifying vulnerabilities within input fields, user interactions, and potential vulnerabilities.
- ✓ The findings were meticulously examined and aligned with the corresponding categories in the OWASP Top 10 framework. Subsequently, corrective measures were recommended to address the identified security risks.

Identified Vulnerabilities

1. **Cross Domain Misconfiguration**

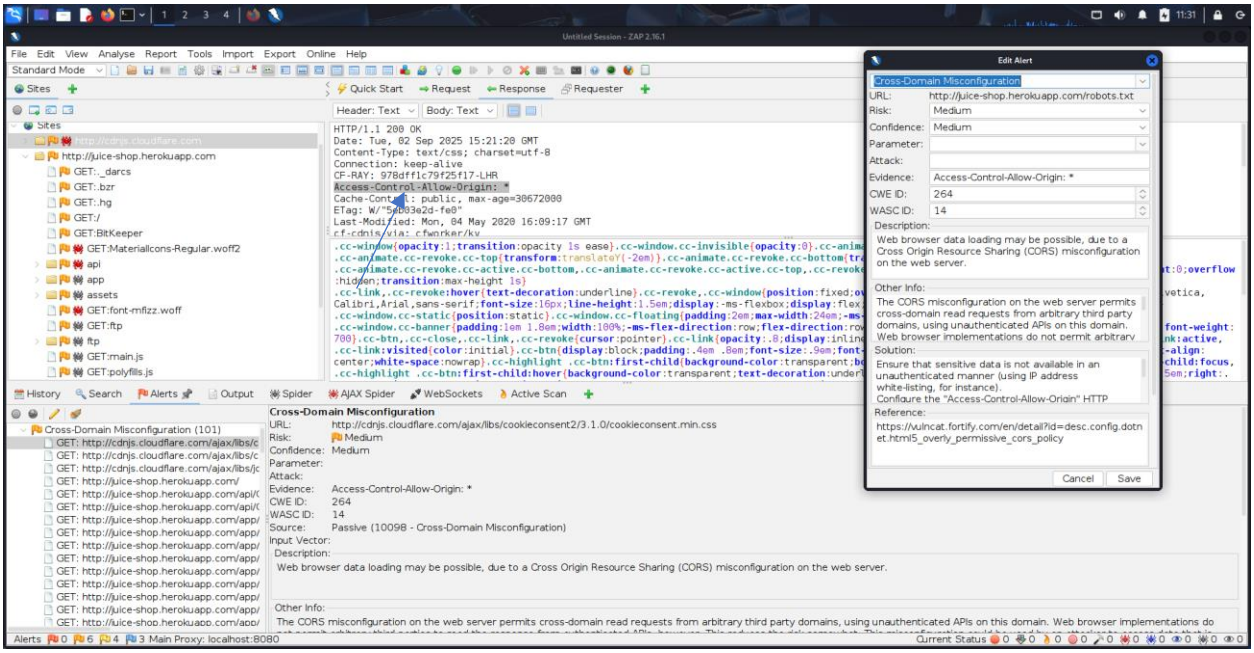
   During the automated assessment with OWASP ZAP, a potential Cross-Domain Misconfiguration was detected in the application's Cross-Origin Resource Sharing (CORS) policy. The server was observed to include overly permissive response headers, such as the Access-Control-Allow-Origin: * header. This configuration allows requests from any external domain, thereby bypassing the browser's same-origin policy. As a result, a malicious website could potentially send crafted requests to the application.

   Severity Level

   This vulnerability was rated with a Medium risk and Medium confidence level by OWASP ZAP. While not a critical vulnerability, this misconfiguration can still lead to data disclosure or enable other attacks if exploited in combination with other vulnerabilities.

   Mitigation Strategies

   - ✓ Restrict the Access-Control-Allow-0rigin Header.
   - ✓ Implement a Secure CORS Policy.
   - ✓ Require Authentication for Cross-Origin Requests.
   - ✓ Avoid Wildcards



2. **Content Security Policy (CSP) Header Not Set**

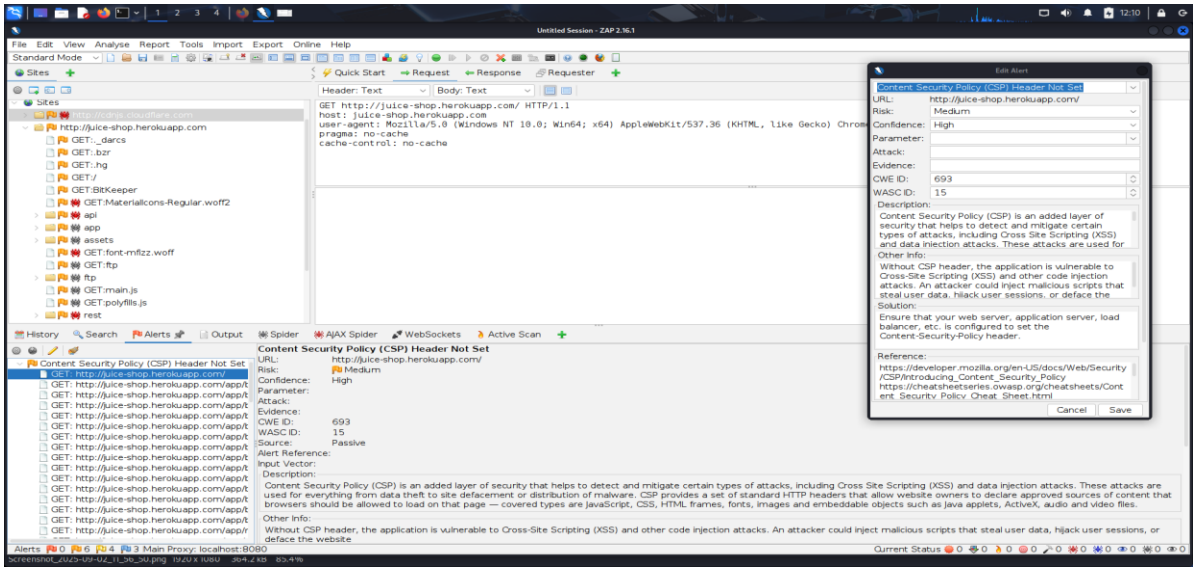
During the automated assessment with OWASP ZAP, a critical security vulnerability was identified due to the absence of a Content Security Policy (CSP) header. The application's HTTP responses lacked this fundamental security layer, which is designed to detect and mitigate various types of attacks, including Cross-Site Scripting (XSS), clickjacking, and data injection. Without a properly configured CSP, the application is exposed to malicious content execution, where an attacker could inject scripts or resources to compromise user data or deface the site.

Severity Level

This vulnerability was rated with a Medium risk and High confidence level by OWASP ZAP. The high confidence indicates a strong certainty in the existence of this issue. While classified as medium risk, the absence of a CSP significantly increases the attack surface, potentially enabling more severe attacks like XSS, which can lead to session hijacking, data theft, and defacement.

Mitigation Strategies

- ✓ Implement a Robust CSP Header.
- ✓ Define a Strict Policy.
- ✓ Use report-uri or report-to for Monitoring.
- ✓ Regularly Review and Update CSP.



3. **Missing Anti-clickjacking Header**

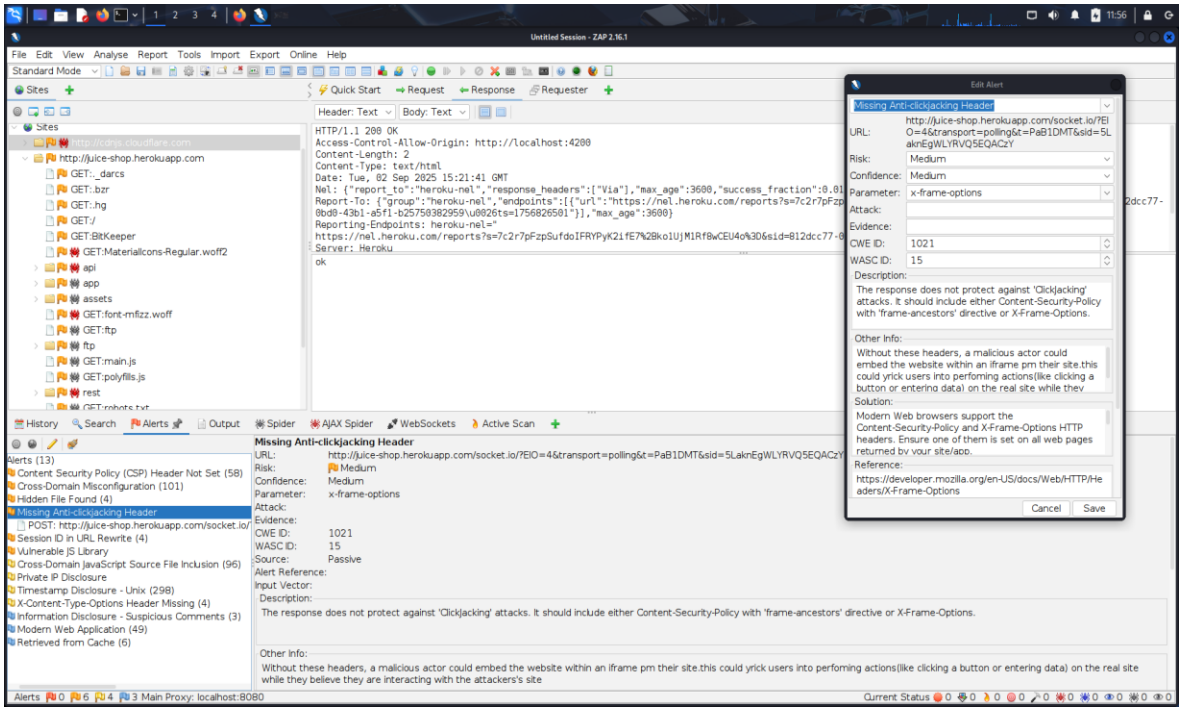
During the automated security assessment with OWASP ZAP, a vulnerability was identified due to the absence of crucial anti-clickjacking headers. The application's HTTP responses lacked either the X-Frame-Options or the Content-Security-Policy with the frame-ancestors directive. This security lapse makes the application susceptible to clickjacking attacks, where an attacker can trick a user into clicking on a hidden element by embedding the vulnerable site within an <iframe> on a malicious webpage.

Severity Level

This vulnerability was rated with a Medium risk and a Medium confidence level by OWASP ZAP. While not the highest-rated risk, it is a significant concern as a successful clickjacking attack can be used to trick users into performing unwanted actions on the legitimate website, such as transferring funds, making unauthorized purchases, or revealing sensitive data.

Mitigation Strategies

- ✓ Implement Content-Security-Policy.
- ✓ Use X-Frame-Options.
- ✓ Ensure Consistent Implementation.
- ✓ Avoid Conflicting Headers.



4. **Vulnerable JS Library**

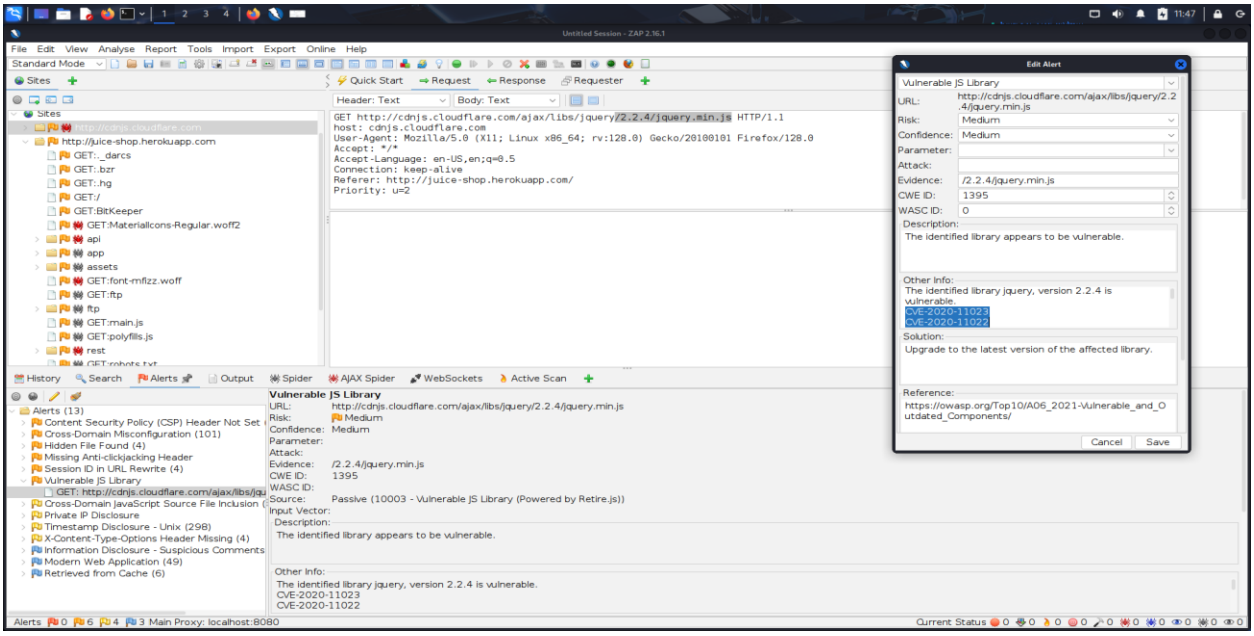
During the automated assessment with OWASP ZAP, a vulnerability was identified related to the use of a publicly known insecure JavaScript library. The application was found to be using jQuery version 2.2.4, which has documented security vulnerabilities. Using outdated and vulnerable third-party components puts the entire application at risk, as attackers can exploit these known flaws to compromise the system, steal user data, or disrupt service. This finding aligns with the OWASP Top 10 category for "Using Components with Known Vulnerabilities."

Severity Level

This vulnerability was rated with a Medium risk and Medium confidence level by OWASP ZAP. The severity stems from the fact that the exploit code for these vulnerabilities is often publicly available, making it easy for even an unsophisticated attacker to compromise the application. The risk is directly dependent on the specific vulnerabilities present in the library version and their potential impact.

Mitigation Strategies

- ✓ Upgrade to the Latest Version.
- ✓ Regularly Update Dependencies.
- ✓ Use Automated Tools.
- ✓ Remove Unused Libraries.



5. **Session ID in URL Rewrite**

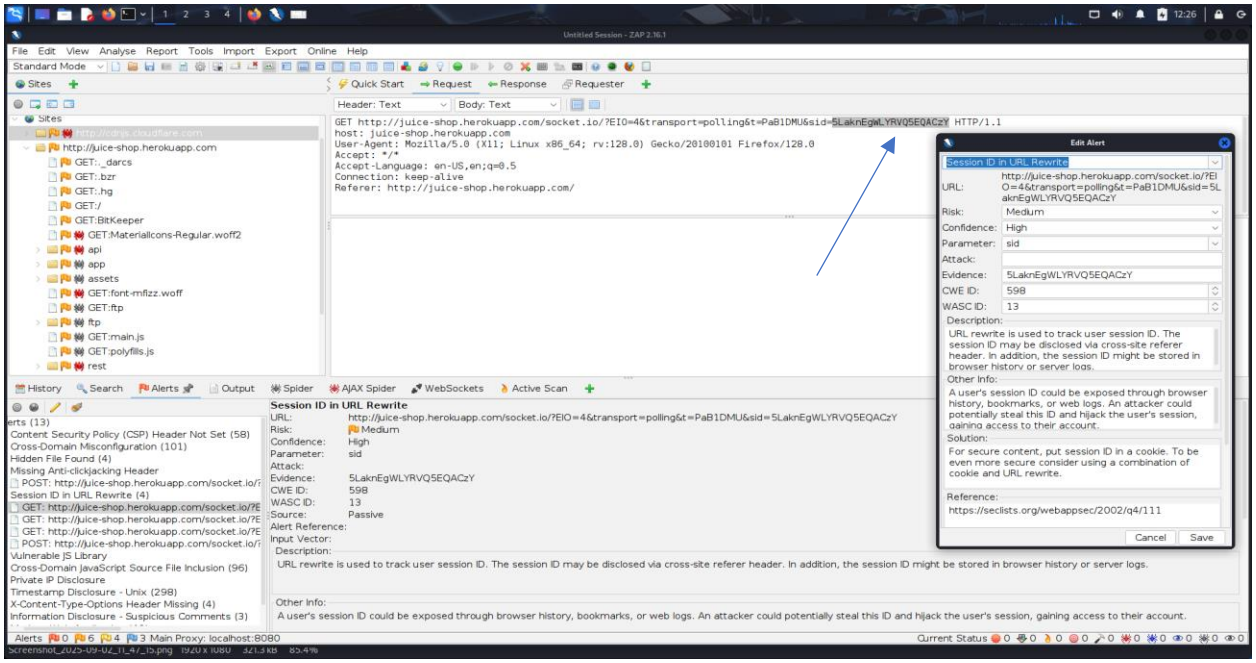
During the automated assessment with OWASP ZAP, a vulnerability was detected related to how the application manages user sessions. The Session ID was found to be included directly in the URL as a query parameter (sid). This practice, known as URL rewriting, is a significant security risk because it exposes a sensitive identifier in a location that is easily captured. The session ID could be disclosed in browser history, bookmarks, or web server logs. An attacker who obtains this ID could hijack a user's session and gain unauthorized access to their account and data.

Severity Level

This vulnerability was rated with a Medium risk and High confidence level by OWASP ZAP. The high confidence indicates that the issue is not a false positive. While it may not be as critical as a SQL Injection, a stolen session ID can lead to complete account compromise, making it a serious security concern that requires immediate attention.

Mitigation Strategies

- ✓ Use Cookies for Session IDs.
- ✓ Avoid URL Rewriting.
- ✓ Implement Short Session Timeouts.
- ✓ Use Unique and Unpredictable IDs.



OWASP Top 10 Mapping

Vulnerability Found	OWASP Top 10 (2021) Category	Risk
Cross-Domain Misconfiguration	A05:2021 - Security Misconfiguration	Medium
Content Security Policy Header Not Set	A05:2021 - Security Misconfiguration	Medium
Missing Anti-clickjacking Header	A05:2021 - Security Misconfiguration	Medium
Vulnerable JS Library	A06:2021 - Vulnerable and Outdated Components	Medium
Session ID in URL Rewrite	A07:2021 - Identification and Authentication Failures	Medium

Conclusion

This security assessment successfully identified multiple vulnerabilities, including security misconfigurations and outdated components. These findings highlight significant security weaknesses that, if left unaddressed, could expose the application to serious attacks. Promptly implementing the recommended mitigation strategies is crucial to securing the application, protecting user data, and building a more resilient system against evolving cyber threats.