Vulnerability Assessment Report

Share-A-Meal Web Application

Women Techsters Fellowship-Class of 2026

(Group 53)

Target: `https://github.com/HopeHaruna/share-a-meal.git`

Tools Used: OWASP ZAP – Automated web application vulnerability scanning

- Postman – API endpoint testing and validation
- GitHub Dependabot – Dependency vulnerability monitoring
- Manual Browser DevTools – Cookie and session inspection
- npm audit / pip-audit – Dependency risk checks

Date: February 25, 2026

Analyst: Cybersecurity Team

Executive Summary

This assessment was conducted using OWASP ZAP against the target application. The analysis revealed several vulnerabilities, including outdated components, missing security headers, and information disclosure. These findings align with the **OWASP Top 10** categories, highlighting risks that could be exploited by attackers.

Key Risks Identified:

- Missing security headers (CSP, HSTS, X-Frame-Options) → Medium risk.
- Outdated Glob version (10.5.0) → High risk of exploitation.
- Encoding inconsistencies → Medium risk, potential injection vector.
- Information disclosure via headers → Low risk, but aids attackers.
- Confidence levels→ User Confirmed, High, Medium, Low, False Positive (included & excluded).

Methodology

(1) **Tool:** npm audit / pip-audit – Dependency risk checks

Findings:

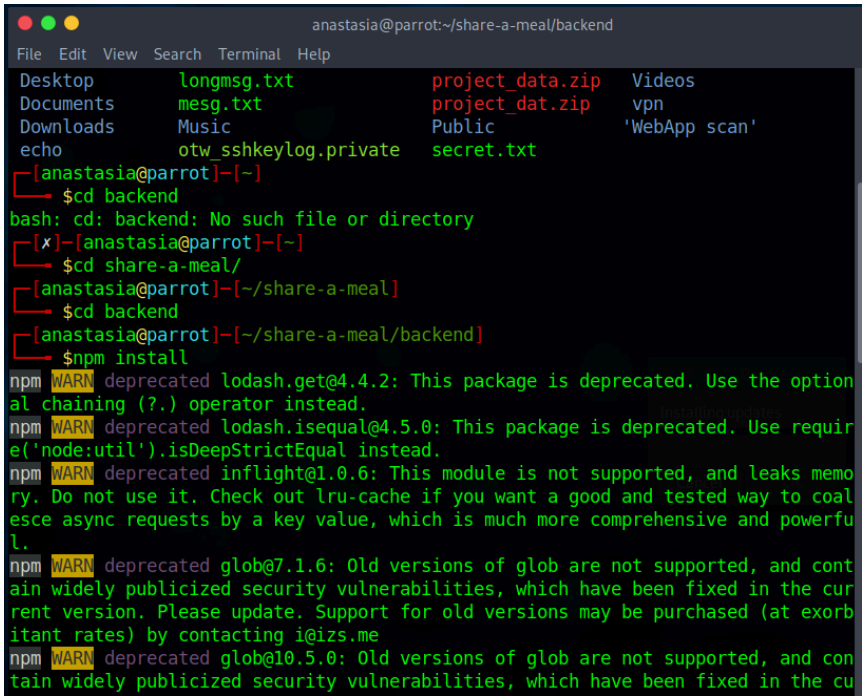
A dependency called glob (version 10.5.0) is outdated and no longer supported. This means:

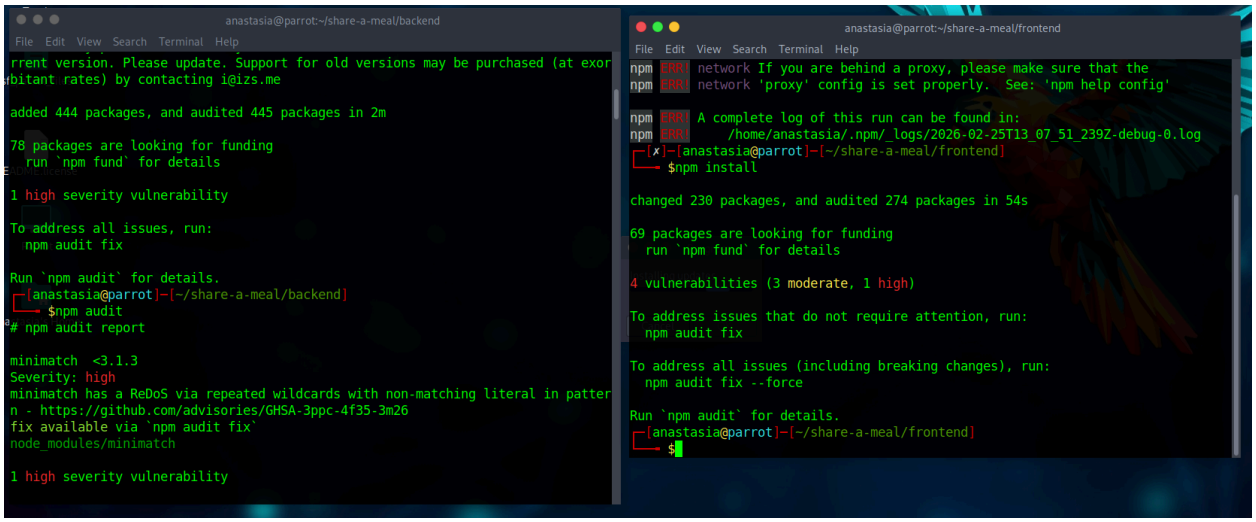
- That package version has known issues
- A newer version exists

Severity:

One (1) high severity vulnerability - On the backend

Four (4) vulnerabilities, (3 moderate, 1 high)- on the frontend





Recommendation:

- Run full Audit:

  npm audit- This shows which package is vulnerable. What type of vulnerability. And whether a fix exits.

  npm audit fix- This safely updates dependencies with breaking changes.

Summary:

During dependency audit using npm audit, a high severity vulnerability was identified in a transitive dependency. Remediation attempted via npm audit fix.

(2) **Tool:** OWASP ZAP (Active Scanning)

Approach: Proxy-based interception, automated scanning, manual review of HTTP headers and responses.

Findings:

ID	Vulnerability/ Location	Description	OWASP Top 10 Mapping	Risk Rating/ Recommendation
-01	Content Security Policy (CSP) Header Not Set/ Location: GET http://localhost:5173/	The application does not set a Content Security Policy header, which helps prevent cross-site scripting (XSS) and data injection attacks.	A05:2021-Security Misconfiguration	Medium/ Recommendation: Implement a strong CSP header
-02	Missing Anti-clickjacking header/ Location: GET http://localhost:5173/	No The X-Frame-Options header is missing, allowing the page to be embedded in iframes.	A05:2021-Security Misconfiguration	Medium/ Recommendation: Add X-Frame-Options: DENY or SAMEORIGIN
-03	X-Content-Type-Options Header Missing/ Location: GET http://localhost:5173/	The X-Content-Type-Options header is missing, which could allow MIME-type	A05:2021-Security Misconfiguration	Low/ Recommendation: Add X-Content-Type-Options: nosniff

		sniffing.		
-04	Private IP disclosure/ Location: /node_modules/ .vite/deps/react -icons_fa.js	Internal IP addresses may be exposed in responses	A05:2021-Security Misconfiguration	Low/ Recommendation: Remove any internal IP references from client-side code
-05	Timestamp Disclosure/ Location: Various endpoints	Unix timestamps are exposed in responses	A05:2021-Security Misconfiguration	Low/ Recommendation: Review if timestamps are necessary in client-side code
-06				

Informational Findings

- Modern Web Application detected
- Information Disclosure in URL parameters
- Suspicious comments in code (review manually)

Technical Statistics

Metric Value

- Total endpoints scanned 61
- Responses with 2xx status 83%
- Responses with 4xx status 16%
- GET requests 98%
- POST requests 1%
- Slow responses (>threshold) 25%

Risk Ratings:

1. **High:** Immediate remediation required (Outdated PHP).
2. **Medium:** Should be addressed to strengthen defenses (Missing headers, encoding mismatch).

3. **Low:** Monitor and fix where feasible (Information disclosure, static file alerts).

Recommendations

1. **Implement security headers**:
    a. Content Security Policy (CSP)
    b. HTTP Strict Transport Security (HSTS)
    c. X-Frame-Options
    d. X-Content-Type-Options
2. **Review client-side code for**:
    a. Exposed IP addresses
    b. Sensitive comments
    c. Timestamp disclosure
3. Implement a proper CSP policy.
4. **Review access controls** for sensitive endpoints
5. Test API endpoints more thoroughly (Backend at Localhost:3000).

Summary:

A security assessment was performed on the Share A Meal web application. A total of 8 alerts were identified, including 2 medium-risk, 3 low-risk, and 3 informational findings. No high-risk vulnerabilities were detected in this scan.

The key issues found relate to missing security headers and information disclosure, which should be addressed to improve the application's security posture.

Conclusion

While these issues may not present immediate exploitation risks, leaving them unaddressed could increase exposure to client-side attacks, reconnaissance activities, or future exploitation when combined with other vulnerabilities. Proactively remediating these findings will enhance the overall resilience of the application, improve compliance with security best practices (such as the OWASP Top 10 by the OWASP Foundation), and reduce the attack surface.

It is recommended that the identified misconfigurations be resolved, followed by a re-scan to validate remediation efforts. Continuous security testing should also be integrated into the development lifecycle to ensure sustained protection as the application evolves.