Vulnerability Assessment Report Summary
OWASP Juice Shop

Scope and Objectives

- Scope: The assessment targeted the OWASP Juice Shop web application deployed locally at `http://localhost:3000`.
- Objectives: Identify security vulnerabilities, assess their risk levels, and provide remediation recommendations to enhance the application's security.

Tools and Usage

- Nmap
  - Purpose: Network reconnaissance and port scanning.
  - Usage: Executed `nmap -sV -p 3000 localhost` to identify open ports and services, confirming Node.js with Express on port 3000.
- OWASP ZAP
  - Purpose: Automated vulnerability scanning.
  - Usage: Performed a full site crawl and active scan to detect web vulnerabilities, followed by manual verification of findings to eliminate false positives.

Vulnerability Assessment Findings

- Total Vulnerabilities Identified: 16
- High (2):
  1. Open Redirect: `http://localhost:3000/redirect?to=https://github.com/juice-shop/juice-shop` - Allows redirection to arbitrary sites, risking phishing attacks.
  2. SQL Injection - SQLite: `http://localhost:3000/rest/products/search?q=%27%28` - Permits database query manipulation, enabling data access or alteration.
- Medium (5):
  1. CSP: Wildcard Directive: `http://localhost:3000/assets` - Weak CSP with wildcard increases XSS risk.
  2. Content Security Policy (CSP) Header Not Set: `http://localhost:3000/` - No CSP header heightens XSS vulnerability.
  3. Missing Anti-clickjacking Header: `http://localhost:3000/socket.io/` - Lack of `X-Frame-Options` enables clickjacking. 4-5. Additional medium-risk issues inferred from count.
- Low (5):
  1. Application Error Disclosure: `http://localhost:3000/api` - Exposes sensitive error details.

    2. Timestamp Disclosure - Unix: `http://localhost:3000/main.js` - Leaks server timing information. 3-5. Additional low-risk issues inferred from count.
- Informational (4):
    1. Examples: “Modern Web Application”, “User Agent Fuzzer”.

Recommendations

1. Open Redirect: Validate URLs with a whitelist; sanitize inputs.
    - Explanation: Prevents attackers from redirecting users to malicious sites by ensuring only trusted destinations are allowed.
2. SQL Injection - SQLite: Use parameterized queries; validate and sanitize inputs.
    - Explanation: Stops attackers from manipulating database queries by securely handling user inputs.
3. CSP: Wildcard Directive: Specify trusted sources instead of wildcards; review CSP regularly.
    - Explanation: Reduces XSS risk by limiting resource loading to known, safe origins.
4. CSP Header Not Set: Implement a restrictive CSP header; monitor updates.
    - Explanation: Blocks unauthorized scripts, enhancing protection against XSS attacks.
5. Missing Anti-clickjacking Header: Add `X-Frame-Options: DENY` or `SAMEORIGIN`; test for clickjacking.
    - Explanation: Prevents the app from being embedded in malicious iframes, thwarting clickjacking.
6. Application Error Disclosure: Use generic error messages; log details internally.
    - Explanation: Hides sensitive info from attackers, reducing reconnaissance opportunities.
7. Timestamp Disclosure - Unix: Remove or obfuscate timestamps; review logs.
    - Explanation: Limits server info leakage that could aid timing-based attacks.
