Vulnerability Assessment Report: JuiceShop

Repository: https://github.com/Akaolisangwu-projects/juice-shop
Analyst: Akaolisa Ngwu

Summary

The purpose of Juice Shop is to simulate real-world insecure coding and misconfiguration patterns found in modern Node.js web applications, build security awareness, test mitigation techniques, and validate automated scanning tools such as Snyk. The current scan and assessment summarize the actual exploitable risks from a production security standpoint, using real Snyk scan data and closed GitHub pull requests that represent partial remediation efforts. This report provides a technical breakdown of vulnerabilities, remediation efforts already addressed and a prioritized action plan for mitigating critical security risks. The analysis draws exclusively from my Snyk scan findings and GitHub repo history.

Package / Component	Version / Path	Priority	Issue Type	Risk	Fix Availability
vm2 (via juicy-chat-bot @0.9.0)	3.9.17	661	Remote Code Execution (RCE)	High — sandbox escape, code injection	No supported fix per Snyk
libxmljs2	0.37.0	512	Type Confusion (2 findings)	High — memory corruption, arbitrary read/write	No supported fix per Snyk
express-ipfilter	1.3.2	502	Server-Side Request Forgery (SSRF)	High — SSRF to internal endpoints	No supported fix per Snyk
marsdb	0.6.11	490	Arbitrary Code Injection	High — execute arbitrary JS	No supported fix per Snyk
grunt-replace-json	0.1.0	472	Prototype Pollution	Medium / High	No supported fix per Snyk
notevil	1.3.3	432	Sandbox Bypass	Medium	No supported fix per Snyk

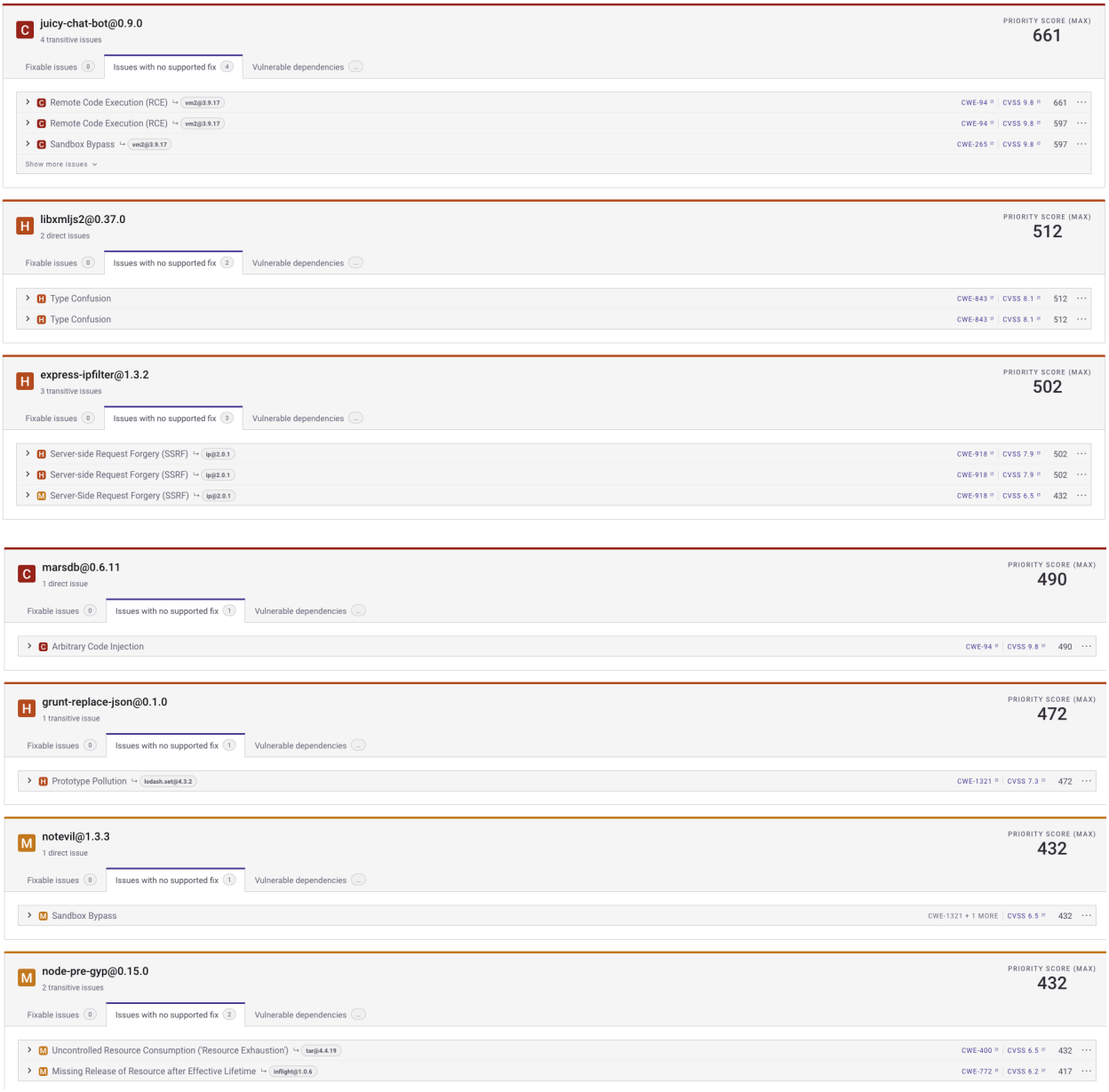
node-pre-gyp	0.15.0	432	Resource Exhaustion / Missing Release	Medium	No supported fix per Snyk
download	8.0.0	422	Zip Slip, ReDoS, Open Redirect	Medium	No supported fix per Snyk
codemirror-solidity	0.2.5	452	ReDoS	Medium	No supported fix per Snyk
In-repo code	lib/insecurity.ts + routes	—	Hardcoded Secrets, NoSQL Injection, Path Traversal	High	Manual fix required

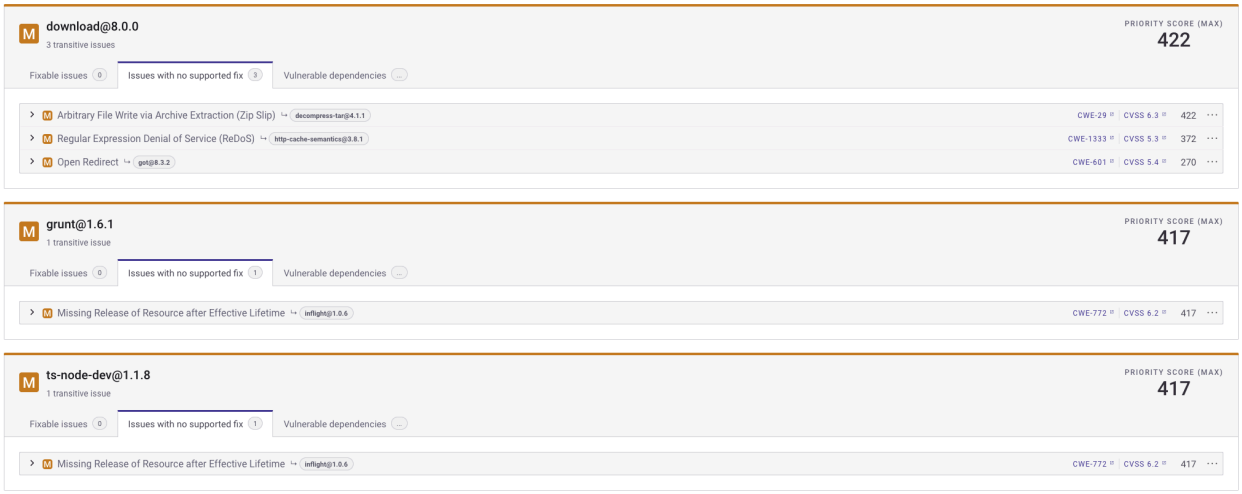
Changes Already Addressed (via Snyk)

Pull request	What It Fixed / Upgraded
Package.json (Fix for 18 vulns)	Bulk upgrades of dependencies flagged by Snyk
Package.json (Fix for 12 vulns)	More dependency patching
package.json (Fix for 4 vulns)	Targeted fixes in vulnerable packages
frontend/package.json (Fix for 3 vulns)	Smaller updates in utility libs
jsonwebtoken 9.0.0 to 9.0.2	Upgraded to safer JWT library version
i18n 0.14.2 to 0.15.1	Fixed i18n vulnerabilities
pdfkit 0.12.3 to 0.17.2	Upgraded PDF generation library
Sanitize-html upgrades (v1 → v2 → v2.17.0)	Hardened HTML sanitization logic
express-jwt 5.3.0 to 6.1.1	Upgraded JWT auth middleware

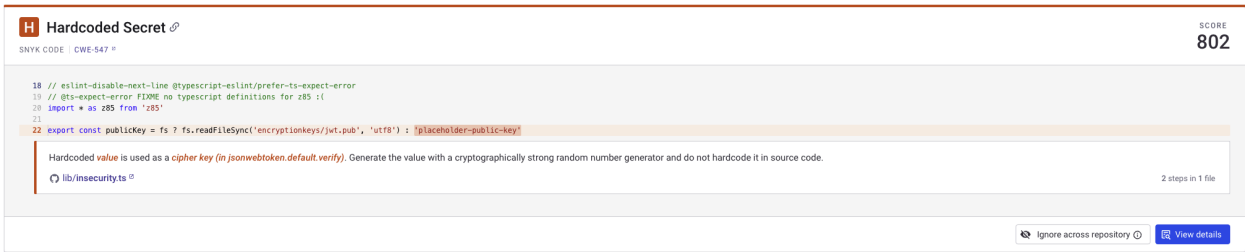
Reference: https://github.com/Akaolisangwu-projects/juice-shop/pulls?q=is%3Apr+is%3Aclosed

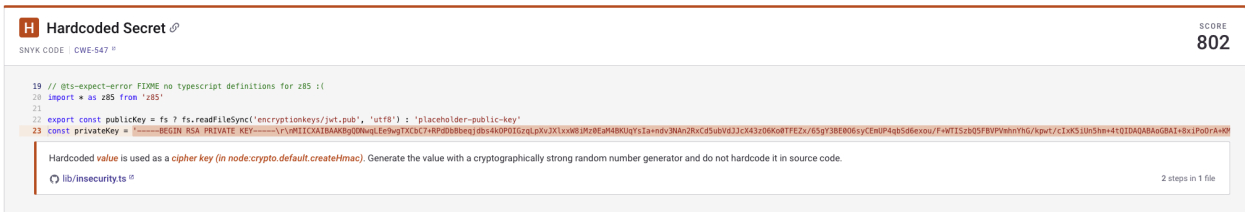
Vulnerabilities in package.json

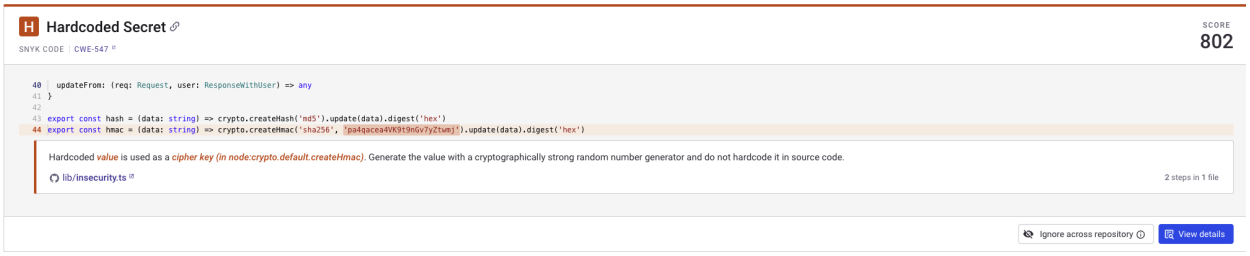


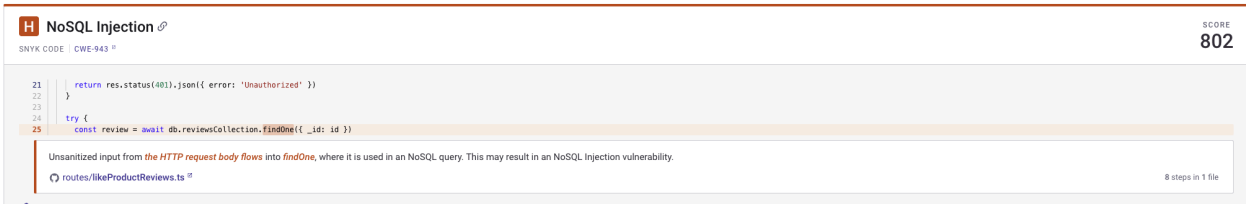


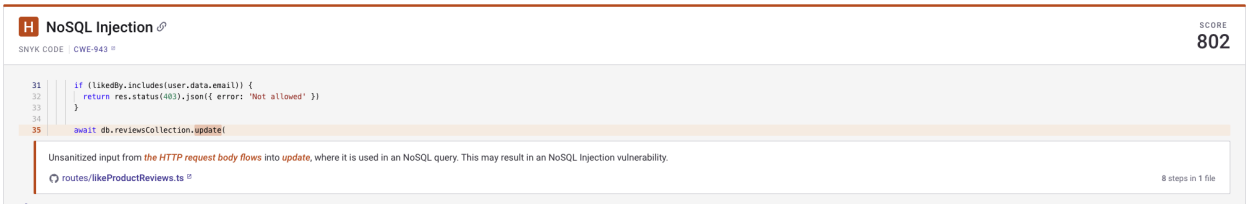
Vulnerabilities in Code Analysis

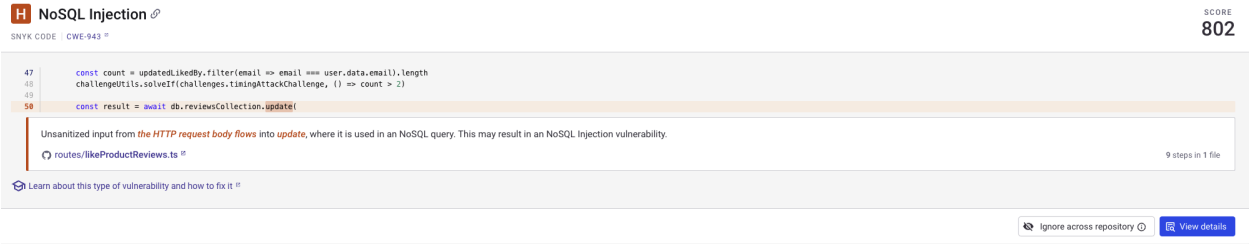




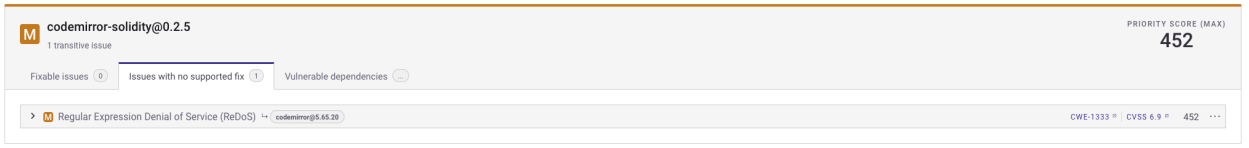








Vulnerabilities in frontend/package.json



★ Dependency-Level Vulnerabilities

Package / Module	Version / Source	Vulnerability	CWE	CVSS / Priority	Impact Summary	Status
vm2 (via juicy-chat-bot@0.9.0)	3.9.17	Remote Code Execution (RCE) – sandbox escape allows arbitrary code execution.	CWE-94 / CWE-265	9.8 / Critical	Full system compromise; attacker executes arbitrary commands in Node runtime.	No supported fix
marsdb	0.6.11	Arbitrary Code Injection due to unsafe eval implementation.	CWE-94	9.8 / Critical	Enables arbitrary JavaScript execution within the database layer.	No supported fix
libxmljs2	0.37.0	Type Confusion (2 findings) – unsafe pointer manipulation and memory corruption.	CWE-843	8.1 / High	Memory corruption may result in denial-of-service or data exposure.	No supported fix
express-ipfilter	1.3.2	Server-Side Request Forgery (SSRF) – improper IP validation.	CWE-918	7.9 / High	Attackers can access internal endpoints (e.g., cloud metadata).	No supported fix
grunt-replace-json	0.1.0	Prototype Pollution vulnerability via JSON object merging.	CWE-1321	7.3 / High	Allows object prototype manipulation and potential privilege escalation.	No supported fix
notevil	1.3.3	Sandbox Bypass – malicious payloads can execute in unrestricted scope.	CWE-265	7.8 / High	Leads to remote code execution when evaluating untrusted input.	No supported fix
node-pre-gyp	0.15.0	Resource Exhaustion – improper resource release under load.	CWE-400	6.5 / Medium	Causes performance degradation or denial-of-service.	No supported fix
download	8.0.0	Zip Slip & Open Redirect – unsafe extraction and URL handling.	CWE-29 / CWE-601	6.3 / Medium	File overwrite or redirect-based phishing attacks.	No supported fix
codemirror-solidity	0.2.5	Regular Expression Denial of Service (ReDoS).	CWE-1333	6.9 / Medium	Crafted input causes excessive CPU consumption, leading to DoS.	No supported fix

★ Code-Level Vulnerabilities

Category	Affected File / Path	Description	CWE	CVSS / Priority	Risk Impact	Status
Hardcoded Secrets	lib/insecurity.ts	RSA and HMAC keys are hardcoded directly into the application source code.	CWE-547	8.8 / High	Exposure of private keys allows attackers to forge JWT tokens or decrypt sensitive data.	Manual fix required
NoSQL Injection	routes/likeProductReviews.ts	Unvalidated user input flows directly into MongoDB queries (findOne, update).	CWE-943	8.8 / High	Enables attackers to exfiltrate or modify arbitrary database records.	Manual fix required
Path Traversal	routes/profileImageUrlUpload.ts	Uncontrolled file path used in fs.createWriteStream for image upload.	CWE-23	8.8 / High	Allows arbitrary file overwrite, leading to possible server compromise.	Manual fix required

Probable High-Risk Issues

Risk	Impact	Recommended Priority
Remote Code Execution (via vm2, marsdb)	Could lead to full takeover of your Node process	P0 – remove or isolate
Hardcoded secrets	Attackers could forge tokens or decrypt sensitive data	P1 – move to secure vault
NoSQL Injection	Direct database manipulation or data theft	P1 – sanitize and validate input
Path Traversal / Arbitrary Write	Overwrite critical system or app files	P1 – lock down file access logic

Recommended Remediation Plan

Phase 1: Immediate fixes (within days)

- ★ **Remove or isolate dangerous dependencies**
    - Remove vm2, marsdb, notevil, express-ipfilter from runtime.
    - If you need sandboxing, use remote service or container isolation.
- ★ **Refactor hardcoded secrets**
    - Remove all embedded RSA / HMAC keys.
    - Use environment variables or secret management (Vault, AWS Secret Manager).
    - Rotate keys.
- ★ **Sanitize database input**
    - Validate all input (e.g. IDs) before passing to findOne, update.
    - Use safe object mapping or query builders, reject extraneous fields.
- ★ **Secure file writes**
    - Always resolve with path.resolve(base, ...) and check startsWith(base).
    - Restrict allowed file extensions.

Phase 2: Dependency remediation & compensating controls

- ★ **Replace vulnerable modules with safer alternatives**
    - libxmljs2 to fast-xml-parser or xml2js with entity protection.
    - grunt-replace-json to custom or safer JSON utilities.
- ★ **Add runtime protections**
    - Use a Web Application Firewall (WAF) to block SSRF, path attacks.
    - Deny all outbound traffic except whitelisted domains.
    - Rate-limit high-risk endpoints (file upload, review).
- ★ **CI / monitoring / scanning**
    - Add snyk test to your CI pipeline; fail on new critical issues.
    - Enable Snyk monitoring (snyk monitor).

- Use Dependabot (or similar) to auto-propose updates.

★ **Penetration testing / security audit**

- After fixes, run dynamic tests focusing on upload, review, sandbox areas.
- Validate that no RCE, path, SSRF, DB injection remain.

Compliance Mapping

Standard	Affected Controls	Implication
OWASP Top 10	A03-Injection, A05-Misconfiguration, A06-Vulnerability Components, A09-Security Logging	Currently violate multiple top-10 categories
NIST SP 800-53	SI-10 (Input Validation), CM-6 (Configuration Management), RA-5 (Vulnerability Management)	Needs improved defense and patching
ISO 27001:2022	A.14.2 (Secure System Engineering), A.8.8 (Configuration Management)	Hardcoded keys and unsafe code violate secure coding norms
PCI DSS 4.0	6.2 / 6.3.2 (secure coding / patching), 3.6 (key management)	Sensitive data handling and key exposure risk non-compliance
SOC 2 – Security	CC6.1, CC6.6	Demonstrate vulnerability management and secure configuration

Key Hardening Checklist

Area	Action Item	Area
☐ Secrets & Keys	Move all secrets out of code into secure vaults/env. Rotate keys.	Secrets & Keys
☐ Input Validation	Use a validation library (Joi, zod, express-validator). Block unexpected fields.	Input Validation
☐ Safe Querying	Convert IDs to typed objects; forbid direct insertion of request body into queries.	Safe Querying
☐ File System Access	Always use path.resolve; verify target path is within allowed base; restrict file types.	File System Access
☐ Dependency Hygiene	Remove unmaintained modules; prefer actively maintained ones.	Dependency Hygiene
☐ Network Controls	Deny outbound traffic by default; whitelist only needed services.	Network Controls
☐ Runtime Hardening	Use process-level sandboxing, container isolation, seccomp.	Runtime Hardening
☐ Monitoring & Alerts	Use Snyk or other scanners; alert on new critical issues.	Monitoring & Alerts

Final Summary

My closed pull requests have already plugged many lower-level holes (especially via bulk dependency upgrades) which is excellent progress. But major risks remain:

- RCE / sandbox escape via vm2 and marsdb
- Hardcoded secrets embedded in your code
- NoSQL injection paths
- Path traversal allowing file overwrite
- SSRF in express-ipfilter

These require manual code remediation, dependency removal or replacement, and strong runtime controls