

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

<div><div>C</div><div>juicy-chat-bot@0.9.0</div><div>4 transitive issues</div></div> <div><div>Fixable issues</div><div>Issues with no supported fix</div><div>Vulnerable dependencies</div></div> <div><div><div>></div><div>C</div><div>Remote Code Execution (RCE)</div><div>↔</div><div>vm2@3.9.17</div><div></div><div>CWE-94</div><div>CVSS 9.8</div><div>661</div><div>...</div></div><div><div>></div><div>C</div><div>Remote Code Execution (RCE)</div><div>↔</div><div>vm2@3.9.17</div><div></div><div>CWE-94</div><div>CVSS 9.8</div><div>597</div><div>...</div></div><div><div>></div><div>C</div><div>Sandbox Bypass</div><div>↔</div><div>vm2@3.9.17</div><div></div><div>CWE-265</div><div>CVSS 9.8</div><div>597</div><div>...</div></div></div> <div><div>Show more issues</div></div>	<div>PRIORITY SCORE (MAX)</div> <div>661</div>
<div><div>H</div><div>libxmljs2@0.37.0</div><div>2 direct issues</div></div> <div><div>Fixable issues</div><div>Issues with no supported fix</div><div>Vulnerable dependencies</div></div> <div><div><div>></div><div>H</div><div>Type Confusion</div><div></div><div>CWE-843</div><div>CVSS 8.1</div><div>512</div><div>...</div></div><div><div>></div><div>H</div><div>Type Confusion</div><div></div><div>CWE-843</div><div>CVSS 8.1</div><div>512</div><div>...</div></div></div>	<div>PRIORITY SCORE (MAX)</div> <div>512</div>
<div><div>H</div><div>express-ipfilter@1.3.2</div><div>3 transitive issues</div></div> <div><div>Fixable issues</div><div>Issues with no supported fix</div><div>Vulnerable dependencies</div></div> <div><div><div>></div><div>H</div><div>Server-side Request Forgery (SSRF)</div><div>↔</div><div>ip@2.0.1</div><div></div><div>CWE-918</div><div>CVSS 7.9</div><div>502</div><div>...</div></div><div><div>></div><div>H</div><div>Server-side Request Forgery (SSRF)</div><div>↔</div><div>ip@2.0.1</div><div></div><div>CWE-918</div><div>CVSS 7.9</div><div>502</div><div>...</div></div><div><div>></div><div>M</div><div>Server-Side Request Forgery (SSRF)</div><div>↔</div><div>ip@2.0.1</div><div></div><div>CWE-918</div><div>CVSS 6.5</div><div>432</div><div>...</div></div></div>	<div>PRIORITY SCORE (MAX)</div> <div>502</div>
<div><div>C</div><div>marsdb@0.6.11</div><div>1 direct issue</div></div> <div><div>Fixable issues</div><div>Issues with no supported fix</div><div>Vulnerable dependencies</div></div> <div><div><div>></div><div>C</div><div>Arbitrary Code Injection</div><div></div><div>CWE-94</div><div>CVSS 9.8</div><div>490</div><div>...</div></div></div>	<div>PRIORITY SCORE (MAX)</div> <div>490</div>
<div><div>H</div><div>grunt-replace-json@0.1.0</div><div>1 transitive issue</div></div> <div><div>Fixable issues</div><div>Issues with no supported fix</div><div>Vulnerable dependencies</div></div> <div><div><div>></div><div>H</div><div>Prototype Pollution</div><div>↔</div><div>lodash.set@4.3.2</div><div></div><div>CWE-1321</div><div>CVSS 7.3</div><div>472</div><div>...</div></div></div>	<div>PRIORITY SCORE (MAX)</div> <div>472</div>
<div><div>M</div><div>notevil@1.3.3</div><div>1 direct issue</div></div> <div><div>Fixable issues</div><div>Issues with no supported fix</div><div>Vulnerable dependencies</div></div> <div><div><div>></div><div>M</div><div>Sandbox Bypass</div><div></div><div>CWE-1321 + 1 MORE</div><div>CVSS 6.5</div><div>432</div><div>...</div></div></div>	<div>PRIORITY SCORE (MAX)</div> <div>432</div>
<div><div>M</div><div>node-pre-gyp@0.15.0</div><div>2 transitive issues</div></div> <div><div>Fixable issues</div><div>Issues with no supported fix</div><div>Vulnerable dependencies</div></div> <div><div><div>></div><div>M</div><div>Uncontrolled Resource Consumption (Resource Exhaustion)</div><div>↔</div><div>tar@4.4.19</div><div></div><div>CWE-400</div><div>CVSS 4.5</div><div>432</div><div>...</div></div><div><div>></div><div>M</div><div>Missing Release of Resource after Effective Lifetime</div><div>↔</div><div>utf8@3.1.0.6</div><div></div><div>CWE-772</div><div>CVSS 6.2</div><div>417</div><div>...</div></div></div>	<div>PRIORITY SCORE (MAX)</div> <div>432</div>

download@8.0.0

3 transitive issues

Fixable issues

Issues with no supported fix

Vulnerable dependencies

> Arbitrary File Write via Archive Extraction (Zip Slip)

decompress-tar@4.1.1

CWE-29 CVSS 6.3 422

> Regular Expression Denial of Service (ReDoS)

http-cache-semantics@3.8.1

CWE-1333 CVSS 5.3 372

> Open Redirect

got@8.3.2

CWE-601 CVSS 5.4 270

grunt@1.6.1

1 transitive issue

Fixable issues

Issues with no supported fix

Vulnerable dependencies

> Missing Release of Resource after Effective Lifetime

inflight@1.0.6

CWE-772 CVSS 6.2 417

ts-node-dev@1.1.8

1 transitive issue

Fixable issues

Issues with no supported fix

Vulnerable dependencies

> Missing Release of Resource after Effective Lifetime

inflight@1.0.6

CWE-772 CVSS 6.2 417

Vulnerabilities in Code Analysis

Hardcoded Secret

SNYK CODE | CWE-547

SCORE

802

```
18 // eslint-disable-next-line @typescript-eslint/prefer-ts-expect-error
19 // @ts-expect-error FIXME no typescript definitions for z85 :(
20 import * as z85 from 'z85'
21
22 export const publicKey = fs ? fs.readFileSync('encryptionkeys/jwt.pub', 'utf8') : 'placeholder-public-key'
```

Hardcoded value is used as a cipher key (in jsonwebtoken.default.verify). Generate the value with a cryptographically strong random number generator and do not hardcode it in source code.
lib/insecurity.ts

2 steps in 1 file

Ignore across repository

View details

Hardcoded Secret

SNYK CODE | CWE-547

SCORE

802

```
19 // @ts-expect-error FIXME no typescript definitions for z85 :(
20 import * as z85 from 'z85'
21
22 export const publicKey = fs ? fs.readFileSync('encryptionkeys/jwt.pub', 'utf8') : 'placeholder-public-key'
23 const privateKey = '-----BEGIN RSA PRIVATE KEY-----\nMIICBAIBAAKCCQNuqTeWqTXC8C7HPD0bDeq\n-----\n0s440P912zqLpXvJX1xw0HrE6PH8NUqT5ia+ndvJNn2Rc45uV8JJCA4320K08TPEZz/65gY3B8B06syCmUP4q5d6eou/FwtIS2bQSFbVPWnHhG/kpvt/CLxK3Ln3hN+4tQ1DAQ4Aa2A1+bx1P0r7AeH'
```

Hardcoded value is used as a cipher key (in node.crypto.default.createHmac). Generate the value with a cryptographically strong random number generator and do not hardcode it in source code.
lib/insecurity.ts

2 steps in 1 file

Ignore across repository

View details

Hardcoded Secret

SNYK CODE | CWE-547

SCORE

802

```
40 | updateFrom: (req: Request, user: ResponseWithUser) => any
41 | }
42
43 export const hash = (data: string) => crypto.createHash('md5').update(data).digest('hex')
44 export const hmac = (data: string) => crypto.createHmac('sha256', 'pa4qacea4W9t9n6vZyZhmj').update(data).digest('hex')
```

Hardcoded value is used as a cipher key (in node.crypto.default.createHmac). Generate the value with a cryptographically strong random number generator and do not hardcode it in source code.
lib/insecurity.ts

2 steps in 1 file

Ignore across repository

View details

NoSQL Injection

SNYK CODE | CWE-943

SCORE

802

```
21 | return res.status(401).json({ error: 'Unauthorized' })
22 | }
23
24 | try {
25 |   const review = await db.reviewsCollection.findOne({ _id: id })
```

Unsanitized input from the HTTP request body flows into findOne, where it is used in a NoSQL query. This may result in a NoSQL injection vulnerability.
routes/likeProductReviews.ts

8 steps in 1 file

NoSQL Injection

SNYK CODECWE-943

SCORE802

```
40 // Artificial wait for timing attack challenge
41 await sleep(150)
42 try {
43   const updatedReview: Review = await db.reviewsCollection.findOne({ _id: id })

```

Unsanitized input from the HTTP request body flows into findOne, where it is used in an NoSQL query. This may result in an NoSQL Injection vulnerability.

[routes/likeProductReviews.ts](#)

8 steps in 1 file

Learn about this type of vulnerability and how to fix it

Ignore across repository

View details

NoSQL Injection

SNYK CODECWE-943

SCORE802

```
31 if (likedBy.includes(user.data.email)) {
32   return res.status(403).json({ error: 'Not allowed' })
33 }
34
35 await db.reviewsCollection.update(

```

Unsanitized input from the HTTP request body flows into update, where it is used in an NoSQL query. This may result in an NoSQL Injection vulnerability.

[routes/likeProductReviews.ts](#)

8 steps in 1 file

NoSQL Injection

SNYK CODECWE-943

SCORE802

```
47 const count = updated.likedBy.filter(email => email === user.data.email).length
48 challengeUtils.solveIf(challenges.timingAttackChallenge, () => count > 2)
49
50 const result = await db.reviewsCollection.update(

```

Unsanitized input from the HTTP request body flows into update, where it is used in an NoSQL query. This may result in an NoSQL Injection vulnerability.

[routes/likeProductReviews.ts](#)

9 steps in 1 file

Learn about this type of vulnerability and how to fix it

Ignore across repository

View details

Path Traversal

SNYK CODECWE-23

SCORE802

```
25 if (!response.ok || !response.body) {
26   throw new Error('url returned a non-OK status code or an empty body')
27 }
28 const ext = ['jpg', 'jpeg', 'png', 'svg', 'gif'].includes(url.split('.').slice(-1)[0].toLowerCase()) ? url.split('.').slice(-1)[0].toLowerCase() : 'jpg'
29 const fileStream = fs.createWriteStream('frontend/dist/frontend/assets/public/images/uploads/${loggedUser.data.id}.${ext}', { flags: 'w' })

```

Unsanitized input from cookies flows into node:fs.default.createWriteStream, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

[routes/profileImageUrlUpload.ts](#)

14 steps in 1 file

Vulnerabilities in frontend/package.json

codemirror-solidity@0.2.5

1 transitive issue

Fixable issuesIssues with no supported fixVulnerable dependencies

PRIORITY SCORE (MAX)452

Regular Expression Denial of Service (ReDoS)

codemirror@5.65.20

CWE-1333CVSS 6.9452

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