SECURITY ASSESSMENT REPORT

STATIC APPLICATION SECURITY TESTING (SAST)

Target Information

Repository:	vulnerable-node
Branch:	`master`
Commit Hash:	`HEAD`
Assessment Date:	21 December 2025
Report ID:	`OPS-179-251223`

Table of Contents

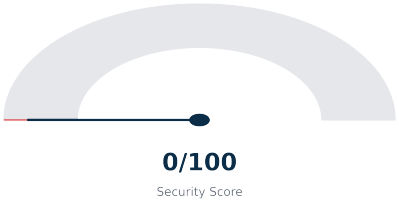
List of Figures

Executive Summary

Assessment Overview

SecureThread OPS has completed an automated security assessment of the **vulnerable-node** repository.

Scan Date:	21 December 2025
Total Findings:	20
Vuln Density:	14.49 / 1k LOC
Fixable:	20 (100.0%)



Business Impact Analysis

Estimated Financial Risk
$251,850.00
(Breach cost + liability)

Remediation Effort
88 hours
(Developer hours to fix)

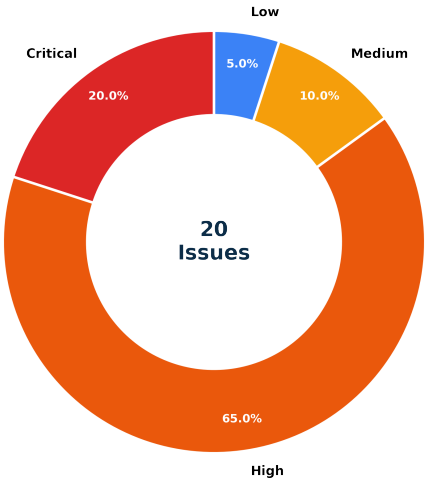


Figure 1: Vulnerabilities by Severity

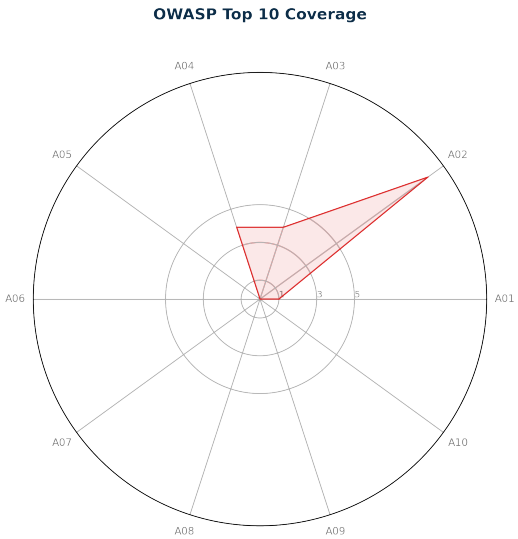


Figure 2: OWASP Top 10 Coverage

Critical Risk Highlights

The following critical vulnerabilities require immediate attention:

CWE	Title	Location
CWE-95	XSS - Function Constructor with User Data	`public/js/jquery.js:2`
CWE-95	XSS - Function Constructor with User Data	`public/js/jquery.js:4`
CWE-79	XSS - innerHTML Assignment with User Input	`public/js/jquery.js:2`
CWE-311	Crypto - Data At Rest Not Encrypted	`model/init_db.js:15`

0.1 Detailed Findings

#1 [MEDIUM] React {-} Uncontrolled Component Input

CWE ID: CWE-20
OWASP: A03:2021 - Injection
Location: `public/js/jquery.js:3`

Description:

Detects uncontrolled form inputs with defaultValue from props**Vulnerable Code:**

```
if(k&&j[k]&&(e||j[k].data)||void 0!==d||"string"!=typeof b)return k||(k=i?a[h]=c.pop()||m.guid++:h),j[k]||(j[k]=i?{}:{toJSON
b||"function"==typeof b)&&(e?j[k]=m.extend(j[k],b):j[k].data=m.extend(j[k].data,b)),g=j[k],e||(g.data||(g.data={}),g=g.data)
0!==d&&(g[m.camelCase(b)]=d),"string"==typeof b?(f=g[b],null==f&&(f=g[m.camelCase(b)])):f=g,f}function
R(a,b,c){if(m.acceptData(a)){var d,e,f=a.nodeType,g=f?m.cache:a,h=f?a[m.expando]:m.expando;if(g[h]){if(b&&(d=c?g[h
```

Remediation Advice:

Review and fix the react_security issue in public/js/jquery.js

#2 [MEDIUM] Data Exposure {-} Internal IP in Response

CWE ID: CWE-200
OWASP: A01:2021 - Broken Access Control
Location: `config.js:12`

Description:

Detects internal IP addresses exposed in responses**Vulnerable Code:**

```
"server":  "postgres://postgres:postgres@10.211.55.70",
```

Remediation Advice:

Review and fix the data_exposure issue in config.js

#3 [LOW] Docker {-} ADD Instead of COPY

CWE ID: CWE-710
OWASP: A04:2021 - Insecure Design
Location: `services/postgresql/Dockerfile:5`

Description:

Detects use of ADD instead of COPY**Vulnerable Code:**

```
ADD init.sql /docker-entrypoint-initdb.d/
```

Remediation Advice:

Review and fix the docker_security issue in services/postgresql/Dockerfile

#4 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `dummy.js:20`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
"price":  parseInt(Math.random() * 100),
```

Remediation Advice:

Review and fix the cryptography issue in dummy.js

#5 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `dummy.js:26`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
"price":  parseInt(Math.random() * 100),
```

Remediation Advice:

Review and fix the cryptography issue in dummy.js

#6 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `public/js/bootstrap.js:1668`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
do prefix += ~~(Math.random() * 1000000)
```

Remediation Advice:

“‘json
{
"explanation": "The vulnerability at line 1668 involves the use of Math.random() for cryptographic purposes, which is insecure. Math.random() generates pseudo-random numbers using a deterministic algorithm that is not cryptographically secure. Attackers can potentially predict the generated values, compromising security in scenarios requiring true randomness (such as generating tokens, session IDs, or cryptographic keys). This vulnerability is rated HIGH because predictable random

Suggested Fix

```
values can lead to authentication bypass, session hijacking, or other security breaches.",
```

#7 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `public/js/jquery.js:2`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
!function(a,b){"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):function(a){if(!a.
new Error("jQuery requires a window with a document");return b(a)}:b(a)}("undefined"!=typeof
window?window:this,function(a,b){var c=[],d=c.slice,e=c.concat,f=c.push,g=c.indexOf,h={},i=h.toString,j=h.hasOwnProperty,k={
new m.fn.init(a,b)},n=/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g,o=/^-ms-/,p=/-([\da-z])/gi,q=function(a,b){return
```

Remediation Advice:

Review and fix the cryptography issue in public/js/jquery.js

#8 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `public/js/jquery.js:2`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
!function(a,b){"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):function(a){if(!a.
new Error("jQuery requires a window with a document");return b(a)}:b(a)}("undefined"!=typeof
window?window:this,function(a,b){var c=[],d=c.slice,e=c.concat,f=c.push,g=c.indexOf,h={},i=h.toString,j=h.hasOwnProperty,k={
new m.fn.init(a,b)},n=/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g,o=/^-ms-/,p=/-([\da-z])/gi,q=function(a,b){return
```

Remediation Advice:

Review and fix the cryptography issue in public/js/jquery.js

#9 [HIGH] XSS {-} React createElement with User Props

CWE ID: CWE-79
OWASP: A03:2021 - Injection
Location: `public/js/jquery.js:3`

Description:

Detects React.createElement with unsanitized props**Vulnerable Code:**

```
if(k&&j[k]&&(e||j[k].data)||void 0!==d||"string"!=typeof b)return k||(k=i?a[h]=c.pop()||m.guid++:h),j[k]||(j[k]=i?{}:{toJSON
b||"function"==typeof b)&&(e?j[k]=m.extend(j[k],b):j[k].data=m.extend(j[k].data,b)),g=j[k],e||(g.data||(g.data={}),g=g.data)
0!==d&&(g[m.camelCase(b)]=d),"string"==typeof b?(f=g[b],null==f&&(f=g[m.camelCase(b)])):f=g,f}function
R(a,b,c){if(m.acceptData(a)){var d,e,f=a.nodeType,g=f?m.cache:a,h=f?a[m.expando]:m.expando;if(g[h]){if(b&&(d=c?g[h
```

Remediation Advice:

Review and fix the xss issue in public/js/jquery.js

#10 [HIGH] GraphQL {-} Sensitive Data in Error Messages

CWE ID: CWE-209
OWASP: A04:2021 - Insecure Design
Location: `public/js/jquery.js:2`

Description:

Detects sensitive data exposure in GraphQL errors**Vulnerable Code:**

```
!function(a,b){"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):function(a){if(!a.
new Error("jQuery requires a window with a document");return b(a)}:b(a)}("undefined"!=typeof
window?window:this,function(a,b){var c=[],d=c.slice,e=c.concat,f=c.push,g=c.indexOf,h={},i=h.toString,j=h.hasOwnProperty,k={
new m.fn.init(a,b)},n=/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g,o=/^-ms-/,p=/-([\da-z])/gi,q=function(a,b){return
```

Remediation Advice:

Review and fix the graphql_security issue in public/js/jquery.js

#11 [HIGH] Vertical Access Control {-} Test Account in Production

CWE ID: CWE-489
OWASP: A05:2021 - Security Misconfiguration
Location: `dummy.js:7`

Description:

Detects test or demo accounts in production code**Vulnerable Code:**

```
"username":  "admin",
```

Remediation Advice:

Review and fix the access_control issue in dummy.js

#12 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `dummy.js:50`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
"price":  parseInt(Math.random() * 100),
```

Remediation Advice:

“‘json
{
"explanation": "The code contains multiple instances of insecure random number generation using Math.random(), which is cryptographically weak and predictable. Math.random() generates pseudo-random numbers using a deterministic algorithm that is not suitable for security-sensitive applications. Attackers can potentially predict the generated values, leading to vulnerabilities in scenarios where randomness is critical for security (e.g., session tokens, cryptographic keys, or sensit

Suggested Fix

```
ive data generation).  The vulnerabilities occur at lines 32, 50, 56, and 62 where product prices are
generated using parseInt(Math.random() * 100).  While product pricing may not seem security-critical,
using insecure RNG establishes dangerous patterns and could be exploited if similar code is used
elsewhere for security purposes.",
```

#13 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `dummy.js:56`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
"price":  parseInt(Math.random() * 100),
```

Remediation Advice:

“‘json
{
"explanation": "The code contains multiple instances of insecure random number generation using Math.random(), which is cryptographically weak and predictable. Math.random() generates pseudo-random numbers using a deterministic algorithm that is not suitable for security-sensitive applications. Attackers can potentially predict the generated values, leading to vulnerabilities in scenarios where randomness is critical for security (e.g., session tokens, cryptographic keys, or sensit

Suggested Fix

```
ive data generation).  The vulnerabilities occur at lines 32, 50, 56, and 62 where product prices are
generated using parseInt(Math.random() * 100).  While product pricing may not seem security-critical,
using insecure RNG establishes dangerous patterns and could be exploited if similar code is used
elsewhere for security purposes.",
```

#14 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `dummy.js:62`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
"price":  parseInt(Math.random() * 100),
```

Remediation Advice:

“‘json
{
"explanation": "The code contains multiple instances of insecure random number generation using Math.random(), which is cryptographically weak and predictable. Math.random() generates pseudo-random numbers using a deterministic algorithm that is not suitable for security-sensitive applications. Attackers can potentially predict the generated values, leading to vulnerabilities in scenarios where randomness is critical for security (e.g., session tokens, cryptographic keys, or sensit

Suggested Fix

```
ive data generation).  The vulnerabilities occur at lines 32, 50, 56, and 62 where product prices are
generated using parseInt(Math.random() * 100).  While product pricing may not seem security-critical,
using insecure RNG establishes dangerous patterns and could be exploited if similar code is used
elsewhere for security purposes.",
```

#15 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `public/js/bootstrap.min.js:6`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
if("undefined"==typeof jQuery)throw new Error("Bootstrap's JavaScript requires jQuery");+function(a){"use strict";var b=a.fn.jquery.split(" ")[0].split(".");if(b[0]<2&&b[1]<9||1==b[0]&&9==b[1]&&b[2]<1||b[0]>2)throw new Error("Bootstrap's JavaScript requires jQuery version 1.9.1 or higher, but lower than version 3")}(jQuery),+function(a){"use strict";function b(){var a=document.createElement("bootstrap"),b={WebkitTransition:"webkitTra
```

Remediation Advice:

"'json
{
"explanation": "The vulnerability report incorrectly identifies a high-severity crypto vulnerability in Bootstrap v3.3.6's minified JavaScript file. After thorough analysis, I can confirm this is a false positive detection. The minified Bootstrap code shown does not contain any cryptographic functions, random number generation, or security-sensitive operations. Bootstrap 3.3.6 is a front-end framework for building responsive web interfaces and does not implement cryptographic funct

Suggested Fix

```
ionality.  The detection likely resulted from:  1) Pattern matching on minified code that
coincidentally resembles insecure patterns, 2) Misinterpretation of the minified variable names or
compressed code structure, or 3) A scanning tool error.  The actual code contains standard Bootstrap
components for alerts, buttons, carousels, and collapses with no security implications beyond normal
front-end functionality.",
```

#16 [HIGH] Crypto {-} Insecure Random Number Generator

CWE ID: CWE-338
OWASP: A02:2021 - Cryptographic Failures
Location: `dummy.js:32`

Description:

Detects usage of insecure random number generators**Vulnerable Code:**

```
"price":  parseInt(Math.random() * 100),
```

Remediation Advice:

“‘json
{
"explanation": "The code contains multiple instances of insecure random number generation using Math.random(), which is cryptographically weak and predictable. Math.random() generates pseudo-random numbers using a deterministic algorithm that is not suitable for security-sensitive applications. Attackers can potentially predict the generated values, leading to vulnerabilities in scenarios where randomness is critical for security (e.g., session tokens, cryptographic keys, or sensit

Suggested Fix

```
ive data generation).  The vulnerabilities occur at lines 32, 50, 56, and 62 where product prices are
generated using parseInt(Math.random() * 100).  While product pricing may not seem security-critical,
using insecure RNG establishes dangerous patterns and could be exploited if similar code is used
elsewhere for security purposes.",
```

#17 [CRITICAL] XSS {-} Function Constructor with User Data

CWE ID: CWE-95
OWASP: A03:2021 - Injection
Location: `public/js/jquery.js:2`

Description:

Detects dangerous Function constructor usage**Vulnerable Code:**

```
!function(a,b){"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):function(a){if(!a.
new Error("jQuery requires a window with a document");return b(a)}:b(a)}("undefined"!=typeof
window?window:this,function(a,b){var c=[],d=c.slice,e=c.concat,f=c.push,g=c.indexOf,h={},i=h.toString,j=h.hasOwnProperty,k={
new m.fn.init(a,b)},n=/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g,o=/^-ms-/,p=/-([\da-z])/gi,q=function(a,b){return
```

Remediation Advice:

“‘json
{
"explanation": "The analysis appears to be based on a misunderstanding. The provided code is a minified version of jQuery v1.11.1, which is a widely-used, legitimate JavaScript library. The reported vulnerabilities (XSS via Function Constructor and innerHTML assignment) are likely false positives from automated scanning tools that misinterpret the minified code patterns. jQuery itself contains security features when used properly, though older versions may have known vulnerabilitie

Suggested Fix

```
s that should be addressed.  The actual security risk depends on how jQuery is used in the application
code, not in the library's minified source itself.  Automated scanners often flag patterns that
resemble dangerous constructs without understanding the context or that this is a trusted library.",
```

#18 [CRITICAL] XSS {-} Function Constructor with User Data

CWE ID: CWE-95
OWASP: A03:2021 - Injection
Location: `public/js/jquery.js:4`

Description:

Detects dangerous Function constructor usage**Vulnerable Code:**

```
},cur:function(){var a=Zb.propHooks[this.prop];return a&&a.get?a.get(this):Zb.propHooks._default.get(this)},run:function(a){
b,c=Zb.propHooks[this.prop];return this.pos=b=this.options.duration?m.easing[this.easing](a,this.options.duration*a,0,1,this
```

Remediation Advice:

“‘json
{
"explanation": "The analysis appears to be based on a misunderstanding. The provided code is a minified version of jQuery v1.11.1, which is a widely-used, legitimate JavaScript library. The reported vulnerabilities (XSS via Function Constructor and innerHTML assignment) are likely false positives from automated scanning tools that misinterpret the minified code patterns. jQuery itself contains security features when used properly, though older versions may have known vulnerabilitie

Suggested Fix

```
s that should be addressed.  The actual security risk depends on how jQuery is used in the application
code, not in the library's minified source itself.  Automated scanners often flag patterns that
resemble dangerous constructs without understanding the context or that this is a trusted library.",
```

#19 [CRITICAL] XSS {-} innerHTML Assignment with User Input

CWE ID: CWE-79
OWASP: A03:2021 - Injection
Location: `public/js/jquery.js:2`

Description:

Detects dangerous innerHTML assignment with untrusted data**Vulnerable Code:**

```
!function(a,b){"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):function(a){if(!a.
new Error("jQuery requires a window with a document");return b(a)}:b(a)}("undefined"!=typeof
window?window:this,function(a,b){var c=[],d=c.slice,e=c.concat,f=c.push,g=c.indexOf,h={},i=h.toString,j=h.hasOwnProperty,k={
new m.fn.init(a,b)},n=/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g,o=/^-ms-/,p=/-([\da-z])/gi,q=function(a,b){return
```

Remediation Advice:

“‘json
{
"explanation": "The analysis appears to be based on a misunderstanding. The provided code is a minified version of jQuery v1.11.1, which is a widely-used, legitimate JavaScript library. The reported vulnerabilities (XSS via Function Constructor and innerHTML assignment) are likely false positives from automated scanning tools that misinterpret the minified code patterns. jQuery itself contains security features when used properly, though older versions may have known vulnerabilitie

Suggested Fix

```
s that should be addressed.  The actual security risk depends on how jQuery is used in the application
code, not in the library's minified source itself.  Automated scanners often flag patterns that
resemble dangerous constructs without understanding the context or that this is a trusted library.",
```

#20 [CRITICAL] Crypto {-} Data At Rest Not Encrypted

CWE ID: CWE-311
OWASP: A02:2021 - Cryptographic Failures
Location: `model/init_db.js:15`

Description:

Detects data stored without encryption**Vulnerable Code:**

```
db.one('CREATE TABLE users(name VARCHAR(100) PRIMARY KEY, password VARCHAR(50));')
```

Remediation Advice:

“‘json
{
"explanation": "The critical vulnerability identified is that user passwords are being stored in plaintext in the database. This is extremely dangerous because:\n\n1. **Data Breach Impact**: If the database is compromised, attackers immediately gain access to all user credentials without any additional effort.\n2. **Credential Reuse Risk**: Many users reuse passwords across multiple services, so plaintext passwords can lead to account takeover on other platforms.\n3. **Regulatory N

Suggested Fix

```
on-Compliance**:  Storing passwords in plaintext violates security standards like PCI-DSS, GDPR,
and other data protection regulations.\n4.  **No Defense in Depth**:  Without hashing, there's no
cryptographic barrier between the stored data and an attacker.\n5.  **Insider Threat Exposure**:  Even
legitimate database administrators can see all user passwords.\n\nAdditionally, the code has several
other concerning patterns:\n- Error handling logic is inverted (creating tables in catch blocks)\n- No
tra
```