




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 Java


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



JavaScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVASCRIPT code

All rules 285

Vulnerability 29

Bug 62

Security Hotspot 43

Code Smell 151

Quick Fix 41

Tags ▾

Search by name... 🔍

Expanding archive files without controlling resource consumption is security-sensitive

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot

Disabling CSRF protections is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is security-sensitive

Security Hotspot

Dynamically executing code is security-sensitive

Security Hotspot

Equality operators should not be used in "for" loop termination conditions

Code Smell

Tests should not execute any code after "done()" is called

Code Smell

"default" clauses should be last

Code Smell

"await" should only be used with promises

Code Smell

A conditionally executed single line should be denoted by indentation

Code Smell

Conditionals should start on new lines

Code Smell

Cognitive Complexity of functions should not be too high

JWT should be signed and verified with strong cipher algorithms

Analyze your code

Vulnerability Critical cwe privacy owasp

If a JSON Web Token (JWT) is not signed with a strong cipher algorithm (or not signed at all) an attacker can forge it and impersonate user identities.

- Don't use none algorithm to sign or verify the validity of a token.
- Don't use a token without verifying its signature before.

Noncompliant Code Example

jsonwebtoken library:

```
const jwt = require('jsonwebtoken');

let token = jwt.sign({ foo: 'bar' }, key, { algorithm: 'none' });

jwt.verify(token, key, { expiresIn: 360000 * 5, algorithms: 'none' });
```

Compliant Solution

jsonwebtoken library:

```
const jwt = require('jsonwebtoken');

let token = jwt.sign({ foo: 'bar' }, key, { algorithm: 'HS256' });

jwt.verify(token, key, { expiresIn: 360000 * 5, algorithms: 'HS256' });
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- MITRE, CWE-347 - Improper Verification of Cryptographic Signature

Available In:






sonarlint | sonarcloud | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

[Privacy Policy](#)

https://rules.sonarsource.com/javascript/RSPEC-5659

1/2

 Code Smell
"void" should not be used  Code Smell
Loop counters should not be assigned to from within the loop body  Code Smell
"for" loop increment clauses should modify the loops' counters  Code Smell
Functions should not be empty  Code Smell