sonar RULES

Products ⌄

- 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...

Object literal shorthand syntax should be used
Code Smell

Strings and non-strings should not be added
Code Smell

Object literal syntax should be used
Code Smell

"undefined" should not be assigned
Code Smell

Trailing commas should not be used
Code Smell

Array constructors should not be used
Code Smell

Quotes for string literals should be used consistently
Code Smell

Statements should end with semicolons
Code Smell

Comments should not be located at the end of lines of code
Code Smell

Loops should not contain more than a single "break" or "continue" statement
Code Smell

Variable, property and parameter names should comply with a naming convention
Code Smell

Lines should not end with trailing whitespaces
Code Smell

Allowing mixed-content is security-sensitive

Analyze your code

Security Hotspot ✓ Minor ? express.js owasp

A mixed-content is when a resource is loaded with the HTTP protocol, from a website accessed with the HTTPs protocol, thus mixed-content are not encrypted and exposed to MITM attacks and could break the entire level of protection that was desired by implementing encryption with the HTTPs protocol.

The main threat with mixed-content is not only the confidentiality of resources but the whole website integrity:

- A passive mixed-content (eg: *<img src="http://example.com/picture.png">*) allows an attacker to access and replace only these resources, like images, with malicious ones that could lead to successful phishing attacks.
- With active mixed-content (eg: *<script src="http://example.com/library.js">*) an attacker can compromise the entire website by injecting malicious javascript code for example (accessing and modifying the DOM, steal cookies, etc).

Ask Yourself Whether

- The HTTPS protocol is in place and external resources are fetched from the website pages.

There is a risk if you answered yes to this question.

Recommended Secure Coding Practices

Implement content security policy *block-all-mixed-content* directive which is supported by all modern browsers and will block loading of mixed-contents.

Sensitive Code Example

In Express.js application the code is sensitive if the helmet-csp or helmet middleware is used without the `blockAllMixedContent` directive:

```
const express = require('express');
const helmet = require('helmet');

let app = express();

app.use(
  helmet.contentSecurityPolicy({
    directives: {
      "default-src": ["'self'", 'example.com', 'code.jquery.
    } // Sensitive: blockAllMixedContent directive is missin
  })
);
```

Compliant Solution

In Express.js application a standard way to block mixed-content is to put in place the helmet-csp or helmet middleware with the `blockAllMixedContent` directive:

Files should contain an empty newline at the end

Code Smell

An open curly brace should be located at the end of a line

Code Smell

Tabulation characters should not be used

Code Smell

Function and method names should comply with a naming convention

Code Smell

```
const express = require('express');
const helmet = require('helmet');

let app = express();

app.use(
  helmet.contentSecurityPolicy({
    directives: {
      "default-src": ["'self'", 'example.com', 'code.jquery.
      blockAllMixedContent: [] // Compliant
    }
  })
);
```

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- developer.mozilla.org - Mixed-content
- developer.mozilla.org - Content Security Policy (CSP)
- w3.org - Content Security Policy Level 3

Available In:

sonarcloud sonarqube

Privacy Policy