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

JS

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...

Vulnerability

NoSQL operations should not be vulnerable to injection attacks

Vulnerability

HTTP request redirections should not be open to forging attacks

Vulnerability

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks

Vulnerability

Database queries should not be vulnerable to injection attacks

Vulnerability

XML parsers should not be vulnerable to XXE attacks

Vulnerability

I/O function calls should not be vulnerable to path injection attacks

Vulnerability

OS commands should not be vulnerable to command injection attacks

Vulnerability

Callbacks of array methods should have return statements

Bug

Loops should not be infinite

Bug

Disabling Vue.js built-in escaping is security-sensitive

Security Hotspot

Disabling Angular built-in sanitization is security-sensitive

Security Hotspot

Extracting archives should not lead to zip slip vulnerabilities

Analyze your code

Vulnerability | Blocker | injection cwe owasp sans-top25

File names of the entries in a zip archive should be considered untrusted, tainted and should be validated before being used for file system operations. Indeed, file names can contain specially crafted values, such as '../', that change the initial path and, when accessed, resolve to a path on the filesystem where the user should normally not have access.

A successful attack might give an attacker the ability to read, modify, or delete sensitive information from the file system and sometimes even execute arbitrary operating system commands. This special case of path injection vulnerabilities is called "zip slip".

The mitigation strategy should be based on the whitelisting of allowed paths or characters.

Noncompliant Code Example

```
const AdmZip = require('adm-zip');
const fs = require('fs');

const zip = new AdmZip("zip-slip.zip");
const zipEntries = zip.getEntries();
zipEntries.forEach(function (zipEntry) {
  fs.createWriteStream(zipEntry.entryName); // Noncompliant
});
```

Compliant Solution

```
const AdmZip = require('adm-zip');
const pathmodule = require('path');
const fs = require('fs');

const zip = new AdmZip("zip-slip.zip");
const zipEntries = zip.getEntries();
zipEntries.forEach(function (zipEntry) {
  let resolvedPath = pathmodule.join(__dirname + '/archive_t

  if (resolvedPath.startsWith(__dirname + '/archive_tmp')) {
    // the file cannot be extracted outside of the "archive_
    fs.createWriteStream(resolvedPath); // Compliant
  }
});
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection
- snyk - Zip Slip Vulnerability
- MITRE, CWE-20 - Improper Input Validation

Hard-coded credentials are security-sensitive

Security Hotspot

Function returns should not be invariant

Code Smell

Assertions should be complete

Code Smell

Variables should be declared explicitly

Code Smell

Tests should include assertions

- MITRE, CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- MITRE, CWE-99 - Improper Control of Resource Identifiers ('Resource Injection')
- MITRE, CWE-641 - Improper Restriction of Names for Files and Other Resources
- SANS Top 25 - Risky Resource Management

Available In:

sonarcloud sonarqube Developer Edition