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

TypeScript static code analysis

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

All rules 279 | Vulnerability 27 | Bug 51 | Security Hotspot 43 | Code Smell 158 | Quick Fix 50

Tags

Search by name...

The global "this" object should not be used
Code Smell

"catch" clauses should do more than rethrow
Code Smell

Boolean checks should not be inverted
Code Smell

Deprecated APIs should not be used
Code Smell

Wrapper objects should not be used for primitive types
Code Smell

Multiline string literals should not be used
Code Smell

Local variables should not be declared and then immediately returned or thrown
Code Smell

Function call arguments should not start on new lines
Code Smell

"switch" statements should have at least 3 "case" clauses
Code Smell

A "while" loop should be used instead of a "for" loop
Code Smell

Unnecessary imports should be removed
Code Smell

Boolean literals should not be used in comparisons

Setting loose POSIX file permissions is security-sensitive

Analyze your code

Security Hotspot Major cwe sans-top25 owasp

In Unix, "`others`" class refers to all users except the owner of the file and the members of the group assigned to this file.

Granting permissions to this group can lead to unintended access to files.

Ask Yourself Whether

- The application is designed to be run on a multi-user environment.
- Corresponding files and directories may contain confidential information.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

The most restrictive possible permissions should be assigned to files and directories.

Sensitive Code Example

Node.js `fs`

```
const fs = require('fs');

fs.chmodSync("/tmp/fs", 0o777); // Sensitive
```

```
const fs = require('fs');
const fsPromises = fs.promises;

fsPromises.chmod("/tmp/fsPromises", 0o777); // Sensitive
```

```
const fs = require('fs');
const fsPromises = fs.promises

async function fileHandler() {
  let filehandle;
  try {
    filehandle = fsPromises.open('/tmp/fsPromises', 'r');
    filehandle.chmod(0o777); // Sensitive
  } finally {
    if (filehandle !== undefined)
      filehandle.close();
  }
}
```

Node.js `process.umask`

```
const process = require('process');
```

Code Smell

Extra semicolons should be removed

Code Smell

Class names should comply with a naming convention

Code Smell

Track uses of "TODO" tags

Code Smell

Web SQL databases should not be used

Vulnerability

```
process.umask(0o000); // Sensitive
```

Compliant Solution

Node.js fs

```
const fs = require('fs');

fs.chmodSync("/tmp/fs", 0o770); // Compliant
```

```
const fs = require('fs');
const fsPromises = fs.promises;

fsPromises.chmod("/tmp/fsPromises", 0o770); // Compliant
```

```
const fs = require('fs');
const fsPromises = fs.promises

async function fileHandler() {
  let filehandle;
  try {
    filehandle = fsPromises.open('/tmp/fsPromises', 'r');
    filehandle.chmod(0o770); // Compliant
  } finally {
    if (filehandle !== undefined)
      filehandle.close();
  }
}
```

Node.js `process.umask`

```
const process = require('process');

process.umask(0o007); // Compliant
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2021 Category A4 - Insecure Design
- OWASP Top 10 2017 Category A5 - Broken Access Control
- OWASP File Permission
- MITRE, CWE-732 - Incorrect Permission Assignment for Critical Resource
- MITRE, CWE-266 - Incorrect Privilege Assignment
- SANS Top 25 - Porous Defenses

Available In:

sonarcloud sonarqube