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

TS

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...

Results of "in" and "instanceof" should be negated rather than operands
Bug

A compare function should be provided when using "Array.prototype.sort()"
Bug

Jump statements should not occur in "finally" blocks
Bug

Using slow regular expressions is security-sensitive
Security Hotspot

Using publicly writable directories is security-sensitive
Security Hotspot

Using clear-text protocols is security-sensitive
Security Hotspot

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

Switch cases should end with an unconditional "break" statement

Analyze your code

Code Smell Blocker cwe suspicious

When the execution is not explicitly terminated at the end of a switch case, it continues to execute the statements of the following case. While this is sometimes intentional, it often is a mistake which leads to unexpected behavior.

Noncompliant Code Example

```
switch (myVariable) {
  case 1:
    foo();
    break;
  case 2:  // Both 'doSomething()' and 'doSomethingElse()' w
    doSomething();
  default:
    doSomethingElse();
    break;
}
```

Compliant Solution

```
switch (myVariable) {
  case 1:
    foo();
    break;
  case 2:
    doSomething();
    break;
  default:
    doSomethingElse();
    break;
}
```

Exceptions

This rule is relaxed in the following cases:

```
switch (myVariable) {
  case 0:                                // Empty case used
  case 1:
    doSomething();
    break;
  case 2:                                // Use of return st
    return;
  case 3:                               // Ends with comment
    console.log("this case falls through")
    // fall through
  case 4:                                // Use of throw sta
    throw new IllegalStateException();
  case 5:                                // Use of continue
```

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

Union and intersection types should not be defined with duplicated elements

Code Smell

"default" clauses should be last

Code Smell

```
    continue;
  default:                              // For the last cas
    doSomethingElse();
}
```

See

- MITRE, CWE-484 - Omitted Break Statement in Switch

Available In:

sonarlint sonarcloud sonarqube