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...

Special identifiers should not be bound or assigned
Bug

Values should not be uselessly incremented
Bug

Related "if/else if" statements should not have the same condition
Bug

Objects should not be created to be dropped immediately without being used
Bug

Identical expressions should not be used on both sides of a binary operator
Bug

All code should be reachable
Bug

Loops with at most one iteration should be refactored
Bug

Variables should not be self-assigned
Bug

Bitwise operators should not be used in boolean contexts
Bug

Constructing arguments of system commands from user input is security-sensitive
Security Hotspot

Allowing requests with excessive content length is security-sensitive
Security Hotspot

Statically serving hidden files is

"default" clauses should be last

Analyze your code

Code Smell   Critical

`switch` can contain a `default` clause for various reasons: to handle unexpected values, to show that all the cases were properly considered.

For readability purpose, to help a developer to quickly find the default behavior of a `switch` statement, it is recommended to put the `default` clause at the end of the `switch` statement. This rule raises an issue if the `default` clause is not the first or the last one of the `switch`'s cases.

Noncompliant Code Example

```
switch (param) {
  case 0:
    doSomething();
    break;
  default: // default clause should be the first or last one
    error();
    break;
  case 1:
    doSomethingElse();
    break;
}
```

Compliant Solution

```
switch (param) {
  case 0:
    doSomething();
    break;
  case 1:
    doSomethingElse();
    break;
  default:
    error();
    break;
}
```

Available In:

sonarlint | sonarcloud | sonarqube

security-sensitive

Security Hotspot

Using intrusive permissions is security-sensitive

Security Hotspot

Disabling auto-escaping in template engines is security-sensitive

Security Hotspot

Using shell interpreter when executing OS commands is security-sensitive

Security Hotspot

Setting loose POSIX file permissions