




 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

Security Hotspot
Creating cookies without the "secure" flag is security-sensitive
Security Hotspot
Using hardcoded IP addresses is security-sensitive
Security Hotspot
Regular expression quantifiers and character classes should be used concisely
Code Smell
Regular expression literals should be used when possible
Code Smell
"await" should not be used redundantly
Code Smell
Redundant casts and non-null assertions should be avoided
Code Smell
Type aliases should be used
Code Smell
Type guards should be used
Code Smell
"module" should not be used
Code Smell
"for of" should be used with Iterables
Code Smell
Imports from the same modules should be merged
Code Smell
Jump statements should not be redundant

Tags ▾

Search by name... 🔍

Bitwise operators should not be used in boolean contexts

Analyze your code

Bug Major ?

The bitwise operators `&`, `|` can be mistaken for the boolean operators `&&` and `||`.
This rule raises an issue when `&` or `|` is used in a boolean context.

Noncompliant Code Example

```
if (a & b) { ... } // Noncompliant; & used in error
```

Compliant Solution





```
if (a && b) { ... }
```

Exceptions

When a file contains other bitwise operations, (`^`, `<<`, `>>`, `>>`, `~`, `&=`, `^=`, `|=`, `<<=`, `>>=`, `>>=` and `&` or `|` used with a numeric literal as the right operand) all issues in the file are ignored, because it is evidence that bitwise operations are truly intended in the file.

Available In:
sonarlint | sonarcloud | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)

 Code Smell
Default export names and file names should match  Code Smell
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