C# static code analysis

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

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

All rules `409` | 🔒 Vulnerability `34` | 🐛 Bug `76` | Security Hotspot `28` | Code Smell `271` | Quick Fix `52`

Tags ⌄   Search by name...

be used

⊘ Code Smell

Unchanged local variables should be "const"

⊘ Code Smell

"ConfigureAwait(false)" should be used

⊘ Code Smell

"interface" instances should not be cast to concrete types

⊘ Code Smell

Literal boolean values should not be used in assertions

⊘ Code Smell

Optional parameters should not be used

⊘ Code Smell

Public constant members should not be used

⊘ Code Smell

Array covariance should not be used

⊘ Code Smell

"nameof" should be used

⊘ Code Smell

Modulus results should not be checked for direct equality

⊘ Code Smell

"for" loop increment clauses should modify the loops' counters

⊘ Code Smell

"switch" statements should not be nested

⊘ Code Smell

"GC.SuppressFinalize" should not be called

Analyze your code

⊘ Code Smell   🔺 Major ⍰

`GC.SuppressFinalize` requests that the system not call the finalizer for the specified object. This should only be done when implementing `Dispose` as part of the Dispose Pattern.

This rule raises an issue when `GC.SuppressFinalize` is called outside that pattern.

Available In:

sonarlint ∞ | sonarcloud ⬡ | sonarqube 〉

Methods and properties should not be too complex

⊗ Code Smell

Control flow statements "if", "switch", "for", "foreach", "while", "do" and "try" should not be nested too deeply

⊗ Code Smell

"switch/Select" statements should contain a "default/Case Else" clauses

⊗ Code Smell

"if ... else if" constructs should end with "else" clauses

⊗ Code Smell