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

C# static code analysis

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

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

Tags

Search by name...

Code Smell

"static" fields should be initialized inline
Code Smell

Classes that provide "Equals(<T>)" should implement "IEquatable<T>"
Code Smell

Jump statements should not be redundant
Code Smell

Member initializer values should not be redundant
Code Smell

Unassigned members should be removed
Code Smell

Empty "case" clauses that fall through to the "default" should be omitted
Code Smell

Parameters with "[DefaultParameterValue]" attributes should also be marked "[Optional]"
Code Smell

Interfaces should not simply inherit from base interfaces with colliding members
Code Smell

Variables should not be checked against the values they're about to be assigned
Code Smell

Methods should not return constants
Code Smell

Attribute, EventArgs, and Exception type names should end with the type being extended

Doubled prefix operators "!!" and "~~" should not be used

Analyze your code

Bug Major ? Quick Fix ?

Calling the `!` or `~` prefix operator twice does nothing: the second invocation undoes the first. Such mistakes are typically caused by accidentally double-tapping the key in question without noticing.

Either this is a bug, if the operator was actually meant to be called once, or misleading if done on purpose.

Noncompliant Code Example

```
int v1 = 0;
bool v2 = false;

var v3 = !!v1; // Noncompliant
var v4 = ~~v2; // Noncompliant
```

Compliant Solution

```
int v1 = 0;
bool v2 = false;

var v3 = !v1;
var v4 = ~v2;
```

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

Loops should be simplified with "LINQ" expressions

Code Smell

Namespaces should not be empty

Code Smell

Non-derived "private" classes and records should be "sealed"

Code Smell

"string.IsNullOrEmpty" should be used

Code Smell