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

Enumeration type names should not have "Flags" or "Enum" suffixes
Code Smell

Enumeration types should comply with a naming convention
Code Smell

Trivial properties should be auto-implemented
Code Smell

Runtime type checking should be simplified
Code Smell

Boolean checks should not be inverted
Code Smell

Inheritance list should not be redundant
Code Smell

Redundant casts should not be used
Code Smell

Strings should not be concatenated using '+' in a loop
Code Smell

Unused local variables should be removed
Code Smell

Private fields only used as local variables in methods should become local variables
Code Smell

A "while" loop should be used instead of a "for" loop
Code Smell

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

Analyze your code

Bug Major

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code, and should be simplified. In the case of bitwise operators and most binary mathematical operators, having the same value on both sides of an operator yields predictable results, and should be simplified.

Noncompliant Code Example

```
if ( a == a ) // always true
{
  doZ();
}
if ( a != a ) // always false
{
  doY();
}
if ( a == b && a == b ) // if the first one is true, the sec
{
  doX();
}
if ( a == b || a == b ) // if the first one is true, the sec
{
  doW();
}

int j = 5 / 5; //always 1
int k = 5 - 5; // always 0

c.Equals(c);    //always true
Object.Equals(c, c); //always true
```

Exceptions

This rule ignores *, +, =, <<, and >>.

See

- {rule:csharpsquid:S1656} - Implements a check on =.

Available In:

sonarlint | sonarcloud | sonarqube

"Equals" and the comparison operators should be overridden when implementing "IComparable"

Code Smell

Nested code blocks should not be used

Code Smell

Overriding members should do more than simply call the same member in the base class

Code Smell

"Any()" should be used to test for emptiness

Code Smell