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#

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...

Classes should not have only "private" constructors
Bug

Expressions used in "Debug.Assert" should not produce side effects
Bug

Caller information parameters should come at the end of the parameter list
Bug

Static fields should appear in the order they must be initialized
Bug

Classes directly extending "object" should not call "base" in "GetHashCode" or "Equals"
Bug

Anonymous delegates should not be used to unsubscribe from Events
Bug

Delegates should not be subtracted
Bug

"async" methods should not return "void"
Bug

"ThreadStatic" should not be used on non-static fields
Bug

"IDisposables" created in a "using" statement should not be returned
Bug

"ThreadStatic" fields should not be initialized
Bug

"Object.ReferenceEquals" should not

Parameter names should match base declaration and other partial definitions

Analyze your code

Code Smell   Critical   suspicious

The name of a parameter in an externally visible. This rule raises an issue when method override does not match the name of the parameter in the base declaration of the method, or the name of the parameter in the interface declaration of the method or the name of any other `partial` definition.

Noncompliant Code Example

```
partial class Point
{
  partial void MoveVertically(int z);
}

partial class Point
{
  int x = 0;
  int y = 0;
  int z = 0;

  partial void MoveVertically(int y)  // Noncompliant
  {
    this.y = y;
  }
}

interface IFoo
{
  void Bar(int i);
}

class Foo : IFoo
{
  void Bar(int z) // Noncompliant, parameter name should be
  {
  }
}
```

Compliant Solution

```
partial class Point
{
  partial void MoveVertically(int z);
}

partial class Point
{
  int x = 0;
  int y = 0;
  int z = 0;
```

be used for value types

Bug

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

Bug

"=+" should not be used instead of "+="

Bug

"NaN" should not be used in comparisons

Bug

```
  partial void MoveVertically(int z)
  {
    this.z = z;
  }
}

interface IFoo
{
  void Bar(int i);
}

class Foo : IFoo
{
  void Bar(int i)
  {
  }
}
```

Available In:

sonarlint sonarcloud sonarqube