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...

"protected" members
Code Smell

Underscores should be used to make large numbers readable
Code Smell

"ToString()" calls should not be redundant
Code Smell

"==" should not be used when "Equals" is overridden
Code Smell

An abstract class should have both abstract and concrete methods
Code Smell

Multiple variables should not be declared on the same line
Code Smell

Culture should be specified for "string" operations
Code Smell

"switch" statements should have at least 3 "case" clauses
Code Smell

break statements should not be used except for switch cases
Code Smell

String literals should not be duplicated
Code Smell

Files should contain an empty newline at the end
Code Smell

Unused "using" should be removed
Code Smell

Overloads with a "CultureInfo" or an "IFormatProvider" parameter should be used

Analyze your code

Code Smell | Minor | localisation pitfall

When a `System.Globalization.CultureInfo` or `IFormatProvider` object is not supplied, the default value that is supplied by the overloaded member might not have the effect that you want in all locales.

You should supply culture-specific information according to the following guidelines:

- If the value will be displayed to the user, use the current culture. See `CultureInfo.CurrentCulture`.
- If the value will be stored and accessed by software (persisted to a file or database), use the invariant culture. See `CultureInfo.InvariantCulture`.
- If you do not know the destination of the value, have the data consumer or provider specify the culture.

This rule raises an issue when a method or constructor calls one or more members that have overloads that accept a `System.IFormatProvider` parameter, and the method or constructor does not call the overload that takes the `IFormatProvider` parameter. This rule ignores calls to .NET Framework methods that are documented as ignoring the `IFormatProvider` parameter as well as the following methods:

- `Activator.CreateInstance`
- `ResourceManager.GetObject`
- `ResourceManager.GetString`

Noncompliant Code Example

```
using System;

namespace MyLibrary
{
    public class Foo
    {
        public void Bar(String string1)
        {
            if(string.Compare(string1, string2, false) == 0)
            {
                Console.WriteLine(string3.ToLower()); // Non
            }
        }
    }
}
```

Compliant Solution

```
using System;
using System.Globalization;

namespace MyLibrary
{
```

A close curly brace should be located at the beginning of a line

Code Smell

Tabulation characters should not be used

Code Smell

Methods and properties should be named in PascalCase

Code Smell

Track uses of in-source issue suppressions

Code Smell

```
    public class Foo
    {
        public void Bar(String string1, String string2, Stri
        {
            if(string.Compare(string1, string2, false,
                              CultureInfo.InvariantCulture)
            {
                Console.WriteLine(string3.ToLower(CultureInf
            }
        }
    }
}
```

Exceptions

This rule will not raise an issue when the overload is marked as obsolete.

Available In:

sonarlint | sonarcloud | sonarqube