

- 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 "StringComparison" parameter should be used

Analyze your code

Code Smell Minor ?

Many string operations, the Compare and Equals methods in particular, provide an overload that accepts a StringComparison enumeration value as a parameter. Calling these overloads and explicitly providing this parameter makes your code clearer and easier to maintain.

This rule raises an issue when a string comparison operation doesn't use the overload that takes a StringComparison parameter.

Noncompliant Code Example

```
using System;





namespace MyLibrary
{
    public class Foo
    {
        public bool HaveSameNames(string name1, string name2)
        {
            return string.Compare(name1, name2) == 0; // Noncompliant
        }
    }
}
```

Compliant Solution

```
using System;

namespace MyLibrary
{
    public class Foo
    {
        public bool HaveSameNames(string name1, string name2)
        {
            return string.Compare(name1, name2, StringComparison.Ordinal) == 0;
        }
    }
}
```

Available In: sonarlint | sonarcloud | sonarqube

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