

- 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

Utility classes should not have public constructors

Analyze your code

Code Smell Major design

Utility classes, which are collections of static members, are not meant to be instantiated.

C# adds an implicit public constructor to every class which does not explicitly define at least one constructor. Hence, at least one protected constructor should be defined if you wish to subclass this utility class. Or the static keyword should be added to the class declaration to prevent subclassing.

Noncompliant Code Example

```
public class StringUtils // Noncompliant
{
    public static string Concatenate(string s1, string s2)
    {
        return s1 + s2;
    }
}
```

Compliant Solution

```
public static class StringUtils
{
    public static string Concatenate(string s1, string s2)
    {
        return s1 + s2;
    }
}
```

or

```
public class StringUtils
{
    protected StringUtils()
    {
    }
    public static string Concatenate(string s1, string s2)
    {
        return s1 + s2;
    }
}
```

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