

- 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

Array covariance should not be used

Analyze your code

Code Smell Critical ? pitfall

Array covariance is the principle that if an implicit or explicit reference conversion exists from type A to B, then the same conversion exists from the array type A[] to B[].

While this array conversion can be useful in readonly situations to pass instances of A[] where B[] is expected, it must be used with care, since assigning an instance of B into an array of A will cause an `ArrayTypeMismatchException` to be thrown at runtime.

Noncompliant Code Example

```
abstract class Fruit { }
class Apple : Fruit { }
class Orange : Fruit { }

class Program
{
    static void Main(string[] args)
    {
        Fruit[] fruits = new Apple[1]; // Noncompliant - array covariance
        FillWithOranges(fruits);
    }

    // Just looking at the code doesn't reveal anything suspicious
    static void FillWithOranges(Fruit[] fruits)
    {
        for (int i = 0; i < fruits.Length; i++)
        {
            fruits[i] = new Orange(); // Will throw an ArrayTypeMismatchException
        }
    }
}
```

Compliant Solution

```
abstract class Fruit { }
class Apple : Fruit { }
class Orange : Fruit { }

class Program
{
    static void Main(string[] args)
    {
        Orange[] fruits = new Orange[1]; // Compliant
        FillWithOranges(fruits);
    }

    static void FillWithOranges(Orange[] fruits)
    {
    }
}
```

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

```
for (int i = 0; i < fruits.Length; i++)
{
    fruits[i] = new Orange();
}
}
```

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)