

- 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...



Enumeration type names should not have "Flags" or "Enum" suffixes

Code Smell

Enumeration types should comply with a naming convention

Code Smell

Trivial properties should be auto-implemented

Code Smell

Runtime type checking should be simplified

Code Smell

Boolean checks should not be inverted

Code Smell

Inheritance list should not be redundant

Code Smell

Redundant casts should not be used

Code Smell

Strings should not be concatenated using '+' in a loop

Code Smell

Unused local variables should be removed

Code Smell

Private fields only used as local variables in methods should become local variables

Code Smell

A "while" loop should be used instead of a "for" loop

Code Smell

"Equals" and the comparison operators should be overridden when implementing "IComparable"

"Object.ReferenceEquals" should not be used for value types

Analyze your code

Bug Major ?

Using `Object.ReferenceEquals` to compare the references of two value types simply won't return the expected results most of the time because such types are passed by value, not by reference.

Noncompliant Code Example

```
public class MyClass
{
    private MyStruct myStruct;

    public void DoSomething(MyStruct s1) {
        int a = 1;
        int b = 1;

        if (Object.ReferenceEquals(myStruct, s1)) // Noncompliant
        {
            // ...
        }
        else if (Object.ReferenceEquals(a,b)) // Noncompliant
        {
            // ...
        }
    }
}
```

Available In:


sonarlint | sonarcloud | sonarqube

 Code Smell

Nested code blocks should not be used

 Code Smell

Overriding members should do more than simply call the same member in the base class

 Code Smell

"Any()" should be used to test for emptiness

 Code Smell

Boolean literals should not be