
































-  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

Optional parameters should not be used

Analyze your code

Code Smell Critical ? pitfall

The overloading mechanism should be used in place of optional parameters for several reasons:

- Optional parameter values are baked into the method call site code, thus, if a default value has been changed, all referencing assemblies need to be rebuilt, otherwise the original values will be used.
- The Common Language Specification (CLS) allows compilers to ignore default parameter values, and thus require the caller to explicitly specify the values. For example, if you want to consume a method with default argument from another .NET compatible language (for instance C++/CLI), you will have to provide all arguments. When using method overloads, you could achieve similar behavior as default arguments.
- Optional parameters prevent muddying the definition of the function contract. Here is a simple example: if there are two optional parameters, when one is defined, is the second one still optional or mandatory?

Noncompliant Code Example

```
void Notify(string company, string office = "QJZ") // Noncompliant
{
}
```

Compliant Solution

```
void Notify(string company)
{
    Notify(company, "QJZ");
}
void Notify(string company, string office)
{
}
```

Exceptions

The rule ignores non externally visible methods.

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