

- 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

"out" and "ref" parameters should not be used

Analyze your code

Code Smell Critical ? suspicious

Passing a parameter by reference, which is what happens when you use the `out` or `ref` parameter modifiers, means that the method will receive a pointer to the argument, rather than the argument itself. If the argument was a value type, the method will be able to change the argument's values. If it was a reference type, then the method receives a pointer to a pointer, which is usually not what was intended. Even when it is what was intended, this is the sort of thing that's difficult to get right, and should be used with caution.

This rule raises an issue when `out` or `ref` is used on a non-Optional parameter in a public method. Optional parameters are covered by {rule:csharpsquid:S3447}.

Noncompliant Code Example

```
public void GetReply(  
    ref MyClass input, // Noncompliant  
    out string reply) // Noncompliant  
{ ... }
```

Compliant Solution

```
public string GetReply(MyClass input)  
{ ... }  
  
public bool TryGetReply(MyClass input, out string reply)  
{ ... }  
  
public ReplyData GetReply(MyClass input)  
{ ... }  
  
internal void GetReply(ref MyClass input, out string reply)  
{ ... }
```





Exceptions

This rule will not raise issues for:

- non-public methods
- methods with only 'out' parameters, name starting with "Try" and return type bool.
- interface implementation 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