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

Properties should be preferred

Analyze your code

⊙ Code Smell | ⊘ Minor ? | 🏷 convention

Properties are accessed like fields which makes them easier to use.

This rule raises an issue when the name of a `public` or `protected` method starts with `Get`, takes no parameter, and returns a value that is not an array.

Noncompliant Code Example

```
using System;

namespace MyLibrary
{
    public class Foo
    {
        private string name;

        public string GetName()  // Noncompliant
        {
            return name;
        }
    }
}
```

Compliant Solution

```
using System;

namespace MyLibrary
{
    public class Foo
    {
        private string name;

        public string Name
        {
            get
            {
                return name;
            }
        }
    }
}
```

Exceptions

The rule doesn't raise an issue when the method:

- Is a constructor
- Is an `override`
- Is an interface implementation

- Is async
- Returns `Task`, `Task<T>`
- Is named `GetEnumerator`, `GetAwaiter`

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