

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C# 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... 🔍

Code Smell
"IDisposable" should be implemented correctly
Code Smell
"ServiceContract" and "OperationContract" attributes should be used together
Code Smell
Composite format strings should be used correctly
Code Smell
Exceptions should not be explicitly rethrown
Code Smell
"abstract" classes should not have "public" constructors
Code Smell
Assertion arguments should be passed in the correct order
Code Smell
Ternary operators should not be nested
Code Smell
Events should be invoked
Code Smell
"params" should be used on overrides
Code Smell
Generic type parameters should be co/contravariant when possible
Code Smell
Multiple "OrderBy" calls should not be used
Code Smell
Reflection should not be used to

Constructors should only call non-overridable methods

Analyze your code

Code Smell Critical ? pitfall

Calling an overridable method from a constructor could result in failures or strange behaviors when instantiating a subclass which overrides the method.

For example:

- The subclass class constructor starts by calling the parent class constructor.
- The parent class constructor calls the method, which has been overridden in the child class.
- If the behavior of the child class method depends on fields that are initialized in the child class constructor, unexpected behavior (like a `NullReferenceException`) can result, because the fields aren't initialized yet.

Noncompliant Code Example

```
public class Parent
{
    public Parent()
    {
        DoSomething(); // Noncompliant
    }

    public virtual void DoSomething() // can be overridden
    {
        ...
    }
}

public class Child : Parent
{
    private string foo;

    public Child(string foo) // leads to call DoSomething() in
    {
        this.foo = foo;
    }


    public override void DoSomething()
    {
        Console.WriteLine(this.foo.Length);
    }
}
```

Available In:
sonarlint | sonarcloud | sonarqube

increase accessibility of classes, methods, or fields

 Code Smell

Static fields should not be updated in constructors

 Code Smell

"IEnumerable" LINQs should be simplified

 Code Smell

Fields that are only assigned in the constructor should be "readonly"

 Code Smell

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](#)