

- 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...



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 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"

### Method overrides should not change parameter defaults

Analyze your code

Code Smell Critical Quick Fix pitfall

Default arguments are determined by the static type of the object. If a default argument is different for a parameter in an overriding method, the value used in the call will be different when calls are made via the base or derived object, which may be contrary to developer expectations.

Default parameter values are useless in explicit interface implementations, because the static type of the object will always be the implemented interface. Thus, specifying default values is useless and confusing.

#### Noncompliant Code Example

```
using System;

public class Base
{
    public virtual void Write(int i = 42)
    {
        Console.WriteLine(i);
    }
}

public class Derived : Base
{
    public override void Write(int i = 5) // Noncompliant
    {
        Console.WriteLine(i);
    }
}





public class Program
{
    public static void Main()
    {
        var derived = new Derived();
        derived.Write(); // writes 5
        Print(derived); // writes 42; was that expected?
    }

    private static void Print(Base item)
    {
        item.Write();
    }
}
```

#### Compliant Solution

```
using System;

public class Base
```

constructor should be readonly
 Code Smell
Static fields should not be used in generic types
 Code Smell
Multiline blocks should be enclosed in curly braces
 Code Smell
Boolean expressions should not be gratuitous
 Code Smell
Types and methods should not have

```
{
    public virtual void Write(int i = 42)
    {
        Console.WriteLine(i);
    }
}

public class Derived : Base
{
    public override void Write(int i = 42)
    {
        Console.WriteLine(i);
    }
}

public class Program
{
    public static void Main()
    {
        var derived = new Derived();
        derived.Write(); // writes 42
        Print(derived); // writes 42
    }

    private static void Print(Base item)
    {
        item.Write();
    }
}
```

Available In:

**sonarlint**  | **sonarcloud**  | **sonarqube** 