

- 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... 🔍

can be

Code Smell

Objects should not be disposed more than once

Code Smell

Parameter names used into ArgumentException constructors should match an existing one

Code Smell

"ISerializable" should be implemented correctly

Code Smell

"Assembly.Load" should be used

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

### Non-constant static fields should not be visible

Analyze your code

Code Smell Critical ? pitfall

A static field that is neither constant nor read-only is not thread-safe. Correctly accessing these fields from different threads needs synchronization with locks. Improper synchronization may lead to unexpected results, thus publicly visible static fields are best suited for storing non-changing data shared by many consumers. To enforce this intent, these fields should be marked readonly or converted to constants.

#### Noncompliant Code Example

```
public class Math
{
    public static double Pi = 3.14; // Noncompliant
}
```

or

```
public class Shape
{
    public static Shape Empty = new EmptyShape(); // Noncompliant

    private class EmptyShape : Shape
    {
    }
}
```

#### Compliant Solution

```
public class Math
{
    public const double Pi = 3.14;
}
```


or

```
public class Shape
{
    public static readonly Shape Empty = new EmptyShape();

    private class EmptyShape : Shape
    {
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

 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

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, 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](#)