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...

Literal suffixes should be upper case
Code Smell

Null checks should not be used with "is"
Code Smell

Method overloads should be grouped together
Code Smell

"params" should be used instead of "varargs"
Code Smell

"static" fields should be initialized inline
Code Smell

Classes that provide "Equals(<T>)" should implement "IEquatable<T>"
Code Smell

Jump statements should not be redundant
Code Smell

Member initializer values should not be redundant
Code Smell

Unassigned members should be removed
Code Smell

Empty "case" clauses that fall through to the "default" should be omitted
Code Smell

Parameters with "[DefaultParameterValue]" attributes should also be marked "[Optional]"
Code Smell

"ThreadStatic" fields should not be initialized

Analyze your code

Bug | Major | multi-threading

When an object has a field annotated with `ThreadStatic`, that field is shared within a given thread, but unique across threads. Since a class' static initializer is only invoked for the first thread created, it also means that only the first thread will have the expected initial values.

Instead, allow such fields to be initialized to their default values or make the initialization lazy.

Noncompliant Code Example

```
public class Foo
{
  [ThreadStatic]
  public static object PerThreadObject = new object(); // No
}
```

Compliant Solution

```
public class Foo
{
  [ThreadStatic]
  public static object _perThreadObject;
  public static object PerThreadObject
  {
    get
    {
      if (_perThreadObject == null)
      {
        _perThreadObject = new object();
      }
      return _perThreadObject;
    }
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

Interfaces should not simply inherit from base interfaces with colliding members

Code Smell

Variables should not be checked against the values they're about to be assigned

Code Smell

Methods should not return constants

Code Smell

Attribute, EventArgs, and Exception type names should end with the type being extended

Code Smell