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...

Formatting SQL queries is security-sensitive
Security Hotspot

Using hardcoded IP addresses is security-sensitive
Security Hotspot

"goto" statement should not be used
Code Smell

"new Guid()" should not be used
Code Smell

Parameter validation in "async"/"await" methods should be wrapped
Code Smell

Parameter validation in yielding methods should be wrapped
Code Smell

Events should have proper arguments
Code Smell

"P/Invoke" methods should not be visible
Code Smell

Native methods should be wrapped
Code Smell

Methods should not have identical implementations
Code Smell

Non-flags enums should not be marked with "FlagsAttribute"
Code Smell

Classes implementing "IEquatable<T>" should be sealed
Code Smell

Instance members should not write to "static" fields

Analyze your code

Code Smell | Critical | multi-threading

Correctly updating a `static` field from a non-static method is tricky to get right and could easily lead to bugs if there are multiple class instances and/or multiple threads in play.

This rule raises an issue each time a `static` field is updated from a non-static method or property.

Noncompliant Code Example

```
public class MyClass
{
  private static int count = 0;

  public void DoSomething()
  {
    //...
    count++;  // Noncompliant
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

"GC.SuppressFinalize" should not be called

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