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...

Overflow checking should not be disabled for "Enumerable.Sum"
Code Smell

Field-like events should not be virtual
Code Smell

Non-constant static fields should not be visible
Code Smell

Inappropriate casts should not be made
Code Smell

Constructors should only call non-overridable methods
Code Smell

"GC.Collect" should not be called
Code Smell

Methods should not be empty
Code Smell

Exceptions should not be thrown in finally blocks
Code Smell

Method overrides should not change parameter defaults
Code Smell

Types allowed to be deserialized should be restricted
Vulnerability

Server-side requests should not be vulnerable to forging attacks
Vulnerability

Members should not have conflicting transparency annotations
Vulnerability

Non-async "Task/Task<T>" methods should not return null

Analyze your code

Bug | Critical | async-await

Returning `null` from a non-`async` `Task/Task<T>` method will cause a `NullReferenceException` at runtime. This problem can be avoided by returning `Task.FromResult<T>(null)` instead.

Noncompliant Code Example

```
public Task<object> GetFooAsync()
{
    return null; // Noncompliant
}
```

Compliant Solution

```
public Task<object> GetFooAsync()
{
    return Task.FromResult<object>(null);
}
```

Available In:

sonarlint | sonarcloud | sonarqube

"PartCreationPolicyAttribute" should be used with "ExportAttribute"

Bug

"ConstructorArgument" parameters should exist in constructors

Bug

Windows Forms entry points should be marked with STAThread

Bug

Collection elements should not be replaced unconditionally

Bug