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...

"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

"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

Exceptions should not be created without being thrown

"Shared" parts should not be created with "new"

Analyze your code

Bug Critical mef pitfall

Marking a class with `PartCreationPolicy(CreationPolicy.Shared)`, which is part of Managed Extensibility Framework (MEF), means that a single, shared instance of the exported object will be created. Therefore it doesn't make sense to create new instances using the constructor and it will most likely result in unexpected behaviours.

This rule raises an issue when a constructor of a class marked shared with a `PartCreationPolicyAttribute` is invoked.

Noncompliant Code Example

```
[Export(typeof(IFooBar))]
[PartCreationPolicy(CreationPolicy.Shared)]
public class FooBar : IFooBar
{
}

public class Program
{
    public static void Main()
    {
        var fooBar = new FooBar(); // Noncompliant;
    }
}
```

Compliant Solution

```
[Export(typeof(IFooBar))]
[PartCreationPolicy(CreationPolicy.Shared)]
public class FooBar : IFooBar
{
}

public class Program
{
    public static void Main()
    {
        var fooBar = serviceProvider.GetService<IFooBar>();
    }
}
```

Available In:
sonarlint | sonarcloud | sonarqube

Bug

Collection sizes and array length comparisons should make sense

Bug

Serialization event handlers should be implemented correctly

Bug

Deserialization methods should be provided for "OptionalField" members

Bug

All branches in a conditional structure should not have exactly the same