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...

Exceptions should not be thrown from unexpected methods
Code Smell

"operator==" should not be overloaded on reference types
Code Smell

Type should not be examined on "System.Type" instances
Code Smell

Test method signatures should be correct
Code Smell

Method overloads with default parameter values should not overlap
Code Smell

"value" parameters should be used
Code Smell

"is" should not be used with "this"
Code Smell

Methods named "Dispose" should implement "IDisposable.Dispose"
Code Smell

Tests should include assertions
Code Smell

Silly bit operations should not be performed
Code Smell

Public methods should not have multidimensional array parameters
Code Smell

"async" and "await" should not be used as identifiers
Code Smell

Classes should implement their "ExportAttribute" interfaces

Analyze your code

Bug | Blocker | mef pitfall

In the Attributed Programming Model, the `ExportAttribute` declares that a part "exports", or provides to the composition container, an object that fulfills a particular contract. During composition, parts with imports that have matching contracts will have those dependencies filled by the exported object.

If the type doesn't implement the interface it is exporting there will be an issue at runtime (either a cast exception or just a container not filled with the exported type) leading to unexpected behaviors/crashes.

The rule raises an issue when a class doesn't implement or inherit the type declared in the `ExportAttribute`.

Noncompliant Code Example

```
[Export(typeof(ISomeType))]
public class SomeType // Noncompliant; doesn't implement 'IS
{
}
```

Compliant Solution

```
[Export(typeof(ISomeType))]
public class SomeType : ISomeType
{
}
```

Available In:

sonarlint | sonarcloud | sonarqube

TestCases should contain tests

Code Smell

Short-circuit logic should be used in boolean contexts

Code Smell

JWT should be signed and verified with strong cipher algorithms

Vulnerability

Cipher algorithms should be robust

Vulnerability

Encryption algorithms should be used