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...

Track uses of "FIXME" tags
Code Smell

"Obsolete" attributes should include explanations
Code Smell

Assignments should not be made from within sub-expressions
Code Smell

General exceptions should never be thrown
Code Smell

Utility classes should not have public constructors
Code Smell

Local variables should not shadow class fields
Code Smell

Redundant pairs of parentheses should be removed
Code Smell

Inheritance tree of classes should not be too deep
Code Smell

Nested blocks of code should not be left empty
Code Smell

Methods should not have too many parameters
Code Smell

Collapsible "if" statements should be merged
Code Smell

OS commands should not be vulnerable to argument injection

Types should be defined in named namespaces

Analyze your code

Bug Major

Types are declared in namespaces in order to prevent name collisions and as a way to organize them into the object hierarchy. Types that are defined outside any named namespace are in a global namespace that cannot be referenced in code.

Noncompliant Code Example

```
public class Foo // Noncompliant
{
}

public struct Bar // Noncompliant
{
}
```

Compliant Solution

```
namespace SomeSpace
{
    public class Foo
    {
    }

    public struct Bar
    {
    }
}
```

Available In:

sonarlint sonarcloud sonarqube

attacks

Vulnerability

Logging should not be vulnerable to injection attacks

Vulnerability

Empty collections should not be accessed or iterated

Bug

Mutable, non-private fields should not be "readonly"

Bug

"string.ToCharArray()" should not be