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#

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...

Optional parameters should not be used
Code Smell

Public constant members should not be used
Code Smell

Array covariance should not be used
Code Smell

"nameof" should be used
Code Smell

Modulus results should not be checked for direct equality
Code Smell

"for" loop increment clauses should modify the loops' counters
Code Smell

"switch" statements should not be nested
Code Smell

Methods and properties should not be too complex
Code Smell

Control flow statements "if", "switch", "for", "foreach", "while", "do" and "try" should not be nested too deeply
Code Smell

"switch/Select" statements should contain a "default/Case Else" clauses
Code Smell

"if ... else if" constructs should end with "else" clauses
Code Smell

Control structures should use curly braces

"Assembly.Load" should be used

Analyze your code

Code Smell Major unpredictable

The parameter to `Assembly.Load` includes the full specification of the dll to be loaded. Use another method, and you might end up with a dll other than the one you expected.

This rule raises an issue when `Assembly.LoadFrom`, `Assembly.LoadFile`, or `Assembly.LoadWithPartialName` is called.

Noncompliant Code Example

```
static void Main(string[] args)
{
    Assembly.LoadFrom(...); // Noncompliant
    Assembly.LoadFile(...); // Noncompliant
    Assembly.LoadWithPartialName(...); // Noncompliant + dep
}
```

Available In:

sonarlint sonarcloud sonarqube

Code Smell

Expressions should not be too complex

Code Smell

ASP.NET HTTP request validation feature should not be disabled

Vulnerability

Serialization constructors should be secured

Vulnerability

Calculations should not overflow