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...

Code Smell

Trivial properties should be auto-implemented
Code Smell

Runtime type checking should be simplified
Code Smell

Boolean checks should not be inverted
Code Smell

Inheritance list should not be redundant
Code Smell

Redundant casts should not be used
Code Smell

Strings should not be concatenated using '+' in a loop
Code Smell

Unused local variables should be removed
Code Smell

Private fields only used as local variables in methods should become local variables
Code Smell

A "while" loop should be used instead of a "for" loop
Code Smell

"Equals" and the comparison operators should be overridden when implementing "IComparable"
Code Smell

Nested code blocks should not be used
Code Smell

Loops with at most one iteration should be refactored

Analyze your code

Bug Major

A loop with at most one iteration is equivalent to the use of an `if` statement to conditionally execute one piece of code. If the initial intention of the author was really to conditionally execute one piece of code, an `if` statement should be used instead. If that was not the initial intention of the author, the body of the loop should be fixed to use the nested `return`, `break` or `throw` statements in a more appropriate way.

Noncompliant Code Example

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
    break;  // Noncompliant, loop only executes once
}
...
foreach (var item in items)
{
    return item;  // Noncompliant, loop only executes once
}
...
```

Compliant Solution

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
...
var item = items.FirstOrDefault();
if (item != null)
{
    return item;
}
...
```

Available In:

sonarlint | sonarcloud | sonarqube

Overriding members should do more than simply call the same member in the base class

Code Smell

"Any()" should be used to test for emptiness

Code Smell

Boolean literals should not be redundant

Code Smell

Empty statements should be removed

Code Smell