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

Empty "case" clauses that fall through to the "default" should be omitted
Code Smell

Parameters with "[DefaultParameterValue]" attributes should also be marked "[Optional]"
Code Smell

Interfaces should not simply inherit from base interfaces with colliding members
Code Smell

Variables should not be checked against the values they're about to be assigned
Code Smell

Methods should not return constants
Code Smell

Attribute, EventArgs, and Exception type names should end with the type being extended
Code Smell

Loops should be simplified with "LINQ" expressions
Code Smell

Namespaces should not be empty
Code Smell

Non-derived "private" classes and records should be "sealed"
Code Smell

"string.IsNullOrEmpty" should be used
Code Smell

Implementations should be provided for "partial" methods
Code Smell

Conditionally executed code should be reachable

Analyze your code

Bug | Major | cwe unused suspicious pitfall

Conditional expressions which are always `true` or `false` can lead to dead code. Such code is always buggy and should never be used in production.

Noncompliant Code Example

```
public void Sample(bool b)
{
    bool a = false;
    if (a) // Noncompliant
    {
        DoSomething(); // never executed
    }

    if (!a || b) // Noncompliant; "!a" is always "true", "b"
    {
        DoSomething();
    }
    else
    {
        DoSomethingElse(); // never executed
    }

    var d = "xxx";
    var res = d ?? "value"; // Noncompliant, d is always not
}
```

Compliant Solution

```
public void Sample(bool b)
{
    bool a = false;
    if (Foo(a))
    {
        DoSomething();
    }

    if (b)
    {
        DoSomething();
    }
    else
    {
        DoSomethingElse();
    }

    var d = "xxx";
    var res = d;
}
```

Duplicate casts should not be made

Code Smell

Methods should not return values that are never used

Code Smell

Caller information arguments should not be provided explicitly

Code Smell

Method calls should not resolve ambiguously to overloads with "params"

Code Smell

Exceptions

This rule will not raise an issue in either of these cases:

- When the condition is a single `const bool`

```
const bool debug = false;
//...
if (debug)
{
  // Print something
}
```

- When the condition is the literal `true` or `false`.

In these cases it is obvious the code is as intended.

See

- MITRE, CWE-570 - Expression is Always False
- MITRE, CWE-571 - Expression is Always True

Available In:

sonarlint | sonarcloud | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
Privacy Policy