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...

Doubled prefix operators "!!" and "~~" should not be used
Bug

"=+" should not be used instead of "+="
Bug

"NaN" should not be used in comparisons
Bug

Conditionally executed code should be reachable
Bug

Null pointers should not be dereferenced
Bug

For-loop conditions should be true at least once
Bug

A "for" loop update clause should move the counter in the right direction
Bug

"ToString()" method should not return null
Bug

Return values from functions without side effects should not be ignored
Bug

Values should not be uselessly incremented
Bug

Collections should not be passed as arguments to their own methods
Bug

Related "if/else if" statements should

Threads should not lock on objects with weak identity

Analyze your code

Code Smell | Critical | multi-threading pitfall

A thread acquiring a lock on an object that can be accessed across application domain boundaries runs the risk of being blocked by another thread in a different application domain. Objects that can be accessed across application domain boundaries are said to have weak identity. Types with weak identity are:

- `MarshalByRefObject`
- `ExecutionEngineException`
- `OutOfMemoryException`
- `StackOverflowException`
- `String`
- `MemberInfo`
- `ParameterInfo`
- `Thread`

Noncompliant Code Example

```
public class Sample
{
    string myString = "foo";

    public void Go()
    {
        lock (myString) { } // Noncompliant
    }
}
```

Compliant Solution

```
public class Sample
{
    private readonly static string thisLock = new object();

    public void Go()
    {
        lock (thisLock) { }
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

Related "if/else if" statements should not have the same condition

Bug

Objects should not be created to be dropped immediately without being used

Bug

Identical expressions should not be used on both sides of a binary operator

Bug

Loops with at most one iteration should be refactored

Bug