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...

Using Sockets is security-sensitive
Security Hotspot

Encrypting data is security-sensitive
Security Hotspot

Using regular expressions is security-sensitive
Security Hotspot

Interface methods should be callable by derived types
Code Smell

Child class fields should not differ from parent class fields only by capitalization
Code Smell

Pointers to unmanaged memory should not be visible
Code Smell

Number patterns should be regular
Code Smell

"out" and "ref" parameters should not be used
Code Smell

Unchanged local variables should be "const"
Code Smell

"ConfigureAwait(false)" should be used
Code Smell

"interface" instances should not be cast to concrete types
Code Smell

Literal boolean values should not be used in assertions
Code Smell

Native methods should be wrapped

Analyze your code

Code Smell | Major | pitfall

Native methods are functions that reside in libraries outside the virtual machine. Being able to call them is useful for interoperability with applications and libraries written in other programming languages, in particular when performing platform-specific operations. However doing so comes with extra risks since it means stepping out of the security model of the virtual machine. It is therefore highly recommended to take extra steps, like input validation, when invoking native methods. This is best done by making the native method `private` and by providing a wrapper that performs these extra steps and verifications.

This rule raises an issue when a native method is declared `public` or its wrapper is too trivial.

Noncompliant Code Example

```
using System;
using System.Runtime.InteropServices;

namespace MyLibrary
{
  class Foo
  {
    [DllImport("mynativelib")]
    extern public static void Bar(string s, int x); // Nonco
  }
}
```

Compliant Solution

```
using System;
using System.Runtime.InteropServices;

namespace MyLibrary
{
  class Foo
  {
    [DllImport("mynativelib")]
    extern private static void Bar(string s, int x);

    public void BarWrapper(string s, int x)
    {
      if (s != null && x >= 0  && x < s.Length)
      {
        bar(s, x);
      }
    }
  }
}
```

Code Smell

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

Available In:
sonarlint | sonarcloud | sonarqube