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...

Classes should "Dispose" of members from the classes' own "Dispose" methods
Bug

Reading the Standard Input is security-sensitive
Security Hotspot

Using command line arguments is security-sensitive
Security Hotspot

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"

Events should have proper arguments

Analyze your code

Code Smell | Major | event pitfall

When raising an event, two arguments are expected by the `EventHandler` delegate: Sender and event-data. There are three guidelines regarding these parameters:

- Do not pass `null` as the sender when raising a non-static event.
- Do pass `null` as the sender when raising a static event.
- Do not pass `null` as the event-data. If no data should be passed, then `EventArgs.Empty` should be used.

This rule raises an issue when any of these guidelines is not met.

Noncompliant Code Example

```
using System;

namespace MyLibrary
{
  class Foo
  {
    public event EventHandler ThresholdReached;

    protected virtual void OnThresholdReached(EventArgs e)
    {
        ThresholdReached?.Invoke(null, e); // Noncompliant
    }
  }
}
```

Compliant Solution

```
using System;

namespace MyLibrary
{
  class Foo
  {
    public event EventHandler ThresholdReached;

    protected virtual void OnThresholdReached(EventArgs e)
    {
        ThresholdReached?.Invoke(this, e);
    }
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

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

Optional parameters should not be used