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...

"protected" members
Code Smell

Underscores should be used to make large numbers readable
Code Smell

"ToString()" calls should not be redundant
Code Smell

"==" should not be used when "Equals" is overridden
Code Smell

An abstract class should have both abstract and concrete methods
Code Smell

Multiple variables should not be declared on the same line
Code Smell

Culture should be specified for "string" operations
Code Smell

"switch" statements should have at least 3 "case" clauses
Code Smell

break statements should not be used except for switch cases
Code Smell

String literals should not be duplicated
Code Smell

Files should contain an empty newline at the end
Code Smell

Unused "using" should be removed
Code Smell

Classes with "IDisposable" members should implement "IDisposable"

Analyze your code

Bug | Blocker | cwe denial-of-service

An `IDisposable` object should be disposed (there are some rare exceptions where not disposing is fine, most notably `Task`). If a class has an `IDisposable` field, there can be two situations:

- The class observes a field that is under the responsibility of another class.
- The class owns the field, and is therefore responsible for calling `Dispose` on it.

In the second case, the safest way for the class to ensure `Dispose` is called is to call it in its own `Dispose` function, and therefore to be itself `IDisposable`. A class is considered to own an `IDisposable` field resource if it created the object referenced by the field.

Noncompliant Code Example

```
public class ResourceHolder   // Noncompliant; doesn't imple
{
  private FileStream fs;  // This member is never Disposed
  public void OpenResource(string path)
  {
    this.fs = new FileStream(path, FileMode.Open); // I crea
  }
  public void CloseResource()
  {
    this.fs.Close();
  }
}
```

Compliant Solution

```
public class ResourceHolder : IDisposable
{
  private FileStream fs;
  public void OpenResource(string path)
  {
    this.fs = new FileStream(path, FileMode.Open); // I crea
  }
  public void CloseResource()
  {
    this.fs.Close();
  }

  public void Dispose()
  {
    this.fs.Dispose();
  }
}
```

See

A close curly brace should be located at the beginning of a line

Code Smell

Tabulation characters should not be used

Code Smell

Methods and properties should be named in PascalCase

Code Smell

Track uses of in-source issue suppressions

Code Smell

- MITRE, CWE-459 - Incomplete Cleanup

Available In:

sonarlint | sonarcloud | sonarqube