

- 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...



Track uses of "TODO" tags

Code Smell

Classes with "IDisposable" members should implement "IDisposable"

Bug

Calls to "async" methods should not be blocking

Code Smell

Child class fields should not shadow parent class fields

Code Smell

Track lack of copyright and license headers

Code Smell

Exit methods should not be called

Code Smell

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

"new Guid()" should not be used

Analyze your code

Code Smell Major ?

When the syntax `new Guid()` (i.e. parameterless instantiation) is used, it must be that one of three things is wanted:

1. An empty GUID, in which case `Guid.Empty` is clearer.
2. A randomly-generated GUID, in which case `Guid.NewGuid()` should be used.
3. A new GUID with a specific initialization, in which case the initialization parameter is missing.

This rule raises an issue when a parameterless instantiation of the `Guid` struct is found.

Noncompliant Code Example

```
public void Foo()
{
    var g1 = new Guid(); // Noncompliant - what's the intent
    Guid g2 = new(); // Noncompliant

    var g3 = default(Guid); // Noncompliant
    Guid g4 = default; // Noncompliant
}
```

Compliant Solution

```
public void Foo(byte[] bytes)
{
    var g1 = Guid.Empty;
    var g2 = Guid.NewGuid();
    var g3 = new Guid(bytes);
}
```

Available In:

sonarlint | sonarcloud | sonarqube

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