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...

too many lines of code
Code Smell

Magic numbers should not be used
Code Smell

Standard outputs should not be used directly to log anything
Code Smell

Files should not have too many lines of code
Code Smell

Lines should not be too long
Code Smell

HTTP response headers should not be vulnerable to injection attacks
Vulnerability

Console logging should not be used
Vulnerability

Generic parameters not constrained to reference types should not be compared to "null"
Bug

The length returned from a stream read should be checked
Bug

Method parameters, caught exceptions and foreach variables' initial values should not be ignored
Bug

Controlling permissions is security-sensitive
Security Hotspot

Writing cookies is security-sensitive
Security Hotspot

Unused method parameters should be removed

Analyze your code

Code Smell | Major | Quick Fix | unused

Unused parameters are misleading. Whatever the values passed to such parameters, the behavior will be the same.

This rule raises an issue when a `private` method or constructor of a class/struct takes a parameter without using it.

Noncompliant Code Example

```
private void DoSomething(int a, int b) // "b" is unused
{
    Compute(a);
}

private void DoSomething2(int a) // value of "a" is unused
{
    a = 10;
    Compute(a);
}
```

Compliant Solution

```
private void DoSomething(int a)
{
    Compute(a);
}

private void DoSomething2()
{
    var a = 10;
    Compute(a);
}
```

Exceptions

This rule doesn't raise any issue in the following contexts:

- The `this` parameter of extension methods.
- Methods decorated with attributes.
- Empty methods.
- Methods which only throw `NotImplementedException`.
- Main methods.
- `virtual`, `override` methods.
- interface implementations.

Available In:

sonarlint | sonarcloud | sonarqube

Methods should be named according to their synchronicities

Code Smell

Extensions should be in separate namespaces

Code Smell

Extension methods should not extend "object"

Code Smell

Operator overloads should have named alternatives

Code Smell