

- 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 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

Optional parameters should not be used

Code Smell

Public constant members should not be used

Methods should not have identical implementations

Analyze your code

Code Smell Major ? confusing duplicate suspicious

When two methods have the same implementation, either it was a mistake - something else was intended - or the duplication was intentional, but may be confusing to maintainers. In the latter case, one implementation should invoke the other.

Noncompliant Code Example

```
private const string CODE = "bounteous";
private int callCount = 0;

public string GetCode()
{
    callCount++;
    return CODE;
}

public string GetName() // Noncompliant
{
    callCount++;
    return CODE;
}
```

Compliant Solution

```
private const string CODE = "bounteous";
private int callCount = 0;






public string GetCode()
{
    callCount++;
    return CODE;
}

public string GetName()
{
    return GetCode();
}
```

Exceptions

Empty methods, methods with only one line of code and methods with the same name (overload) are ignored.

Available In: sonarlint | sonarcloud | sonarqube

be used
 Code Smell
Array covariance should not be used
 Code Smell
"nameof" should be used
 Code Smell
Modulus results should not be checked for direct equality
 Code Smell
"for" loop increment clauses should modify the loops' counters
 Code Smell