

- 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... 🔍

Code Smell

Events should have proper arguments

Code Smell

"P/Invoke" methods should not be visible

Code Smell

Native methods should be wrapped

Code Smell

Methods should not have identical implementations

Code Smell

Non-flags enums should not be marked with "FlagsAttribute"

Code Smell

Classes implementing "IEquatable<T>" should be sealed

Code Smell

"GC.SuppressFinalize" should not be called

Code Smell

Objects should not be disposed more than once

Code Smell

Parameter names used into ArgumentException constructors should match an existing one

Code Smell

"ISerializable" should be implemented correctly

Code Smell

"Assembly.Load" should be used

Code Smell

"IDisposable" should be implemented correctly

Properties should not make collection or array copies

Analyze your code

Code Smell Critical ? api-design performance

Most developers expect property access to be as efficient as field access. However, if a property returns a copy of an array or collection, it will be much slower than simple field access, contrary to the caller's likely expectations. Therefore, such properties should be refactored into methods so that callers are not surprised by the unexpectedly poor performance.

This rule detects calls to `ToList`, `ToArray` and `array.Clone`.

Noncompliant Code Example

```
private List<string> _foo = new List<string> { "a", "b", "c" }
public IEnumerable<string> Foo // Noncompliant
{
    get
    {
        return _foo.ToList();
    }
}

private string[] _bar = new string[] { "a", "b", "c" };
public IEnumerable<string> Bar // Noncompliant
{
    get
    {
        return (string[])_bar.Clone();
    }
}
```

Compliant Solution





```
private List<string> _foo = new List<string> { "a", "b", "c" }
private string[] _bar = new string[] { "a", "b", "c" };

public IEnumerable<string> GetFoo()
{
    return _foo.ToList();
}

public IEnumerable<string> GetBar()
{
    return (string[])_bar.Clone();
}
```

Available In:

sonarlint | sonarcloud | sonarqube

 Code Smell
"ServiceContract" and "OperationContract" attributes should be used together  Code Smell
Composite format strings should be used correctly  Code Smell
Exceptions should not be explicitly rethrown  Code Smell
"abstract" classes should not have