

- 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... 🔍

sensitive
Security Hotspot
Searching OS commands in PATH is security-sensitive
Security Hotspot
Creating cookies without the "HttpOnly" flag is security-sensitive
Security Hotspot
Creating cookies without the "secure" flag is security-sensitive
Security Hotspot
Literal suffixes should be upper case
Code Smell
Null checks should not be used with "is"
Code Smell
Method overloads should be grouped together
Code Smell
"params" should be used instead of "varargs"
Code Smell
"static" fields should be initialized inline
Code Smell
Classes that provide "Equals(<T>)" should implement "IEquatable<T>"
Code Smell
Jump statements should not be redundant
Code Smell
Member initializer values should not be redundant

"ThreadStatic" should not be used on non-static fields

Analyze your code

Bug Major Quick Fix unused

When a non-static class field is annotated with `ThreadStatic`, the code seems to show that the field can have different values for different calling threads, but that's not the case, since the `ThreadStatic` attribute is simply ignored on non-static fields.

So `ThreadStatic` should either be removed or replaced with a use of the `ThreadLocal<T>` class, which gives a similar behavior for non-static fields.

Noncompliant Code Example

```
public class MyClass
{
    [ThreadStatic] // Noncompliant
    private int count = 0;

    // ...
}
```

Compliant Solution

```
public class MyClass
{
    private int count = 0;

    // ...
}
```

or

```
public class MyClass
{
    private readonly ThreadLocal<int> count = new ThreadLocal<
    public int Count
    {
        get { return count.Value; }
        set { count.Value = value; }
    }
    // ...
}
```

Available In: sonarlint | sonarcloud | sonarqube

 Code Smell

Unassigned members should be removed

 Code Smell

Empty "case" clauses that fall through to the "default" should be omitted

 Code Smell

Parameters with "[DefaultParameterValue]" attributes should also be marked "[Optional]"

 Code Smell

Interfaces should not simply inherit from base interfaces with colliding