- 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...

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

"IDisposables" created in a "using" statement should not be returned
Bug

"ThreadStatic" fields should not be initialized
Bug

"Object.ReferenceEquals" should not be used for value types
Bug

Doubled prefix operators "!!" and "~~" should not be used
Bug

"=+" should not be used instead of "+="
Bug

"NaN" should not be used in comparisons
Bug

Conditionally executed code should be reachable
Bug

Null pointers should not be dereferenced
Bug

For-loop conditions should be true at least once
Bug

A "for" loop update clause should move the counter in the right direction
Bug

"ToString()" method should not return

Unread "private" fields should be removed

Analyze your code

Code Smell | Critical | cwe unused

Private fields only used to store values without reading them later is a case of dead store. So changing the value of such field is useless and most probably indicates a serious error in the code.

Noncompliant Code Example

```
public class Rectangle
{
  private readonly int length;
  private readonly int width;  // width is written but never

  public Rectangle(int length, int width)
  {
    this.length = length;
    this.width = width;
  }

  public int Surface
  {
    get
    {
      return length * length;
    }
  }
}
```

Compliant Solution

```
public class Rectangle
{
  private readonly int length;
  private readonly int width;

  public Rectangle(int length, int width)
  {
    this.length = length;
    this.width= width;
  }

  public int Surface
  {
    get
    {
      return length * width;
    }
  }
}
```

null

Bug

Return values from functions without side effects should not be ignored

Bug

Values should not be uselessly incremented

Bug

Collections should not be passed as arguments to their own methods

Bug

Related "if/else if" statements should

See

- MITRE, CWE-563 - Assignment to Variable without Use ('Unused Variable')

Available In:

sonarlint | sonarcloud | sonarqube