

- 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

Utility classes should not have public constructors

Code Smell

Local variables should not shadow class fields

Code Smell

Redundant pairs of parentheses should be removed

Code Smell

Inheritance tree of classes should not be too deep

Code Smell

Nested blocks of code should not be left empty

Code Smell

Methods should not have too many parameters

Code Smell

Collapsible "if" statements should be merged

Code Smell

OS commands should not be vulnerable to argument injection attacks

Vulnerability

Logging should not be vulnerable to injection attacks

Vulnerability

Empty collections should not be accessed or iterated

Bug

Mutable, non-private fields should not be "readonly"

### Nullable type comparison should not be redundant

Analyze your code

Bug Major redundant

Calling `GetType()` on a nullable object returns the underlying value type. Thus, comparing the returned `Type` object to `typeof(Nullable<SomeType>)` doesn't make sense. The comparison either throws an exception or the result can be known at compile time.

#### Noncompliant Code Example





```
int? nullable = 42;
bool comparison = nullable.GetType() == typeof(Nullable<int>)
comparison = nullable.GetType() != typeof(Nullable<int>); //

nullable = null;
comparison = nullable.GetType() != typeof(Nullable<int>); //
```

Available In:

sonarlint | sonarcloud | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.  
[Privacy Policy](#)

 Bug
<p>"string.ToCharArray()" should not be called redundantly</p> <p> Bug</p>
<p>"base.Equals" should not be used to check for reference equality in "Equals" if "base" is not "object"</p> <p> Bug</p>
<p>Property assignments should not be made for "readonly" fields not constrained to reference types</p> <p> Bug</p>
Flags enumerations should explicitly