C#

C# static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C# code

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

Bug

"string.ToCharArray()" should not be called redundantly

Bug

"base.Equals" should not be used to check for reference equality in "Equals" if "base" is not "object"

Bug

Property assignments should not be made for "readonly" fields not constrained to reference types

Bug

Flags enumerations should explicitly