C# static code analysis

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

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 null
Bug

Return values from functions without side effects should not be ignored
Bug

Base class methods should not be hidden

Analyze your code

Code Smell Critical pitfall

When a method in a derived class has the same name as a method in the base class but with a signature that only differs by types that are weakly derived (e.g. `object` vs `string`), the result is that the base method becomes hidden.

Noncompliant Code Example

```
using System;

namespace MyLibrary
{
  class Foo
  {
    internal void SomeMethod(string s1, string s2) { }
  }

  class Bar : Foo
  {
    internal void SomeMethod(string s1, object o2) { }  // N
  }
}
```

Compliant Solution

```
using System;

namespace MyLibrary
{
  class Foo
  {
    internal void SomeMethod(string s1, string s2) { }
  }

  class Bar : Foo
  {
    internal void SomeOtherMethod(string s1, object o2) { }
  }
}
```

Available In:

sonarlint sonarcloud sonarqube

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 not have the same condition

Bug

Objects should not be created to be dropped immediately without being used

Bug