C# static code analysis

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

"Explicit" conversions of "foreach" loops should not be used

Analyze your code

Code Smell   Critical   Quick Fix   suspicious

The `foreach` statement was introduced in the C# language prior to generics to make it easier to work with the non-generic collections available at that time such as `ArrayList`. The `foreach` statements allows you to downcast elements of a collection of `Objects` to any other type. The problem is that to achieve the cast, the `foreach` statements silently performs `explicit` type conversion, which at runtime can result in an `InvalidCastException`.

C# code iterating on generic collections or arrays should not rely on `foreach` statement's silent `explicit` conversions.

Noncompliant Code Example

```
public class Fruit { }
public class Orange : Fruit { }
public class Apple : Fruit { }

class MyTest
{
  public void Test()
  {
    var fruitBasket = new List<Fruit>();
    fruitBasket.Add(new Orange());
    fruitBasket.Add(new Orange());
    // fruitBasket.Add(new Apple());  // uncommenting this l

    foreach (Fruit fruit in fruitBasket)
    {
      var orange = (Orange)fruit; // This "explicit" convers
      ...
    }

    foreach (Orange orange in fruitBasket) // Noncompliant
    {
      ...
    }
  }
}
```

Compliant Solution

```
var fruitBasket = new List<Orange>();
fruitBasket.Add(new Orange());
fruitBasket.Add(new Orange());
// fruitBasket.Add(new Apple());  // uncommenting this line

foreach (Orange orange in fruitBasket)
{
  ...
}
```

Code Smell

Classes implementing "IEquatable<T>" should be sealed

Code Smell

"GC.SuppressFinalize" should not be called

Code Smell

Objects should not be disposed more than once

Code Smell

Parameter names used into ArgumentException constructors

or

```
var fruitBasket = new List<Fruit>();
fruitBasket.Add(new Orange());
fruitBasket.Add(new Orange());
fruitBasket.Add(new Apple());

foreach (Orange orange in fruitBasket.OfType<Orange>())
{
  ...
}
```

Exceptions

The rule ignores iterations on collections of `objects`. This includes legacy code that uses `ArrayList`. Furthermore, the rule does not report on cases when user defined conversions are being called.

Available In:

sonarlint | sonarcloud | sonarqube