C# static code analysis

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

Array covariance should not be used

Analyze your code

Code Smell   Critical   pitfall

Array covariance is the principle that if an implicit or explicit reference conversion exits from type `A` to `B`, then the same conversion exists from the array type `A[]` to `B[]`.

While this array conversion can be useful in readonly situations to pass instances of `A[]` where `B[]` is expected, it must be used with care, since assigning an instance of `B` into an array of `A` will cause an `ArrayTypeMismatchException` to be thrown at runtime.

Noncompliant Code Example

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

class Program
{
  static void Main(string[] args)
  {
    Fruit[] fruits = new Apple[1]; // Noncompliant - array c
    FillWithOranges(fruits);
  }

  // Just looking at the code doesn't reveal anything suspic
  static void FillWithOranges(Fruit[] fruits)
  {
    for (int i = 0; i < fruits.Length; i++)
    {
      fruits[i] = new Orange(); // Will throw an ArrayTypeMi
    }
  }
}
```

Compliant Solution

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

class Program
{
  static void Main(string[] args)
  {
    Orange[] fruits = new Orange[1]; // Compliant
    FillWithOranges(fruits);
  }

  static void FillWithOranges(Orange[] fruits)
  {
```

A close curly brace should be located at the beginning of a line

Code Smell

Tabulation characters should not be used

Code Smell

Methods and properties should be named in PascalCase

Code Smell

Track uses of in-source issue suppressions

Code Smell

```
      for (int i = 0; i < fruits.Length; i++)
      {
        fruits[i] = new Orange();
      }
    }
  }
```

Available In:

sonarlint | sonarcloud | sonarqube