C# static code analysis

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

Flags enumerations should explicitly initialize all their members

Analyze your code

Bug Minor

Flags enumerations should not rely on the language to initialize the values of their members. Implicit initialization will set the first member to 0, and increment the value by one for each subsequent member. This implicit behavior does not allow members to be combined using the bitwise or operator in a useful way.

Instead, 0 and powers of two (i.e. 1, 2, 4, 8, 16, ...) should be used to explicitly initialize all the members.

Noncompliant Code Example

```
[Flags]
enum FruitType    // Noncompliant
{
  None,
  Banana,
  Orange,
  Strawberry
}
class Program
{
    static void Main()
    {
        var bananaAndStrawberry = FruitType.Banana | FruitTy
        // Will display only Strawberry!
        Console.WriteLine(bananaAndStrawberry.ToString());
    }
}
```

Compliant Solution

```
[Flags]
enum FruitType
{
  None = 0,
  Banana = 1,
  Orange = 2,
  Strawberry = 4
}
class Program
{
    static void Main()
    {
        var bananaAndStrawberry = FruitType.Banana | FruitTy
        // Will display Banana and Strawberry, as expected.
        Console.WriteLine(bananaAndStrawberry.ToString());
    }
}
```

Exceptions

The default initialization of 0, 1, 2, 3, 4, … matches 0, 1, 2, 4, 8 … in the first three values, so no issue is reported if the first three members of the enumeration is not initialized.

Available In:

sonarlint | sonarcloud | sonarqube

Underscores should be used to make large numbers readable

Code Smell

"ToString()" calls should not be redundant

Code Smell

"==" should not be used when "Equals" is overridden

Code Smell

An abstract class should have both abstract and concrete methods

Code Smell