C#

C# static code analysis

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

Delegates should not be subtracted
Bug

"async" methods should not return "void"
Bug

"ThreadStatic" should not be used on non-static fields
Bug

"IDisposables" created in a "using" statement should not be returned
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

"default" clauses should be first or last

Analyze your code

Code Smell Critical

`switch` can contain a `default` clause for various reasons: to handle unexpected values, to show that all the cases were properly considered.

For readability purpose, to help a developer to quickly find the default behavior of a `switch` statement, it is recommended to put the `default` clause at the end of the `switch` statement. This rule raises an issue if the `default` clause is not the first or the last one of the `switch`'s cases.

Noncompliant Code Example

```
switch (param)
{
    case 0:
      DoSomething();
      break;
    default: // default clause should be the first or last o
      Error();
      break;
    case 1:
      DoSomethingElse();
      break;
}
```

Compliant Solution

```
switch (param)
{
    case 0:
      DoSomething();
      break;
    case 1:
      DoSomethingElse();
      break;
    default:
      Error();
      break;
}
```

Available In:

sonarlint | sonarcloud | sonarqube

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

Values should not be uselessly incremented