# C# static code analysis

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

All rules 409 | Vulnerability 34 | Bug 76 | Security Hotspot 28 | Code Smell 271 | Quick Fix 52

Tags

Search by name...

"const" instead
Code Smell

Strings or integral types should be used for indexers
Code Smell

Parameter names should not duplicate the names of their methods
Code Smell

Track use of "NotImplementedException"
Code Smell

Empty "default" clauses should be removed
Code Smell

Redundant property names should be omitted in anonymous classes
Code Smell

Declarations and initializations should be as concise as possible
Code Smell

Default parameter values should not be passed as arguments
Code Smell

Constructor and destructor declarations should not be redundant
Code Smell

Method parameters should be declared with base types
Code Smell

The simplest possible condition syntax should be used
Code Smell

Redundant parentheses should not be used
Code Smell

## Mutable, non-private fields should not be "readonly"

Analyze your code

Bug   Minor

Using the `readonly` keyword on a field means that it can't be changed after initialization. However, when applied to collections or arrays, that's only partly true. `readonly` enforces that another instance can't be assigned to the field, but it cannot keep the contents from being updated. That means that in practice, the field value really can be changed, and the use of `readonly` on such a field is misleading, and you're likely to not be getting the behavior you expect.

This rule raises an issue when a non-private, `readonly` field is an array or collection.

### Noncompliant Code Example

```
public class MyClass
{
  public readonly string[] strings;  // Noncompliant

  // ...
```

### Compliant Solution

```
public class MyClass
{
  public string[] strings;

  // ...
```

or

```
public class MyClass
{
  public readonly ImmutableArray<string> strings;

  // ...
```

or

```
public class MyClass
{
  private readonly string[] strings;

  // ...
```

Available In:

sonarlint sonarcloud sonarqube

"GC.SuppressFinalize" should not be invoked for types without destructors

Code Smell

Members should not be initialized to default values

Code Smell

Sequential tests should not check the same condition

Code Smell

Redundant modifiers should not be used

Code Smell