Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

# 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...

should not call "base" in "GetHashCode" or "Equals"
Bug

Anonymous delegates should not be used to unsubscribe from Events
Bug

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

## String offset-based methods should be preferred for finding substrings from offsets

Analyze your code

Code Smell | Critical | performance

Looking for a given substring starting from a specified offset can be achieved by such code: `str.Substring(startIndex).IndexOf(char1)`. This works well, but it creates a new `string` for each call to the `Substring` method. When this is done in a loop, a lot of `strings` are created for nothing, which can lead to performance problems if `str` is large.

To avoid performance problems, `string.Substring(startIndex)` should not be chained with the following methods:

- `IndexOf`
- `IndexOfAny`
- `LastIndexOf`
- `LastIndexOfAny`

For each of these methods, another method with an additional parameter is available to specify an offset.

Using these methods gives the same result while avoiding the creation of additional `String` instances.

### Noncompliant Code Example

```
str.Substring(StartIndex).IndexOf(char1); // Noncompliant; a
```

### Compliant Solution

```
str.IndexOf(char1, startIndex) - startIndex;
```

Available In:

sonarlint | sonarcloud | sonarqube

Bug

Null pointers should not be dereferenced

Bug

For-loop conditions should be true at least once

Bug

A "for" loop update clause should move the counter in the right direction

Bug

"ToString()" method should not return null