

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C# 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...



"protected" members

Code Smell

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

Multiple variables should not be declared on the same line

Code Smell

Culture should be specified for "string" operations

Code Smell

"switch" statements should have at least 3 "case" clauses

Code Smell

break statements should not be used except for switch cases

Code Smell

String literals should not be duplicated

Code Smell

Files should contain an empty newline at the end

Code Smell

Unused "using" should be removed

Code Smell

### URI Parameters should not be strings

Analyze your code

Code Smell Major ?

String representations of URIs or URLs are prone to parsing and encoding errors which can lead to vulnerabilities. The `System.Uri` class is a safe alternative and should be preferred. At minimum, an overload of the method taking a `System.Uri` as a parameter should be provided in each class that contains a method with an apparent Uri passed as a string.

This rule raises issues when a method has a string parameter with a name containing "uri", "Uri", "urn", "Urn", "url" or "Url", and the type doesn't declare a corresponding overload taking an `System.Uri` parameter instead.

#### Noncompliant Code Example

```
using System;

namespace MyLibrary
{
    public class MyClass
    {
        public void FetchResource(string uriString) { } // Noncompliant
    }
}
```

#### Compliant Solution





```
using System;

namespace MyLibrary
{
    public class MyClass
    {
        public void FetchResource(string uriString)
        {
            FetchResource(new Uri(uriString));
        }

        public void FetchResource(Uri uri) { }
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

<b>A close curly brace should be located at the beginning of a line</b>  Code Smell
<b>Tabulation characters should not be used</b>  Code Smell
<b>Methods and properties should be named in PascalCase</b>  Code Smell
<b>Track uses of in-source issue suppressions</b>  Code Smell