

- 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...



security-sensitive

Security Hotspot

"goto" statement should not be used

Code Smell

"new Guid()" should not be used

Code Smell

Parameter validation in "async"/"await" methods should be wrapped

Code Smell

Parameter validation in yielding methods should be wrapped

Code Smell

Events should have proper arguments

Code Smell

"P/Invoke" methods should not be visible

Code Smell

Native methods should be wrapped

Code Smell

Methods should not have identical implementations

Code Smell

Non-flags enums should not be marked with "FlagsAttribute"

Code Smell

Classes implementing "IComparable<T>" should be sealed

Code Smell

"GC.SuppressFinalize" should not be called

Code Smell

Objects should not be disposed more

Whitespace and control characters in string literals should be explicit

Analyze your code

Code Smell Critical pitfall

Non-encoded control characters and whitespace characters are often injected in the source code because of a bad manipulation. They are either invisible or difficult to recognize, which can result in bugs when the string is not what the developer expects. If you actually need to use a control character use their encoded version (ex: ASCII \n, \t, ... or Unicode U+000D, U+0009, ...).

This rule raises an issue when the following characters are seen in a literal string:

- [ASCII control character](#). (character index < 32 or = 127)
- Unicode [whitespace characters](#).
- Unicode [C0 control characters](#)
- Unicode characters U+200B, U+200C, U+200D, U+2060, U+FEFF, U+2028, U+2029

No issue will be raised on the simple space character. Unicode U+0020, ASCII 32.

Noncompliant Code Example

```
string tabInside = "A B"; // Noncompliant, contains a tab
string zeroWidthSpaceInside = "foobar"; // Noncompliant, it
Console.WriteLine(zeroWidthSpaceInside); // Prints "foo?bar"
```

Compliant Solution





```
string tabInside = "A\tB"; // Compliant, uses escaped value
string zeroWidthSpaceInside = "foo\u200Bbar"; // Compliant,
Console.WriteLine(zeroWidthSpaceInside); // Prints "foo?bar"
```

Exceptions

Verbatim string literals have no escape character mechanism.

Available In:

sonarlint | sonarcloud | sonarqube

than once
 Code Smell
Parameter names used into ArgumentException constructors should match an existing one
 Code Smell
"ISerializable" should be implemented correctly
 Code Smell
"Assembly.Load" should be used
 Code Smell
"IDisposable" should be implemented