

- 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...



Code Smell

"System.Uri" arguments should be used instead of strings
Code Smell

Collection properties should be readonly
Code Smell

Disposable types should declare finalizers
Code Smell

String URI overloads should call "System.Uri" overloads
Code Smell

URI properties should not be strings
Code Smell

URI return values should not be strings
Code Smell

URI Parameters should not be strings
Code Smell

Custom attributes should be marked with "System.AttributeUsageAttribute"
Code Smell

Assemblies should explicitly specify COM visibility
Code Smell

Assemblies should be marked as CLS compliant
Code Smell

"Generic.List" instances should not be part of public APIs
Code Smell

Collections should implement the generic interface

"IEnumerable" LINQs should be simplified

Analyze your code

Code Smell Major clumsy

In the interests of readability, code that can be simplified should be simplified. To that end, there are several ways IEnumerable language integrated queries (LINQ) can be simplified

- Use OfType instead of using Select with as to type cast elements and then null-checking in a query expression to choose elements based on type.
- Use OfType instead of using Where and the is operator, followed by a cast in a Select
- Use an expression in Any instead of Where(element => [expression]).Any().
- Use Count instead of Count() when it's available.
- Don't call ToArray() or ToList() in the middle of a query chain.

Using EntityFramework may require enforcing client evaluations. Such queries should use AsEnumerable() instead of ToArray() or ToList() in the middle of a query chain.

Noncompliant Code Example

```
seq1.Select(element => element as T).Any(element => element
seq2.Select(element => element as T).Any(element => element
seq3.Where(element => element is T).Select(element => elemen
seq4.Where(element => element is T).Select(element => (T)ele
seq5.Where(element => [expression]).Any(); // Noncompliant;

var num = seq6.Count(); // Noncompliant
var arr = seq.ToList().ToArray(); //Noncompliant
var count = seq.ToList().Count(x=>[condition]); //Noncompliant
```





Compliant Solution

```
seq1.OfType<T>().Any();
seq2.OfType<T>().Any(element => CheckCondition(element));
seq3.OfType<T>();
seq4.OfType<T>();
seq5.Any(element => [expression])

var num = seq6.Count;
var arr = seq.ToArray();
var count = seq.Count(x=>[condition]);
```

Available In:

sonarlint | sonarcloud | sonarqube

generic interface  Code Smell
Generic event handlers should be used  Code Smell
Event Handlers should have the correct signature  Code Smell
"Assembly.GetExecutingAssembly" should not be called  Code Smell
Arguments of public methods should