

- 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 Hotspot
Setting loose file permissions is security-sensitive
Security Hotspot
Formatting SQL queries is security-sensitive
Security Hotspot
Using hardcoded IP addresses is 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"

"Explicit" conversions of "foreach" loops should not be used

Analyze your code

Code Smell Critical Quick Fix suspicious

The `foreach` statement was introduced in the C# language prior to generics to make it easier to work with the non-generic collections available at that time such as `ArrayList`. The `foreach` statements allows you to downcast elements of a collection of `Objects` to any other type. The problem is that to achieve the cast, the `foreach` statements silently performs explicit type conversion, which at runtime can result in an `InvalidCastException`.

C# code iterating on generic collections or arrays should not rely on `foreach` statement's silent explicit conversions.

Noncompliant Code Example

```
public class Fruit { }
public class Orange : Fruit { }
public class Apple : Fruit { }

class MyTest
{
    public void Test()
    {
        var fruitBasket = new List<Fruit>();
        fruitBasket.Add(new Orange());
        fruitBasket.Add(new Orange());
        // fruitBasket.Add(new Apple()); // uncommenting this line





        foreach (Fruit fruit in fruitBasket)
        {
            var orange = (Orange)fruit; // This "explicit" conversion is noncompliant
            ...
        }

        foreach (Orange orange in fruitBasket) // Noncompliant
        {
            ...
        }
    }
}
```

Compliant Solution

```
var fruitBasket = new List<Orange>();
fruitBasket.Add(new Orange());
fruitBasket.Add(new Orange());
// fruitBasket.Add(new Apple()); // uncommenting this line

foreach (Orange orange in fruitBasket)
{
    ...
}
```

 Code Smell
Classes implementing "IEquatable<T>" should be sealed
 Code Smell
"GC.SuppressFinalize" should not be called
 Code Smell
Objects should not be disposed more than once
 Code Smell
Parameter names used into ArgumentException constructors

or

```
var fruitBasket = new List<Fruit>();
fruitBasket.Add(new Orange());
fruitBasket.Add(new Orange());
fruitBasket.Add(new Apple());

foreach (Orange orange in fruitBasket.OfType<Orange>())
{
    ...
}
```

Exceptions

The rule ignores iterations on collections of objects. This includes legacy code that uses ArrayList. Furthermore, the rule does not report on cases when user defined conversions are being called.

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 