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

"GC.SuppressFinalize" should not be called
Code Smell

Objects should not be disposed more 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 correctly
Code Smell

"ServiceContract" and "OperationContract" attributes should be used together
Code Smell

Composite format strings should be used correctly
Code Smell

Exceptions should not be explicitly rethrown
Code Smell

"abstract" classes should not have "public" constructors
Code Smell

Assertion arguments should be passed in the correct order
Code Smell

Field-like events should not be virtual

Code Smell | Critical | Quick Fix

Field-like events are events that do not have explicit `add` and `remove` methods. The compiler generates a `private delegate` field to back the event, as well as generating the implicit `add` and `remove` methods.

When a `virtual` field-like `event` is overridden by another field-like `event`, the behavior of the C# compiler is to generate a new `private delegate` field in the derived class, separate from the parent's field. This results in multiple and separate events being created, which is rarely what's actually intended.

To prevent this, remove the `virtual` designation from the parent class event.

Noncompliant Code Example

```
abstract class Car
{
  public virtual event EventHandler OnRefueled; // Noncompli

  public void Refuel()
  {
    // This OnRefueled will always be null
    if (OnRefueled != null)
    {
      OnRefueled(this, null);
    }
  }
}

class R2 : Car
{
  public override event EventHandler OnRefueled;
}

class Program
{
  static void Main(string[] args)
  {
    var r2 = new R2();
    r2.OnRefueled += new EventHandler((o, a) =>
    {
      Console.WriteLine("This event will never be called");
    });
    r2.Refuel();
  }
}
```

Compliant Solution

```
abstract class Car
{
  public event EventHandler OnRefueled; // Compliant
```

Ternary operators should not be nested

Code Smell

Events should be invoked

Code Smell

"params" should be used on overrides

Code Smell

Generic type parameters should be co/contravariant when possible

Code Smell

Multiple "OrderBy" calls should not be

```
  public void Refuel()
  {
    if (OnRefueled != null)
    {
      OnRefueled(this, null);
    }
  }
}

class R2 : Car {}

class Program
{
  static void Main(string[] args)
  {
    var r2 = new R2();
    r2.OnRefueled += new EventHandler((o, a) =>
    {
      Console.WriteLine("This event will be called");
    });
    r2.Refuel();
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube