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...

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

Ternary operators should not be nested
Code Smell

Events should be invoked
Code Smell

"params" should be used on overrides

Inappropriate casts should not be made

Analyze your code

Code Smell | Critical | cwe suspicious

Inappropriate casts are issues that will lead to unexpected behavior or runtime errors, such as `InvalidCastExceptions`. The compiler will catch bad casts from one class to another, but not bad casts to interfaces. Nor will it catch nullable values that are known to be null but that are cast to their underlying value types anyway.

It is much better to use the `as` operator because it will return `null` instead of throwing an exception.

Noncompliant Code Example

```
public interface IMyInterface
{ /* ... */ }

public class Implementer : IMyInterface
{ /* ... */ }

public class MyClass
{ /* ... */ }

public static class Program
{
  public static void Main()
  {
    var myclass = new MyClass();
    var x = (IMyInterface) myclass; // Noncompliant, Invalid
    var b = myclass is IMyInterface; // Noncompliant, always

    int? i = null;
    var ii = (int)i; // Noncompliant, InvalidOperationExcept
  }
}
```

Compliant Solution

```
public interface IMyInterface
{ /* ... */ }

public class Implementer : IMyInterface
{ /* ... */ }

public class MyClass
{ /* ... */ }

public static class Program
{
  public static void Main()
  {
    var myclass = new MyClass();
```

Code Smell

Generic type parameters should be co/contravariant when possible

Code Smell

Multiple "OrderBy" calls should not be used

Code Smell

Reflection should not be used to increase accessibility of classes, methods, or fields

Code Smell

Static fields should not be updated in

```
    var x = myclass as IMyInterface; // Compliant, but will
    var b = false;

    int? i = null;
    if (i.HasValue)
    {
      var ii = (int)i;
    }
  }
}
```

Exceptions

No issue is reported if the interface has no implementing class in the assembly.

See

- MITRE, CWE-588 - Attempt to Access Child of a Non-structure Pointer
- MITRE, CWE-704 - Incorrect Type Conversion or Cast

Available In:

sonarlint | sonarcloud | sonarqube