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

"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
Code Smell

Generic type parameters should be

"GC.Collect" should not be called

Analyze your code

Code Smell | Critical | performance unpredictable bad-practice

Calling `GC.Collect` is rarely necessary, and can significantly affect application performance. That's because it triggers a blocking operation that examines *every object in memory* for cleanup. Further, you don't have control over when this blocking cleanup will actually run.

As a general rule, the consequences of calling this method far outweigh the benefits unless perhaps you've just triggered some event that is unique in the run of your program that caused a lot of long-lived objects to die.

This rule raises an issue when `GC.Collect` is invoked.

Noncompliant Code Example

```
static void Main(string[] args)
{
  // ...
  GC.Collect(2, GCCollectionMode.Optimized); // Noncompliant
}
```

Available In:

sonarlint | sonarcloud | sonarqube

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 constructors

Code Smell