

- 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

OS commands should not be vulnerable to argument injection attacks

Vulnerability

Logging should not be vulnerable to injection attacks

Vulnerability

Empty collections should not be accessed or iterated

Bug

Mutable, non-private fields should not be "readonly"

Bug

"string.ToArray()" should not be called redundantly

Bug

"base.Equals" should not be used to check for reference equality in "Equals" if "base" is not "object"

Bug

Property assignments should not be made for "readonly" fields not constrained to reference types

Bug

Flags enumerations should explicitly initialize all their members

Bug

"GetHashCode" should not reference mutable fields

Bug

Results of integer division should not be assigned to floating point variables

Bug

Integral numbers should not be shifted by zero or more than their

Expressions used in "Debug.Assert" should not produce side effects

Analyze your code

Bug Major ?

An assertion is a piece of code that's used during development when the compilation debug mode is activated. It allows a program to check itself as it runs. When an assertion is true, that means everything is operating as expected.

In non-debug mode, all `Debug.Assert` are automatically left out. So, by contract, the boolean expressions that are evaluated by those assertions must absolutely not contain any side effects. Otherwise, when leaving the Debug mode, the functional behavior of the application is not the same anymore.

The rule will raise if the method name starts with any of the following `remove`, `delete`, `add`, `pop`, `update`, `retain`, `insert`, `push`, `append`, `clear`, `dequeue`, `enqueue`, `dispose`, `put`, or `set`, although `SetEquals` will be ignored.

Noncompliant Code Example





```
Debug.Assert(list.Remove("dog"));
```

Compliant Solution

```
bool result = list.Remove("dog");
Debug.Assert(result);
```

Available In:

sonarlint | sonarcloud | sonarqube

number of bits-1  Bug
"Equals(Object)" and "GetHashCode()" should be overridden in pairs  Bug
Having a permissive Cross-Origin Resource Sharing policy is security-sensitive  Security Hotspot
Delivering code in production with debug features activated is security-sensitive  Security Hotspot