sonar RULES

Products

- 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...

Classes that provide "Equals(<T>)" should implement "IEquatable<T>"
Code Smell

Jump statements should not be redundant
Code Smell

Member initializer values should not be redundant
Code Smell

Unassigned members should be removed
Code Smell

Empty "case" clauses that fall through to the "default" should be omitted
Code Smell

Parameters with "[DefaultParameterValue]" attributes should also be marked "[Optional]"
Code Smell

Interfaces should not simply inherit from base interfaces with colliding members
Code Smell

Variables should not be checked against the values they're about to be assigned
Code Smell

Methods should not return constants
Code Smell

Attribute, EventArgs, and Exception type names should end with the type being extended
Code Smell

Loops should be simplified with "LINQ" expressions
Code Smell

"=+" should not be used instead of "+="

Analyze your code

Bug Major

The use of operators pairs (=+, =- or =!) where the reversed, single operator was meant (+=, -= or !=) will compile and run, but not produce the expected results.

This rule raises an issue when =+, =-, or =! is used without any spacing between the two operators and when there is at least one whitespace character after.

Noncompliant Code Example

```
int target = -5;
int num = 3;

target =- num;  // Noncompliant; target = -3. Is that really
target =+ num; // Noncompliant; target = 3
```

Compliant Solution

```
int target = -5;
int num = 3;

target = -num;  // Compliant; intent to assign inverse value
target += num;
```

Available In:

sonarlint | sonarcloud | sonarqube

Namespaces should not be empty

Code Smell

Non-derived "private" classes and records should be "sealed"

Code Smell

"string.IsNullOrEmpty" should be used

Code Smell

Implementations should be provided for "partial" methods

Code Smell

Duplicate casts should not be made