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...

Assignments should not be made from within sub-expressions
Code Smell

General exceptions should never be thrown
Code Smell

Utility classes should not have public constructors
Code Smell

Local variables should not shadow class fields
Code Smell

Redundant pairs of parentheses should be removed
Code Smell

Inheritance tree of classes should not be too deep
Code Smell

Nested blocks of code should not be left empty
Code Smell

Methods should not have too many parameters
Code Smell

Collapsible "if" statements should be merged
Code Smell

OS commands should not be vulnerable to argument injection attacks
Vulnerability

Logging should not be vulnerable to injection attacks
Vulnerability

Empty nullable value should not be accessed

Analyze your code

Bug Major cwe

Nullable value types can hold either a value or `null`. The value held in the nullable type can be accessed with the `Value` property, but `.Value` throws an `InvalidOperationException` when the value is `null`. To avoid the exception, a nullable type should always be tested before `.Value` is accessed.

Noncompliant Code Example

```
int? nullable = null;
...
UseValue(nullable.Value); // Noncompliant
```

Compliant Solution

```
int? nullable = null;
...
if (nullable.HasValue)
{
  UseValue(nullable.Value);
}
```

or

```
int? nullable = null;
...
if (nullable != null)
{
  UseValue(nullable.Value);
}
```

See

- MITRE, CWE-476 - NULL Pointer Dereference

Available In:
sonarlint sonarcloud sonarqube

Empty collections should not be accessed or iterated

Bug

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

Bug

"string.ToCharArray()" 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