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...

Bug

"NaN" should not be used in comparisons

Bug

Conditionally executed code should be reachable

Bug

Null pointers should not be dereferenced

Bug

For-loop conditions should be true at least once

Bug

A "for" loop update clause should move the counter in the right direction

Bug

"ToString()" method should not return null

Bug

Return values from functions without side effects should not be ignored

Bug

Values should not be uselessly incremented

Bug

Collections should not be passed as arguments to their own methods

Bug

Related "if/else if" statements should not have the same condition

Bug

Objects should not be created to be dropped immediately without being used

Conditionals should start on new lines

Analyze your code

Code Smell Critical suspicious

Code is clearest when each statement has its own line. Nonetheless, it is a common pattern to combine on the same line an `if` and its resulting *then* statement. However, when an `if` is placed on the same line as the closing `}` from a preceding *then*, *else* or *else if* part, it is either an error - `else` is missing - or the invitation to a future error as maintainers fail to understand that the two statements are unconnected.

Noncompliant Code Example

```
if (condition1) {
  // ...
} if (condition2) {  // Noncompliant
  //...
}
```

Compliant Solution

```
if (condition1) {
  // ...
} else if (condition2) {
  //...
}
```

Or

```
if (condition1) {
  // ...
}

if (condition2) {
  //...
}
```

Available In:

sonarlint sonarcloud sonarqube

Bug

Identical expressions should not be used on both sides of a binary operator

Bug

Loops with at most one iteration should be refactored

Bug

Variables should not be self-assigned

Bug

Constructing arguments of system commands from user input is security-sensitive