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

Exceptions should not be thrown from property getters
Code Smell

Unused type parameters should be removed
Code Smell

Parameters should be passed in the correct order
Code Smell

Two branches in a conditional structure should not have exactly the same implementation
Code Smell

Unused assignments should be removed
Code Smell

Tests should not be ignored
Code Smell

"switch" statements should not have too many "case" clauses
Code Smell

Sections of code should not be commented out
Code Smell

Unused method parameters should be removed
Code Smell

Empty arrays and collections should be returned instead of null
Code Smell

Unused private types or members should be removed
Code Smell

Collection elements should not be replaced unconditionally

Analyze your code

Bug | Major | suspicious

It is highly suspicious when a value is saved for a key or index and then unconditionally overwritten. Such replacements are likely errors.

Noncompliant Code Example

```
list[index] = "value 1";
list[index] = "value 2";  // Noncompliant

dictionary.Add(key, "value 1");
dictionary[key] = "value 2"; // Noncompliant
```

Available In:

sonarlint | sonarcloud | sonarqube

Track uses of "FIXME" tags

Code Smell

"Obsolete" attributes should include explanations

Code Smell

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