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#

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

Types should not have members with visibility set higher than the type's visibility

Code Smell

Fields should be private

Code Smell

"try" statements with identical "catch" and/or "finally" blocks should be merged

Code Smell

NullReferenceException should not be caught

Code Smell

Functions should not have too many lines of code

Code Smell

"for" loop stop conditions should be invariant

Code Smell

Statements should be on separate lines

Code Smell

Classes should not be coupled to too many other classes (Single Responsibility Principle)

Code Smell

"switch case" clauses should not have too many lines of code

Code Smell

Magic numbers should not be used

Code Smell

Standard outputs should not be used directly to log anything

Code Smell

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

Analyze your code

Code Smell Major design suspicious

Having two `cases` in the same `switch` statement or branches in the same `if` structure with the same implementation is at best duplicate code, and at worst a coding error. If the same logic is truly needed for both instances, then in an `if` structure they should be combined, or for a `switch`, one should fall through to the other.

Noncompliant Code Example

```
switch (i)
{
  case 1:
    DoFirst();
    DoSomething();
    break;
  case 2:
    DoSomethingDifferent();
    break;
  case 3:  // Noncompliant; duplicates case 1's implementati
    DoFirst();
    DoSomething();
    break;
  default:
    DoTheRest();
}

if (a >= 0 && a < 10)
{
  DoFirst();
  DoTheThing();
}
else if (a >= 10 && a < 20)
{
  DoTheOtherThing();
}
else if (a >= 20 && a < 50)   // Noncompliant; duplicates fi
{
  DoFirst();
  DoTheThing();
}
```

Exceptions

Blocks in an `if` chain that contain a single line of code are ignored, as are blocks in a `switch` statement that contain a single line of code with or without a following `break`.

```
if (a >= 0 && a < 10)
{
  DoTheThing();
```

Files should not have too many lines of code

Code Smell

Lines should not be too long

Code Smell

HTTP response headers should not be vulnerable to injection attacks

Vulnerability

Console logging should not be used

Vulnerability

Generic parameters not constrained to

```
}
else if (a >= 10 && a < 20)
{
  DoTheOtherThing();
}
else if (a >= 20 && a < 50)    //no issue, usually this is d
{
  DoTheThing();
}
```

But this exception does not apply to `if` chains without `else`-s, or to `switch`-es without default clauses when all branches have the same single line of code. In case of `if` chains with `else`-s, or of `switch`-es with default clauses, rule {rule:csharpsquid:S3923} raises a bug.

```
if(a == 1)
{
  doSomething();  //Noncompliant, this might have been done
}
else if (a == 2)
{
  doSomething();
}
```

Available In:

sonarlint | sonarcloud | sonarqube