sonar RULES

Products

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...

"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

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 reference types should not be compared to "null"
Bug

The length returned from a stream read should be checked

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

Analyze your code

Code Smell · Major · brain-overload

When `switch` statements have large sets of case clauses, it is usually an attempt to map two sets of data. A `Dictionary` should be used instead to make the code more readable and maintainable.

Noncompliant Code Example

With a "Maximum number of case" set to 4

```
public class TooManyCase
{
    public int switchCase(char ch)
    {
        switch(ch) {  // Noncompliant
            case 'a':
                return 1;
            case 'b':
            case 'c':
                return 2;
            case 'd':
                return 3;
            case 'e':
                return 4;
            case 'f':
            case 'g':
            case 'h':
                return 5;
            default:
                return 6;
        }
    }
}
```

Compliant Solution

```
using System.Collections.Generic;

public class TooManyCase
{
    Dictionary<char, int> matching = new Dictionary<char, in
    {
        {'a', 1}, {'b', 2}, {'c', 2}, {'d', 3},
        {'e', 4}, {'f', 5}, {'g', 5}, {'h', 5}
    };

    public int withDictionary(char ch)
    {
        int value;
        if (this.matching.TryGetValue(ch, out value)) {
            return value;
```

Bug

Method parameters, caught exceptions and foreach variables' initial values should not be ignored

Bug

Controlling permissions is security-sensitive

Security Hotspot

Writing cookies is security-sensitive

Security Hotspot

Methods should be named according to their synchronicities

```
        } else {
            return 6;
        }
    }
}
```

Exceptions

This rule ignores `switches` over `Enums` and empty, fall-through cases.

Available In:

sonarlint sonarcloud sonarqube