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...

"protected" members
Code Smell

Underscores should be used to make large numbers readable
Code Smell

"ToString()" calls should not be redundant
Code Smell

"==" should not be used when "Equals" is overridden
Code Smell

An abstract class should have both abstract and concrete methods
Code Smell

Multiple variables should not be declared on the same line
Code Smell

Culture should be specified for "string" operations
Code Smell

"switch" statements should have at least 3 "case" clauses
Code Smell

break statements should not be used except for switch cases
Code Smell

String literals should not be duplicated
Code Smell

Files should contain an empty newline at the end
Code Smell

Unused "using" should be removed
Code Smell

## Reading the Standard Input is security-sensitive

Analyze your code

Security Hotspot | Critical

Reading Standard Input is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2005-2337
- CVE-2017-11449

It is common for attackers to craft inputs enabling them to exploit software vulnerabilities. Thus any data read from the standard input (stdin) can be dangerous and should be validated.

This rule flags code that reads from the standard input.

### Ask Yourself Whether

- data read from the standard input is not sanitized before being used.

You are at risk if you answered yes to this question.

### Recommended Secure Coding Practices

Sanitize all data read from the standard input before using it.

### Sensitive Code Example

```
using System;
public class C
{
    public void Main()
    {
        Console.In; // Sensitive
        var code = Console.Read(); // Sensitive
        var keyInfo = Console.ReadKey(...); // Sensitive
        var text = Console.ReadLine(); // Sensitive
        Console.OpenStandardInput(...); // Sensitive
    }
}
```

### Exceptions

This rule does not raise issues when the return value of the `Console.Read` `Console.ReadKey` or `Console.ReadLine` methods is ignored.

```
using System;
public class C
{
    public void Main()
    {
        Console.ReadKey(...); // Return value is ignored
        Console.ReadLine(); // Return value is ignored
    }
}
```

A close curly brace should be located at the beginning of a line

Code Smell

Tabulation characters should not be used

Code Smell

Methods and properties should be named in PascalCase

Code Smell

Track uses of in-source issue suppressions

Code Smell

**See**

- MITRE, CWE-20 - Improper Input Validation

**Deprecated**

This rule is deprecated, and will eventually be removed.

Available In:

sonarcloud | sonarqube