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

Using command line arguments is security-sensitive

Analyze your code

Security Hotspot | Critical

Using command line arguments is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2018-7281
- CVE-2018-12326
- CVE-2011-3198

Command line arguments can be dangerous just like any other user input. They should never be used without being first validated and sanitized.

Remember also that any user can retrieve the list of processes running on a system, which makes the arguments provided to them visible. Thus passing sensitive information via command line arguments should be considered as insecure.

This rule raises an issue when on every program entry points (`main` methods) when command line arguments are used. The goal is to guide security code reviews.

Ask Yourself Whether

- any of the command line arguments are used without being sanitized first.
- your application accepts sensitive information via command line arguments.

If you answered yes to any of these questions you are at risk.

Recommended Secure Coding Practices

Sanitize all command line arguments before using them.

Any user or application can list running processes and see the command line arguments they were started with. There are safer ways of providing sensitive information to an application than exposing them in the command line. It is common to write them on the process' standard input, or give the path to a file containing the information.

Sensitive Code Example

```
namespace MyNamespace
{
    class Program
    {
        static void Main(string[] args) // Sensitive if ther
        {
            string myarg = args[0];
            // ...
        }
    }
}
```

See

- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-88 - Argument Injection or Modification

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

- MITRE, CWE-214 - Information Exposure Through Process Environment
- SANS Top 25 - Insecure Interaction Between Components

Deprecated

This rule is deprecated, and will eventually be removed.

Available In:

sonarcloud | sonarqube