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

Assemblies should be marked with "NeutralResourcesLanguageAttribute"
Code Smell

Interfaces should not be empty
Code Smell

Enumerations should have "Int32" storage
Code Smell

Generic methods should provide type parameters
Code Smell

Multidimensional arrays should not be used
Code Smell

"static readonly" constants should be "const" instead
Code Smell

Strings or integral types should be used for indexers
Code Smell

Parameter names should not duplicate the names of their methods
Code Smell

Track use of "NotImplementedException"
Code Smell

Empty "default" clauses should be removed
Code Smell

Redundant property names should be omitted in anonymous classes
Code Smell

Declarations and initializations should

OS commands should not be vulnerable to argument injection attacks

Analyze your code

Vulnerability | Minor | injection cwe owasp sans-top25

Applications that allow execution of operating system commands from user-controlled data should control the arguments passed to the command, otherwise an attacker can inject additional arbitrary arguments which can change the behavior of the command.

User-controlled arguments should be sanitized by neutralizing argument delimiters (eg: ', space, –) and thus preventing injection of unwanted additional arguments. A single user-controlled argument may still lead to vulnerabilities if it corresponds to a dangerous option supported by the command, such as `–exec` available with find, in that case, mark end of option processing on the command line using `––` (double-dash) or restrict the options to only trusted values.

Noncompliant Code Example

```
using System.Diagnostics;
using Microsoft.AspNetCore.Mvc;

namespace WebApplicationDotNetCore.Controllers
{
    public class NoncompliantController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Run(string args)
        {
            Process p = new Process();
            p.StartInfo.FileName = "/usr/bin/file.exe";
            p.StartInfo.Arguments = args; // Noncompliant
            p.StartInfo.RedirectStandardOutput = true;
            p.Start();
            string output = p.StandardOutput.ReadToEnd();
            p.Dispose();

            return View();
        }
    }
}
```

Compliant Solution

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Microsoft.AspNetCore.Mvc;

namespace WebApplicationDotNetCore.Controllers
{
```

be as concise as possible

Code Smell

Default parameter values should not be passed as arguments

Code Smell

Constructor and destructor declarations should not be redundant

Code Smell

Method parameters should be declared with base types

Code Smell

The simplest possible condition

```
    public class CompliantController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Run(string arg)
        {
            Process p = new Process();
            p.StartInfo.FileName = "/usr/bin/file.exe";
            p.StartInfo.ArgumentList.Add(arg); // Compliant
            p.StartInfo.RedirectStandardOutput = true;
            p.Start();
            string output = p.StandardOutput.ReadToEnd();
            p.Dispose();

            return View();
        }
    }
}
```

See

- OWASP Top 10 2021 Category A3 - Injection
- OWASP OS Command Injection Defense Cheat Sheet
- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-88 - Argument Injection or Modification
- SANS Top 25 - Insecure Interaction Between Components

Available In:

sonarcloud | sonarqube Developer Edition