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 Sockets is security-sensitive

Analyze your code

Security Hotspot | Critical

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

- CVE-2011-178
- CVE-2017-5645
- CVE-2018-6597

Sockets are vulnerable in multiple ways:

- They enable a software to interact with the outside world. As this world is full of attackers it is necessary to check that they cannot receive sensitive information or inject dangerous input.
- The number of sockets is limited and can be exhausted. Which makes the application unresponsive to users who need additional sockets.

This rules flags code that creates sockets. It matches only the direct use of sockets, not use through frameworks or high-level APIs such as the use of http connections.

Ask Yourself Whether

- sockets are created without any limit every time a user performs an action.
- input received from sockets is used without being sanitized.
- sensitive data is sent via sockets without being encrypted.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- In many cases there is no need to open a socket yourself. Use instead libraries and existing protocols.
- Encrypt all data sent if it is sensitive. Usually it is better to encrypt it even if the data is not sensitive as it might change later.
- Sanitize any input read from the socket.
- Limit the number of sockets a given user can create. Close the sockets as soon as possible.

Sensitive Code Example

```
using System.Net.Sockets;

class TestSocket
{
    public static void Run()
    {
        // Sensitive
        Socket socket = new Socket(AddressFamily.InterNetwor

        // TcpClient and UdpClient simply abstract the detai
        TcpClient client = new TcpClient("example.com", 80);
        UdpClient listener = new UdpClient(80); // Sensitive
    }
}
```

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

- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-400 - Uncontrolled Resource Consumption ('Resource Exhaustion')
- MITRE, CWE-200 - Exposure of Sensitive Information to an Unauthorized Actor
- SANS Top 25 - Risky Resource Management
- SANS Top 25 - Porous Defenses

Deprecated

This rule is deprecated, and will eventually be removed.

Available In:

sonarcloud | sonarqube