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

Inner class members should not shadow outer class "static" or type members
Code Smell

"Explicit" conversions of "foreach" loops should not be used
Code Smell

Instance members should not write to "static" fields
Code Smell

"IndexOf" checks should not be for positive numbers
Code Smell

Whitespace and control characters in string literals should be explicit
Code Smell

Properties should not make collection or array copies
Code Smell

Flags enumerations zero-value members should be named "None"
Code Smell

Overflow checking should not be disabled for "Enumerable.Sum"
Code Smell

Field-like events should not be virtual
Code Smell

Non-constant static fields should not be visible
Code Smell

Inappropriate casts should not be made
Code Smell

Cryptographic keys should be robust

Analyze your code

Vulnerability | Critical | cwe privacy owasp

Most of cryptographic systems require a sufficient key size to be robust against brute-force attacks.

NIST recommendations will be checked for these use-cases:

Digital Signature Generation and **Verification:**

- p ≥ 2048 AND q ≥ 224 for DSA (`p` is key length and `q` the modulus length)
- n ≥ 2048 for RSA (`n` is the key length)

Key Agreement:

- p ≥ 2048 AND q ≥ 224 for DH and MQV
- n ≥ 224 for ECDH and ECMQV (Examples: `secp192r1` is a non-compliant curve (`n` < 224) but `secp224k1` is compliant (`n` >= 224))

Symmetric keys:

- key length ≥ 128 bits

This rule will not raise issues for ciphers that are considered weak (no matter the key size) like `DES`, `Blowfish`.

Noncompliant Code Example

```
using System;
using System.Security.Cryptography;

namespace MyLibrary
{
    public class MyCryptoClass
    {
        static void Main()
        {
            var dsa1 = new DSACryptoServiceProvider(); // No
            dsa1.KeySize = 2048; // Noncompliant - the sette

            var dsa2 = new DSACryptoServiceProvider(2048); /

            var rsa1 = new RSACryptoServiceProvider(); // No
            rsa1.KeySize = 2048; // Noncompliant - the sette

            var rsa2 = new RSACng(1024); // Noncompliant

            // ...
        }
    }
}
```

KeySize property of DSACryptoServiceProvider and RSACryptoServiceProvider does not change the value of underlying KeySize for the algorithm. Property setter is ignored without error and KeySize can be changed only by using constructor overload. See:

Constructors should only call non-overridable methods
Code Smell

"GC.Collect" should not be called
Code Smell

Methods should not be empty
Code Smell

Exceptions should not be thrown in finally blocks
Code Smell

Method overrides should not change

- DSACryptoServiceProvider.KeySize Property
- RSACryptoServiceProvider.KeySize Property

Compliant Solution

```
using System;
using System.Security.Cryptography;

namespace MyLibrary
{
    public class MyCryptoClass
    {
        static void Main()
        {
            var dsa1 = new DSACng(); // Compliant - default
            var dsa2 = new DSACng(2048); // Compliant
            var rsa1 = new RSACryptoServiceProvider(2048); /
            var rsa2 = new RSACng(); // Compliant - default

            // ...
        }
    }
}
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- Mobile AppSec Verification Standard - Cryptography Requirements
- OWASP Mobile Top 10 2016 Category M5 - Insufficient Cryptography
- NIST 800-131A - Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
- MITRE, CWE-326 - Inadequate Encryption Strength