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...

implementation
Bug

Types should be defined in named namespaces
Bug

Empty nullable value should not be accessed
Bug

Nullable type comparison should not be redundant
Bug

Methods with "Pure" attribute should return a value
Bug

One-way "OperationContract" methods should have "void" return type
Bug

Optional parameters should be passed to "base" calls
Bug

Classes should not have only "private" constructors
Bug

Expressions used in "Debug.Assert" should not produce side effects
Bug

Caller information parameters should come at the end of the parameter list
Bug

Static fields should appear in the order they must be initialized
Bug

Classes directly extending "object" should not call "base" in "GetHashCode" or "Equals"

Using non-standard cryptographic algorithms is security-sensitive

Security Hotspot | Critical | cwe sans-top25 owasp

The use of a non-standard algorithm is dangerous because a determined attacker may be able to break the algorithm and compromise whatever data has been protected. Standard algorithms like `AES`, `RSA`, `SHA`, ... should be used instead.

This rule tracks custom implementation of these types from `System.Security.Cryptography` namespace:

- `AsymmetricAlgorithm`
- `AsymmetricKeyExchangeDeformatter`
- `AsymmetricKeyExchangeFormatter`
- `AsymmetricSignatureDeformatter`
- `AsymmetricSignatureFormatter`
- `DeriveBytes`
- `HashAlgorithm`
- `ICryptoTransform`
- `SymmetricAlgorithm`

Recommended Secure Coding Practices

- Use a standard algorithm instead of creating a custom one.

Sensitive Code Example

```
public class CustomHash : HashAlgorithm // Noncompliant
{
    private byte[] result;

    public override void Initialize() => result = null;
    protected override byte[] HashFinal() => result;

    protected override void HashCore(byte[] array, int ibSta
        result ??= array.Take(8).ToArray();
}
```

Compliant Solution

```
SHA256 mySHA256 = SHA256.Create()
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- MITRE, CWE-327 - Use of a Broken or Risky Cryptographic Algorithm
- SANS Top 25 - Porous Defenses
- Derived from FindSecBugs rule MessageDigest is Custom

Available In:

sonarcloud | sonarqube

Bug

Anonymous delegates should not be used to unsubscribe from Events

Bug

Delegates should not be subtracted

Bug

"async" methods should not return "void"

Bug

"ThreadStatic" should not be used on non-static fields

Bug