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...

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"
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

Using pseudorandom number generators (PRNGs) is security-sensitive

Analyze your code

Security Hotspot | Critical | cwe owasp

Using pseudorandom number generators (PRNGs) is security-sensitive. For example, it has led in the past to the following vulnerabilities:

- CVE-2013-6386
- CVE-2006-3419
- CVE-2008-4102

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.

As the `System.Random` class relies on a pseudorandom number generator, it should not be used for security-critical applications or for protecting sensitive data. In such context, the `System.Cryptography.RandomNumberGenerator` class which relies on a cryptographically strong random number generator (RNG) should be used in place.

Ask Yourself Whether

- the code using the generated value requires it to be unpredictable. It is the case for all encryption mechanisms or when a secret value, such as a password, is hashed.
- the function you use generates a value which can be predicted (pseudo-random).
- the generated value is used multiple times.
- an attacker can access the generated value.

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

Recommended Secure Coding Practices

- Only use random number generators which are recommended by OWASP or any other trusted organization.
- Use the generated random values only once.
- You should not expose the generated random value. If you have to store it, make sure that the database or file is secure.

Sensitive Code Example

```
var random = new Random(); // Sensitive use of Random
byte[] data = new byte[16];
random.NextBytes(data);
return BitConverter.ToString(data); // Check if this value i
```

Compliant Solution

```
using System.Security.Cryptography;
...
var randomGenerator = RandomNumberGenerator.Create(); // Com
byte[] data = new byte[16];
```

"IDisposables" created in a "using" statement should not be returned

Bug

"ThreadStatic" fields should not be initialized

Bug

"Object.ReferenceEquals" should not be used for value types

Bug

Doubled prefix operators "!!" and "~~" should not be used

Bug

```
randomGenerator.GetBytes(data);
return BitConverter.ToString(data);
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- Mobile AppSec Verification Standard - Cryptography Requirements
- OWASP Mobile Top 10 2016 Category M5 - Insufficient Cryptography
- MITRE, CWE-338 - Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)
- MITRE, CWE-330 - Use of Insufficiently Random Values
- MITRE, CWE-326 - Inadequate Encryption Strength
- MITRE, CWE-1241 - Use of Predictable Algorithm in Random Number Generator
- Derived from FindSecBugs rule Predictable Pseudo Random Number Generator

Available In:

sonarcloud | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
Privacy Policy