sonar RULES

Products

- 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...

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

Constructors should only call non-overridable methods
Code Smell

Cipher Block Chaining IVs should be unpredictable

Analyze your code

Vulnerability   Critical   cwe owasp

When encrypting data with the Cipher Block Chaining (CBC) mode an Initialization Vector (IV) is used to randomize the encryption, ie under a given key the same plaintext doesn't always produce the same ciphertext. The IV doesn't need to be secret but should be unpredictable to avoid "Chosen-Plaintext Attack".

To generate Initialization Vectors, NIST recommends to use a secure random number generator.

Noncompliant Code Example

```
public void Encrypt(byte[] key, byte[] data, MemoryStream ta
{
    byte[] initializationVector = new byte[] { 1, 2, 3, 4, 5

    using var aes = new AesCryptoServiceProvider();
    var encryptor = aes.CreateEncryptor(key, initializationV

    using var cryptoStream = new CryptoStream(target, encryp
    cryptoStream.Write(data);
}
```

Compliant Solution

```
public byte[] Encrypt(byte[] key, byte[] data, MemoryStream
{
    using var aes = new AesCryptoServiceProvider();
    var encryptor = aes.CreateEncryptor(key, aes.IV); // aes

    using var cryptoStream = new CryptoStream(target, encryp
    cryptoStream.Write(data);

    return aes.IV;
}
```

See

- OWASP Top 10 2021 Category A2 - Cryptographic Failures
- OWASP Top 10 2017 Category A6 - Security Misconfiguration
- Mobile AppSec Verification Standard - Cryptography Requirements
- OWASP Mobile Top 10 2016 Category M5 - Insufficient Cryptography
- MITRE, CWE-329 - Not Using an Unpredictable IV with CBC Mode
- MITRE, CWE-330 - Use of Insufficiently Random Values
- MITRE, CWE-340 - Generation of Predictable Numbers or Identifiers
- MITRE, CWE-1204 - Generation of Weak Initialization Vector (IV)
- NIST, SP-800-38A - Recommendation for Block Cipher Modes of Operation

Available In:

"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 parameter defaults

Code Smell

sonarlint | 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