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...

Serialization event handlers should be implemented correctly
Bug

Deserialization methods should be provided for "OptionalField" members
Bug

All branches in a conditional structure should not have exactly the same 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

Using weak hashing algorithms is security-sensitive

Analyze your code

Security Hotspot | Critical | cwe spring owasp sans-top25

Cryptographic hash algorithms such as `MD2`, `MD4`, `MD5`, `MD6`, `HAVAL-128`, `HMAC-MD5`, `DSA` (which uses `SHA-1`), `RIPEMD`, `RIPEMD-128`, `RIPEMD-160`, `HMACRIPEMD160` and `SHA-1` are no longer considered secure, because it is possible to have `collisions` (little computational effort is enough to find two or more different inputs that produce the same hash).

Ask Yourself Whether

The hashed value is used in a security context like:

- User-password storage.
- Security token generation (used to confirm e-mail when registering on a website, reset password, etc ...).
- To compute some message integrity.

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

Recommended Secure Coding Practices

Safer alternatives, such as `SHA-256`, `SHA-512`, `SHA-3` are recommended, and for password hashing, it's even better to use algorithms that do not compute too "quickly", like `bcrypt`, `scrypt`, `argon2` or `pbkdf2` because it slows down `brute force attacks`.

Sensitive Code Example

```
var hashProvider1 = new MD5CryptoServiceProvider(); // Sensi
var hashProvider2 = (HashAlgorithm)CryptoConfig.CreateFromNa
var hashProvider3 = new SHA1Managed(); // Sensitive
var hashProvider4 = HashAlgorithm.Create("SHA1"); // Sensiti
```

Compliant Solution

```
var hashProvider1 = new SHA512Managed(); // Compliant
var hashProvider2 = (HashAlgorithm)CryptoConfig.CreateFromNa
var hashProvider3 = HashAlgorithm.Create("SHA512Managed"); /
```

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
- MITRE, CWE-1240 - Use of a Risky Cryptographic Primitive
- SANS Top 25 - Porous Defenses

Available In:

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

sonarcloud | sonarqube