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...

Silly bit operations should not be performed
Code Smell

Public methods should not have multidimensional array parameters
Code Smell

"async" and "await" should not be used as identifiers
Code Smell

TestCases should contain tests
Code Smell

Short-circuit logic should be used in boolean contexts
Code Smell

JWT should be signed and verified with strong cipher algorithms
Vulnerability

Cipher algorithms should be robust
Vulnerability

Encryption algorithms should be used with secure mode and padding scheme
Vulnerability

Insecure temporary file creation methods should not be used
Vulnerability

Server certificates should be verified during SSL/TLS connections
Vulnerability

LDAP connections should be authenticated
Vulnerability

Cryptographic keys should be robust

## Type inheritance should not be recursive

Analyze your code

Bug | Blocker

Recursion is acceptable in methods, where you can break out of it. But with class types, you end up with code that will compile but not run if you try to instantiate the class.

### Noncompliant Code Example

```
class C1<T>
{
}
class C2<T> : C1<C2<C2<T>>> // Noncompliant
{
}

...
var c2 = new C2<int>();
```

Available In:

sonarlint | sonarcloud | sonarqube

Vulnerability

Weak SSL/TLS protocols should not be used

Vulnerability

Cipher Block Chaining IVs should be unpredictable

Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks

Vulnerability

Hashes should include an unpredictable salt