

- 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... 🔍

"System.Type" instances

Code Smell

Test method signatures should be correct

Code Smell

Method overloads with default parameter values should not overlap

Code Smell

"value" parameters should be used

Code Smell

"is" should not be used with "this"

Code Smell

Methods named "Dispose" should implement "IDisposable.Dispose"

Code Smell

Tests should include assertions

Code Smell

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

Neither "Thread.Resume" nor "Thread.Suspend" should be used

Analyze your code

Bug Blocker multi-threading unpredictable

`Thread.Suspend` and `Thread.Resume` can give unpredictable results, and both methods have been deprecated. Indeed, if `Thread.Suspend` is not used very carefully, a thread can be suspended while holding a lock, thus leading to a deadlock. Other safer synchronization mechanisms should be used, such as `Monitor`, `Mutex`, and `Semaphore`.

Noncompliant Code Example

```
static void Main(string[] args)
{
    // ...
    Thread.CurrentThread.Suspend(); // Noncompliant
    Thread.CurrentThread.Resume(); // Noncompliant
}
```


See

- [Thread.Resume Method \(\)](#)
- [Thread.Suspend Method \(\)](#)

Available In:

sonarlint sonarcloud sonarqube

Weak 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