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...

"BeginInvoke" should be paired with calls to "EndInvoke"
Bug

"Shared" parts should not be created with "new"
Bug

Getters and setters should access the expected fields
Bug

Right operands of shift operators should be integers
Bug

Shared resources should not be used for locking
Bug

Locks should be released
Bug

Using publicly writable directories is security-sensitive
Security Hotspot

Using clear-text protocols is security-sensitive
Security Hotspot

Expanding archive files without controlling resource consumption is security-sensitive
Security Hotspot

Configuring loggers is security-sensitive
Security Hotspot

Using weak hashing algorithms is security-sensitive
Security Hotspot

Disabling CSRF protections is security-sensitive

"value" parameters should be used

Analyze your code

Code Smell | Blocker | pitfall

In property and indexer `set` methods, and in event `add` and `remove` methods, the implicit `value` parameter holds the value the accessor was called with. Not using the `value` means that the accessor ignores the caller's intent which could cause unexpected results at runtime.

Noncompliant Code Example

```
private int count;
public int Count
{
  get { return count; }
  set { count = 42; } // Noncompliant
}
```

Compliant Solution

```
private int count;
public int Count
{
  get { return count; }
  set { count = value; }
}
```

or

```
public int Count
{
  get { return count; }
  set { throw new InvalidOperationException(); }
}
```

Exceptions

This rule doesn't raise an issue when the setter is empty and part of the implementation of an `interface` . The assumption is that this part of the interface is not meaningful to that particular implementation. A good example of that would be a "sink" logger that discards any logs.

Available In:

sonarlint | sonarcloud | sonarqube

Security Hotspot

Using non-standard cryptographic algorithms is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is security-sensitive

Security Hotspot

Parameter names should match base declaration and other partial definitions

Code Smell

"ValueTask" should be consumed