

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C# 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...

Non-derived "private" classes and records should be "sealed"

Code Smell

"string.IsNullOrEmpty" should be used

Code Smell

Implementations should be provided for "partial" methods

Code Smell

Duplicate casts should not be made

Code Smell

Methods should not return values that are never used

Code Smell

Caller information arguments should not be provided explicitly

Code Smell

Method calls should not resolve ambiguously to overloads with "params"

Code Smell

"catch" clauses should do more than rethrow

Code Smell

Generic exceptions should not be ignored

Code Smell

Mutable fields should not be "public static"

Code Smell

Enumeration type names should not have "Flags" or "Enum" suffixes

Code Smell

Enumeration types should comply with a naming convention

Return values from functions without side effects should not be ignored

Analyze your code

Bug Major

When the call to a function doesn't have any side effects, what is the point of making the call if the results are ignored? In such case, either the function call is useless and should be dropped or the source code doesn't behave as expected.

This rule raises an issue when the results of the following methods are ignored:

- LINQ method,
- [Pure] method,
- any method on string, int, ..., System.Collections.Immutable.ImmutableArray<T>, ImmutableHashSet<T>, ...

Notes:

- although string.Intern has a side effect, ignoring its return value is still suspicious as it is the only reference ensured to point to the intern pool.
- Link methods can have side effects if they are misused. Example:

```
tests.All(c => { c.myfield = "foo"; return true; });
```

Such code should be rewritten as a normal loop.

Noncompliant Code Example

```
coll.Where(i => i > 5).Select(i => i*i); // Noncompliant
"this string".Equals("other string"); // Noncompliant
```

Compliant Solution





```
var res = coll.Where(i => i > 5).Select(i => i*i);
var isEqual = "this string".Equals("other string");
```

Exceptions

This rule doesn't report issues on method calls with out or ref arguments.

Available In:

sonarlint | sonarcloud | sonarqube

 Code Smell
<b>Trivial properties should be auto-implemented</b>  Code Smell
<b>Runtime type checking should be simplified</b>  Code Smell
<b>Boolean checks should not be inverted</b>  Code Smell
<b>Inheritance list should not be redundant</b>