

- 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... 🔍

Standard outputs should not be used directly to log anything

Code Smell

Files should not have too many lines of code

Code Smell

Lines should not be too long

Code Smell

HTTP response headers should not be vulnerable to injection attacks

Vulnerability

Console logging should not be used

Vulnerability

Generic parameters not constrained to reference types should not be compared to "null"

Bug

The length returned from a stream read should be checked

Bug

Method parameters, caught exceptions and foreach variables' initial values should not be ignored

Bug

Controlling permissions is security-sensitive

Security Hotspot

Writing cookies is security-sensitive

Security Hotspot

Methods should be named according to their synchronicities

Code Smell

Extensions should be in separate namespaces

Code Smell

Empty arrays and collections should be returned instead of null

Analyze your code

Code Smell Major ?

Returning null instead of an actual array, collection or map forces callers of the method to explicitly test for nullity, making them more complex and less readable.

Moreover, in many cases, null is used as a synonym for empty.

Noncompliant Code Example

```
public Result[] GetResults()
{
    return null; // Noncompliant
}

public IEnumerable<Result> GetResults()
{
    return null; // Noncompliant
}

public IEnumerable<Result> GetResults() => null; // Noncompliant

public IEnumerable<Result> Results
{
    get
    {
        return null; // Noncompliant
    }
}

public IEnumerable<Result> Results => null; // Noncompliant
```





Compliant Solution

```
public Result[] GetResults()
{
    return new Result[0];
}

public IEnumerable<Result> GetResults()
{
    return Enumerable.Empty<Result>();
}

public IEnumerable<Result> GetResults() => Enumerable.Empty<Result>()

public IEnumerable<Result> Results
{
    get
    {
        return Enumerable.Empty<Result>();
    }
}
```

Extension methods should not extend "object"  Code Smell
Operator overloads should have named alternatives  Code Smell
Non-abstract attributes should be sealed  Code Smell
Overloads with a "StringComparison" parameter should be used  Code Smell

```
}
```

```
public IEnumerable<Result> Results => Enumerable.Empty<Resul
```

Exceptions

Although `string` is a collection, the rule won't report on it.

Available In:

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)