

- 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... 🔍

visibility

Code Smell

Fields should be private

Code Smell

"try" statements with identical "catch" and/or "finally" blocks should be merged

Code Smell

NullReferenceException should not be caught

Code Smell

Functions should not have too many lines of code

Code Smell

"for" loop stop conditions should be invariant

Code Smell

Statements should be on separate lines

Code Smell

Classes should not be coupled to too many other classes (Single Responsibility Principle)

Code Smell

"switch case" clauses should not have too many lines of code

Code Smell

Magic numbers should not be used

Code Smell

Standard outputs should not be used directly to log anything

Code Smell

Files should not have too many lines of code

Code Smell

Tests should not be ignored

Analyze your code

Code Smell Major tests bad-practice confusing

When a test fails due, for example, to infrastructure issues, you might want to ignore it temporarily. But without some kind of notation about why the test is being ignored, it may never be reactivated. Such tests are difficult to address without comprehensive knowledge of the project, and end up polluting their projects.

This rule raises an issue for each ignored test that does not have a `WorkItem` attribute nor a comment about why it is being skipped on the right side of the `Ignore` attribute.

Noncompliant Code Example

```
[TestMethod]
[Ignore]
public void Test_DoTheThing()
{
    // ...
}
```

Compliant Solution

```
[TestMethod]
[Ignore] // renable when TCKT-1234 is fixed
public void Test_DoTheThing()
{
    // ...
}
```

or

```
[TestMethod]
[Ignore]
[WorkItem(1234)]
public void Test_DoTheThing()
{
    // ...
}
```

Exceptions

The rule doesn't raise an issue if:

- the test method is also marked with `WorkItem` attribute
- there is a comment on the right side of the `Ignore` attribute

Available In:

sonarlint | sonarcloud | sonarqube

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