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...

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

Lines should not be too long
Code Smell

Unused assignments should be removed

Analyze your code

Code Smell | Major | cwe unused

A dead store happens when a local variable is assigned a value that is not read by any subsequent instruction. Calculating or retrieving a value only to then overwrite it or throw it away, could indicate a serious error in the code. Even if it's not an error, it is at best a waste of resources. Therefore all calculated values should be used.

Noncompliant Code Example

```
i = a + b; // Noncompliant; calculation result not used befo
i = compute();
```

Compliant Solution

```
i = a + b;
i += compute();
```

Exceptions

No issue is reported when

- the analyzed method body contains `try` blocks,
- a lambda expression captures the local variables, or
- the variable is unused (case covered by Rule {rule:csharpsquid:S1481})
- initializations to `-1`, `0`, `1`, `null`, `true`, `false`, `""` and `string.Empty`.

See

- MITRE, CWE-563 - Assignment to Variable without Use ('Unused Variable')

Available In:

sonarlint | sonarcloud | sonarqube

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