

- 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... 🔍

exceptions and foreach variables' initial values should not be ignored



Controlling permissions is security-sensitive



Writing cookies is security-sensitive



Methods should be named according to their synchronicities



Extensions should be in separate namespaces



Extension methods should not extend "object"



Operator overloads should have named alternatives



Non-abstract attributes should be sealed



Overloads with a "StringComparison" parameter should be used



Overloads with a "CultureInfo" or an "IFormatProvider" parameter should be used



Types should not extend outdated base types



Properties should be preferred



Assignments should not be made from within sub-expressions

Analyze your code

Code Smell Major cwe suspicious

Assignments within sub-expressions are hard to spot and therefore make the code less readable. Ideally, sub-expressions should not have side-effects.

Noncompliant Code Example

```
if (string.IsNullOrEmpty(result = str.Substring(index, length)
{
    //...
}
```

Compliant Solution

```
var result = str.Substring(index, length);
if (string.IsNullOrEmpty(result)
{
    //...
}
```

Exceptions

Assignments inside lambda and delegate expressions are allowed.

Furthermore, the following patterns are also accepted:





```
var a = b = c = 10;
```

```
while ((val = GetNewValue()) > 0)
{
    ...
}
```

```
private MyClass instance;
public MyClass Instance
{
    get
    {
        return instance ?? (instance = new MyClass());
    }
}
```

See

- [MITRE, CWE-481](#) - Assigning instead of Comparing

Generics should be used when appropriate  Code Smell
Type names should not match namespaces  Code Smell
Strings should be normalized to uppercase  Code Smell
Exceptions should provide standard constructors  Code Smell