

- 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... 🔍

Methods should not be empty
Code Smell
Exceptions should not be thrown in finally blocks
Code Smell
Method overrides should not change parameter defaults
Code Smell
Types allowed to be deserialized should be restricted
Vulnerability
Server-side requests should not be vulnerable to forging attacks
Vulnerability
Members should not have conflicting transparency annotations
Vulnerability
"PartCreationPolicyAttribute" should be used with "ExportAttribute"
Bug
"ConstructorArgument" parameters should exist in constructors
Bug
Windows Forms entry points should be marked with STAThread
Bug
Collection elements should not be replaced unconditionally
Bug
Exceptions should not be created without being thrown
Bug
Collection sizes and array length comparisons should make sense

## Right operands of shift operators should be integers

Analyze your code

Bug Critical ?





Numbers can be shifted with the << and >> operators, but the right operand of the operation needs to be an int or a type that has an implicit conversion to int. However, with dynamic, the compiler's type checking is turned off, so you can pass anything to a shift operator and have it compile. And if the argument can't be converted to int at runtime, then a RuntimeBinderException will be raised.

### Noncompliant Code Example

```
dynamic d = 5;
var x = d >> 5.4; // Noncompliant
x = d >> null; // Noncompliant
x <<= new object(); // Noncompliant
```

Available In:

sonarlint sonarcloud sonarqube

 Bug
<p>Serialization event handlers should be implemented correctly</p> <p> Bug</p>
<p>Deserialization methods should be provided for "OptionalField" members</p> <p> Bug</p>
<p>All branches in a conditional structure should not have exactly the same implementation</p> <p> Bug</p>
<p>Types should be defined in named namespaces</p>