sonar RULES

Products

- 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...

Empty collections should not be accessed or iterated
Bug

Mutable, non-private fields should not be "readonly"
Bug

"string.ToCharArray()" should not be called redundantly
Bug

"base.Equals" should not be used to check for reference equality in "Equals" if "base" is not "object"
Bug

Property assignments should not be made for "readonly" fields not constrained to reference types
Bug

Flags enumerations should explicitly initialize all their members
Bug

"GetHashCode" should not reference mutable fields
Bug

Results of integer division should not be assigned to floating point variables
Bug

Integral numbers should not be shifted by zero or more than their number of bits-1
Bug

"Equals(Object)" and "GetHashCode()" should be overridden in pairs
Bug

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

## Caller information parameters should come at the end of the parameter list

Analyze your code

Bug  Major  api-design

Caller information attributes (`CallerFilePathAttribute`, `CallerLineNumberAttribute`, and `CallerMemberNameAttribute`) provide a way to get information about the caller of a method through optional parameters. But they only work right if their values aren't provided explicitly. So if you define a method with caller info attributes in the middle of the parameter list, you put your callers in a bad position: they are forced to use named arguments if they want to use the method properly.

### Noncompliant Code Example

```
void TraceMessage([CallerMemberName] string memberName = "",
  [CallerFilePath] string filePath = "",
  [CallerLineNumber] int lineNumber = 0,
  string message = null)  // Noncompliant
{
  /* ... */
}
```

### Compliant Solution

```
void TraceMessage(string message = null,
  [CallerMemberName] string memberName = "",
  [CallerFilePath] string filePath = "",
  [CallerLineNumber] int lineNumber = 0)
{
  /* ... */
}
```

Available In:

sonarlint | sonarcloud | sonarqube

Security Hotspot

**Delivering code in production with debug features activated is security-sensitive**

Security Hotspot

**Searching OS commands in PATH is security-sensitive**

Security Hotspot

**Creating cookies without the "HttpOnly" flag is security-sensitive**

Security Hotspot

**Creating cookies without the "secure" flag is security-sensitive**