C#

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...

security-sensitive
Security Hotspot

Creating cookies without the "HttpOnly" flag is security-sensitive
Security Hotspot

Creating cookies without the "secure" flag is security-sensitive
Security Hotspot

Literal suffixes should be upper case
Code Smell

Null checks should not be used with "is"
Code Smell

Method overloads should be grouped together
Code Smell

"params" should be used instead of "varargs"
Code Smell

"static" fields should be initialized inline
Code Smell

Classes that provide "Equals(<T>)" should implement "IEquatable<T>"
Code Smell

Jump statements should not be redundant
Code Smell

Member initializer values should not be redundant
Code Smell

Unassigned members should be removed
Code Smell

"IDisposables" created in a "using" statement should not be returned

Analyze your code

Bug Major

Typically you want to use `using` to create a local `IDisposable` variable; it will trigger disposal of the object when control passes out of the block's scope. The exception to this rule is when your method returns that `IDisposable`. In that case `using` disposes of the object before the caller can make use of it, likely causing exceptions at runtime. So you should either remove `using` or avoid returning the `IDisposable`.

Noncompliant Code Example

```
public FileStream WriteToFile(string path, string text)
{
  using (var fs = File.Create(path)) // Noncompliant
  {
    var bytes = Encoding.UTF8.GetBytes(text);
    fs.Write(bytes, 0, bytes.Length);
    return fs;
  }
}
```

Compliant Solution

```
public FileStream WriteToFile(string path, string text)
{
  var fs = File.Create(path);
  var bytes = Encoding.UTF8.GetBytes(text);
  fs.Write(bytes, 0, bytes.Length);
  return fs;
}
```

Available In:
sonarlint sonarcloud sonarqube

Code Smell

Empty "case" clauses that fall through to the "default" should be omitted

Code Smell

Parameters with "[DefaultParameterValue]" attributes should also be marked "[Optional]"

Code Smell

Interfaces should not simply inherit from base interfaces with colliding members

Code Smell

Variables should not be checked