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#

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...

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

"async" methods should not return "void"

Analyze your code

Bug Major multi-threading async-await

An `async` method with a `void` return type is a "fire and forget" method best reserved for event handlers because there's no way to wait for the method's execution to complete and respond accordingly. There's also no way to `catch` exceptions thrown from the method.

Having an `async void` method that is not an event handler could mean your program works some times and not others because of timing issues. Instead, `async` methods should return `Task`.

This rule raises an issue when non-event handler methods are both `async` and `void`.

Noncompliant Code Example

```
class HttpPrinter
{
  private string content;

  public async void CallNetwork(string url) //Noncompliant
  {
    var client = new HttpClient();
    var response = await client.GetAsync(url);
    content = await response.Content.ReadAsStringAsync();
  }

  public async Task PrintContent(string url)  // works corre
  {
    CallNetwork(url);
    await Task.Delay(1000);
    Console.Write(content);
  }
}
```

Compliant Solution

```
class HttpPrinter
{
  private string content;

  public async Task CallNetwork(string url)
  {
    var client = new HttpClient();
    var response = await client.GetAsync(url);
    content = await response.Content.ReadAsStringAsync();
  }

  public async Task PrintContent(string url)
  {
    await CallNetwork(url); // <----- call changed here. If
    await Task.Delay(1000);
```

Jump statements should not be redundant

Code Smell

Member initializer values should not be redundant

Code Smell

Unassigned members should be removed

Code Smell

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

Code Smell

```
      Console.Write(content);
    }
  }
```

Exceptions

Event handlers, i.e. methods with two arguments, first one matching type `object` or name `sender` and the second being or inheriting from `EventArgs`, are ignored.

Methods named as `OnSomething` are also ignored.

Available In:

sonarlint | sonarcloud | sonarqube