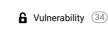




409

C# static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C# code





Security Hotspot

Tags

⊗ Code (28) Smell

Search by name.

Quick 52 Fix

Empty nullable value should not be accessed

Bug Bug

Nullable type comparison should not be redundant

₩ Bug

Methods with "Pure" attribute should return a value

🔐 Bug

One-way "OperationContract" methods should have "void" return type

📆 Bug

Optional parameters should be passed to "base" calls

₩ Bug

Classes should not have only "private" constructors

🖟 Bug

Expressions used in "Debug.Assert" should not produce side effects

Rug Bug

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

R Bug

Static fields should appear in the order they must be initialized

₩ Bug

Classes directly extending "object" should not call "base" in "GetHashCode" or "Equals"

R Bug

Anonymous delegates should not be used to unsubscribe from Events

📆 Bug

Delenates should not be subtracted



Analyze your code







cwe sans-top25 owasp

A cross-site request forgery (CSRF) attack occurs when a trusted user of a web application can be forced, by an attacker, to perform sensitive actions that he didn't intend, such as updating his profile or sending a message, more generally anything that can change the state of the application.

The attacker can trick the user/victim to click on a link, corresponding to the privileged action, or to visit a malicious web site that embeds a hidden web request and as web browsers automatically include cookies, the actions can be authenticated and sensitive.

Ask Yourself Whether

- The web application uses cookies to authenticate users.
- There exist sensitive operations in the web application that can be performed when the user is authenticated.
- The state / resources of the web application can be modified by doing HTTP POST or HTTP DELETE requests for example.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- · Protection against CSRF attacks is strongly recommended:
 - o to be activated by default for all unsafe HTTP methods.
 - implemented, for example, with an unquessable CSRF token
- Of course all sensitive operations should not be performed with safe HTTP methods like GET which are designed to be used only for information retrieval.

Noncompliant Code Example

```
public void ConfigureServices(IServiceCollection services)
    services.AddControllersWithViews(options => options.Filt
}
```

[HttpPost, IgnoreAntiforgeryToken] // Sensitive public IActionResult ChangeEmail(ChangeEmailModel model) =>

Compliant Solution

```
public void ConfigureServices(IServiceCollection services)
    services.AddControllersWithViews(options => options.Filt
    services.AddControllersWithViews(options => options.Filt
```

async" methods should not return
"void"

和 Bug

"ThreadStatic" should not be used on non-static fields

Delegates silvulu livi de subtracteu

"ThreadStatic" fields should not be

"IDisposables" created in a "using"

statement should not be returned

👬 Bug

// ...
}

[HttpPost]
[AutoValidateAntiforgeryToken]
public IActionResult ChangeEmail(ChangeEmailModel model) =>

See

• OWASP Top 10 2021 Category A1 - Broken Access Control
• MITRE, CWE-352 - Cross-Site Request Forgery (CSRF)
• OWASP Top 10 2017 Category A6 - Security Misconfiguration

SANS Top 25 - Insecure Interaction Between Components

Available In: sonarcloud sonarqube

OWASP: Cross-Site Request Forgery

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy