



SAP ABAP

APEX Apex

C C

© C++

CloudFormation

COBOL

C# C#

₹ CSS

-**GO** Go

HTML

👙 Java

Js JavaScript

Kotlin

Objective C

PHP

PL/I

PL/SQL PL/SQL

Python

RPG RPG

Ruby

Scala

Swift

Terraform

■ Text

Ts TypeScript

T-SQL

VB VB.NET

VB6 VB6

XML XML



C# static code analysis

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

All rules









Security
Hotspot









sealed classes should not have "protected" members

Code Smell

Underscores should be used to make large numbers readable

Code Smell

"ToString()" calls should not be redundant

Code Smell

"==" should not be used when "Equals" is overridden

Code Smell

An abstract class should have both abstract and concrete methods

Code Smell

Multiple variables should not be declared on the same line

各 Code Smell

Culture should be specified for "string" operations

Code Smell

"switch" statements should have at least 3 "case" clauses

Code Smell

break statements should not be used except for switch cases

各 Code Smell

String literals should not be duplicated

Code Smell

Files should contain an empty newline at the end

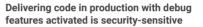
各 Code Smell

Unused "using" should be removed

各 Code Smell

A close curly brace should be located at the beginning of a line

Code Smell



Tags

Analyze your code

Security Hotspot





Search by name.

Delivering code in production with debug features activated is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2018-1999007
- CVE-2015-5306
- CVE-2013-2006

An application's debug features enable developers to find bugs more easily and thus facilitate also the work of attackers. It often gives access to detailed information on both the system running the application and users.

Ask Yourself Whether

- the code or configuration enabling the application debug features is deployed on production servers or distributed to end users.
- the application runs by default with debug features activated.

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

Recommended Secure Coding Practices

Do not enable debug features on production servers.

The .Net Core framework offers multiple features which help during debug.

 ${\tt Microsoft.AspNetCore.Builder.IApplicationBuilder.UseDeveloperExceptionPage} \ and \ \\$

Microsoft.AspNetCore.Builder.IApplicationBuilder.UseDatabaseErrorPage are two of them. Make sure that those features are disabled in production.

Use if (env.IsDevelopment()) to disable debug code.

Sensitive Code Example

This rule raises issues when the following .Net Core methods are called:

Microsoft.AspNetCore.Builder.IApplicationBuilder.UseDeveloperExceptionPage
Microsoft.AspNetCore.Builder.IApplicationBuilder.UseDatabaseErrorPage.

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;

namespace mvcApp
{
   public class Startup2
   {
      public void Configure(IApplicationBuilder app, IHostingEnviro {
            // Those calls are Sensitive because it seems that they vapp.UseDeveloperExceptionPage(); // Sensitive app.UseDatabaseErrorPage(); // Sensitive
      }
   }
}
```

Compliant Solution

Tabulation characters should not be used

Code Smell

Methods and properties should be named in PascalCase

Code Smell

Track uses of in-source issue suppressions

Code Smell

Exceptions

This rule does not analyze configuration files. Make sure that debug mode is not enabled by default in those files.

See

- OWASP Top 10 2021 Category A5 Security Misconfiguration
- OWASP Top 10 2017 Category A3 Sensitive Data Exposure
- MITRE, CWE-489 Active Debug Code
- MITRE, CWE-215 Information Exposure Through Debug Information

Available In:

sonarcloud 🔗 | sonarqube

© 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