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...

"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
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
Code Smell

String literals should not be duplicated
Code Smell

Files should contain an empty newline at the end
Code Smell

Unused "using" should be removed
Code Smell

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

Analyze your code

Security Hotspot | Minor | cwe privacy sans-top25 owasp

When a cookie is protected with the `secure` attribute set to *true* it will not be send by the browser over an unencrypted HTTP request and thus cannot be observed by an unauthorized person during a man-in-the-middle attack.

Ask Yourself Whether

- the cookie is for instance a *session-cookie* not designed to be sent over non-HTTPS communication.
- it's not sure that the website contains mixed content or not (ie HTTPS everywhere or not)

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

Recommended Secure Coding Practices

- It is recommended to use `HTTPs` everywhere so setting the `secure` flag to *true* should be the default behaviour when creating cookies.
- Set the `secure` flag to *true* for session-cookies.

Sensitive Code Example

When the `HttpCookie.Secure` property is set to `false` then the cookie will be send during an unencrypted HTTP request:

```
HttpCookie myCookie = new HttpCookie("Sensitive cookie");
myCookie.Secure = false; //  Sensitive: a security-sensitive
```

The default value of `Secure` flag is `false`, unless overwritten by an application's configuration file:

```
HttpCookie myCookie = new HttpCookie("Sensitive cookie");
//  Sensitive: a security-sensitive cookie is created with t
```

Compliant Solution

Set the `HttpCookie.Secure` property to `true`:

```
HttpCookie myCookie = new HttpCookie("Sensitive cookie");
myCookie.Secure = true; // Compliant
```

Or change the default flag values for the whole application by editing the Web.config configuration file:

```
<httpCookies httpOnlyCookies="true" requireSSL="true" />
```

- the `requireSSL` attribute corresponds programmatically to the `Secure` field.
- the `httpOnlyCookies` attribute corresponds programmatically to the `httpOnly` field.

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

Code Smell

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

See

- OWASP Top 10 2021 Category A4 - Insecure Design
- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- MITRE, CWE-311 - Missing Encryption of Sensitive Data
- MITRE, CWE-315 - Cleartext Storage of Sensitive Information in a Cookie
- MITRE, CWE-614 - Sensitive Cookie in HTTPS Session Without 'Secure' Attribute
- SANS Top 25 - Porous Defenses

Available In:

sonarcloud | sonarqube