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

Writing cookies is security-sensitive

Analyze your code

Security Hotspot | Minor

Using cookies is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2018-11639
- CVE-2016-6537

Attackers can use widely-available tools to read cookies. Any sensitive information they may contain will be exposed.

This rule flags code that writes cookies.

Ask Yourself Whether

- sensitive information is stored inside the cookie.

You are at risk if you answered yes to this question.

Recommended Secure Coding Practices

Cookies should only be used to manage the user session. The best practice is to keep all user-related information server-side and link them to the user session, never sending them to the client. In a very few corner cases, cookies can be used for non-sensitive information that need to live longer than the user session.

Do not try to encode sensitive information in a non human-readable format before writing them in a cookie. The encoding can be reverted and the original information will be exposed.

Using cookies only for session IDs doesn't make them secure. Follow OWASP best practices when you configure your cookies.

As a side note, every information read from a cookie should be Sanitized.

Sensitive Code Example

```
// === .Net Framework ===

HttpCookie myCookie = new HttpCookie("UserSettings");
myCookie["CreditCardNumber"] = "1234 1234 1234 1234"; // Sen
myCookie.Values["password"] = "5678"; // Sensitive
myCookie.Value = "mysecret"; // Sensitive
...
Response.Cookies.Add(myCookie);


// === .Net Core ===

Response.Headers.Add("Set-Cookie", ...); // Sensitive
Response.Cookies.Append("mykey", "myValue"); // Sensitive
```

See

- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure

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

- MITRE, CWE-312 - Cleartext Storage of Sensitive Information
- MITRE, CWE-315 - Cleartext Storage of Sensitive Information in a Cookie
- Derived from FindSecBugs rule COOKIE_USAGE

Deprecated

This rule is deprecated, and will eventually be removed.

Available In:

sonarcloud | sonarqube