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 "HttpOnly" flag is security-sensitive

Analyze your code

Security Hotspot | Minor | cwe sans-top25 privacy owasp

When a cookie is configured with the `HttpOnly` attribute set to *true*, the browser guaranties that no client-side script will be able to read it. In most cases, when a cookie is created, the default value of `HttpOnly` is *false* and it's up to the developer to decide whether or not the content of the cookie can be read by the client-side script. As a majority of Cross-Site Scripting (XSS) attacks target the theft of session-cookies, the `HttpOnly` attribute can help to reduce their impact as it won't be possible to exploit the XSS vulnerability to steal session-cookies.

Ask Yourself Whether

- the cookie is sensitive, used to authenticate the user, for instance a *session-cookie*
- the `HttpOnly` attribute offer an additional protection (not the case for an *XSRF-TOKEN cookie* / CSRF token for example)

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

Recommended Secure Coding Practices

- By default the `HttpOnly` flag should be set to *true* for most of the cookies and it's mandatory for session / sensitive-security cookies.

Sensitive Code Example

When the `HttpCookie.HttpOnly` property is set to `false` then the cookie can be accessed by client side code:

```
HttpCookie myCookie = new HttpCookie("Sensitive cookie");
myCookie.HttpOnly = false; // Sensitive: this cookie is crea
```

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

```
HttpCookie myCookie = new HttpCookie("Sensitive cookie");
// Sensitive: this cookie is created without the httponly fl
```

Compliant Solution

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

```
HttpCookie myCookie = new HttpCookie("Sensitive cookie");
myCookie.HttpOnly = true; // Compliant: the sensitive cookie
```

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.

See

- OWASP Top 10 2021 Category A5 - Security Misconfiguration
- OWASP HttpOnly
- OWASP Top 10 2017 Category A7 - Cross-Site Scripting (XSS)
- MITRE, CWE-1004 - Sensitive Cookie Without 'HttpOnly' Flag
- SANS Top 25 - Insecure Interaction Between Components
- Derived from FindSecBugs rule HTTPONLY_COOKIE

Available In:

sonarcloud | sonarqube

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