

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C# 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... 🔍

Overriding members should do more than simply call the same member in the base class

Code Smell

"Any()" should be used to test for emptiness

Code Smell

Boolean literals should not be redundant

Code Smell

Empty statements should be removed

Code Smell

Fields should not have public accessibility

Code Smell

URIs should not be hardcoded

Code Smell

Types should be named in PascalCase

Code Smell

Track uses of "TODO" tags

Code Smell

Classes with "IDisposable" members should implement "IDisposable"

Bug

Calls to "async" methods should not be blocking

Code Smell

Child class fields should not shadow parent class fields

Code Smell

Track lack of copyright and license headers

Code Smell

Setting loose file permissions is security-sensitive

Analyze your code

Security Hotspot Major cwe sans-top25 owasp

In Unix, "others" class refers to all users except the owner of the file and the members of the group assigned to this file.

In Windows, "Everyone" group is similar and includes all members of the Authenticated Users group as well as the built-in Guest account, and several other built-in security accounts.

Granting permissions to these groups can lead to unintended access to files.

Ask Yourself Whether

- The application is designed to be run on a multi-user environment.
- Corresponding files and directories may contain confidential information.

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

Recommended Secure Coding Practices

The most restrictive possible permissions should be assigned to files and directories.

Sensitive Code Example

.Net Framework:

```
var unsafeAccessRule = new FileSystemAccessRule("Everyone",  
  
var fileSecurity = File.GetAccessControl("path");  
fileSecurity.AddAccessRule(unsafeAccessRule); // Sensitive  
fileSecurity.SetAccessRule(unsafeAccessRule); // Sensitive  
File.SetAccessControl("fileName", fileSecurity);
```

.Net / .Net Core

```
var fileInfo = new FileInfo("path");  
var fileSecurity = fileInfo.GetAccessControl();  
  
fileSecurity.AddAccessRule(new FileSystemAccessRule("Everyon  
fileInfo.SetAccessControl(fileSecurity);
```


.Net / .Net Core using Mono.Posix.NETStandard

```
var fileSystemEntry = UnixFileSystemInfo.GetFileSystemEntry(  
fileSystemEntry.FileAccessPermissions = FileAccessPermission
```

Compliant Solution

.Net Framework

Exit methods should not be called

 Code Smell

Classes should "Dispose" of members from the classes' own "Dispose" methods

 Bug

Reading the Standard Input is security-sensitive

 Security Hotspot

Using command line arguments is security-sensitive

 Security Hotspot

```
var safeAccessRule = new FileSystemAccessRule("Everyone", Fi  
  
var fileSecurity = File.GetAccessControl("path");  
fileSecurity.AddAccessRule(safeAccessRule);  
File.SetAccessControl("path", fileSecurity);
```

.Net / .Net Core

```
var safeAccessRule = new FileSystemAccessRule("Everyone", Fi  
  
var fileInfo = new FileInfo("path");  
var fileSecurity = fileInfo.GetAccessControl();  
fileSecurity.SetAccessRule(safeAccessRule);  
fileInfo.SetAccessControl(fileSecurity);
```

.Net / .Net Core using Mono.Posix.NETStandard

```
var fs = UnixFileSystemInfo.GetFileSystemEntry("path");  
fs.FileAccessPermissions = FileAccessPermissions.UserExecute
```

See

- [OWASP Top 10 2021 Category A1](#) - Broken Access Control
- [OWASP Top 10 2021 Category A4](#) - Insecure Design
- [OWASP Top 10 2017 Category A5](#) - Broken Access Control
- [OWASP File Permission](#)
- [MITRE, CWE-732](#) - Incorrect Permission Assignment for Critical Resource
- [MITRE, CWE-266](#) - Incorrect Privilege Assignment
- [SANS Top 25](#) - Porous Defenses

Available In:

sonarcloud  sonarqube 