

- 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...

Exception types should be "public"

Code Smell

Cognitive Complexity of methods should not be too high

Code Smell

"params" should not be introduced on overrides

Code Smell

"[DefaultValue]" should not be used when "[DefaultParameterValue]" is meant

Code Smell

"[Optional]" should not be used on "ref" or "out" parameters

Code Smell

Non-flags enums should not be used in bitwise operations

Code Smell

Inner class members should not shadow outer class "static" or type members

Code Smell

"Explicit" conversions of "foreach" loops should not be used

Code Smell

Instance members should not write to "static" fields

Code Smell

"IndexOf" checks should not be for positive numbers

Code Smell

Whitespace and control characters in string literals should be explicit

Code Smell

Insecure temporary file creation methods should not be used

Analyze your code

Vulnerability Critical cwe owasp

Creating temporary files using insecure methods exposes the application to race conditions on filenames: a malicious user can try to create a file with a predictable name before the application does. A successful attack can result in other files being accessed, modified, corrupted or deleted. This risk is even higher if the application run with elevated permissions.

In the past, it has led to the following vulnerabilities:

- [CVE-2014-1858](#)
- [CVE-2014-1932](#)

`Path.GetTempFileName()` generates predictable file names and is inherently unreliable and insecure. Additionally, the method will raise an `IOException` if it is used to create more than 65535 files without deleting previous temporary files.

Recommended Secure Coding Practices

Out of the box, .NET is missing secure-by-design APIs to create temporary files. To overcome this, one of the following options can be used:

- Use a dedicated sub-folder with tightly controlled permissions
- Created temporary files in a publicly writable folder and make sure:
 - Generated filename is unpredictable
 - File is readable and writable only by the creating user ID
 - File descriptor is not inherited by child processes
 - File is destroyed as soon as it is closed

Noncompliant Code Example

```
var tempPath = Path.GetTempFileName(); // Noncompliant

using (var writer = new StreamWriter(tempPath))
{
    writer.WriteLine("content");
}
```








Compliant Solution

```
var randomPath = Path.Combine(Path.GetTempPath(), Path.GetRandomFileName());

// Creates a new file with write, non inheritable permission
using (var fileStream = new FileStream(randomPath, FileMode.Create))
using (var writer = new StreamWriter(fileStream))
{
    writer.WriteLine("content");
}
```

See

- [OWASP Top 10 2021 Category A1](#) - Broken Access Control

<div>Properties should not make collection or array copies</div> <div> Code Smell</div>	<div><ul style="list-style-type: none">• OWASP Top 10 2017 Category A9 - Using Components with Known Vulnerabilities• MITRE, CWE-377 - Insecure Temporary File• MITRE, CWE-379 - Creation of Temporary File in Directory with Incorrect Permissions• OWASP, Insecure Temporary File</div> <div>Available In:</div> <div>  </div>
<div>Flags enumerations zero-value members should be named "None"</div> <div> Code Smell</div>	
<div>Overflow checking should not be disabled for "Enumerable.Sum"</div> <div> Code Smell</div>	
<div>Field-like events should not be virtual</div> <div> Code Smell</div>	
<div>Non-constant static fields should not</div>	<div><p>© 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</p></div>