

- 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...



SQL keywords should be delimited by whitespace



Composite format strings should not lead to unexpected behavior at runtime



Recursion should not be infinite



Destructors should not throw exceptions



Hard-coded credentials are security-sensitive



Exceptions should not be thrown from unexpected methods



"operator==" should not be overloaded on reference types



Type should not be examined on "System.Type" instances



Test method signatures should be correct



Method overloads with default parameter values should not overlap



"value" parameters should be used



"is" should not be used with "this"



### I/O function calls should not be vulnerable to path injection attacks

Analyze your code

Vulnerability Blocker injection cwe owasp sans-top25

User-provided data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Constructing file system paths directly from tainted data could enable an attacker to inject specially crafted values, such as ' . / ' , that change the initial path and, when accessed, resolve to a path on the filesystem where the user should normally not have access.

A successful attack might give an attacker the ability to read, modify, or delete sensitive information from the file system and sometimes even execute arbitrary operating system commands. This is often referred to as a "path traversal" or "directory traversal" attack.

The mitigation strategy should be based on the whitelisting of allowed paths or characters.

#### Noncompliant Code Example

```
using System;
using Microsoft.AspNetCore.Mvc;

namespace WebApplicationDotNetCore.Controllers
{
    public class RSPEC2083IOInjectionNoncompliantController
    {
        public IActionResult Index()
        {
            return View();
        }

        public IActionResult DeleteFile(string fileName)
        {
            System.IO.File.Delete(fileName); // Noncompliant

            return Content("File " + fileName + " deleted");
        }
    }
}
```

#### Compliant Solution


```
using System;
using System.IO;
using Microsoft.AspNetCore.Mvc;

namespace WebApplicationDotNetCore.Controllers
{
    public class RSPEC2083IOInjectionCompliantController : Controller
    {
        public IActionResult Index()
        {
        }
```

Methods named "Dispose" should implement "IDisposable.Dispose"

 Code Smell


Tests should include assertions

 Code Smell

Silly bit operations should not be performed

 Code Smell

Public methods should not have multidimensional array parameters

 Code Smell

```
        return View();
    }

    public IActionResult DeleteFile(string fileName)
    {
        string destDirectory = "~/CustomersData/";

        string destFileName = Path.GetFullPath(System.IO
        string fullDestDirPath = Path.GetFullPath(destDi

        if (destFileName.StartsWith(fullDestDirPath, Str
        {
            System.IO.File.Delete(destFileName); // Comp
            return Content("File " + fileName + " delete
        } else
        {
            return BadRequest();
        }
    }
}
```

#### See

- [OWASP Top 10 2021 Category A1](#) - Broken Access Control
- [OWASP Top 10 2021 Category A3](#) - Injection
- [OWASP Top 10 2017 Category A1](#) - Injection
- [OWASP Top 10 2017 Category A5](#) - Broken Access Control
- [MITRE, CWE-20](#) - Improper Input Validation
- [MITRE, CWE-22](#) - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- [MITRE, CWE-99](#) - Improper Control of Resource Identifiers ('Resource Injection')
- [MITRE, CWE-641](#) - Improper Restriction of Names for Files and Other Resources
- [SANS Top 25](#) - Risky Resource Management

Available In:

**sonarcloud**  | **sonarqube**  Developer Edition