sonar RULES

Products

- 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#

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...

Vulnerability

HTTP request redirections should not be open to forging attacks
Vulnerability

Deserialization should not be vulnerable to injection attacks
Vulnerability

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks
Vulnerability

"CoSetProxyBlanket" and "CoInitializeSecurity" should not be used
Vulnerability

Database queries should not be vulnerable to injection attacks
Vulnerability

XML parsers should not be vulnerable to XXE attacks
Vulnerability

A secure password should be used when connecting to a database
Vulnerability

XPath expressions should not be vulnerable to injection attacks
Vulnerability

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

LDAP queries should not be vulnerable to injection attacks
Vulnerability

OS commands should not be vulnerable to command injection attacks

Extracting archives should not lead to zip slip vulnerabilities

Analyze your code

Vulnerability Blocker injection cwe owasp sans-top25

File names of the entries in a zip archive should be considered untrusted, tainted and should be validated before being used for file system operations. Indeed, file names can contain 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 special case of path injection vulnerabilities is called "zip slip".

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

Noncompliant Code Example

```
using System.Collections.Generic;
using System.IO;
using System.IO.Compression;

namespace ZipSlip
{
    public class ZipSlipNoncompliant
    {
        public void ExtractEntry(IEnumerator<ZipArchiveEntry
        {
            var entry = entriesEnumerator.Current;
            var destinationPath = Path.Combine(destinationDi
            entry.ExtractToFile(destinationPath); // Noncomp
        }
    }
}
```

Compliant Solution

```
using System.Collections.Generic;
using System.IO;
using System.IO.Compression;

namespace ZipSlip
{
    public class ZipSlipCompliant
    {
        public void ExtractEntry(IEnumerator<ZipArchiveEntry
        {
            var entry = entriesEnumerator.Current;
            var destinationDirectoryFullPath = Path.GetFullP
            var destinationPath = Path.Combine(destinationDi
            var destinationFullPath = Path.GetFullPath(desti
            if (!destinationFullPath.StartsWith(destinationD
```

Vulnerability

Classes should implement their "ExportAttribute" interfaces

Bug

Neither "Thread.Resume" nor "Thread.Suspend" should be used

Bug

"SafeHandle.DangerousGetHandle" should not be called

Bug

Type inheritance should not be recursive

```
            {
                throw new IOException("Attempting to extract
            }
            entry.ExtractToFile(destinationFullPath); // OK
        }
    }
}
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection
- snyk - Zip Slip Vulnerability
- 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