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...

Multiple "OrderBy" calls should not be used
Code Smell

Reflection should not be used to increase accessibility of classes, methods, or fields
Code Smell

Static fields should not be updated in constructors
Code Smell

"IEnumerable" LINQs should be simplified
Code Smell

Fields that are only assigned in the constructor should be "readonly"
Code Smell

Static fields should not be used in generic types
Code Smell

Multiline blocks should be enclosed in curly braces
Code Smell

Boolean expressions should not be gratuitous
Code Smell

Types and methods should not have too many generic parameters
Code Smell

Write-only properties should not be used
Code Smell

Exceptions should not be thrown from property getters
Code Smell

Server-side requests should not be vulnerable to forging attacks

Analyze your code

Vulnerability | Major | injection cwe sans-top25 owasp

User-supplied data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Performing requests from user-controlled data could allow attackers to make arbitrary requests on the internal network or to change their original meaning and thus to retrieve or delete sensitive information.

The problem could be mitigated in any of the following ways:

- Validate the user-provided data, such as the URL and headers, used to construct the request.
- Redesign the application to not send requests based on user-provided data.

Noncompliant Code Example

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

namespace WebApplicationDotNetCore.Controllers
{
    public class RSPEC5144SSRFNoncompliantController : Contr
    {
        public IActionResult Index()
        {
            return View();
        }

        public IActionResult ReadContentOfURL(string url)
        {
            HttpWebRequest request = (HttpWebRequest)WebRequ

            HttpWebResponse response  = (HttpWebResponse)req
            Stream dataStream         = response.GetResponse
            StreamReader reader       = new StreamReader(dat
            string responseFromServer = reader.ReadToEnd();

            reader.Close();
            dataStream.Close();
            response.Close();
            return Content(responseFromServer);
        }
    }
}
```

Compliant Solution

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

Unused type parameters should be removed

Code Smell

Parameters should be passed in the correct order

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

Unused assignments should be removed

Code Smell

```
namespace WebApplicationDotNetCore.Controllers
{
    public class RSPEC5144SSRFCompliantController : Controll
    {
        public IActionResult Index()
        {
            return View();
        }

        private readonly string[] whiteList = { "www.example

        public IActionResult ReadContentOfURL(string url)
        {
            // Extract the hostname from the URL
            URI remoteUrl     = new Uri(url);
            string remoteHost = remoteUrl.Host;

            // Match the incoming URL against a whitelist
            if (!whiteList.Contains(remoteHost))
            {
                return BadRequest();
            }

            HttpWebRequest request = (HttpWebRequest)WebRequ

            HttpWebResponse response  = (HttpWebResponse)req
            Stream dataStream         = response.GetResponse
            StreamReader reader       = new StreamReader(dat
            string responseFromServer = reader.ReadToEnd();

            reader.Close();
            dataStream.Close();
            response.Close();
            return Content(responseFromServer);
        }
    }
}
```

See

- OWASP Top 10 2021 Category A10 - Server-Side Request Forgery (SSRF)
- OWASP Attack Category - Server Side Request Forgery
- OWASP Top 10 2017 Category A5 - Broken Access Control
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-641 - Improper Restriction of Names for Files and Other Resources
- MITRE, CWE-918 - Server-Side Request Forgery (SSRF)
- SANS Top 25 - Risky Resource Management

Available In:

sonarcloud | sonarqube Developer Edition