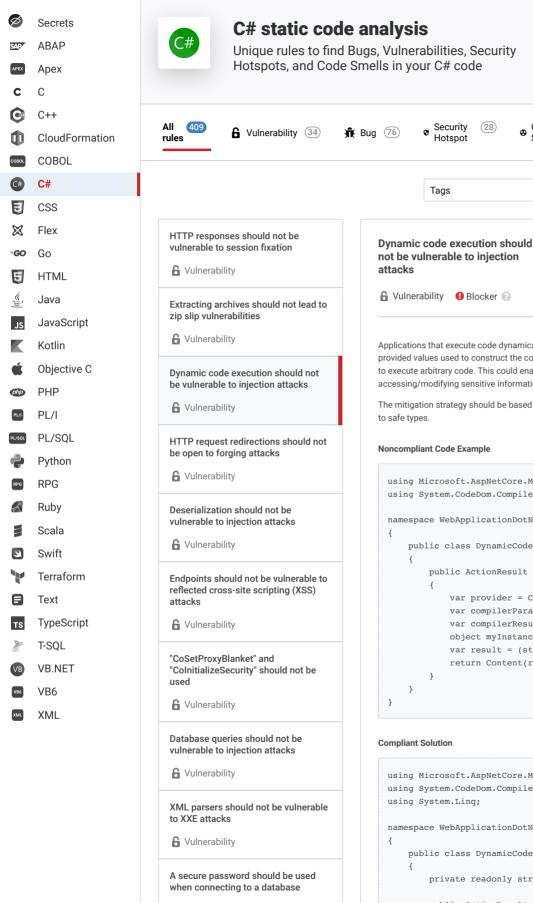
Analyze your code



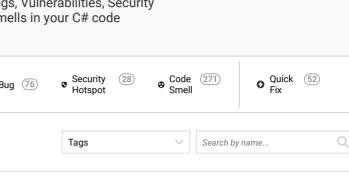


Vulnerability

6 Vulnerability

XPath expressions should not be

vulnerable to injection attacks



Blocker ? injection cwe owasp sans-top25 Applications that execute code dynamically should neutralize any externallyprovided values used to construct the code. Failure to do so could allow an attacker to execute arbitrary code. This could enable a wide range of serious attacks like accessing/modifying sensitive information or gain full system access. The mitigation strategy should be based on whitelisting of allowed values or casting

Noncompliant Code Example

```
using Microsoft.AspNetCore.Mvc:
using System.CodeDom.Compiler;
namespace WebApplicationDotNetCore.Controllers
   public class DynamicCodeExecutionNoncompliantController
       public ActionResult UnsafeCodeExecution(string code)
       {
            var provider = CodeDomProvider.CreateProvider("C
            var compilerParameters = new CompilerParameters
            var compilerResults = provider.CompileAssemblyFr
            object myInstance = compilerResults.CompiledAsse
            var result = (string)myInstance.GetType().GetMet
            return Content(result);
```

Compliant Solution

```
using Microsoft.AspNetCore.Mvc;
using System.CodeDom.Compiler;
using System.Linq;
namespace WebApplicationDotNetCore.Controllers
    public class DynamicCodeExecutionCompliantController : C
        private readonly string[] allowedInnerInvocations =
        public ActionResult SafeCodeExecution(string innerIn
            // Match the input against a whitelist
            if (!allowedInnerInvocations.Contains(innerInvoc
            {
                return BadRequest();
            // Code created is based on controlled template
```

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

Vulnerability

```
Classes should implement their 
"ExportAttribute" interfaces
```

I/O function calls should not be

🙀 Bug

```
var code = CreateFromTemplate(innerInvocationCod

var provider = CodeDomProvider.CreateProvider("C
    var compilerParameters = new CompilerParameters
    var compilerResults = provider.CompileAssemblyFr
    object myInstance = compilerResults.CompiledAsse
    var result = (string)myInstance.GetType().GetMet
    return Content(result);
}

private string CreateFromTemplate(string innerInvoca
{
    // Create code to be compiled from known templat
    // ...
}
}
```

See

- OWASP Top 10 2021 Category A3 Injection
- OWASP Top 10 2017 Category A1 Injection
- MITRE, CWE-20 Improper Input Validation
- <u>MITRE, CWE-95</u> Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')
- SANS Top 25 Risky Resource Management

Available In:

sonarcloud 🐼 sonarqube Developer Edition

© 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