

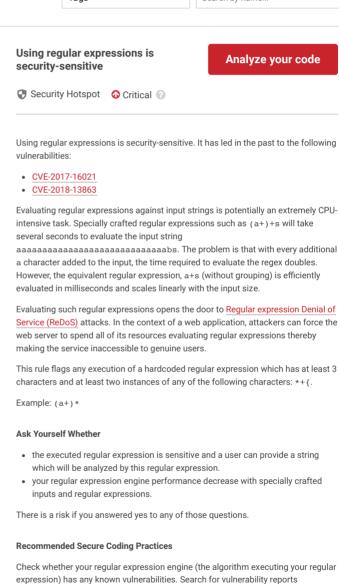
at the end

Code Smell

Code Smell

Unused "using" should be removed





If the regular expression is vulnerable to ReDos attacks, mitigate the risk by using a

"match timeout" to limit the time spent running the regular expression.

Remember also that a ReDos attack is possible if a user-provided regular expression is executed. This rule won't detect this kind of injection.

mentioning the one engine you're are using.

using System.Collections.Generic;

using System.Runtime.Serialization;

using System.Text.RegularExpressions;

Sensitive Code Example

using System;

using System.Ling;

using System.Web;

A close curly brace should be located at the beginning of a line

Code Smell

Tabulation characters should not be used

Code Smell

Methods and properties should be named in PascalCase

Code Smell

Track uses of in-source issue suppressions

Code Smell

```
namespace N
   public class RegularExpression
       void Foo(RegexOptions options, TimeSpan matchTimeout
                 string replacement, MatchEvaluator evaluato
        {
            // All the following instantiations are Sensitiv
            new System.Text.RegularExpressions.Regex("(a+)+"
            new System.Text.RegularExpressions.Regex("(a+)+"
            new System.Text.RegularExpressions.Regex("(a+)+"
            // All the following static methods are Sensitiv
            System.Text.RegularExpressions.Regex.IsMatch(inp
            System.Text.RegularExpressions.Regex.IsMatch(inp
            System.Text.RegularExpressions.Regex.IsMatch(inp
            System.Text.RegularExpressions.Regex.Match(input
            System.Text.RegularExpressions.Regex.Match(input
            System.Text.RegularExpressions.Regex.Match(input
            System.Text.RegularExpressions.Regex.Matches(inp
            System.Text.RegularExpressions.Regex.Matches(inp
            System.Text.RegularExpressions.Regex.Matches(inp
            System.Text.RegularExpressions.Regex.Replace(inp
            System.Text.RegularExpressions.Regex.Replace(inp
            System.Text.RegularExpressions.Regex.Replace(inp
            System.Text.RegularExpressions.Regex.Replace(inp
            System.Text.RegularExpressions.Regex.Replace(inp
            System.Text.RegularExpressions.Regex.Replace(inp
            System.Text.RegularExpressions.Regex.Split(input
            System.Text.RegularExpressions.Regex.Split(input
            System.Text.RegularExpressions.Regex.Split(input
   }
}
```

Exceptions

Some corner-case regular expressions will not raise an issue even though they might be vulnerable. For example: (a | aa) +, (a | a?) +.

It is a good idea to test your regular expression if it has the same pattern on both side of a " $\!\!\!\!|$ ".

See

- OWASP Top 10 2017 Category A1 Injection
- MITRE, CWE-624 Executable Regular Expression Error
- OWASP Regular expression Denial of Service ReDoS

Deprecated

This rule is deprecated; use {rule:roslyn.sonaranalyzer.security.cs:S2631} instead.

Available In:

sonarcloud <equation-block> 🛮 sonarqube

© 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