

Affecting [org.webjars.bower:codemirror](#) package, versions [5.58.2)

Share

Upgrade `org.webjars.bower:codemirror` to version 5.58.2 or higher.

Affected versions of this package are vulnerable to Regular Expression Denial of Service (ReDoS). The vulnerable regular expression is located in <https://github.com/codemirror/CodeMirror/blob/cdb228ac736369c685865b122b736cd0d397836c/mode/javascript/javascript.js#L129>. The ReDoS vulnerability of the regex is mainly due to the sub-pattern `(\s|/|*,*\?/*) *`

```
var value = "asynj**/*//*/**/*/*//**/**//**/*//**/*//**/*//**/*  
/**//**/*//**/*//**/*//**/*//**/*//**/*//**/*//**/*//**/*  
/**//**/*//**/*//**/*//**/*//**/*//**/*//**/*//**/*//**/*!  
var editor =  
CodeMirror(document.body.getElementsByTagName("article")[0], { value: value, lineNumber: true, mode:  
"javascript", keyMap: "sublime", autoCloseBrackets: true, matchBrackets: true, showCursorWhenSelecting:  
true, theme: "monokai", tabSize: 2 });
```

Let's take the following regular expression as an example:

```
regex = /A(B|C+)+D/
```

- A The string must start with the letter 'A'
- (8|+)+ The string must then follow the letter A with either the letter 'B' or some number of occurrences of the letter 'C' (the + matches one or more times). The + at the end of this section states that we can look for one or more matches of this section.
- d Finally, we ensure this section of the string ends with a 'D'

In most cases, it doesn't take very long for a regex engine to find a match:

```
$ time node -e '/A(B|C)+/.test("ACCCCCCCCCCCCCCCCCCCCCCCCCCD")' 0.04s user 0.01s system 95% cpu 0.052
total
```

```
$ time node -e '/A(B|C)+D/.test("ACCCCCCCCCCCCCCCCCCCCCCCCCCX")' 1.79s user 0.02s system 99% cpu 1.812
total
```

1. CCC
2. CC+C
3. C+CC

[See more](#)

7.5 HIGH

Test your applications

Snyk ID	SNYK-JAVA-ORGWEBJARSBOWER-1024445
Published	30 Oct 2020
Disclosed	9 Oct 2020
Credit	Yeting Li

Found a mistake?

4. C+C+C.

The engine has to try each of those combinations to see if any of them potentially match against the expression. When you combine that with the other steps the engine must take, we can use [RegEx 101 debugger](#) to see the engine has to take a total of 38 steps before it can determine the string doesn't match.

From there, the number of steps the engine must use to validate a string just continues to grow.

String	Number of C's	Number of steps
ACCCX	3	38
ACCCCX	4	71
ACCCCCX	5	136
ACCCCCCCCCCCCCX	14	65,553

By the time the string includes 14 C's, the engine has to take over 65,000 steps just to see if the string is valid. These extreme situations can cause them to work very slowly (exponentially related to input size, as shown above), allowing an attacker to exploit this and can cause the service to excessively consume CPU, resulting in a Denial of Service.

References

- [GitHub Commit](#)

PRODUCT

[Snyk Open Source](#)

[Snyk Code](#)

[Snyk Container](#)

[Snyk Infrastructure as Code](#)

[Test with Github](#)

[Test with CLI](#)

RESOURCES

[Vulnerability DB](#)

[Documentation](#)

[Disclosed Vulnerabilities](#)

[Blog](#)

[FAQs](#)

COMPANY

[About](#)

[Jobs](#)

[Contact](#)

[Policies](#)

[Do Not Sell My Personal Information](#)

CONTACT US

[Support](#)

[Report a new vuln](#)

[Press Kit](#)

[Events](#)

FIND US ONLINE

TRACK OUR DEVELOPMENT

