




# Paralyzing the Node one RegEx at a time





# \$ whoami && who is Snyk-labs?

Security Research Team at Snyk 

 Skateboarding /  Snowboarding

 Cats

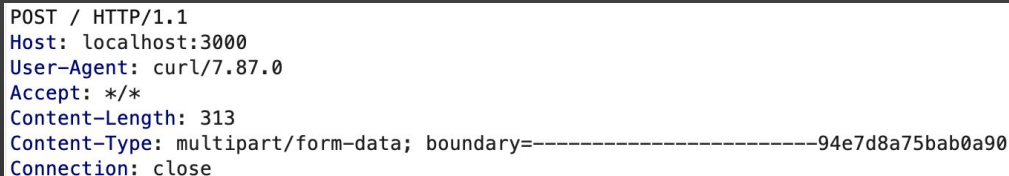
 Craft beer

 UK ➡  Zurich

- Snyk (pronounced sneak) is a developer security platform for securing code, dependencies, containers, and infrastructure as code.
- Snyk Security Lab aims to position Snyk as a leader in open source security through innovative research
- Contribute security expertise to strategic company directions
- Snyk product improvements



/ (Re) DoS / g



value1

```

~
~!. ....~. ..~.7??^57...
:!:~. . .7!::...^ .!^
^J?J?J?7!. ~G#&&##&#G7. . :!55JJJ5Y5Y75G5
J5Y:..755^ 7G&&B?^??^B&&#Y?P?~:..:~!YPP?YJ7JY5P?.
.55Y77?YY^ ~&&&7 .: ?&&&5!7!!~!^!77~::~. . . . .
!5Y:..:YYJ Y&## G&P B#&? .~:. . . . .^ .: .^ .: .^
Y5Y?7?!!~: ~G&SJ .:^^. .... . . . . .
.:~. .B#&SJ .:~7JJJ: J5Y. ~5YYYYY5Y!: :YYYYYYYYY? !YYJJY5J.
.#B#~ !&# :Y##B.J55!... :55? JYY. :J55~ 75Y^..... 7P57...
.#&&P ^ .#&&P .!7JYY?7: !YY: :557 7Y5^ YYYJJJJJ^ .7?Y5YJ7:
:B&&B?~::~YBBB5... .755! YYJ !7J^ .!5Y! ^YY!..... ^....75PJ
7GB&&&&&&B5~ !7JJ?JY?^ ^Y5! .Y?7JJJ?!. ???7JJJY: !Y5YJ?YY!

```

-----94e7d8a75bab0a90-----



1 x

2 x

3 x

+

Send

Cancel

< ▾

> ▾

Target: <http://localhost:3000> HTTP/1

### Request

PrettyRawHex


ln

```
2 Host: localhost:3000
3 User-Agent: curl/7.87.0
4 Accept: */*
5 Connection: close
6
7 |
```

Search..

0 highlights

### Response



Waiting



# \$ What are Regular Expressions?

**Definition:** A Regular Expression is a sequence of characters that forms a search pattern.

## Purpose:

- **Search:** Find specific content within text
- **Match:** Check if a string follows a particular pattern
- **Replace:** Change certain parts of a string
- **Split:** Break a string into smaller pieces

```
/https?:\/\/[^\s]+/
```

URL

```
/^\d{4}[-\s]\d{9}$/
```

CH number

```
 /^[^\s@]+@[^\s]+\.[^\s]+$/
```

Email address

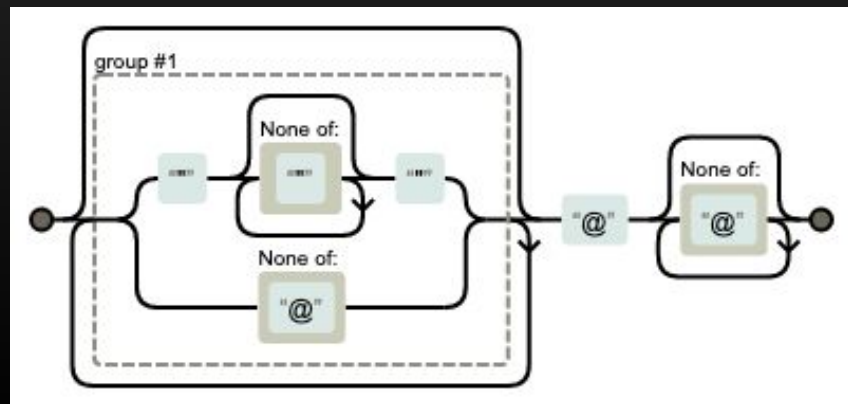
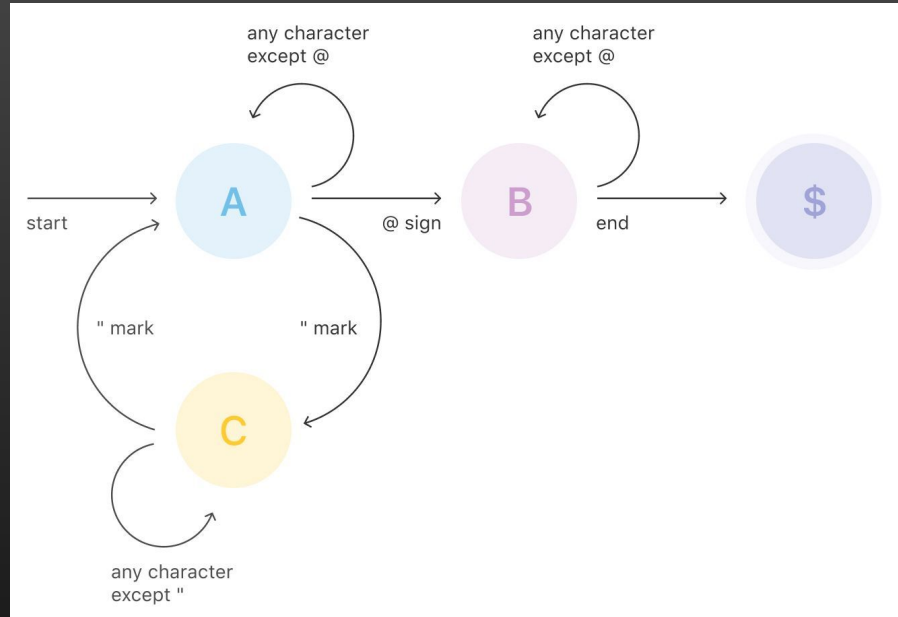






Pattern: `/ (" [^"]*" | [^@]) * @ [^@] * /`

- Input 1: [patch@snyk.io](mailto:patch@snyk.io)
  - We reach **State A**, and go to **State B** after the @ symbol
- Input 2: [pa"tch@snyk.io](mailto:pa)
  - **State A** for first to characters **P + A**
  - When we reach third character **"**, do we:
    - Match under "any character except @"?
    - Enter State C to allow any character within a pair of quotes?



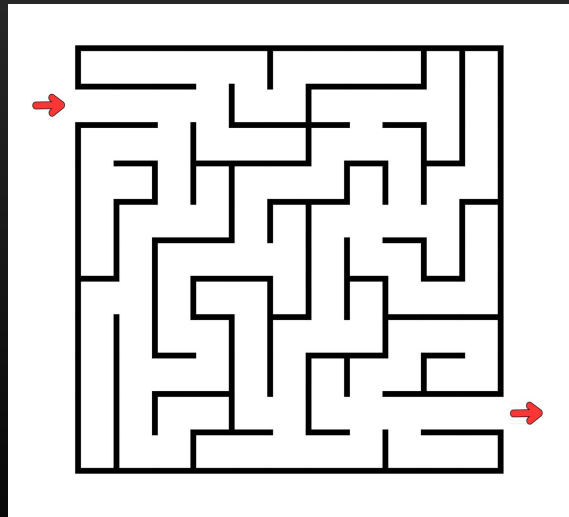


# \$ Backtracking

**Definition:** Backtracking is a trial-and-error-based problem-solving algorithm.

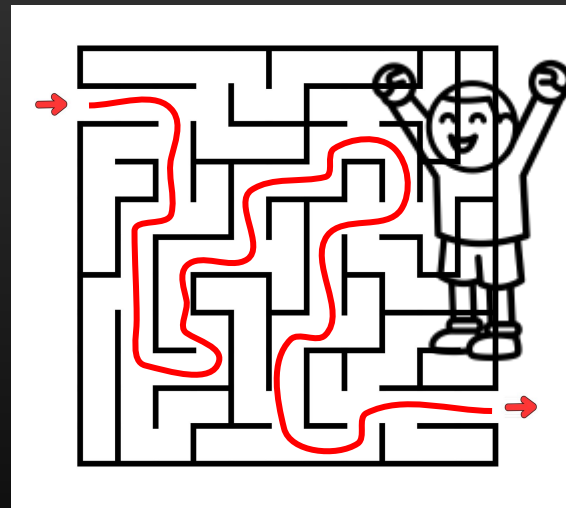
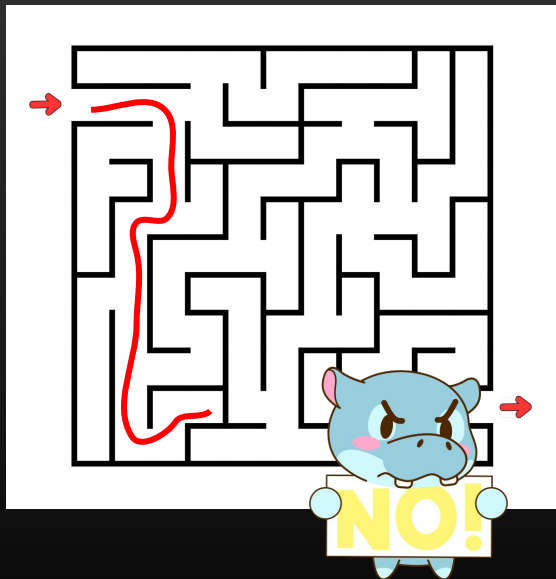
It's like trying to find your way through a maze. You go down one path, and if you hit a dead-end, you go back and try a different one.

- When you reach a junction, you make a choice.
- Not all choices are valid; must meet certain criteria.
- We backtrack when we can't reach the goal via the current path.





# \$ Backtracking





+



0) *, chrs=12
1) *, chrs=11
2) *, chrs=10
3) *, chrs=9
4) *, chrs=8
5) *, chrs=7
6) *, chrs=6
7) *, chrs=4
8) *, chrs=3
9) *, chrs=2
10) *, chrs=1
11) *, chrs=0

Chrs Matched: 13

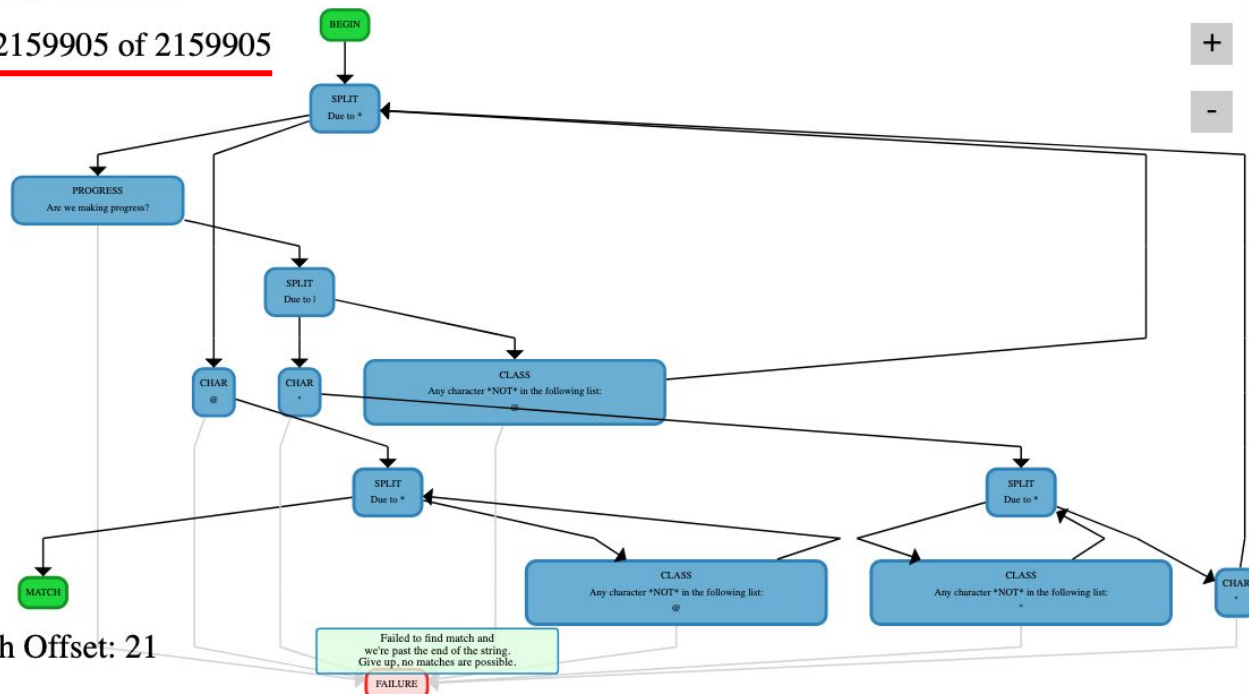
p a t c h @ s n y k . i o



( " [ ^ " ] \* " | [ ^ @ ] ) \* @ [ ^ @ ] \*



Step 2159905 of 2159905



Search Offset: 21

Chrs Matched: 0

" " " " " " " " " " " " " " " " " "



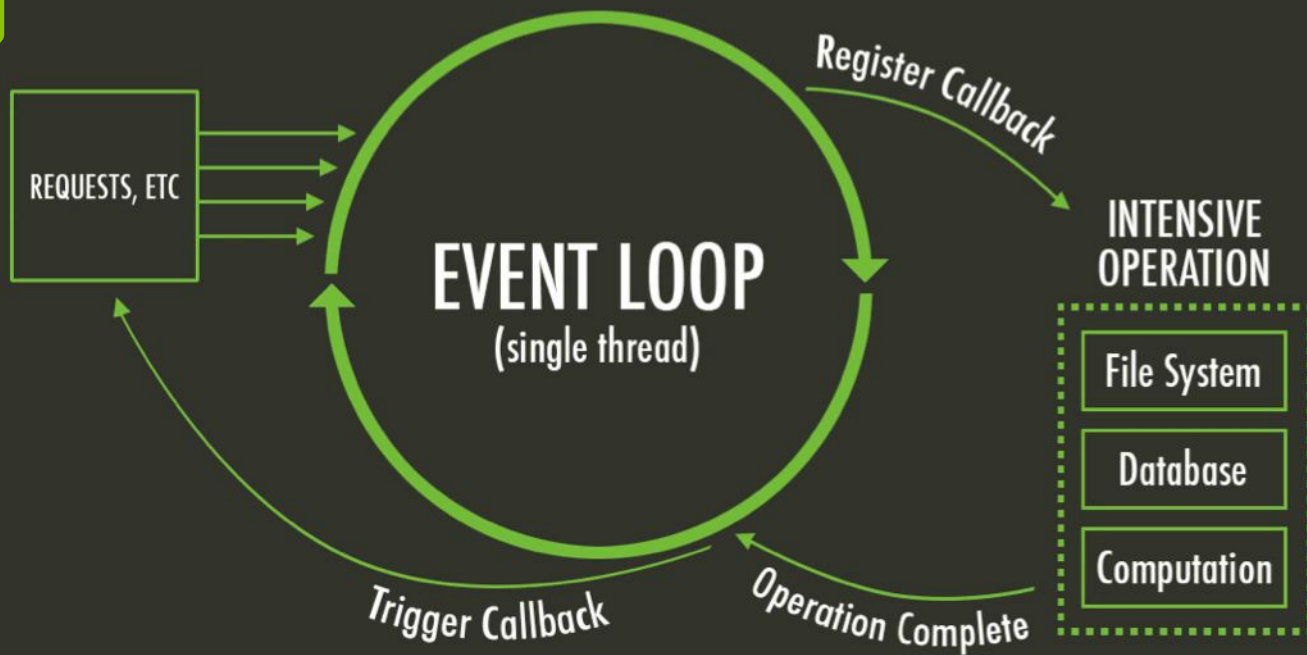
# \$ How does this cause a DoS?

- Shouldn't this only affect the single attacker request?

# \$ Platform threading



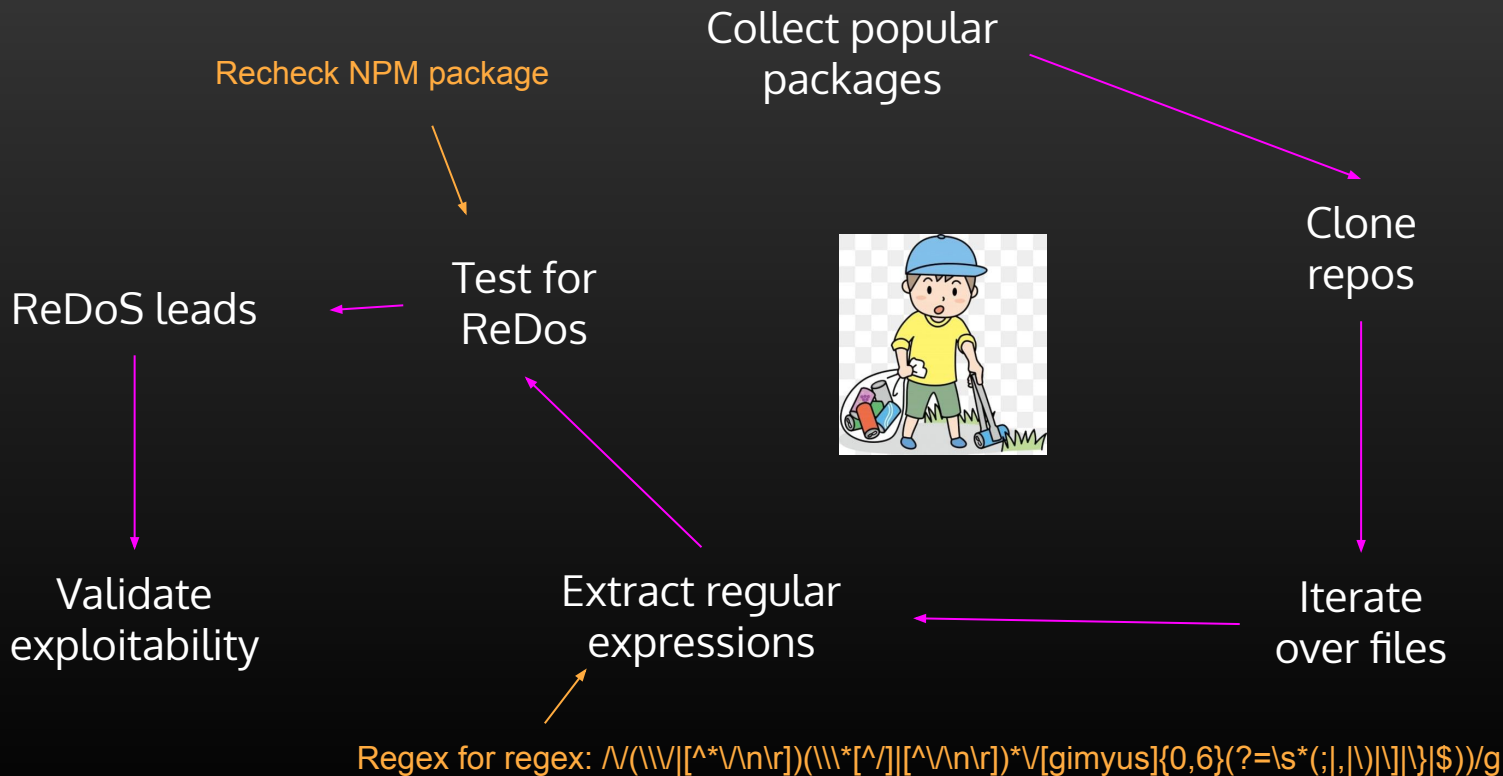
Language / Framework	Request Handling Model	Details
Java (Spring)	Thread-per-request	Uses a thread pool to manage incoming HTTP requests. Each request is handled by a separate thread from the pool.
PHP (mod_php)	Process-per-request	In an Apache setup with mod_php, each worker process handles a single PHP request.
PHP (PHP-FPM)	Process-per-request	Each PHP-FPM worker process is responsible for handling a single request. PHP-FPM manages a pool of these worker processes.
<b>Python (Django, Flask)</b>	<b>Process- or Thread-per-request</b>	<b>Django and Flask can be configured to use different types of server setups, such as WSGI servers like Gunicorn or uWSGI, which can use either worker processes or threads to handle requests.</b>
<b>Node.js</b>	<b>Event Loop</b>	<b>Node.js uses a single-threaded event loop to handle all requests. Asynchronous, non-blocking I/O allows it to efficiently manage multiple concurrent requests.</b>
Ruby (Rails)	Process- or Thread-per-request	Depending on the server (e.g., Puma, Unicorn), Rails can use either a process-per-request or thread-per-request model. Puma, for example, supports both.
Go	Goroutine-per-request	Each request is handled by a separate Goroutine, which is a lightweight thread managed by the Go runtime.







# Scanning NPM for ReDoS





## \$ results

From the initial 150 top NPM packages **1432** vulnerable regular expressions

Does NOT mean all are exploitable... may not be reachable and how its used makes a difference (test, match, exec, replace, etc)

Take sample of results and validate the modules API contains path from user input to vulnerable regex

# \$ results

Exploitable ReDos conditions across many package types for:

- Parsing headers of incoming requests
- Parsing file uploads
- Markdown parsers
- CSS parsers
- Other generic utility parsers

Used by 2.7m



+ 2,744,120

Used by 13.2m



+ 13,205,042

Used by 4.6m



+ 4,608,129



Package	Weekly downloads
semver	283152638
unfixed	13034595
ua-parser-js	10968947
unfixed	7864313
marked	7619555
css-tools	5775025
unfixed	741700
unfixed	724629
unfixed	451882
+ Many more	



# \$ cat mitigations.txt

Regular expressions can be tricky...

- Avoid using regular expressions where possible
- Use pre-vetted regular expressions
- Scan regular expressions for ReDos with Recheck + our script
- Avoid specific patterns
  1. `(a+)+` : Nesting quantifiers
  2. `(a|a)+` : Quantified overlapping disjunctions
  3. `\d+\d+` : Quantified Overlapping Adjacencies

<https://github.com/mowzk/redos-scan>



# \$ thanks for listening

1. Why are ReDos issues often overlooked? Is it complexity, awareness, or something else?
2. Should you care about non exploitable regexes, and how to deal with overwhelming number of issues?
3. Have you ever checked your code (or dependencies) for problematic regular expressions?
4. What responsibilities do open-source maintainers have to prevent ReDoS vulnerabilities? Should they be held to a higher standard than proprietary software?



\$ exit