

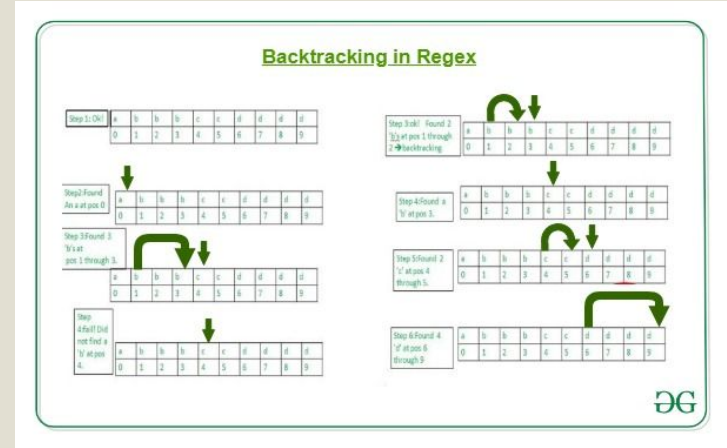


Improving Regular Expressions

Soujanya Namburi
Senior Security Research Engineer -
Harness io

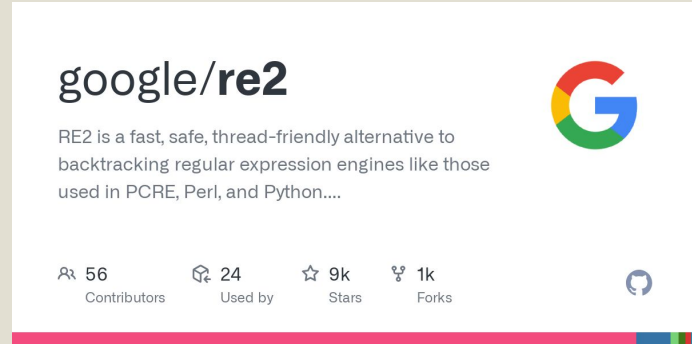
Regex Before

- PCRE, Perl, Java regex
- Backtracking
- Legacy WAFs, grep-like tools
- NFAs



Regexes now

- RE2, Hyperscan, Rust, RE-flex
- Subset (DFA-compatible only)
- WAFs, DPI, high-perf matching

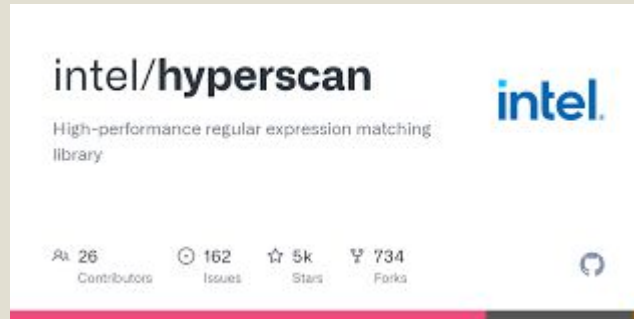


google/re2

RE2 is a fast, safe, thread-friendly alternative to backtracking regular expression engines like those used in PCRE, Perl, and Python....

56 Contributors 24 Used by 9k Stars 1k Forks

The screenshot shows the GitHub repository page for 'google/re2'. It features the Google logo, a description of RE2 as a fast, safe, thread-friendly alternative to backtracking regular expression engines, and statistics for contributors, users, stars, and forks. A GitHub logo is also visible in the bottom right corner.



intel/hyperscan

High-performance regular expression matching library

26 Contributors 162 Issues 5k Stars 734 Forks

The screenshot shows the GitHub repository page for 'intel/hyperscan'. It features the Intel logo, a description of the library as a high-performance regular expression matching library, and statistics for contributors, issues, stars, and forks. A GitHub logo is also visible in the bottom right corner.

How PCRE works

- Execution: Recursive backtracking
- To match, PCRE tries paths depth-first, trying alternatives sequentially
- If one path fails, it backtracks to try other branches
- Backreferences and advanced features make the automaton nondeterministic beyond classical NFAs

How RE2 works

- Regex → DFA/NFA without backtracking
- RE2 converts regex to an optimized Deterministic Finite Automaton (DFA) or sometimes a hybrid with NFA
- It expands all nondeterminism upfront by computing sets of NFA states (powerset construction)
- No backtracking is involved during matching

REDOS!

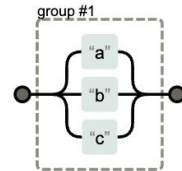
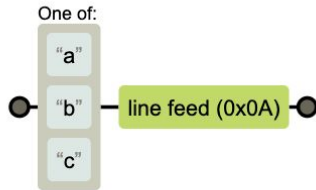
- Relies on backtracking regex engines (e.g., PCRE)
- Malicious input forces the engine to explore exponential possibilities
- WAF or backend becomes unresponsive or crashes
- Bad regex:
 - `".*only.*"`

REDOS!

- $^(a+)+\$$
 - [a][a][a][a]
 - [a][a][aa]
 - [a][aa][a]
 - [aa][a][a]
 - [a][aaa]
 - [aa][aa]
 - [aaa][a]
 - [aaaa]
 -

Better regex

- Reduce ambiguity - $(abc|a) \rightarrow a(bc)?$
- Avoid ambiguous branches, $(a|aa)^*$
- Try to use anchors if possible
- Avoid $(a|b|c) \rightarrow$ use $[abc]$
- Avoid Nested Quantifiers: $(a+)^+ \rightarrow aa^*b$



What IDFs did before

- popular IDSES like Snort and Suricata do a prefilter based regex
 - Specify a string pattern per regex
 - Find words
 - Run if found

Optimisations

- JIT:
- Use the prefiltering as part of the regex match
 - Compiles all your regular expressions into a single, unified automaton.
 - Converts regex patterns into hybrid automata combining:
 - Deterministic Finite Automata (DFA)
 - Nondeterministic Finite Automata (NFA)
 - Bit-parallel (SIMD) engines
 - /FA_n str_n FA_{n-1} str_{n-1} ... str_2 FA_1 str_1 FA_0/

Why can't we use it?

- Designed for hundreds or thousands of simultaneous patterns (e.g., IDS, WAF)
- Rewrite the architecture for it to work on 30 percent of the machines
- Needs CPU SIMD support and sufficient memory

The future

- Fully adaptable engine
- <https://github.com/RadhiFadlillah/go-regex-benchmark>
- <https://rust-leipzig.github.io/regex/2017/03/28/comparison-of-regex-engines/>

Thank you!