

# Fuzz Everything, Everywhere, All at Once

## Advanced QEMU-based fuzzing

Addison Crump <research@addisoncrump.info>  
Andrea Fioraldi <andrea@fioraldi@gmail.com>  
Dominik Maier <mail@dmnk.co>  
Donjia “toka” Zhang <toka@afplusplus.com>  
Marc “vanHauser” Heuse <marc@srlabs.de>



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

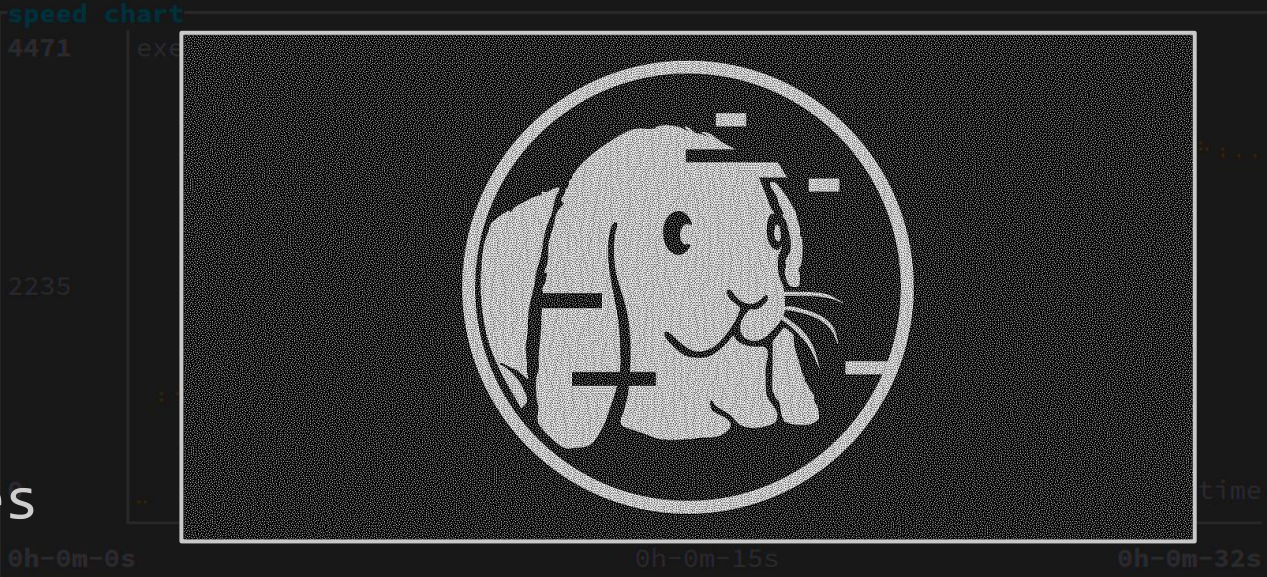
```
generic
run time      0h-0m-32s
clients      2
execut      104314
exec/sec     3271
```

# AFLplusplus Project

client #1 (l/r arrows to switch)

```
executions 104314
exec/sec   3271
corpus     5083
objectives 0
edges     8738/96215 (9%)
stability  96152/96215 (99%)
```

- Started with the AFL fork AFL++
- In 2019
- Added a ton of community features



clients logs ('t' to show/hide)

```
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.615k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```

LibAFL is rewritten from scratch in Rust 🦀🦀🦀

Today we will talk about Fuzzing, LibAFL, and QEMU(lation)



<https://t.me/learningnets>



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```
generic
run time      0h-0m-32s
clients      2
executions   11431
exec/sec     3271
```

speed chart  
4471 exec/sec

voluntary contributors,  
Full-time working/  
researching at:

# The AFLplusplus Project

```
client #1 (l/r arrows to switch)
executions 104314
exec/s     3.640k
corpus     5083
objectives 0
edges     8738/96215 (9%)
stability  96152/96215 (99%)

Marc "vanHauser" Heuse
Andrea Fioraldi
Dominik Maier
```



## Security Research Labs



```
clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```

Donjia "toka" Zhang

Addison Crump

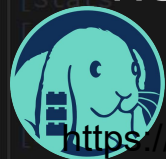
Shmarya Rubenstein

Heiko "hexcoder-" Eissfeldt

and a large community!



and more



<https://t.me/learningnets>

fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time      0h-0m-32s
clients      2
executions   104314
exec/sec     3271

```

# In This Talk

```

client #1 (l/r arrows to switch)
executions      104314
exec/sec        3271
corpus          5083
objectives      0
edges           8738/96215 (9%)
stability       96152/96215 (99%)

```

- Quick Fundamentals of
  - Fuzzing
  - QEMU
  - Binary Instrumentation



```

clients logs
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```

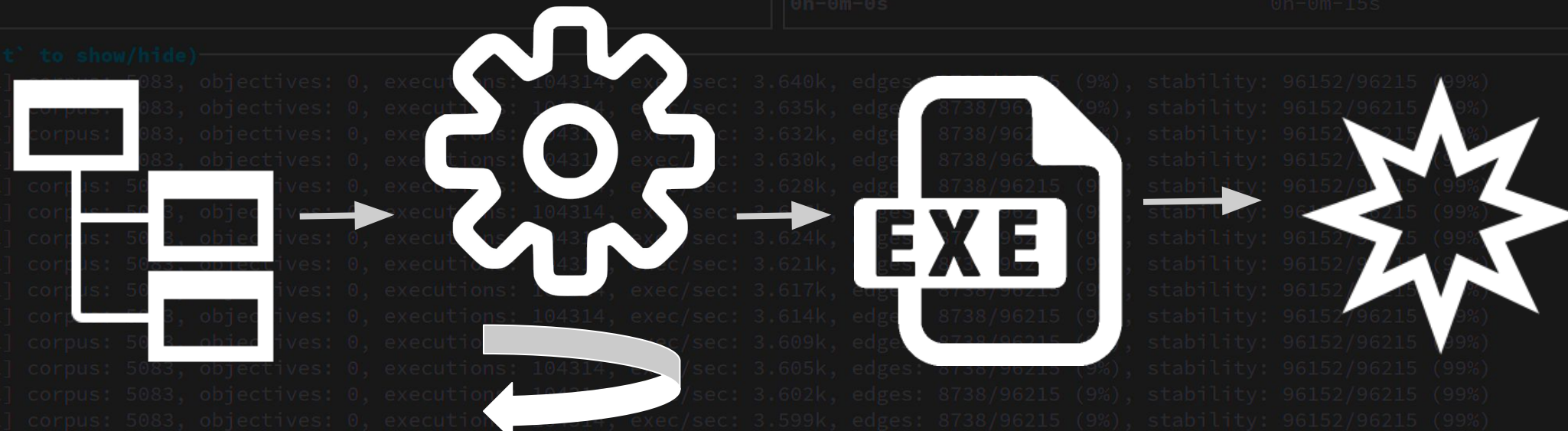


**Fuzz**

**Everything,  
Everywhere,  
All at Once**

# Fuzzing in a Nutshell

Fuzzing delivers a large amount of machine-generated inputs as quickly as possible to the target in order to find some objectives.





fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time      0h-0m-32s
client
executions    3271
exec/sec

```

# Coverage-Guided Fuzzing



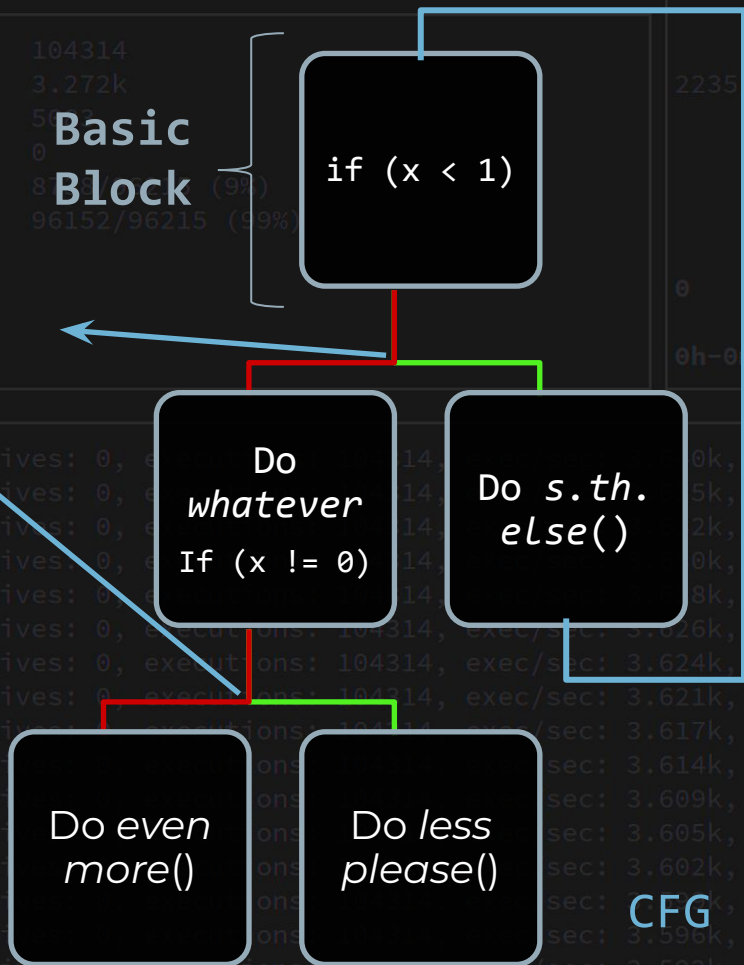
```

client #1 (l/r arrows to switch)
executions    104314
exec/sec      3.272k
corpus        5083
objectives    0
edges         8738
stability     96152/96215 (99%)

```

- Fuzzer takes reached coverage as feedback

Count how often these happen.



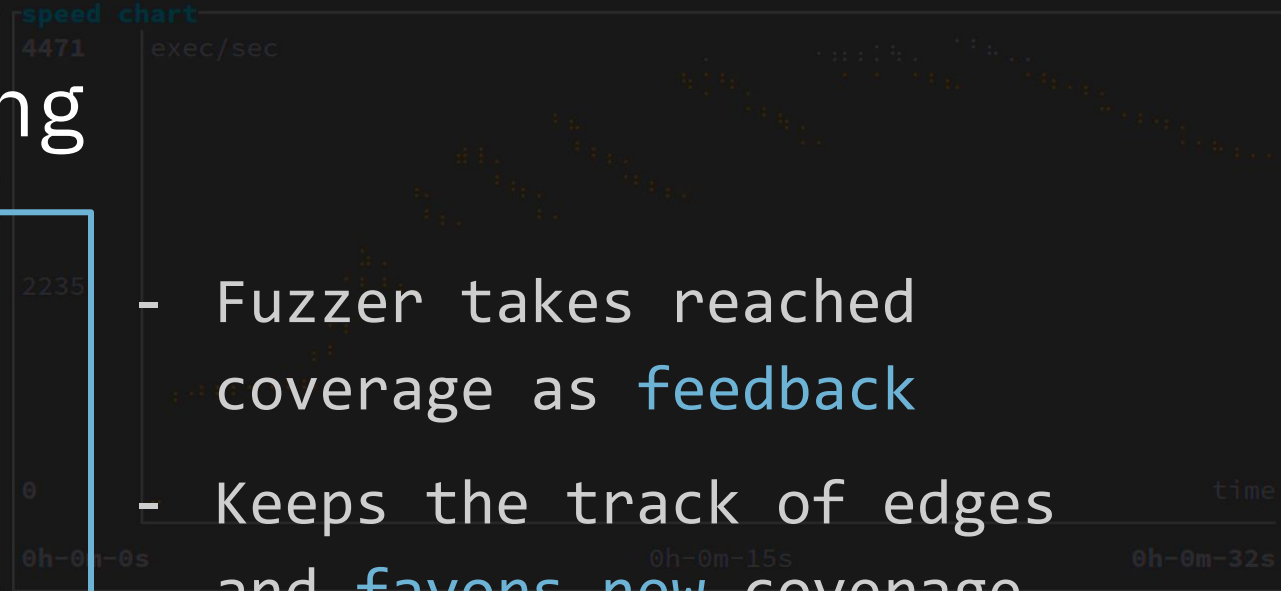
CFG



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

generic  
run time 0h-0m-32s  
client  
executions 3271  
exec/sec

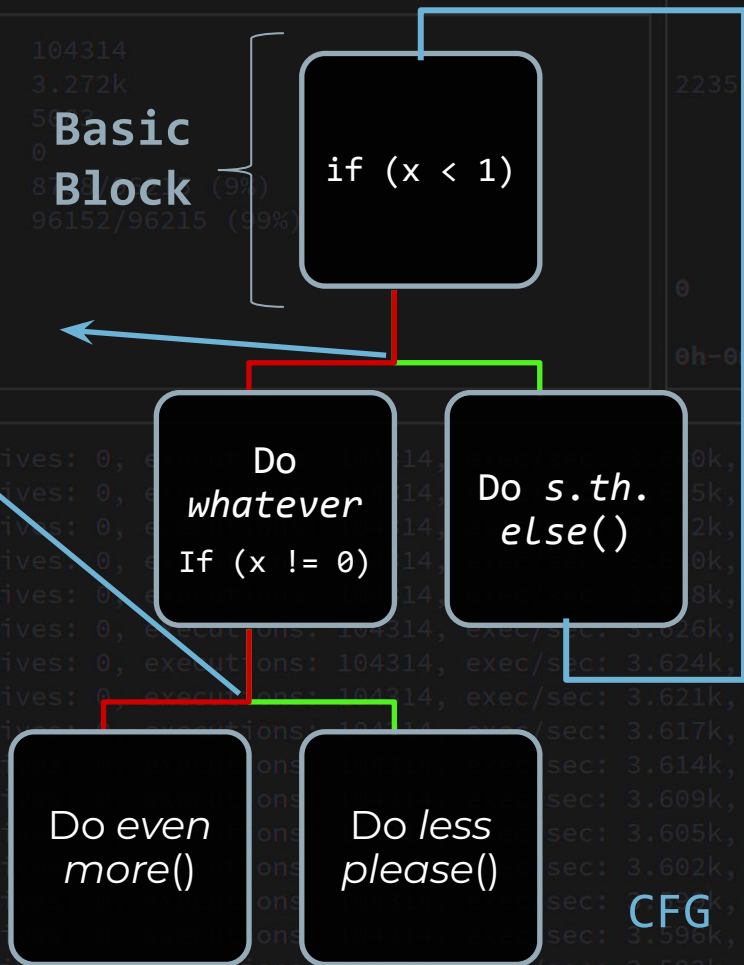


# Coverage-Guided Fuzzing

client #1 (l/r arrows to switch)  
executions 104314  
exec/sec 3.272k  
corpus 5083  
objectives 0  
edges 8738  
stability 96152/96215 (99%)

Count how often these happen.

- ↳ Feedback
- ↳ Favor Inputs leading to new edges



CFG

- Fuzzer takes reached coverage as feedback
- Keeps the track of edges and favors new coverage
- Orders of magnitude faster!



**Fuzz**  
**Everything,**  
**Everywhere,**  
**All at Once**

**Dynamic  
Binary  
Instrumentation**

```
fuzz_regex_match (default)
```

```
charts ('g' switch)
speed | corpus | objectives
```

```
generic
run time      0h-0m-32s
client       2
executions   104314
exec/sec     3171
```



# Meet QEMU, the Quick Emulator

```
client #1 (l/r arrows to switch)
executions   104314
exec/sec     3171
corpus       5083
objectives   0
edges       8738/96215 (9%)
stability    96152/96215 (99%)
```

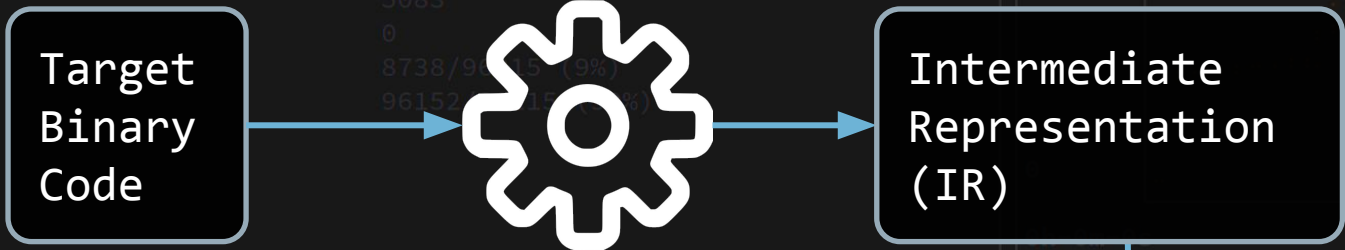
- Very popular full-system emulator
- CPU
- Memory
- Peripherals

```
clients-logs
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.611k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.608k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```

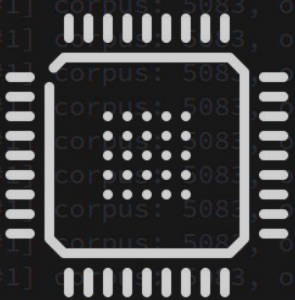
- Support for a variety of ISAs (x86, aarch64, ..., even hexagon)
- user-mode emulation support:
  - Emulation of userspace software
  - System call translation layer
  - Can be (ab-)used to change syscall behavior :)



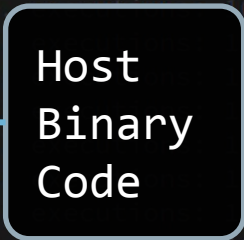
# JIT Code Rewriting



Disassemble & Lift



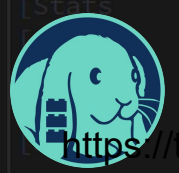
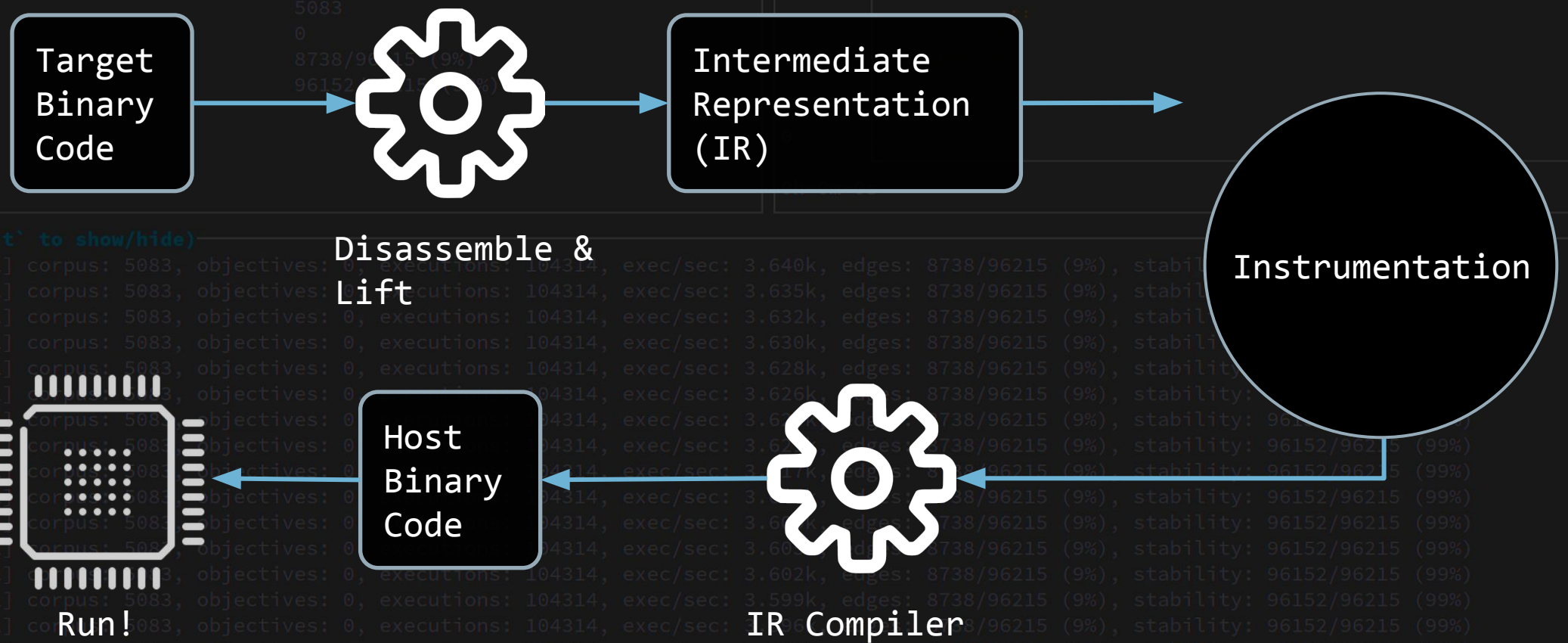
Run!



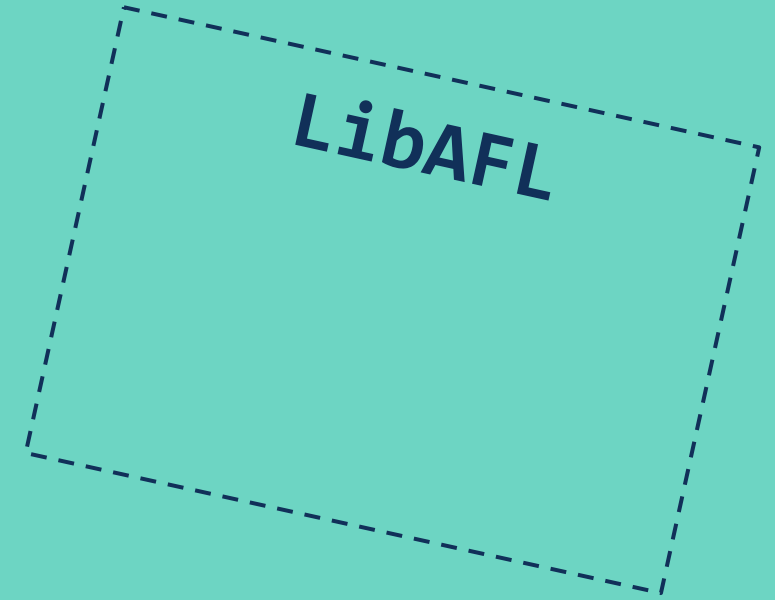
IR Compiler



# JIT Code Instrumentation

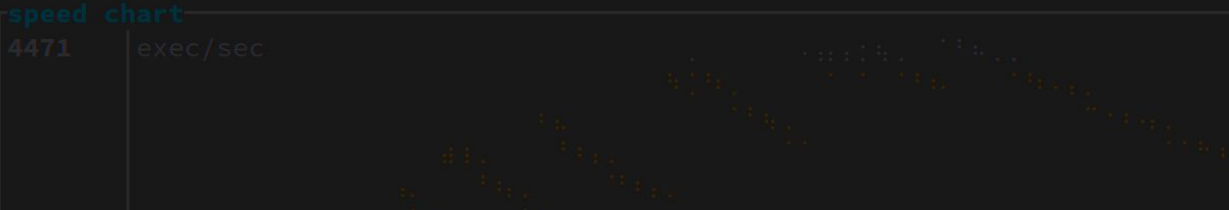


**Fuzz**  
**Everything,**  
Everywhere,  
All at Once



# AFL++ pew pew! bugs!

```
american fuzzy lop ++4.10a {default} (tools/thumbnail) [explore]
process timing
  run time : 0 days, 0 hrs, 5 min, 59 sec
  last new find : 0 days, 0 hrs, 0 min, 25 sec
  last saved crash : 0 days, 0 hrs, 1 min, 51 sec
  last saved hang : 0 days, 0 hrs, 4 min, 59 sec
cycle progress
  now processing : 1151*0 (75.5%)
  runs timed out : 0 (0.00%)
stage progress
  now trying : trim 32/32
  stage execs : 408/514 (79.38%)
  total execs : 1.38M
  exec speed : 37.65/sec (slow!)
fuzzing strategy yields
  bit flips : disabled (default, enable with -D)
  byte flips : disabled (default, enable with -D)
  arithmetics : disabled (default, enable with -D)
  known ints : disabled (default, enable with -D)
  dictionary : n/a
  havoc/splice : 1210/813k, 212/425k
  py/custom/rq : unused, unused, unused, unused
  trim/eff : 9.85%/130k, disabled
strategy: explore state: in progress
overall results
  cycles done : 0
  corpus count : 1525
  saved crashes : 4
  saved hangs : 1
map coverage
  map density : 6.37% / 25.35%
  count coverage : 2.70 bits/tuple
findings in depth
  favored items : 387 (25.38%)
  new edges on : 560 (36.72%)
  total crashes : 11 (4 saved)
  total tmouts : 509 (0 saved)
item geometry
  levels : 15
  pending : 939
  pend fav : 3
  own finds : 1418
  imported : 0
  stability : 100.00%
[cpu000: 6%]
```



AFLplusplus / AFLplusplus Public

Search or jump to... Pulls Issues Marketplace Explore

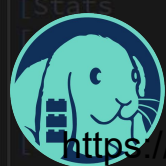
Code Issues 27 Pull requests 1 Discussions Actions Security

stable Go to file Add file Code About

vanhauser-thc Merge pull request #1084 from AFLplusplus... on Sep 1 4,599

- .github Change afl to AFL in \*.md (#1057) 3 months ago
- custom\_mutators Change afl to AFL in \*.md (#1057) 3 months ago
- dictionaries remove docs/README symlink and update ref... 11 months ago
- docs fix regression in class lookup 2 months ago
- frida\_mode Added seccomp support 2 months ago
- include Add unstable coverage support 2 months ago
- instrumentation announce llvm 13 support 2 months ago
- qemu\_mode Fixed spelling of quarantine 2 months ago
- src fix regression in class lookup 2 months ago
- test fix regression in class lookup 2 months ago
- testcases adjust a bit readmes 2 years ago

The screenshot shows the GitHub repository page for AFLplusplus. At the top, there's a search bar and navigation links for Pulls, Issues, Marketplace, and Explore. The repository name 'AFLplusplus / AFLplusplus' is prominently displayed, along with statistics like '60' stars, '2.1k' forks, and '416' issues. Below this, there are tabs for 'Code', 'Issues 27', 'Pull requests 1', 'Discussions', 'Actions', and 'Security'. A dropdown menu shows 'stable' as the selected branch. There are buttons for 'Go to file', 'Add file', and 'Code'. The 'About' section is visible on the right, describing the fuzzer. A list of recent pull requests is shown, with the most recent one by 'vanhauser-thc' merged on Sep 1 with 4,599 commits. The list includes folders like .github, custom\_mutators, dictionaries, docs, frida\_mode, include, instrumentation, qemu\_mode, src, test, and testcases, each with a brief description of the change and the time since it was made.

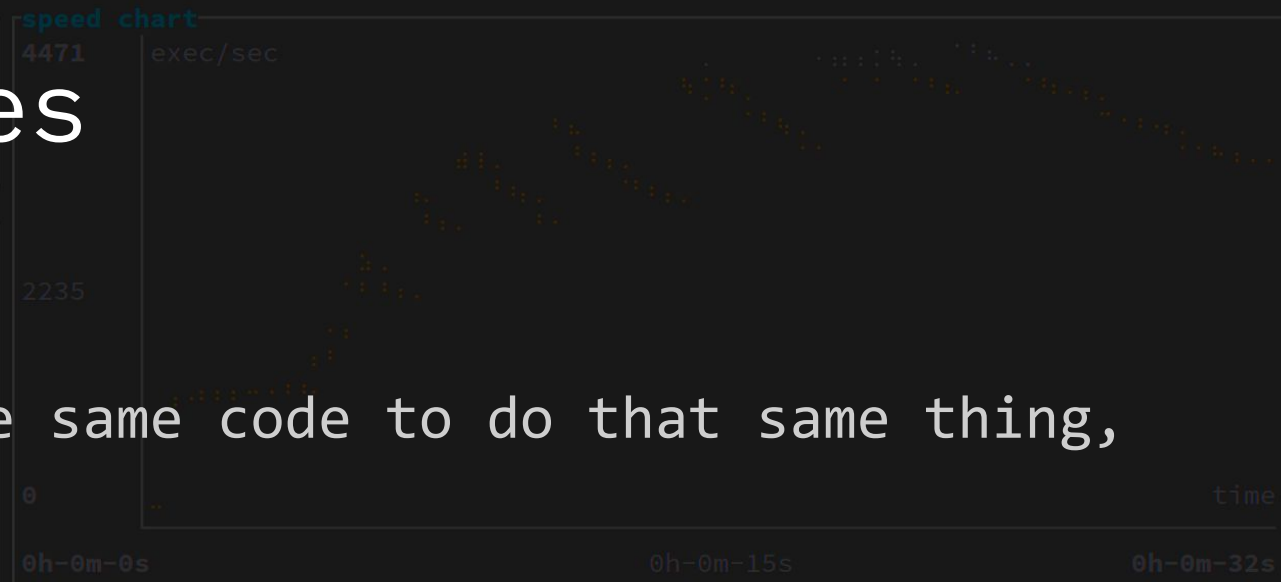


fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

# Custom Fuzzers Issues

```
generic
run time          0h-0m-32s
client #1 (l/r arrows to switch)
executions        104314
exec/sec          3.272k
corpus            5083
objectives        0
edges            8738/96215 (9%)
stability         96152/96215 (99%)
```



- Reinventing the wheel: code the same code to do that same thing, again and again

- Naive design: typically just a mutator

- Scaling: scaling to multi-core or -machine is hard

```
clients logs
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

generic	
run time	0h-0m-32s
clients	2
executions	104314
exec/sec	3271



# LibAFL

client #1 (l/r arrows to switch)	
executions	104314
exec/sec	3.272k
corpus	5083
objectives	0
edges	8738/96215 (9%)
stability	96152/96215 (99%)

- State-of-the-Art
- Portable

clients logs ('*' to show/hide)	
[Stats #0] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.622k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.620k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.618k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.616k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.612k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.610k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.608k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.606k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.604k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	

- Extensible
- Scalable
- **Performant**, thanks to compile-time abstractions in Rust

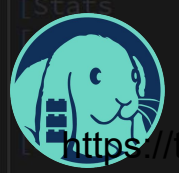


# High level design

- LibAFL **Core**, the main library
- LibAFL **Targets**, the runtime code that lives in the target
- LibAFL **CC**, the library to write compiler wrappers

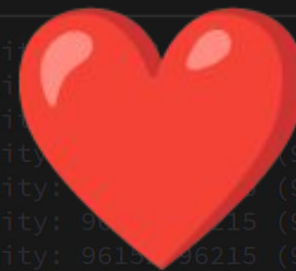
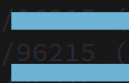


Many instrumentation options



# LibAFL QEMU

Support for user- *and* system-mode (WIP), based on QEMU 8



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```
generic
run time          0h-0m-32s
client #1
executions       104314
exec/sec         3271
```



# Why Emulate?

```
client #1 (l/r arrows to switch)
executions       104314
exec/sec         3271
corpus           5083
objectives       0
edges            8738/96215 (9%)
stability        96152/96215 (99%)
```

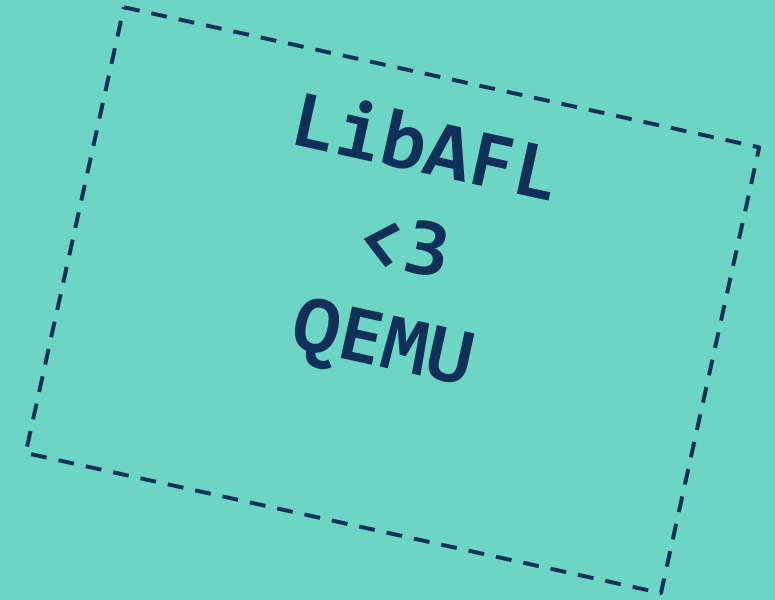
- Why not Compile-Time Instrumentation?
  - Compiling is hard
  - Toolchains are hard
  - Source not always available

```
clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.629k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.611k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.608k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```

- Change instrumentation at runtime
- Advantages over other dynamic binary instrumentation:
  - Cross architecture
  - Reasonably fast while being stable

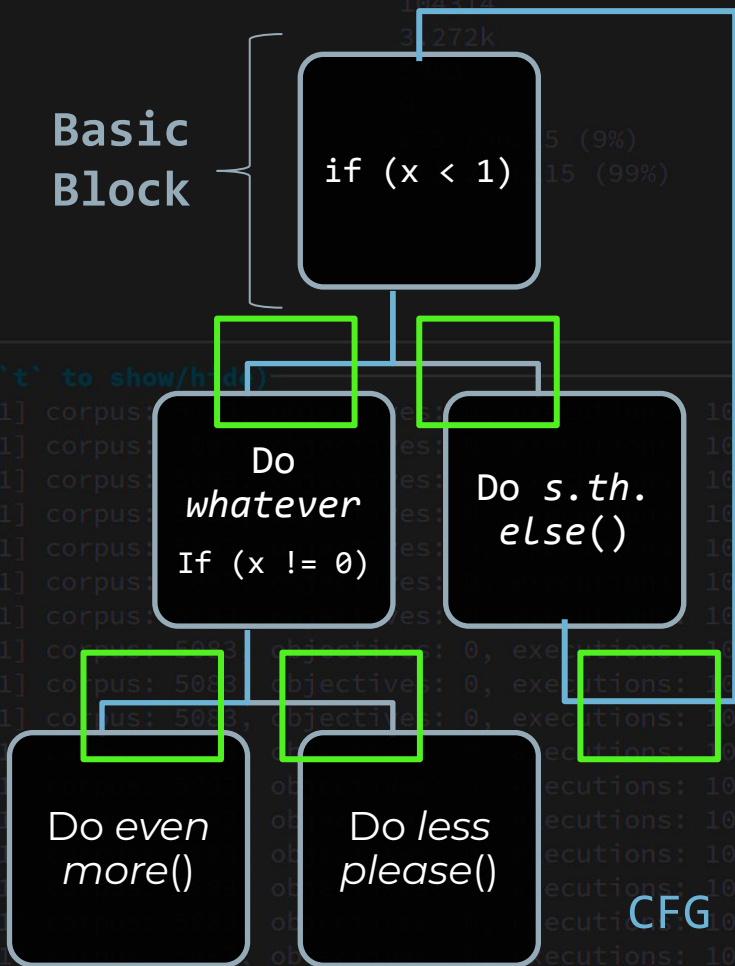


# Fuzz Everything, Everywhere, All at Once



# LibAFL QEMU Hooks

Basic Block



CFG

Example: Edge Hooks

Instrumentation (in JIT) that is running callback functions

Before any jump, reports a unique id for the taken edge to a hook

Generation Hook:  
`fn(&mut Self, Option<&mut S>, src: GuestAddr, dest: GuestAddr) -> Option<u64> { ... }`

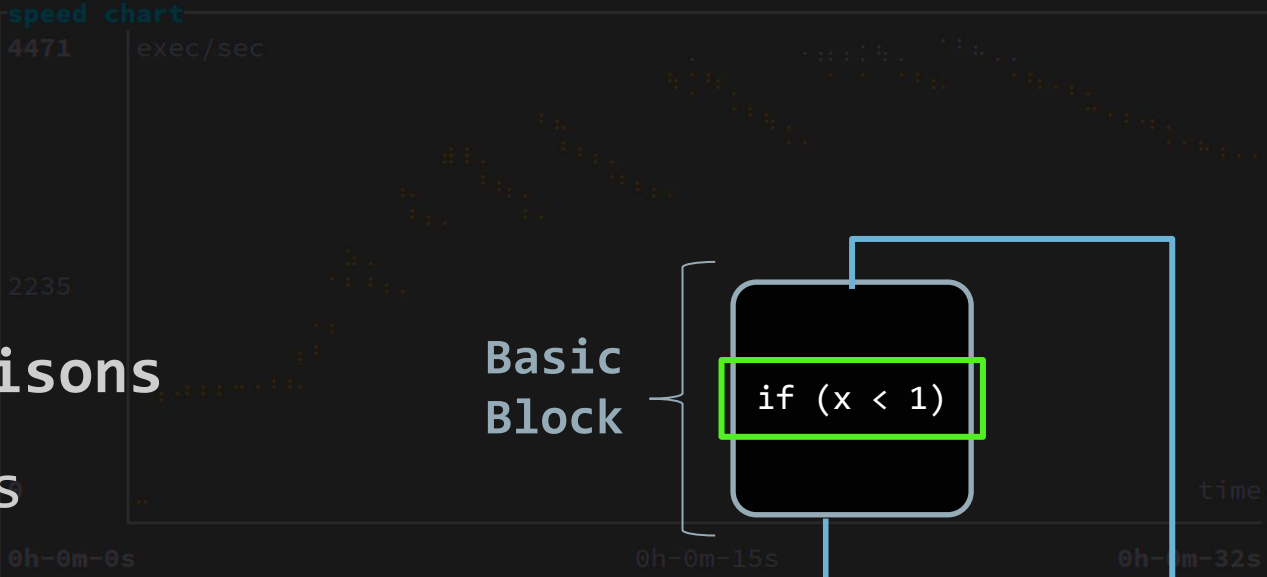
Execution Hook:  
`FnMut(&'a mut Self, Option<&'a mut S>, u64)`



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

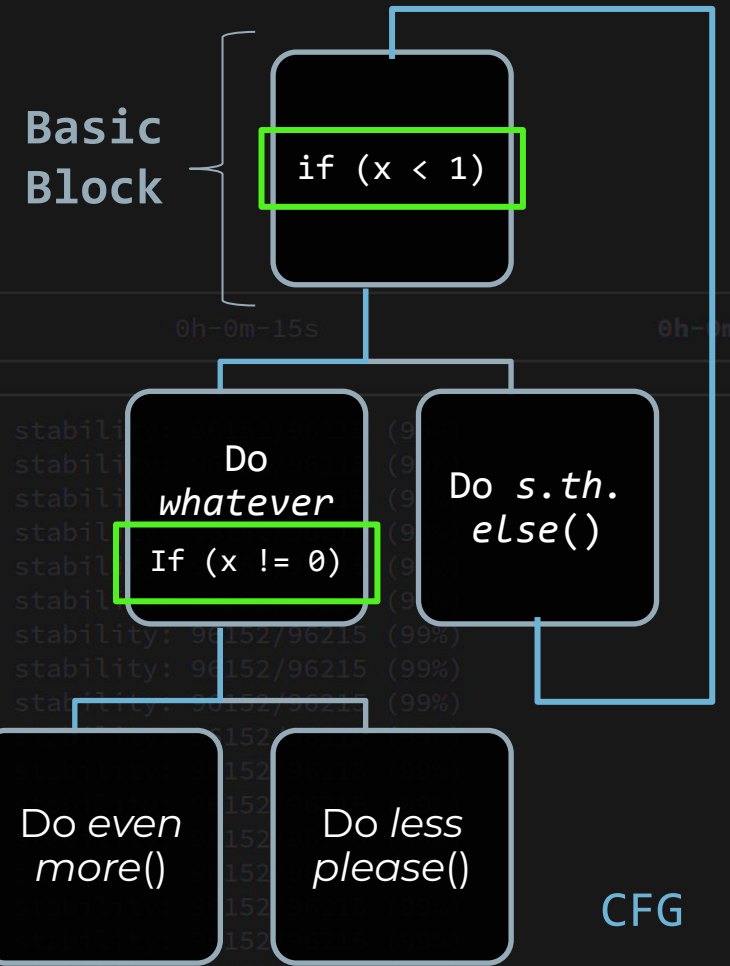
generic	
run time	0h-0m-32s
clients	2
executions	104314
exec/sec	3228



client #1 (l/r arrows to switch)	
executions	104314
exec/sec	3.272k
corpus	5083
objectives	
edges	8738/96215 (9%)
stability	96152/96215 (99%)

# LibAFL QEMU Hooks

- Instructions
- Blocks
- Edges
- Read and write
- Comparisons
- Threads
- Syscalls
- Crashes

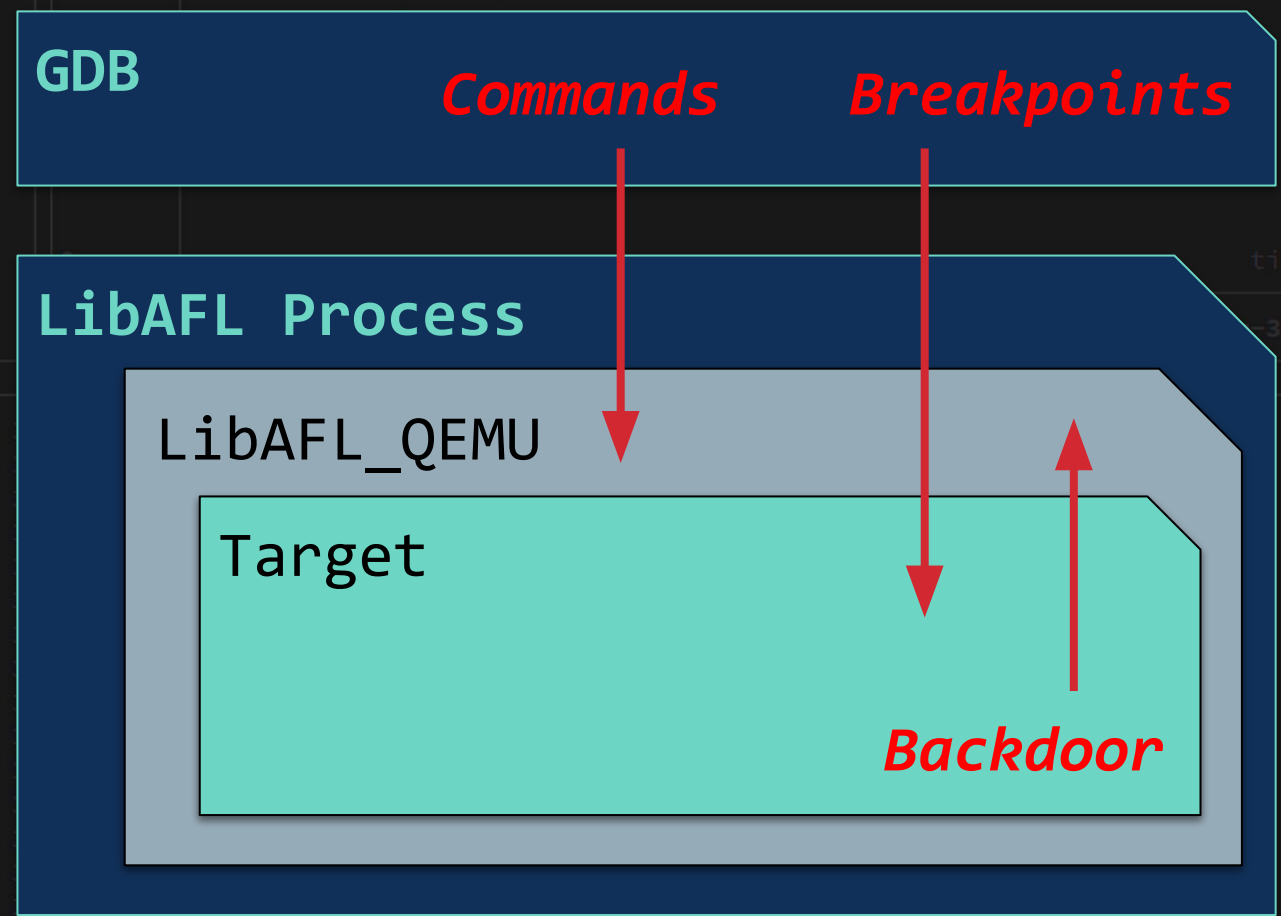


clients logs ('t' to show/hide)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	



# Fuzzing With QEMU: Execution Control

- Backdoor: *target-defined* point at which execution halts
- Breakpoint: *fuzzer-defined* point at which execution halts
- Commands: *custom GDB commands* to interact with the target state



# **Fuzz** **Everything,** **Everywhere,** **All at Once**

**Sanitized  
Android  
Snapshot  
Fuzzing**



# Target Library

## Project Zero

News and updates from the Project Zero team at Google

...n to the Samsung Qmage Codec and

...ring my journey from discovering a vulnerable little-known zero-click MMS attack that worked on the latest Samsung they are completed and will be linked here when complete.

[...e Qmage Codec](#)

[...ry Corruption Primitives](#)

[...ng the ASLR Oracle](#)

[...R, Getting RCE](#)

ashes in a custom Samsung codec called "Qmage", present (version 4.4.4+). This codec is written in C/C++ code, and is

baked deeply into the [Skia](#) graphics library, which is in turn the underlying engine used for nearly all graphics operations in the Android OS. In other words, in addition to the well-known formats such as JPEG and PNG, modern Samsung phones also natively support a proprietary Qmage format, typically denoted by the .qmg file extension. It is automatically enabled for all apps which display images, making it a prime target for remote attacks, as sending pictures is the core functionality of some of the most popular mobile apps.



# Blowing the Cover of Android Binary Fuzzing

Flanker

Senior Researcher, Pangu Team

RWCTF Tech Forum, 2021



<https://t.me/learningnets>

# Reversing

fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

generic  
run time 0h-0m-32s  
clients 2  
executio 104314  
exec/sec 471



client #1 (l/r arrows to switch)  
executions  
exec/sec  
corpus  
objectives  
edges  
stability

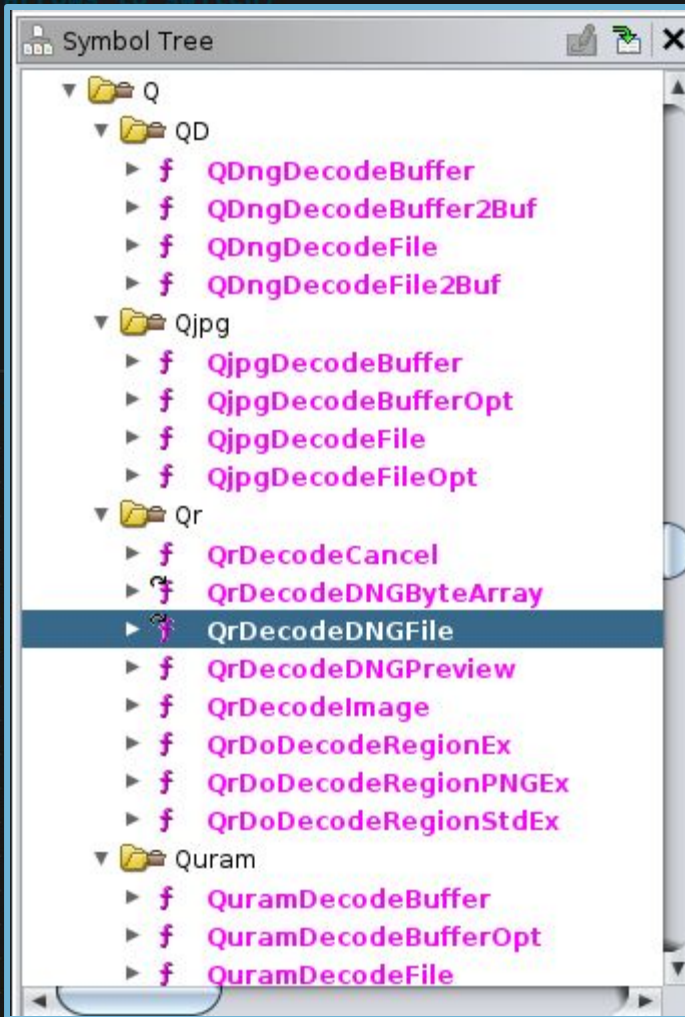
A 'Symbol Tree' window showing a directory structure of functions. The tree is organized into folders: Q, QD, Qjpg, Qr, and Quram. The 'Qr' folder is expanded, and 'QrDecodeDNGFile' is selected. The functions listed are:

- QDngDecodeBuffer
- QDngDecodeBuffer2Buf
- QDngDecodeFile
- QDngDecodeFile2Buf
- QjpgDecodeBuffer
- QjpgDecodeBufferOpt
- QjpgDecodeFile
- QjpgDecodeFileOpt
- QrDecodeCancel
- QrDecodeDNGByteArray
- QrDecodeDNGFile
- QrDecodeDNGPreview
- QrDecodeImage
- QrDoDecodeRegionEx
- QrDoDecodeRegionPNGEx
- QrDoDecodeRegionStdEx
- QuramDecodeBuffer
- QuramDecodeBufferOpt
- QuramDecodeFile

clients logs ('  
[Stats #1 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

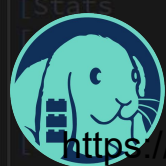


# Reversing



```
Decompile: Java_com_sec_samsung_gallery_decoder_QuramCodecInterface_nativeDeco...
67  if ((iVar5 == 5) || (iVar5 == 3)) {
68  iVar6 = (**(code **)(*param_1 + 800))(param_1,param_4,uVar13);
69  cVar3 = (**(code **)(*param_1 + 0x300))(param_1,param_4,uVar11);
70  uVar4 = (**(code **)(*param_1 + 0x300))(param_1,param_4,uVar12);
71  lVar18 = (**(code **)(*param_1 + 0x2f8))(param_1,param_4,uVar16);
72  uVar10 = (**(code **)(*param_1 + 0x548))(param_1,param_3,0);
73  iVar7 = android_sdk_version();
74  local_lb0 = 0;
75  iStack_lac = 0;
76  local_lb8 = 0;
77  local_lb4 = 0;
78  iVar8 = QuramGetImageInfoFromFile2(uVar10,0,0,&iStack_lac,&local_lb0,&local_lb4,&local_lb8);
79  if (iVar8 == 3) {
80  parseQPNG_icc(uVar10,0,&local_lb4);
81  }
82  if ((cVar3 == '\0') && (0x1b < iVar7 && local_lb4 != 0)) {
83  if (local_lb8 != 0) {
84  uVar2 = local_lb0 * iStack_lac;
85  uVar1 = uVar2 + 0x3f;
86  if (-1 < (int)uVar2) {
87  uVar1 = uVar2;
88  }
89  lVar19 = availableMemory();
90  if (lVar19 < (long)((ulong)uVar1 << 0x20) >> 0x26) goto LAB_0019cfa0;
91  }
92  (**(code **)(*param_1 + 0x2f0))(param_1,uVar9,"outColorSpace","Landroid/graphics/ColorSpace;")
93  ;
94  uVar11 = (**(code **)(*param_1 + 0x30))(param_1,"android/graphics/BitmapFactory");
95  uVar12 = (**(code **)(*param_1 + 0x388))
96  (param_1,uVar11,"decodeFile",
97  "(Ljava/lang/String;Landroid/graphics/BitmapFactory$Options;)Landroid/grap
98  hics/Bitmap;")
99  );
100  lVar19 = (**(code **)(*param_1 + 0x390))(param_1,uVar11,uVar12,param_3,param_4);
101  if ((lVar18 == 0) || (lVar19 != 0)) goto LAB_0019d834;
102  }
103  LAB_0019cfa0:
104  __s = malloc(0x48);
105  if (__s == (void *)0x0) {
106  lVar19 = 0;

```



fuzz\_regex\_match (default)

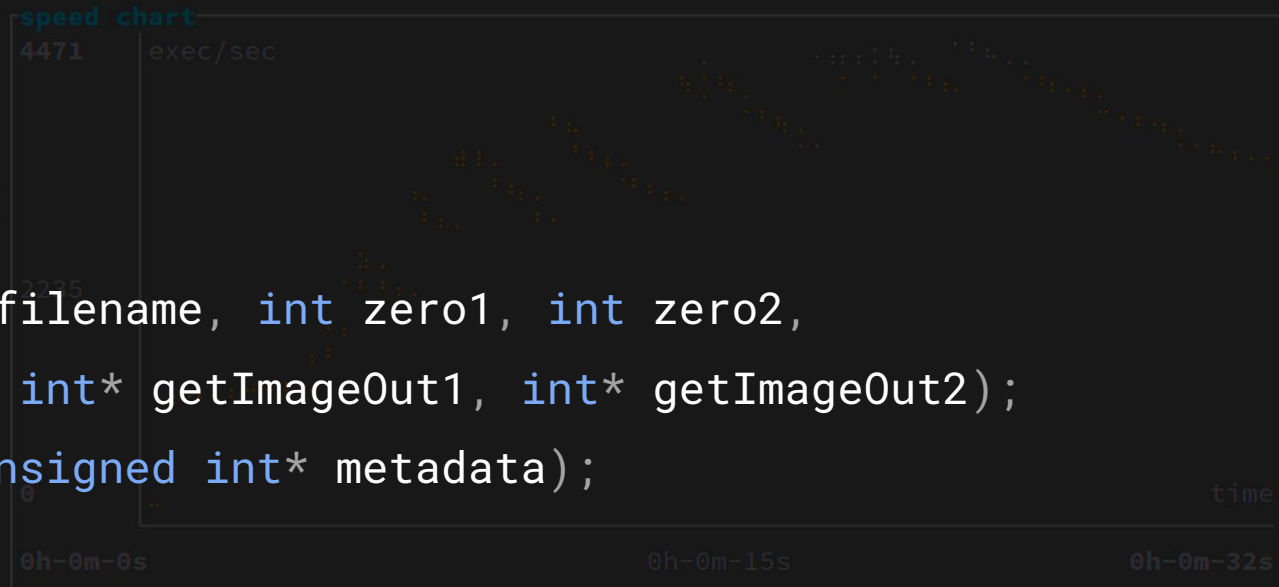
charts ('g' switch)  
speed | corpus | objectives

generic	
run time	0h-0m-32s
clients	2
executions	104314
exec/sec	3271

# Harness

client #1 (l/r arrows to switch)	
executions	104314
exec/sec	3.272k
corpus	5083
objectives	0
edges	8738/96215 (9%)
stability	96152/96215 (99%)

```
int QuramGetImageInfoFromFile2(char *filename, int zero1, int zero2,
int *w, int *h, int* getImageOut1, int* getImageOut2);
int QrParseMetadata(char *filename, unsigned int* metadata);
```

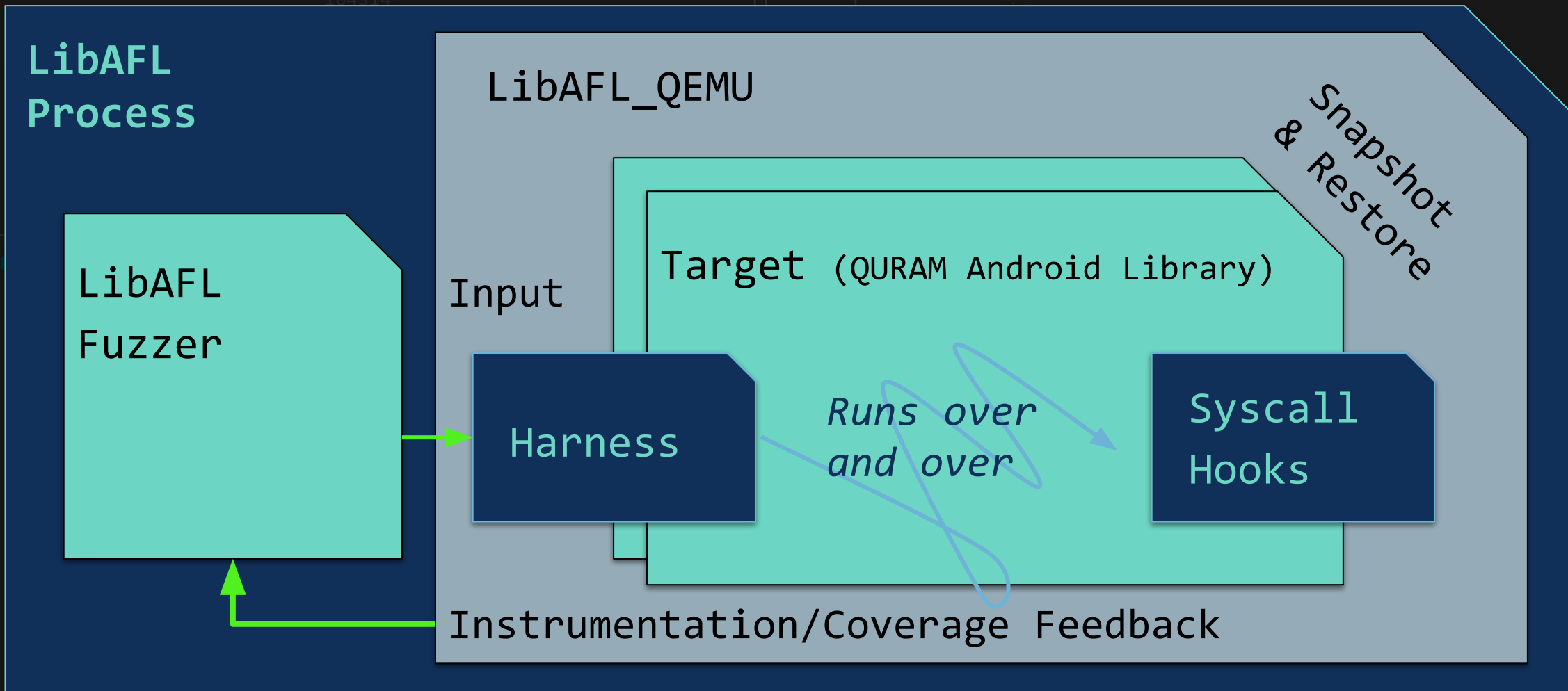


```
clients logs ('t' switch)
void harnessSimple(char* filename) {
int w, h, a, b;
unsigned int metadata[71] = {0};
if (QuramGetImageInfoFromFile2(filename, 0, 0, &w, &h, &a, &b) == 0) {
QrParseMetadata(filename, metadata);
}
}
```

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)



# Fuzzing Android Libs on a Host



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```
generic
run time      0h-0m-32s
clients      2
execution    104314
exec/sec     3272k
```

# A Simple Fuzzer

```
client #1 (l/r arrows to switch)
executions    104314
exec/sec      3.272k
corpus        5083
objectives    0
edges         8738/96215 (9%)
stability     96152/96215 (99%)

let mut args = vec!["qemu".into(), "./harness".into(), MAGIC_FILENAME.into()];
let mut env: Vec<String, String> = env::vars().collect();

let emu = Emulator::new(&mut args, &mut env);

let mut elf_buffer = Vec::new();
let elf = EasyElf::from_file(emu.binary_path(), &mut elf_buffer).unwrap();

let harness_ptr = elf.resolve_symbol(HARNESS_NAME, emu.load_addr())
    .expect(&format!("Symbol {} not found", HARNESS_NAME));
println!("{}", "@{:#x}", HARNESS_NAME, harness_ptr);

emu.set_breakpoint(harness_ptr);
unsafe { emu.run() };

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.622k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.620k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.618k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.616k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.612k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.610k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.608k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.606k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.604k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```



```
clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.622k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.620k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.618k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.616k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.612k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.610k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.608k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.606k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.604k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

generic

run time 0h-0m-32s  
clients 2  
execution 104314  
exec/sec 3272k

speed chart



# A Simple Fuzzer

client #1 (l/r arrows to switch)

executions 104314  
exec/sec 3.272k  
corpus 5083  
objectives 0  
edges 8738  
stability 96152/96215 (99%)

```
let mut args = vec!["qemu".into(), "./harness".into(), MAGIC_FILENAME.into()];  
let mut env: Vec<String, String> = env::vars().collect();  
  
let emu = Emulator::new(&mut args, &mut env);
```

```
let mut elf_buffer = Vec::new();  
let elf = EasyElf::from_file(emu.binary_path(), &mut elf_buffer).unwrap();
```

```
let harness_ptr = elf  
    .resolve_symbol(HARNESS_NAME, emu.load_addr())  
    .expect(&format!("Symbol {} not found", HARNESS_NAME));  
println!("{}", "@{:#x}", HARNESS_NAME, harness_ptr);
```

```
emu.set_breakpoint(harness_ptr);  
unsafe { emu.run() };
```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients           2
execution         104314
exec/sec          3272

```

# A Simple Fuzzer

```

client #1 (l/r arrows to switch)
executions        104314
exec/sec          3.272k
corpus            5083
objectives        0
edges             8738/96215 (9%)
stability         96152/96215 (99%)

let ret_addr: u64 = emu.read_reg(Regs::Lr).unwrap();
println!("Return address = {:#x}", ret_addr);

```



```

clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.622k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.611k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients          2
execution        104314
exec/sec         3272

```

# A Simple Fuzzer



```

client #1 (l/r arrows to switch)
executions      104314
exec/sec        3.272k
corpus          5083
objectives      0
edges           8738/96215 (9%)
stability       96152/96215 (99%)

```

```

let ret_addr: u64 = emu.read_reg(Regs::Lr).unwrap();
println!("Return address = {:#x}", ret_addr);

```

```

clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.622k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```

```

let saved_cpu_states: Vec<_> = (0..emu.num_cpus())
    .map(|i| emu.cpu_from_index(i).save_state())
    .collect();

```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients           2
execution         104314
exec/sec          3267

```

# A Simple Fuzzer

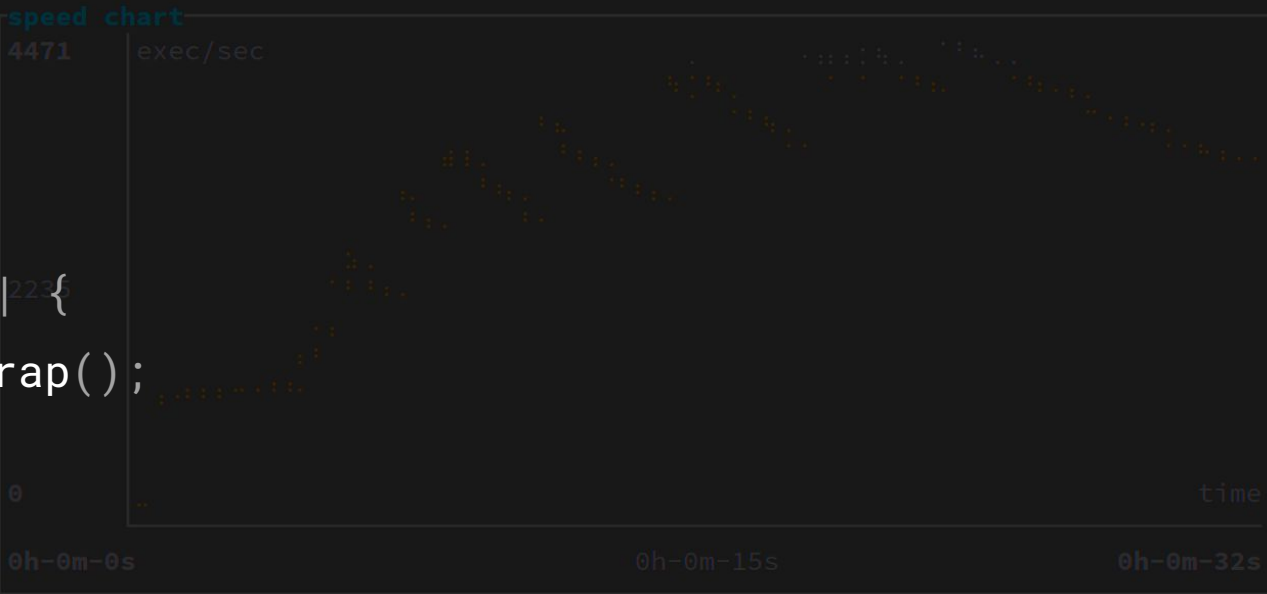
```

client #1 (l/r arrows to switch)
executions        104314
exec/sec          3267
corpus            5083
objectives        0
edges             8738/96215 (9%)
stability         96152/96215 (99%)

let mut harness = |input: &BytesInput| {
    input.to_file(MAGIC_FILENAME).unwrap();

    unsafe { let _ = emu.run() };
}

```



```

clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.638k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```

```

for (i, s) in saved_cpu_states.iter().enumerate() {
    emu.cpu_from_index(i).restore_state(s);
}
ExitKind::Ok
};

```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```
generic
run time      0h-0m-32s
clients      2
execution    104314
exec/sec     3264k
```

# A Simple Fuzzer

```
client #1 (l/r arrows to switch)
executions    104314
exec/sec      3264k
corpus        5083
objectives    0
edges         8738/96215 (9%)
stability     96152/96215 (99%)

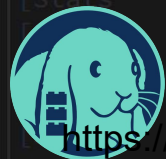
let mut harness = |input: &BytesInput| {
    input.to_file(MAGIC_FILENAME).unwrap();

    unsafe { let _ = emu.run() };
}
```



```
clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

ExitKind::Ok
```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients           2
execution         104314
exec/sec          3267

```

# A Simple Fuzzer

```

client #1 (l/r arrows to switch)
executions        104314
exec/sec          3267
corpus            5083
objectives        0
edges             8738/96215 (9%)
stability         96152/96215 (99%)

let mut hooks = QemuHooks::new(
    emu.clone(),
    tuple_list!(
        QemuEdgeCoverageHelper::default(),
        QemuCmpLogHelper::default(),
    ),
);

```



```

clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

let executor = QemuExecutor::new(
    &mut hooks,
    &mut harness,
    tuple_list!(edges_observer, time_observer),
    &mut fuzzer,
    &mut state,
    &mut mgr,
).expect("Failed to create QemuExecutor");

```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients          2
execution        104314
exec/sec         3277

```

# A Simple Fuzzer

```

client #1 (l/r arrows to switch)
executions        104314
exec/sec          3277
corpus            5083
objectives        0
edges            8738/96215 (9%)
stability         96152/96215 (99%)

let mut hooks = QemuHooks::new(
    emu.clone(),
    tuple_list!(
        QemuEdgeCoverageHelper::default(),
        QemuCmpLogHelper::default(),
    ),
);

```

```

QemuEdgeCoverageHelper::default(),
QemuCmpLogHelper::default(),

```



```

clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

let executor = QemuExecutor::new(
    &mut hooks,
    &mut harness,
    tuple_list!(edges_observer, time_observer),
    &mut fuzzer,
    &mut state,
    &mut mgr,
)
.expect("Failed to create QemuExecutor");

```

```

tuple_list!(edges_observer, time_observer),

```



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time      0h-0m-32s
clients      2
execution    104314
exec/sec     3.599k

```



# A Simple Fuzzer

```

client #1 (l/r arrows to switch)
executions   104314
exec/sec     3.599k

```

```

[Stats #1] (GLOBAL) run time: 47h-59m-32s, clients: 2, corpus: 56011, objectives: 0, executions: 881082769, exec/sec: 5.100k
              (CLIENT) corpus: 56011, objectives: 0, executions: 881082769, exec/sec: 5.100k, edges: 44549/44706 (99%), stability: 43610/44600 (97%)
[Testcase #1] (GLOBAL) run time: 47h-59m-33s, clients: 2, corpus: 56012, objectives: 0, executions: 881092795, exec/sec: 5.100k
              (CLIENT) corpus: 56012, objectives: 0, executions: 881092795, exec/sec: 5.100k, edges: 44549/44706 (99%), stability: 43610/44600 (97%)
[Stats #1] (GLOBAL) run time: 47h-59m-33s, clients: 2, corpus: 56012, objectives: 0, executions: 881094649, exec/sec: 5.100k
              (CLIENT) corpus: 56012, objectives: 0, executions: 881094649, exec/sec: 5.100k, edges: 44549/44706 (99%), stability: 43610/44600 (97%)
[Stats #1] (GLOBAL) run time: 47h-59m-35s, clients: 2, corpus: 56012, objectives: 0, executions: 881094649, exec/sec: 5.100k
              (CLIENT) corpus: 56012, objectives: 0, executions: 881094649, exec/sec: 5.100k, edges: 44550/44707 (99%), stability: 43610/44600 (97%)
[Testcase #1] (GLOBAL) run time: 47h-59m-35s, clients: 2, corpus: 56013, objectives: 0, executions: 881103227, exec/sec: 5.100k
              (CLIENT) corpus: 56013, objectives: 0, executions: 881103227, exec/sec: 5.100k, edges: 44550/44707 (99%), stability: 43610/44600 (97%)
[Stats #1] (GLOBAL) run time: 47h-59m-40s, clients: 2, corpus: 56013, objectives: 0, executions: 881103227, exec/sec: 5.100k
              (CLIENT) corpus: 56013, objectives: 0, executions: 881103227, exec/sec: 5.100k, edges: 44551/44708 (99%), stability: 43610/44600 (97%)
[Testcase #1] (GLOBAL) run time: 47h-59m-40s, clients: 2, corpus: 56014, objectives: 0, executions: 881127950, exec/sec: 5.100k
              (CLIENT) corpus: 56014, objectives: 0, executions: 881127950, exec/sec: 5.100k, edges: 44551/44708 (99%), stability: 43610/44600 (97%)
[Stats #1] (GLOBAL) run time: 47h-59m-42s, clients: 2, corpus: 56014, objectives: 0, executions: 881127950, exec/sec: 5.100k
              (CLIENT) corpus: 56014, objectives: 0, executions: 881127950, exec/sec: 5.100k, edges: 44552/44708 (99%), stability: 43610/44600 (97%)
[Testcase #1] (GLOBAL) run time: 47h-59m-43s, clients: 2, corpus: 56015, objectives: 0, executions: 881137484, exec/sec: 5.100k
              (CLIENT) corpus: 56015, objectives: 0, executions: 881137484, exec/sec: 5.100k, edges: 44552/44708 (99%), stability: 43610/44600 (97%)
[Stats #1] (GLOBAL) run time: 47h-59m-43s, clients: 2, corpus: 56015, objectives: 0, executions: 881137484, exec/sec: 5.100k
              (CLIENT) corpus: 56015, objectives: 0, executions: 881137484, exec/sec: 5.100k, edges: 44552/44708 (99%), stability: 43610/44600 (97%)
[Testcase #1] (GLOBAL) run time: 47h-59m-43s, clients: 2, corpus: 56016, objectives: 0, executions: 881141139, exec/sec: 5.100k
              (CLIENT) corpus: 56016, objectives: 0, executions: 881141139, exec/sec: 5.100k, edges: 44552/44708 (99%), stability: 43610/44600 (97%)
[Stats #1] (GLOBAL) run time: 47h-59m-48s, clients: 2, corpus: 56016, objectives: 0, executions: 881163879, exec/sec: 5.100k
              (CLIENT) corpus: 56016, objectives: 0, executions: 881163879, exec/sec: 5.100k, edges: 44552/44708 (99%), stability: 43610/44600 (97%)

```

```

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.595k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```

5.1k executions per second



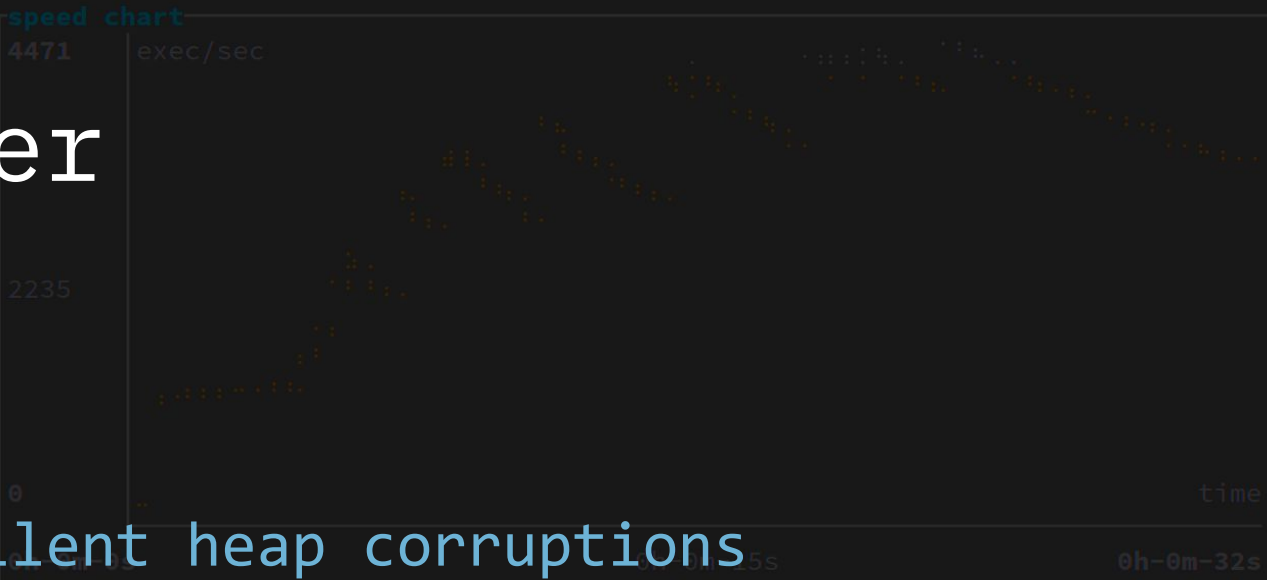
fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients           2
execution         104314
exec/sec          3.272k

```



# A More Complex Fuzzer

```

client #1 (l/r arrows to switch)
executions        104314
exec/sec          3.272k
corpus            5083
objectives        0
edges            8738/96215 (9%)
stability         96152/96215 (99%)

```

- Snapshot-based Fuzzing
- AddressSanitizer to uncover silent heap corruptions

```

clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```

- Scalability over cores



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients          2
executions       104314
exec/sec         3.272k

```

# QASAN Sanitization

```

client #1 (l/r arrows to switch)
executions       104314
exec/sec         3.272k
corpus           5083
objectives       0
edges           8738
stability        96152/96215 (99%)

```

- Sanitizers checks wider range of errors at runtime
- e.g. Illegal memory access
- Track all memory accesses



```

clients logs ('t' to show/hide)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.625k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```

- Hook known libc/allocation functions (malloc/free, strcpy, ...)
- Crash on out-of-bounds accesses, uaf, etc.



fuzz\_regex\_match (default)

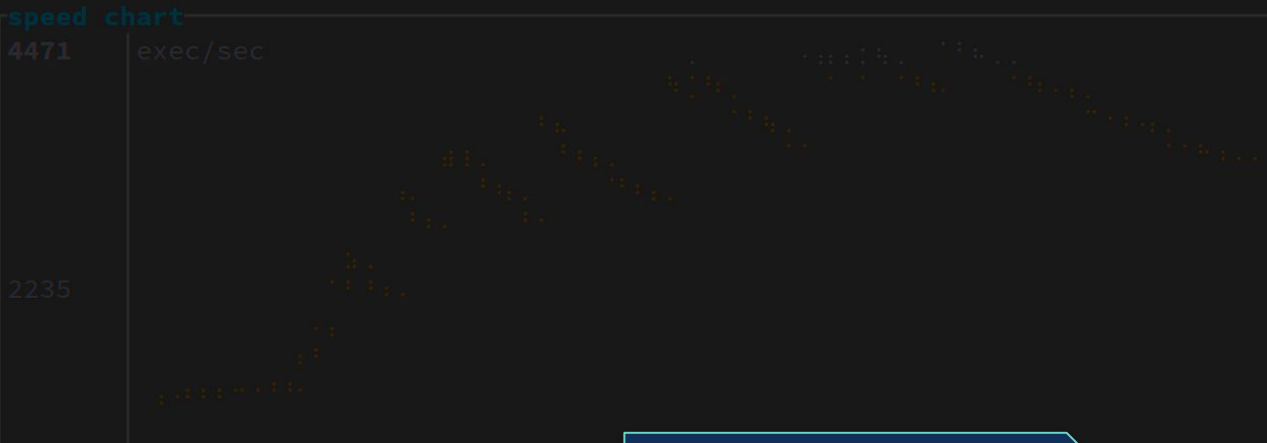
charts ('g' switch)  
speed | corpus | objectives

```

generic
run time      0h-0m-32s
clients      2
executions   104314
exec/sec     3171

```

# Userspace Snapshot



```

client #1 (l/r arrows to switch)
executions   104314
exec/sec     3.272k
corpus       5083
objectives   0
edges        8738/96215 (9%)
stability    96152/96215 (99%)

```

Text

Track  
changed  
pages at  
execution  
*efficiently*

Text

Changed Data

Changed Stack

Reset  
all  
changed  
pages,  
etc.

Text

Reset Data

Reset Stack

```

clients 1
[Stats #0] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```





fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients          2
executions       104314
exec/sec         3271

```



# Profit!

```

client #1 (l/r arrows to switch)
executions       104314

```

```

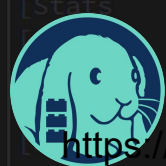
=====
AddressSanitizer Error: Invalid 4 bytes write at 0x4000893f
#0 0x400006ea284c in quram_resize_ (/home/andrea/Desktop/hand_on_2/system/system/lib64/libimagecodec.quram.so+0x7)
#1 0x400002091b38 in __libqasan_malloc /home/andrea/Desktop/LibAFL/libafl_qemu/libqasan/malloc.c:184
#2 0x40000208f78c in malloc /home/andrea/Desktop/LibAFL/libafl_qemu/libqasan/hooks.c:88
#3 0x400007025f9c in _Znwm (/home/andrea/Desktop/hand_on_2/system/system/lib64/libimagecodec.quram.so+0x1fef9c)
#4 0x400007008a98 in _ZN14QuramDngRender8doRenderEP15QuramDngDecoder (/home/andrea/Desktop/hand_on_2/system/system/lib64/libimagecodec.quram.so+0x1e1a98)
Address 0x4000893f5c40 is 118 bytes to the right of the 42-byte chunk [0x4000893f5ba0,0x4000893f5bca)
Allocated at:
#0 0x400004cb1370 in syscall (/home/andrea/Desktop/hand_on_2/system/system/lib64/libc.so+0x1f370)
#1 0x400002091b98 in __libqasan_malloc /home/andrea/Desktop/LibAFL/libafl_qemu/libqasan/malloc.c:197
#2 0x400002091ddc in __libqasan_calloc /home/andrea/Desktop/LibAFL/libafl_qemu/libqasan/malloc.c:258
#3 0x40000208f83c in calloc /home/andrea/Desktop/LibAFL/libafl_qemu/libqasan/hooks.c:98
#4 0x7ffff7fc2e50 in harnessDecode (/home/andrea/Desktop/hand_on_2/harness+0xe50)
Context:
X0: 0x000000000000018      X1: 0x0000000000000214    X2: 0x0000000000000026    X3: 0x0000000fffffe16
X4: 0x000000000000022c    X5: 0x0000000000000241    X6: 0x0000000000000000    X7: 0x0000000000000029
X8: 0x0000000000000008    X9: 0x0000000000000029    X10: 0x000000000000002a   X11: 0x0000000fcbbbbbb
X12: 0x000000000000001d   X13: 0x0000000bd8c8c8c    X14: 0x0000000000000018   X15: 0x00000000000000a8
X16: 0xfffffffffffffff58   X17: 0x00000000000000a0   X18: 0x000000000000002a   X19: 0x0000000000000003
X20: 0x000000000000000e    X21: 0x0000000000000009    X22: 0x000000000000002a   X23: 0x004000893f5ba0
X24: 0x00400089401598      X25: 0x000000000000002d    X26: 0x0000000000000001   X27: 0x00000000000000bc
X28: 0x0000000000000000    X29: 0x004000007fe9f0     X30: 0x00400006ea26f0     Sp: 0x004000007fe990
Pc: 0x00400006ea         Pstate: 0x00555520000000

```

```

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```



# Fuzz

Everything,  
Everywhere,  
All at Once

Scaling to  
cores and  
machines

fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time      0h-0m-32s
client #1
executions    104314
exec/sec      3271

```



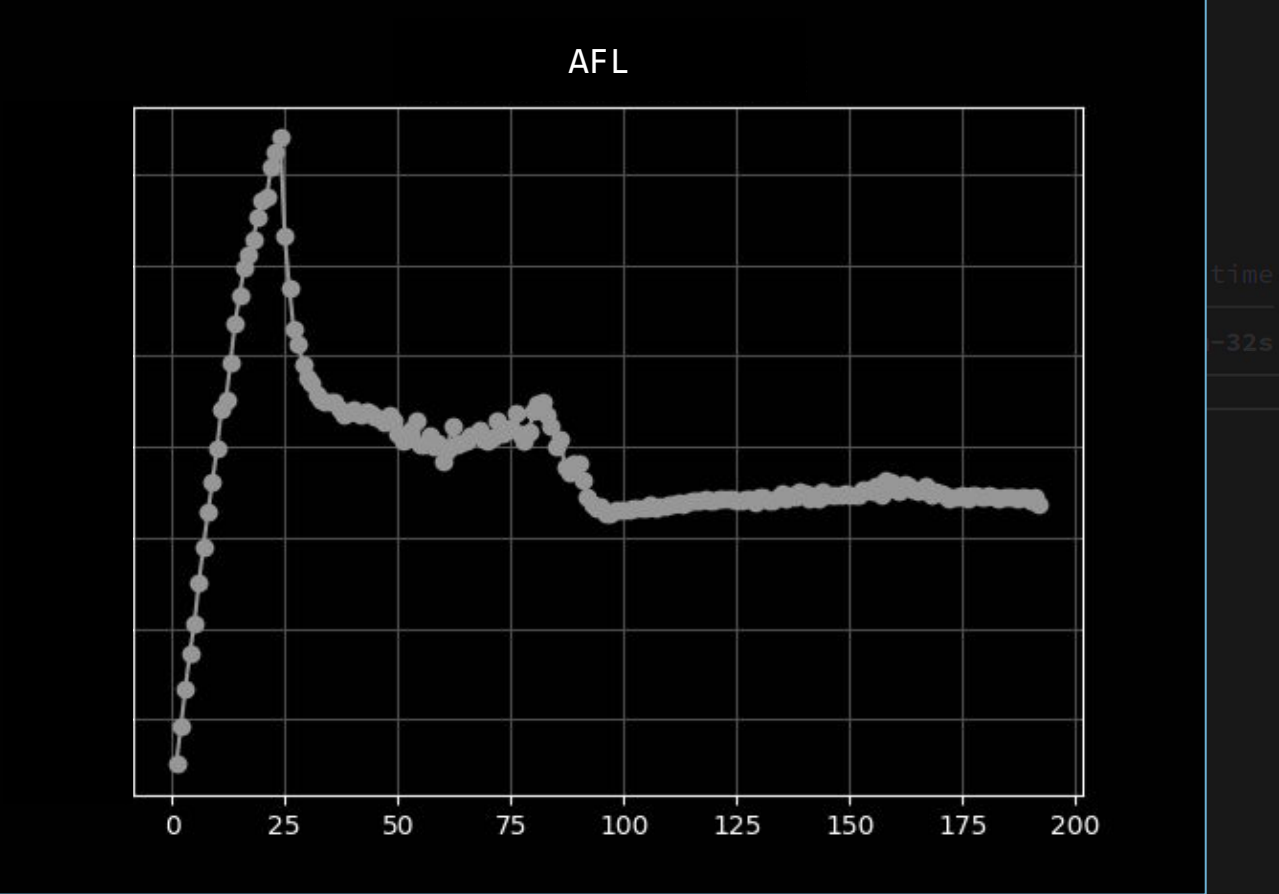
# Fuzzer Scaling

```

client #1 (l/r arrows to switch)
executions    104314
exec/sec      3.572k
corpus        5083
objectives    0
edges        8738
stability     96152/96215 (99%)

```

- Scaling is *hard*
- Not sharing events means lots of duplicated effort
- Communication slows them down



```

clients logs
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```

- Communication via:
  - disk?
  - network?
  - intermittent restarts?
  - something else?



From: <https://github.com/gamozolabs/aflbench>

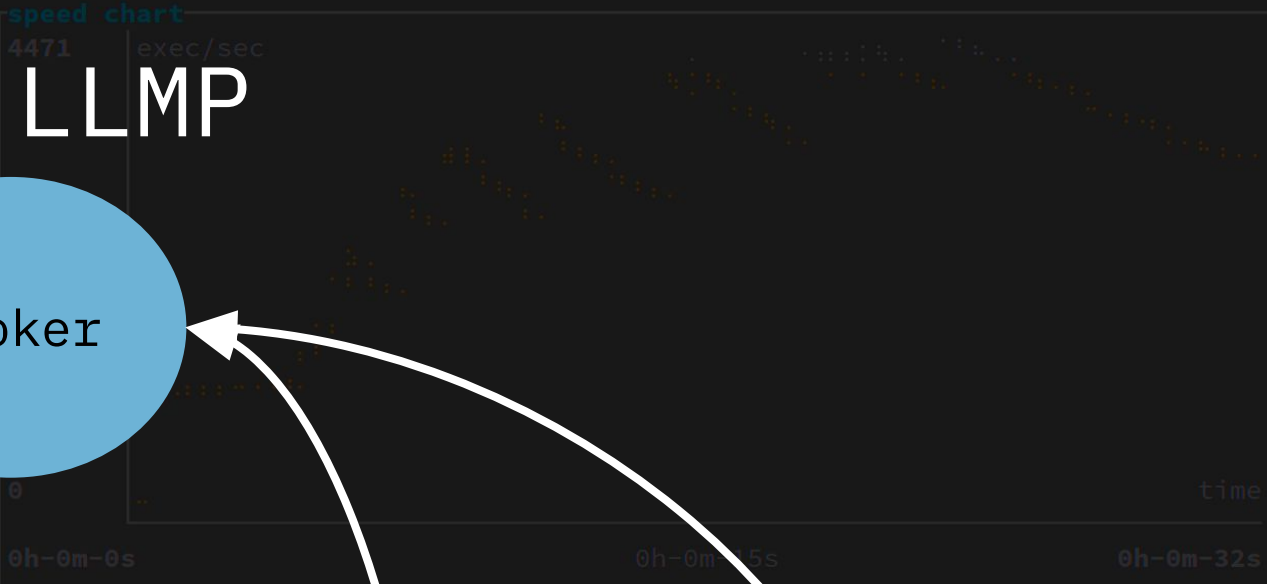
<https://t.me/learningnets>

# Multi-Node Fuzzing: LLMP

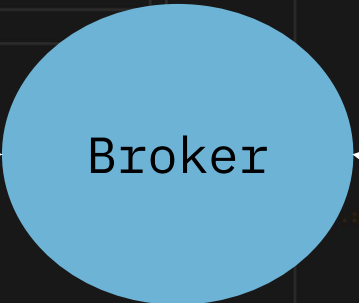
fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

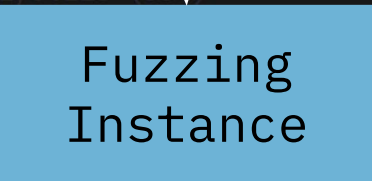
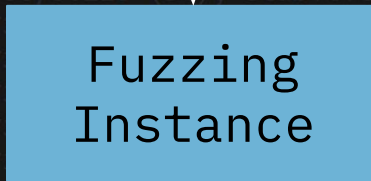
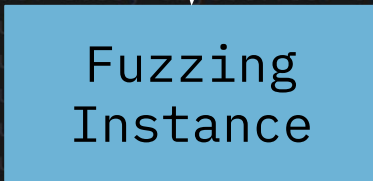
```
generic  
run time 0h-0m-32s  
client #1  
executions 104314  
exec/sec 3271
```



```
client #1 (l/r arrows to switch)  
executions 104314  
exec/sec 3.272k  
corpus 5083  
objectives 0  
edges 8738/96215 (9%)  
stability 96152/96215 (99%)
```



shared memory



```
clients logs ('t' to show/hide)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```

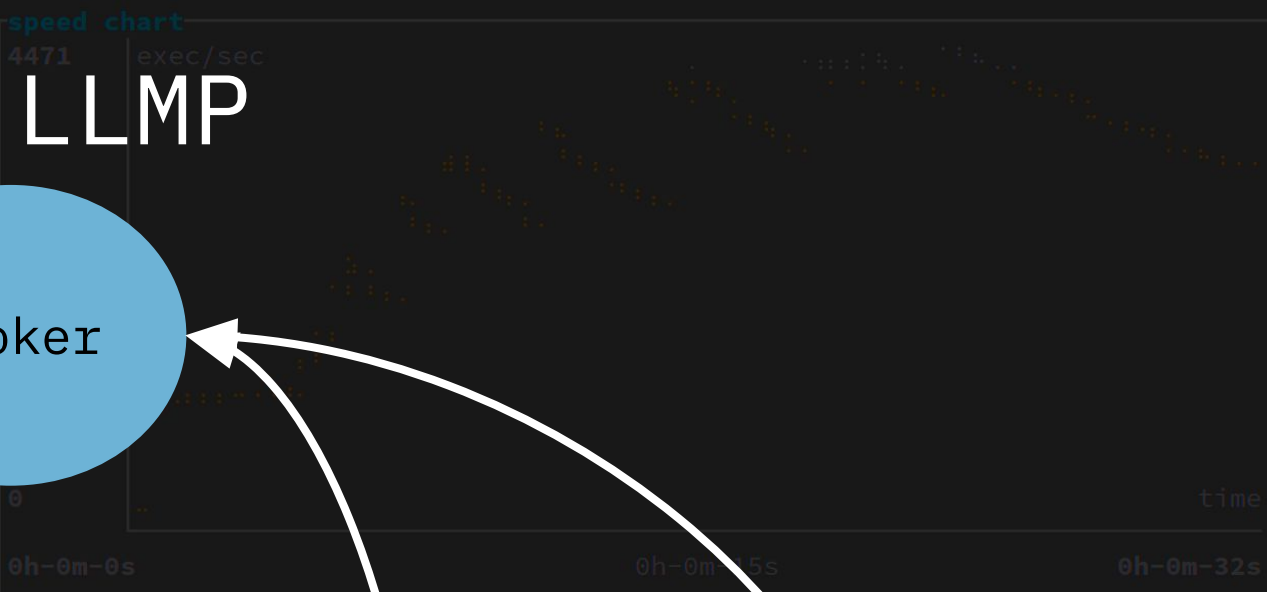


# Multi-Node Fuzzing: LLMP

fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```
generic  
run time      0h-0m-32s  
client #1  
executions    104314  
exec/sec      3271
```



```
client #1 (l/r arrows to switch)  
executions    104314  
exec/sec      3.272k  
corpus        5083  
objectives    0  
edges         8738/96215  
stability     96152/96215
```

```
clients logs ('t' to show/hide)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```

shared memory

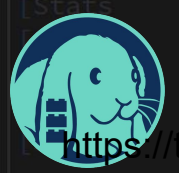
Event

Broker

Fuzzing Instance

Fuzzing Instance

Fuzzing Instance

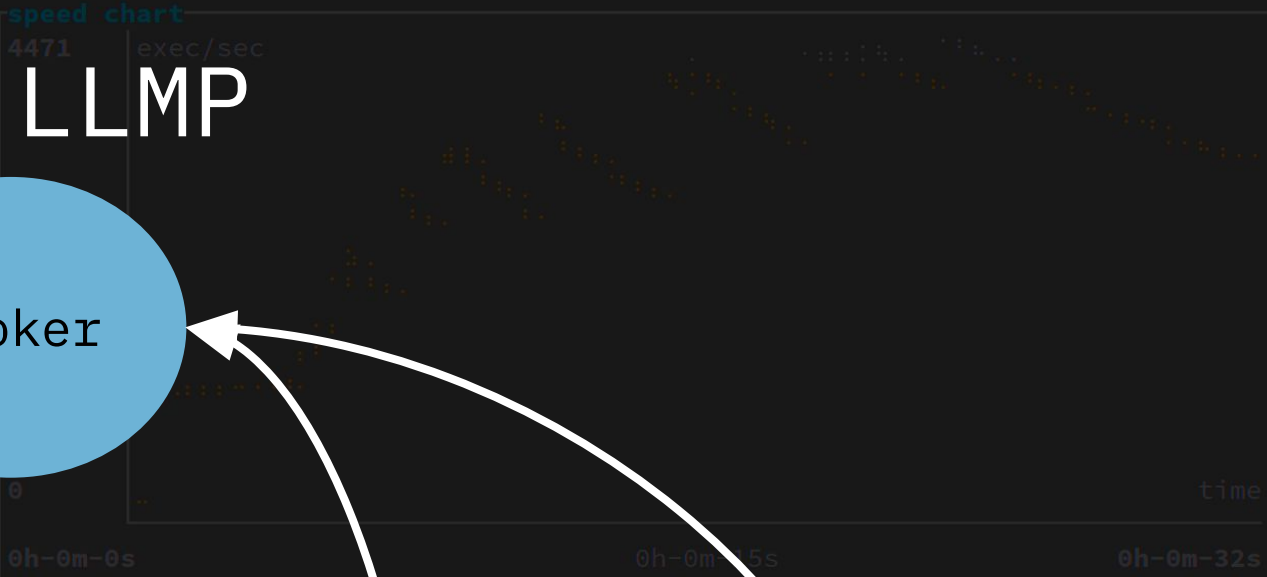


# Multi-Node Fuzzing: LLMP

fuzz\_regex\_match (default)

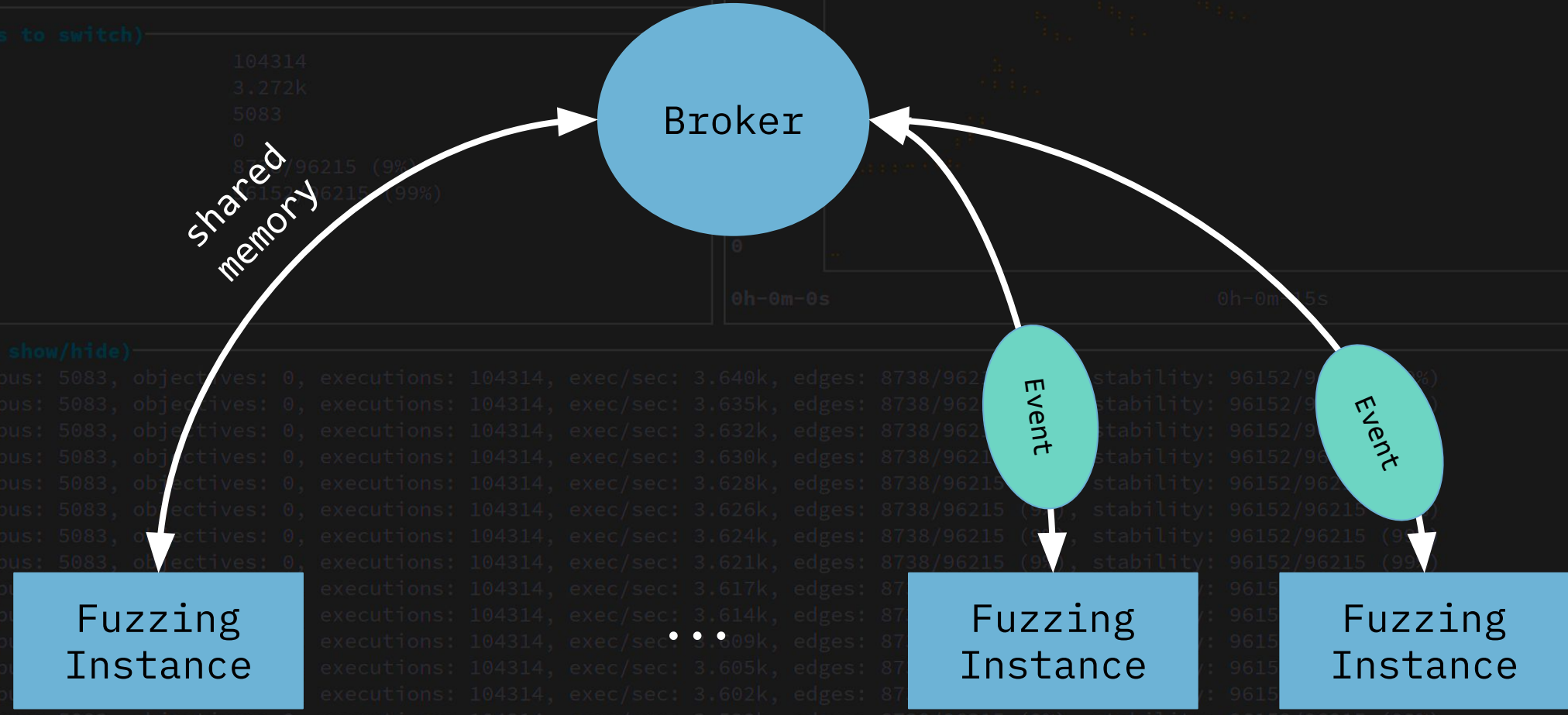
charts ('g' switch)  
speed | corpus | objectives

```
generic  
run time 0h-0m-32s  
client #1  
executions 104314  
exec/sec 3271
```



```
client #1 (l/r arrows to switch)  
executions 104314  
exec/sec 3.272k  
corpus 5083  
objectives 0  
edges 8738/96215 (9%)  
stability 96152/96215 (99%)
```

```
clients logs ('t' to show/hide)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)  
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```



fuzz\_regex\_match (default)

speed | corpus | objectives

```

generic
run time          0h-0m-32s
clients           2
executions       104314
exec/sec         3211

```

# Scaling to 80 cores

```

client #1 (l/r arrows to switch)
executions       104314
exec/sec         3.272k

```



1 [     ] 100.0%	25 [     ] 100.0%	49 [     ] 100.0%	73 [     ] 100.0%
2 [     ] 100.0%	26 [     ] 98.7%	50 [     ] 99.3%	74 [     ] 100.0%
3 [     ] 100.0%	27 [     ] 100.0%	51 [     ] 100.0%	75 [     ] 100.0%
4 [     ] 100.0%	28 [     ] 100.0%	52 [     ] 99.3%	76 [     ] 100.0%
5 [     ] 98.7%	29 [     ] 98.7%	53 [     ] 98.7%	77 [     ] 99.3%
6 [     ] 100.0%	30 [     ] 98.7%	54 [     ] 98.7%	78 [     ] 100.0%
7 [     ] 100.0%	31 [     ] 99.3%	55 [     ] 98.7%	79 [     ] 99.3%
8 [     ] 100.0%	32 [     ] 100.0%	56 [     ] 98.7%	80 [     ] 100.0%
9 [     ] 100.0%	33 [     ] 100.0%	57 [     ] 99.3%	81 [     ] 98.7%
10 [     ] 100.0%	34 [     ] 99.4%	58 [     ] 100.0%	82 [     ] 0.0%
11 [     ] 100.0%	35 [     ] 99.3%	59 [     ] 99.3%	83 [     ] 0.0%
12 [     ] 100.0%	36 [     ] 100.0%	60 [     ] 98.7%	84 [     ] 0.0%
13 [     ] 98.7%	37 [     ] 99.3%	61 [     ] 100.0%	85 [     ] 0.0%
14 [     ] 100.0%	38 [     ] 100.0%	62 [     ] 100.0%	86 [     ] 0.0%
15 [     ] 100.0%	39 [     ] 100.0%	63 [     ] 100.0%	87 [   ] 5.9%
16 [     ] 100.0%	40 [     ] 100.0%	64 [     ] 99.3%	88 [     ] 0.0%
17 [     ] 98.7%	41 [     ] 100.0%	65 [     ] 98.7%	89 [     ] 0.0%
18 [     ] 99.4%	42 [     ] 100.0%	66 [     ] 100.0%	90 [  ] 1.3%
19 [     ] 100.0%	43 [     ] 98.7%	67 [     ] 100.0%	91 [     ] 0.0%
20 [     ] 100.0%	44 [     ] 100.0%	68 [     ] 100.0%	92 [  ] 1.3%
21 [     ] 99.3%	45 [     ] 99.3%	69 [     ] 100.0%	93 [     ] 0.0%
22 [     ] 100.0%	46 [     ] 100.0%	70 [     ] 100.0%	94 [  ] 1.3%
23 [     ] 98.7%	47 [     ] 98.7%	71 [     ] 98.7%	95 [  ] 0.7%
24 [     ] 99.3%	48 [     ] 100.0%	72 [     ] 100.0%	96 [     ] 0.0%

Mem [|||||] 2.73G/252G  
 Swp [|||||] 0K/8.00G  
 Tasks: 215, 240 thr; 82 running  
 Load average: 47.43 29.10 18.34  
 Uptime: 23:58:57

```

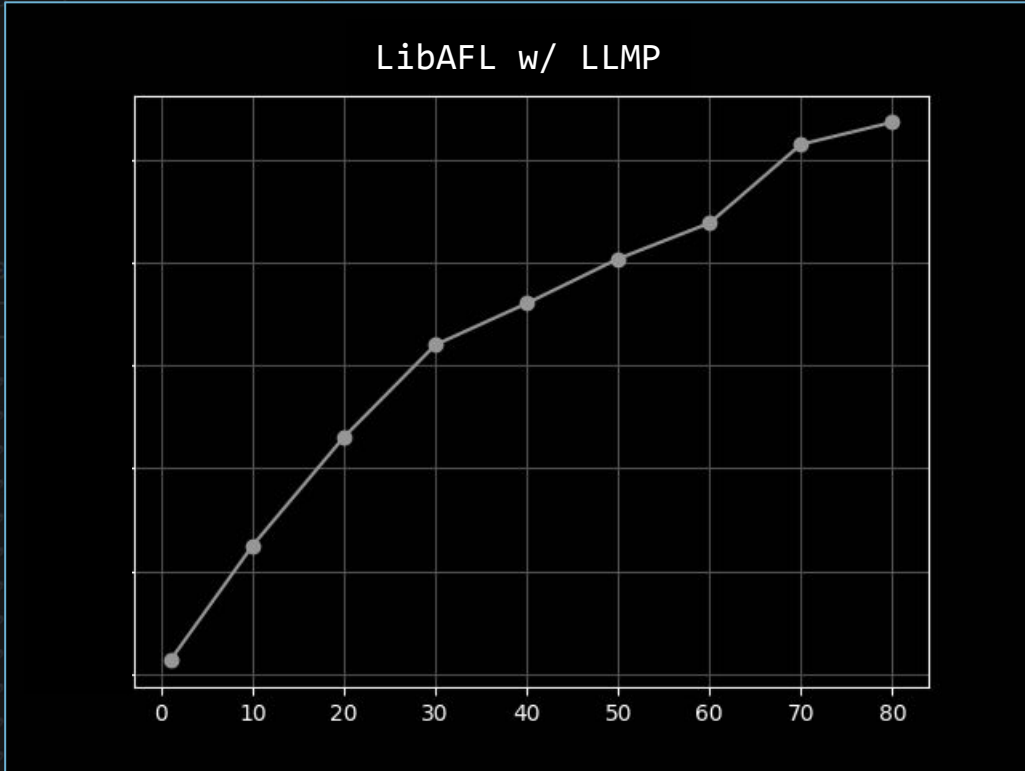
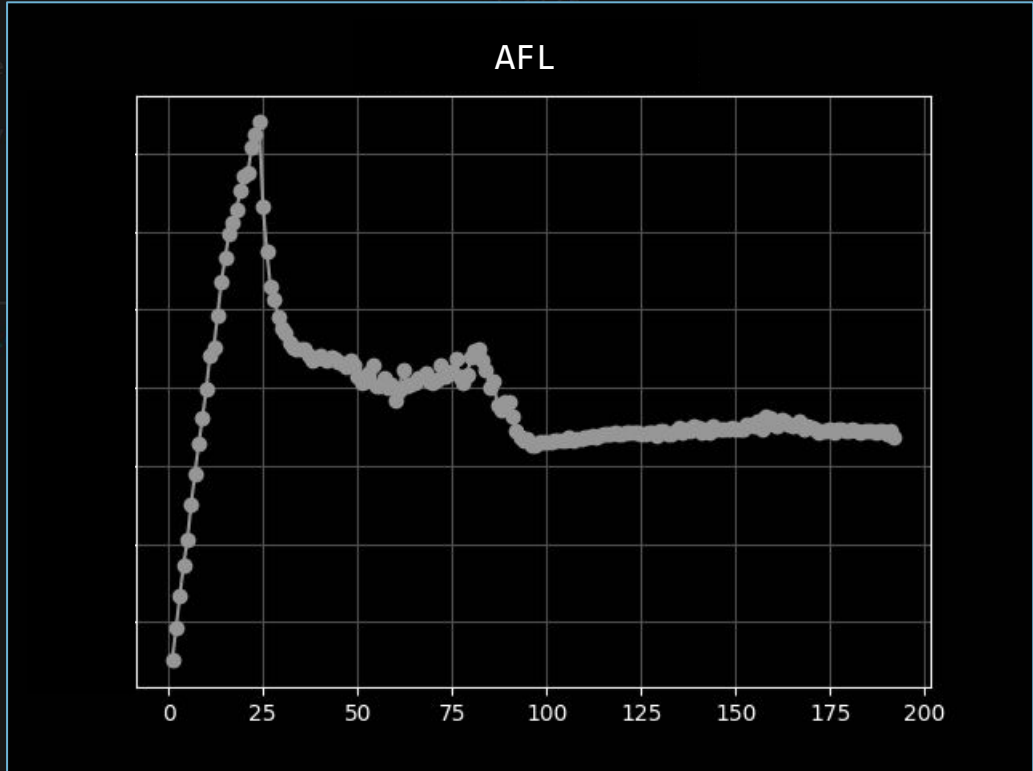
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```



<https://t.me/learningnets>

# Scaling Comparison: AFL and LLMP



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

generic  
run time 0h-0m-32s  
clients 2  
executio 104314  
exec/sec 32.4

# Scaling to 80 cores



```
[Stats #46] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 1007, objectives: 0, executions: 322104, exec/sec: 6.026k, edges: 4755/65536 (7%), stability: 65532/65536 (99%)
[Stats #50] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 973, objectives: 0, executions: 315802, exec/sec: 5.908k, edges: 4757/65536 (7%), stability: 65532/65536 (99%)
[Stats #51] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 890, objectives: 0, executions: 294240, exec/sec: 5.505k, edges: 4757/65536 (7%), stability: 65532/65536 (99%)
[Stats #53] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 913, objectives: 0, executions: 305663, exec/sec: 5.719k, edges: 4755/65536 (7%), stability: 65534/65536 (99%)
[Stats #53] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 913, objectives: 0, executions: 305663, exec/sec: 5.719k, edges: 4757/65536 (7%), stability: 65534/65536 (99%)
[Stats #55] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 884, objectives: 0, executions: 297382, exec/sec: 5.565k, edges: 4750/65536 (7%), stability: 65532/65536 (99%)
[Stats #55] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 884, objectives: 0, executions: 297382, exec/sec: 5.565k, edges: 4755/65536 (7%), stability: 65532/65536 (99%)
[Stats #55] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 884, objectives: 0, executions: 297382, exec/sec: 5.565k, edges: 4757/65536 (7%), stability: 65532/65536 (99%)
[Stats #60] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 854, objectives: 0, executions: 281700, exec/sec: 5.273k, edges: 4757/65536 (7%), stability: 65532/65536 (99%)
[Stats #63] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 1005, objectives: 0, executions: 311398, exec/sec: 5.830k, edges: 4749/65536 (7%), stability: 65532/65536 (99%)
[Stats #63] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 1005, objectives: 0, executions: 311398, exec/sec: 5.830k, edges: 4750/65536 (7%), stability: 65532/65536 (99%)
[Stats #63] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 1005, objectives: 0, executions: 311398, exec/sec: 5.830k, edges: 4750/65536 (7%), stability: 65532/65536 (99%)
[Stats #71] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 834, objectives: 0, executions: 267731, exec/sec: 5.013k, edges: 4757/65536 (7%), stability: 65534/65536 (99%)
[Stats #71] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
(CLIENT) corpus: 834, objectives: 0, executions: 267731, exec/sec: 5.013k, edges: 4757/65536 (7%), stability: 65534/65536 (99%)
[Stats #71] (GLOBAL) run time: 0h-0m-54s, clients: 82, corpus: 74661, objectives: 0, executions: 24985314, exec/sec: 462.8k (Aggregated): edges: 0.726% stability: 99.994%
```

## 462.8k executions per second



**Fuzz**

Everything,

Everywhere,

**All at Once**

catch injections  
& corruptions  
at the same time

fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

```

generic
run time      0h-0m-32s
clients      2
executio     104314
exec/sec     3272

```



# Feedback Fuzzing == Only Crashes(?)

```

client #1 (l/r arrows to switch)
executions  104314
exec/sec    3.272k
corpus      5083
objectives  0
edges       8738/96215 (9%)
stability   96152/96215 (99%)

```

- Coverage-based fuzzing is good at finding crashes like memory corruptions

```

clients logs ('u' to show/hide)
[Stats #0] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.638k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

```

- Unguided fuzzers like sqlmap are great at finding injection vulnerabilities but only work on network targets and have no coverage

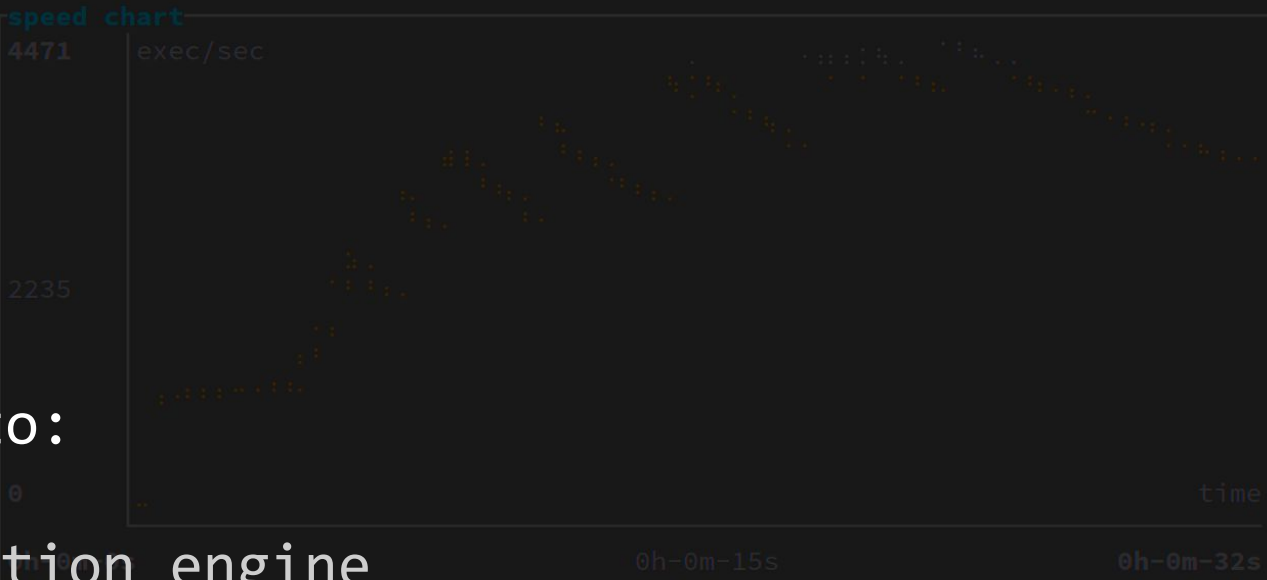
IDEA: Find injection vulnerabilities while doing normal AFL++/libafl style fuzzing!



fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

generic	
run time	0h-0m-32s
clients	2
executions	104314
exec/sec	3271



# The Idea

client #1 (l/r arrows to switch)	
executions	104314
exec/sec	3.272k
corpus	5083
objectives	0
edges	8738/96215 (9%)
stability	96152/96215 (99%)

Use LibAFL-QEMU (usermode) hooks to:

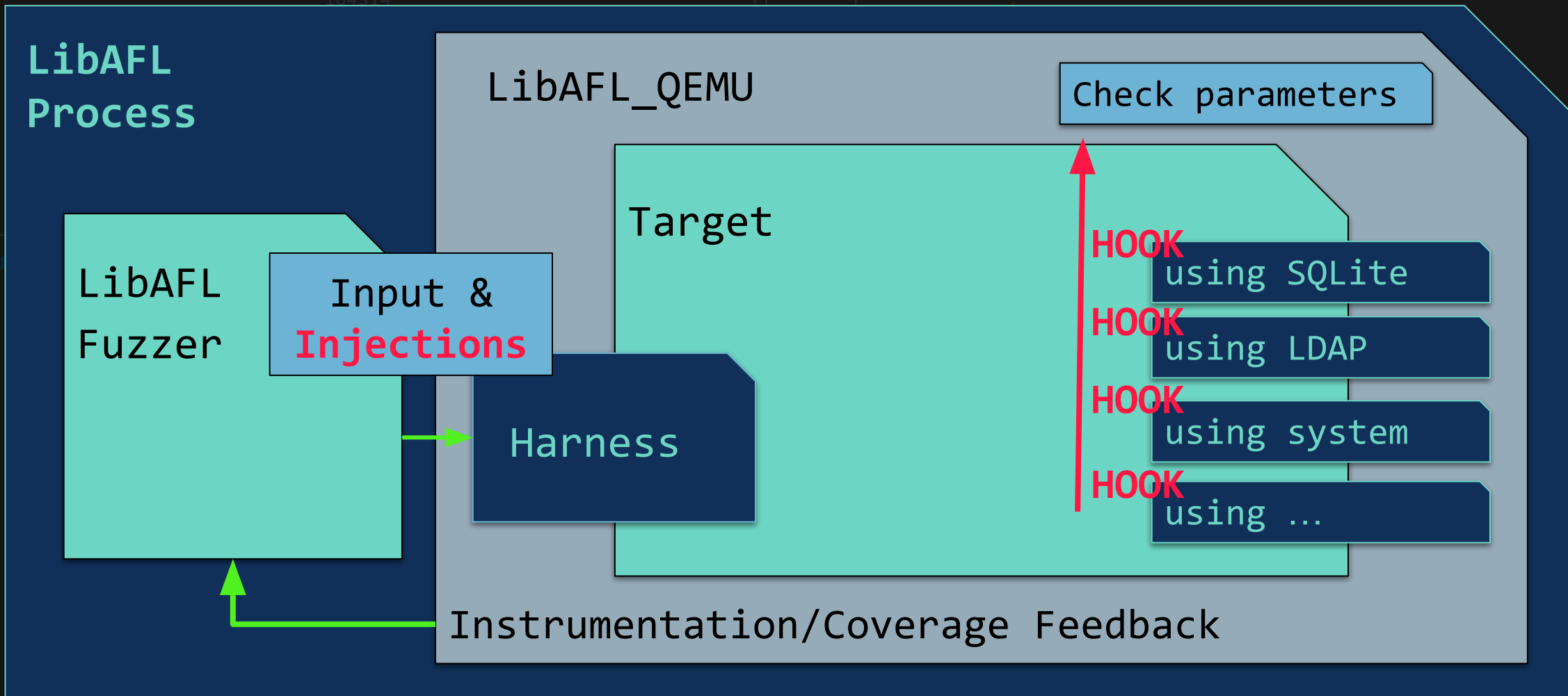
- add injection tests to our mutation engine

clients logs ('t' to show/hide)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.633k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	- hook injection susceptible functions and analyze all queries
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	- crash if injection test is found unsanitized
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.600k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.600k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	
[Stats #] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)	

**Bonus:** Having a flexible configuration script so users can easily modify what they want hunt for - and how



# Fuzzing for Injections



# Example: SQL injection configuration

injections.yml

```
- name: "sql"
  functions:
    - function: "sqlite3_exec"
      parameter: 1
    - function: "mysql_query"
      parameter: 1
  tests:
    - input_value: "'\''"
      match_value: "'\''"
    - input_value: "1\'' OR \'"
      match_value: "1\'' OR"
```

```
sqlite3_exec() - Execute SQL
statements
Definition:
int sqlite3_exec( sqlite3 *db,
const char* sql, ... )
```

Injection into the 2nd parameter!



# Advantages/Disadvantages

- False positives unlikely
- False negatives can happen - depending on your input + match config
- You can hunt for all kinds of injection vulnerabilities

- ... all while doing coverage-guided fuzzing!

All implemented using LibAFL QEMU APIs



```
marc /prg/libafl/fuzzers/qemu_injections (vhqemu) $ █
```

**Fuzz  
Everything,  
Everywhere,  
All at Once**

**Final Words**

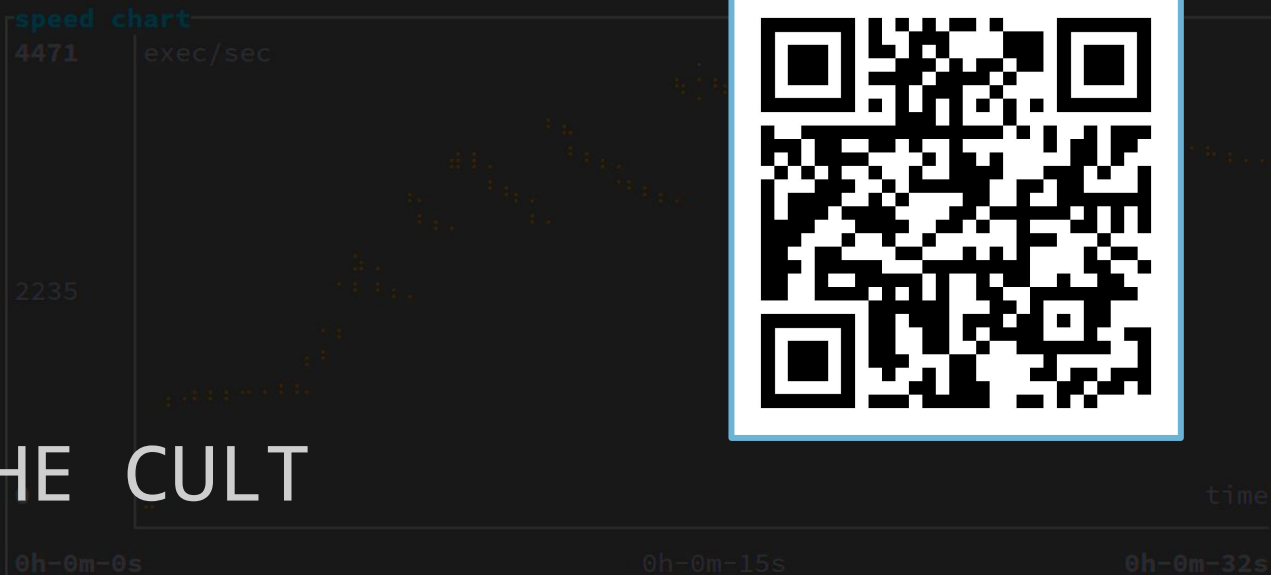
fuzz\_regex\_match (default)

charts ('g' switch)  
speed | corpus | objectives

generic  
run time 0h-0m-32s  
clients 2  
executions 104314  
exec/sec 3.272k

# LibAFL is FOSS!

client #1 (l/r arrows to switch)  
executions 104314  
exec/sec 3.272k  
corpus 5083  
objectives 0  
edges 8738/96215 (9%)  
stability 96152/96215 (99%)



## JOIN THE CULT

clients logs ('t' to show/hide)

```
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.635k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.631k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.628k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.626k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)
```

<https://github.com/AFLplusplus/LibAFL>

108

Contributors

203

Used by

11

Discussions

2k

Stars

231

Forks



<https://t.me/learningnets>

fuzz\_regex\_match (default)

charts ('g' switch)

speed | corpus | objectives

generic

run time 0h-0m-32s  
client 2  
executions 104314  
exec/sec 3271

client #1 (l/r arrows to switch)

executions 104314  
exec/sec 3271  
corpus 5083  
objectives 0  
edges 8738/96215 (9%)  
stability 96152/96215 (99%)

speed chart



# Conclusion

- Fuzz everything, everywhere, all at once
- Extremely scalable fuzzers
- QEMU is amazing

clients logs ('t' to show/hide)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.640k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.632k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.630k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.624k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.621k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.617k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.614k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.609k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.605k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.602k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.599k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.596k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.592k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.590k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.585k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)

[Stats #1] corpus: 5083, objectives: 0, executions: 104314, exec/sec: 3.580k, edges: 8738/96215 (9%), stability: 96152/96215 (99%)



```
while (questions());

char buf[16];
strncpy(buf, " "
        "Thank you for your attention."
        "\n", sizeof(buf));
printf("%s", buf);
```

**Thanks y' a11**

