AFL++

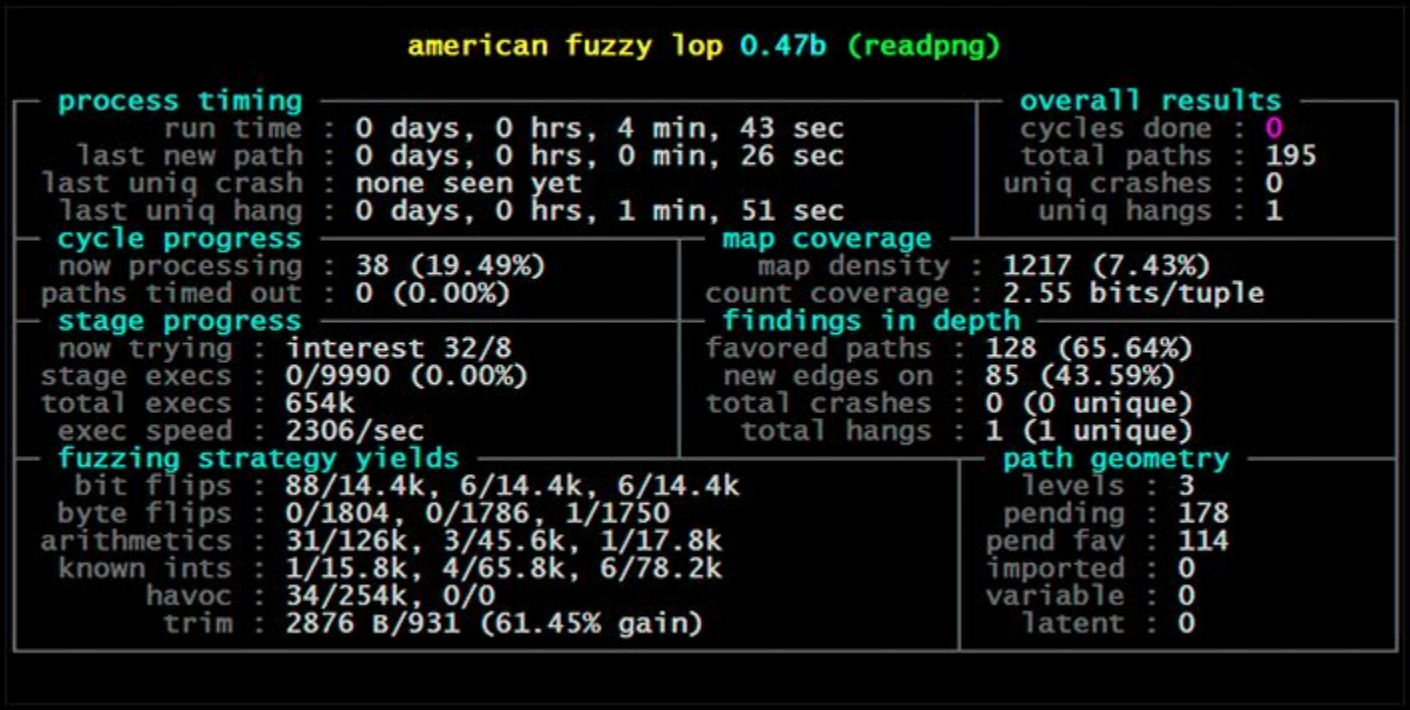
Combining Incremental Steps of Fuzzing Research

Andrea Fioraldi, Dominik Maier, Heiko Eißfeldt, Marc Heuse

@andreafioraldi, @domenuk

{andrea, dominik}@aflplus.plus

American Fuzzy Lop



American Fuzzy Lop

- A legendary tool that proved its effectiveness
- A baseline for a wide range of academic and industrial research
- No new features after 2017

American Fuzzy Lop

- A legendary tool that proved its effectiveness
- A baseline for a wide range of academic and industrial research
- No new features after 2017

Fork it!

A lot of Research Based on AFL

- AFLFast
- AFLSmart
- AFL LAF-Intel
- AFL MOpt
- kAFL
- ...
- Whatever-AFL

Works On Fuzzer Scheduling

- Seed scheduling [AFLFast]

  ⇒ How much time should we fuzz a test case?

- Mutation scheduling [MOpt]

  ⇒ Probability for each mutational operator

Works On Bypassing Roadblocks

- Feedback for comparisons [LAF-Intel]

  ⇒ Split multi-byte comparisons

- Input-to-state replacement [Redqueen (kAFL)]

  ⇒ Guess the input bytes that affect a comparison and replace it with the extracted token

Structured Mutators

- Take input structure into account [AFLSmart]
  - Avoid to generate almost always invalid inputs
  - Stress more deep paths

Speed Enhancements

- Reduce the number of instrumented program points while maintaining the same coverage [Instrim]
- Get rid of fork() and fuzz with snapshots [Opt-AFL]
- Inline instrumentation and re-enable TB linking in QEMU mode [abiondo-AFL]

What if I Want to Use X *AND* Y?

- Orthogonal techniques not easy to combine
- Research fuzzers often unmaintained
- Some techniques are not implemented on top of the original AFL

I created Z *AND* I want X

- If you peak one of the derived fuzzers as baseline you may be incompatible with other orthogonal techniques
- Hard to evaluate techniques without the relation with others (e.g. a new type of coverage without having a roadblock bypassing technique)

Here comes

The AFL++ Project

- Integrates and reimplements fuzzing techniques in a single framework, AFL++
- Ongoing research and new insights about fuzzing using such framework
- We improve the state of the art combining techniques and tuning the implementations

Usability

- All techniques are integrated in afl-fuzz
- Best-effort defaults
- Users familiar with AFL benefit from cutting-edge research without pain

Extensibility

- To enable further research to do cross-comparisons with a reduced effort, we defined a set of API to extend AFL++, the *Custom Mutator API*

Custom Mutator API

afl_custom_fuzz
afl_custom_post_process
afl_custom_trim
afl_custom_havoc_mutation
afl_custom_havoc_mutation_probability
afl_custom_queue_get
afl_custom_queue_new_entry

INSTRUMENT ALL THE THINGS

- We extended techniques to work with other instrumentation backends.
- For Example: QEMU & Unicorn modes can split comparisons in a similar way to LLVM LAF-Intel
- Currently supported instrumentations are LLVM, QEMU, Unicorn, QBDI, GCC plugin, afl-gcc

Runs on Everything

- AFL++ builds and runs on GNU/Linux, Android, iOS, macOS, FreeBSD, OpenBSD, NetBSD, IllumOS, Haiku, Solaris
- It is packaged in popular distributions like Debian, Ubuntu, NixOS, Arch Linux, FreeBSD, Kali Linux, ...

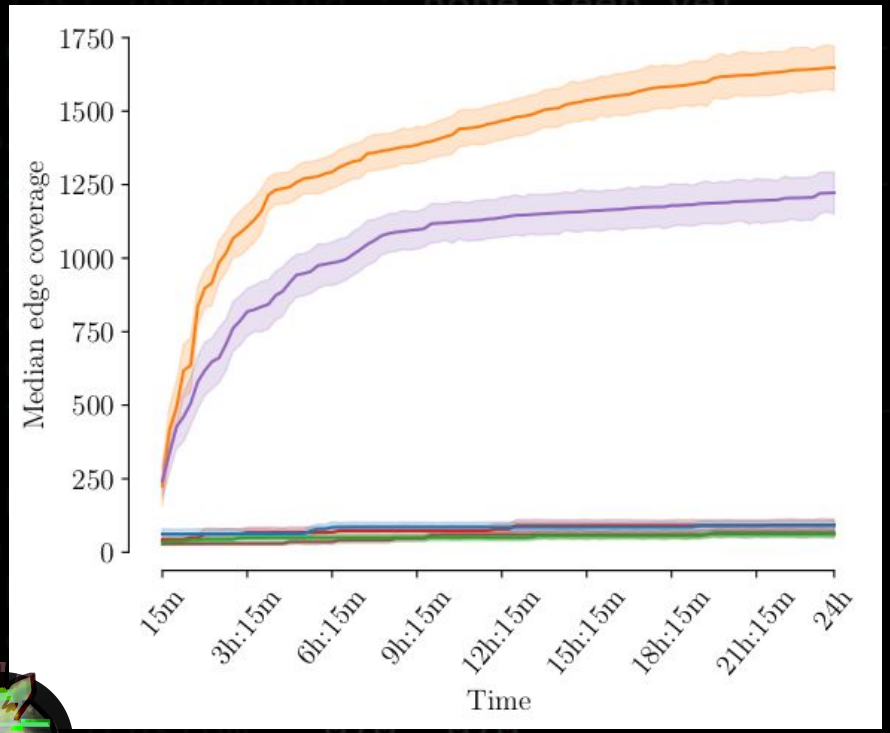
Cross Evaluations

Using AFL++ as baseline gives you immediate access to cross evaluation of your technique combined with pre-existing works

Examples:

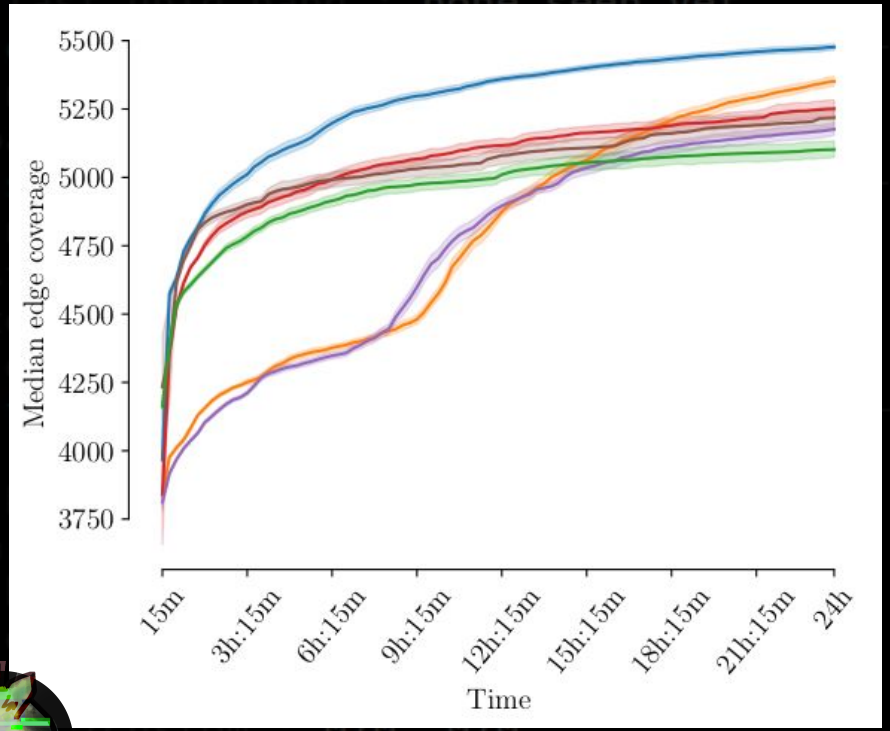
- [Default]
- Ngram4
- MOpt
- Redqueen

Cross-Evaluations (libpcap)



- *Redqueen*
- *Redqueen+MOpt*
- *MOpt*
- *Ngram4*
- *Ngram4+Rare*
- *[Default]*

Cross-Evaluations (bloaty)



- *Redqueen*
- *Redqueen+MOpt*
- *MOpt*
- *Ngram4*
- *Ngram4+Rare*
- *[Default]*

Optimal Configuration

- Observe several runs of AFL++ in different configuration on the same target for a while
- Try to catch blind spots and select the best combination of features
- Profit

Future work

- Static analysis for optimal fuzz settings
- Multicore linear scaling
- Plugin system (executors, queues, feedbacks, …)
- Collision-free instrumentation

Conclusion

- AFL++ enhances comparability of research
- We further improve the state-of-the-art with speed, usability, new features
- AFL++’s custom mutator API can be used to implement novel research in a maintainable way

AFL++ is FOSS!

https://aflplus.plus/

https://github.com/AFLplusplus

Thank you for
your attention.