Technology Innovation Institute

MISOFuzz

A Modular Infrastructure for Scalable Fuzzing Orchestration

Nicole Gervasoni

tii.ae

Who Are we?

Nicole
Researcher
nicole.gervasoni@tii.ae

Lucas
Director
lucas.cordeiro@tii.ae

Mike
Senior Engineer
mikhail.lubinets@tii.ae

Ridhi
Researcher
ridhi.jain@tii.ae

Norbert
Lead researcher
norbert.tihanyi@tii.ae

Motivation

- Plug And Play fuzzing
- Make model Checking and SW verification easily usable for security researchers
- Simplify the integration of fuzzing tools to support cooperative fuzzing (Modular approach)

Motivation

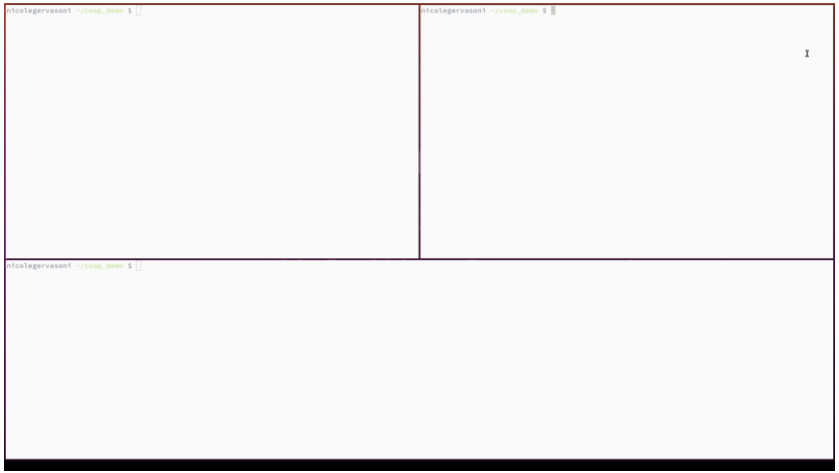
```
1  void crash(char* A, char* B){
2      if (A == "problem1") {
3          if (B == "problem2") {
4              # BUG
5          } else {
6          # nothing
7          }
8      } else if (A == "problem2") {
9          if (B == "problem1"){
10             # BUG
11         } else {
12             # nothing
13         }
14     }
15 }
```

- **Fuzzer1** is very good at solving problem of type *problem1*
- **Fuzzer2** is very good at solving problem of type *problem2*

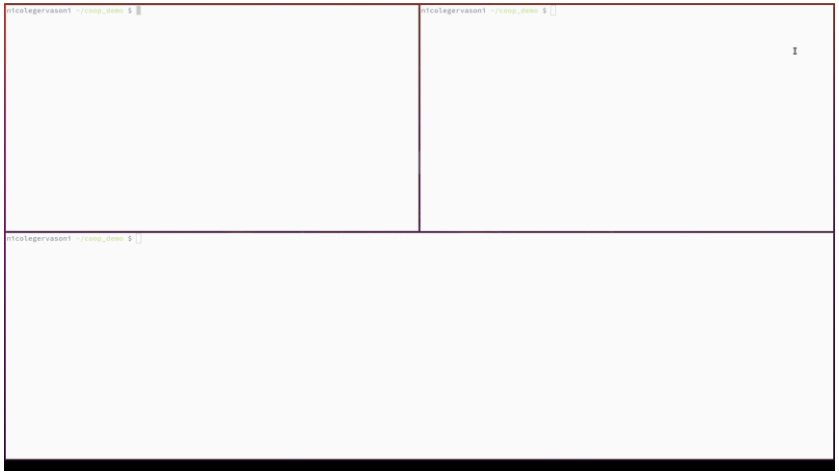
Strategy \ Line	2	3-4	5-6	8	9-10	11-12
Fuzzer1	✓		✓			
Fuzzer2				✓		✓
Fuzzer1 + Fuzzer2	✓		✓	✓		✓
Cooperative Fuzzing	✓	✓	✓	✓	✓	✓

Motivation

2 AFL instances + symcc

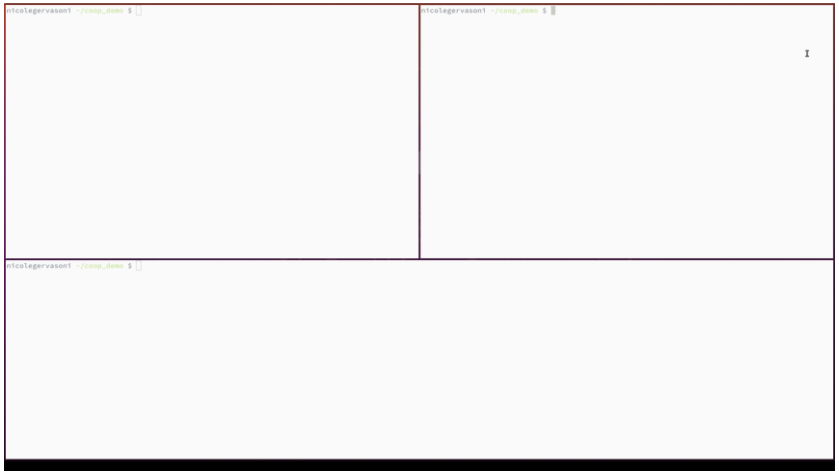


3 AFL++ instances



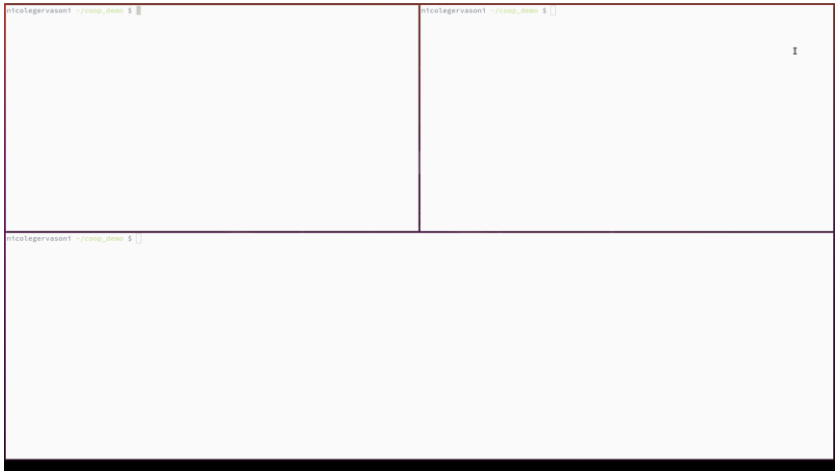
Motivation

2 AFL instances + symcc



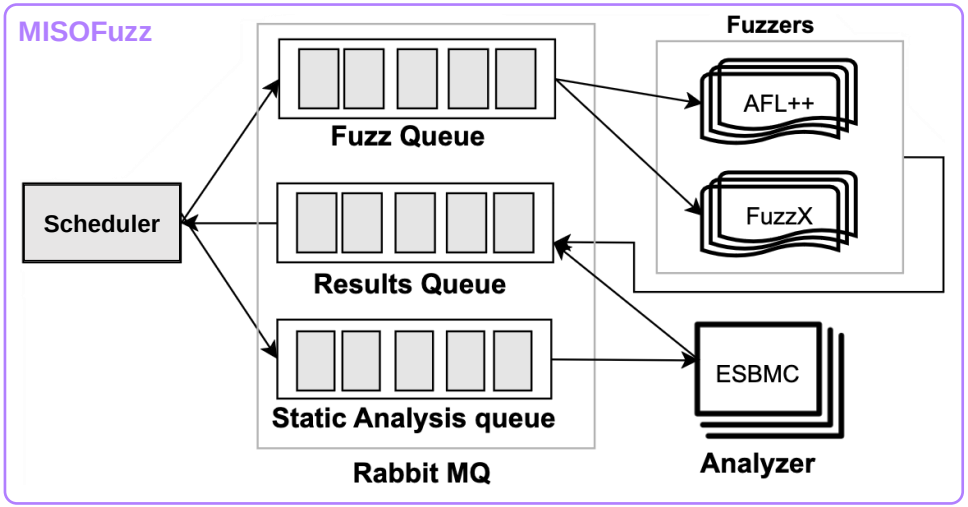
Found 1 crash as soon as symcc start

3 AFL++ instances



Still nothing after 20 min

Architecture
High Level Design



Key features

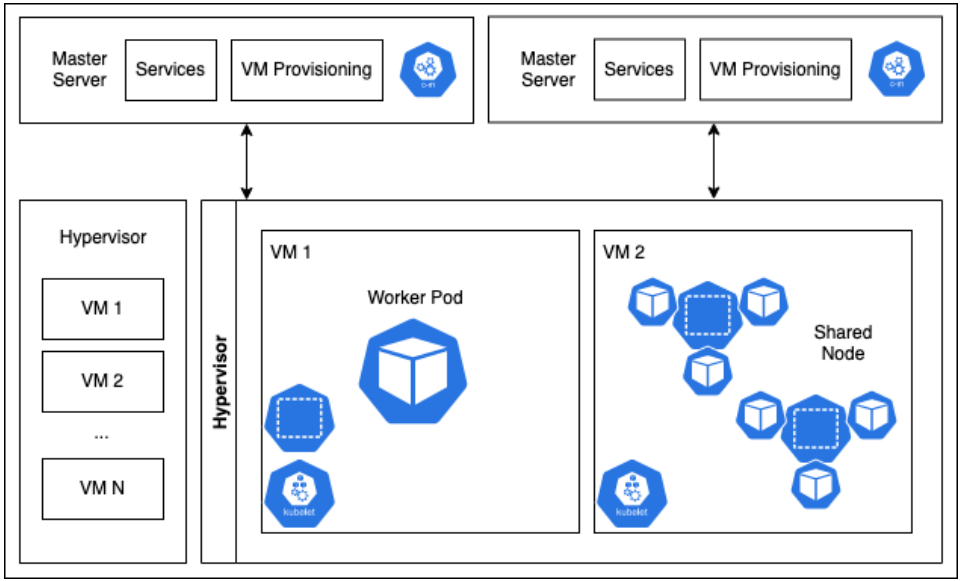
Scalability

Flexibility

- A distributed cooperative fuzzing and software analysis framework
- Easy to scale up and down
- Efficient hardware resource utilization
- Cloud-Native with support of Private Clouds
- Heuristic-based scheduling powered by FuSeBMC

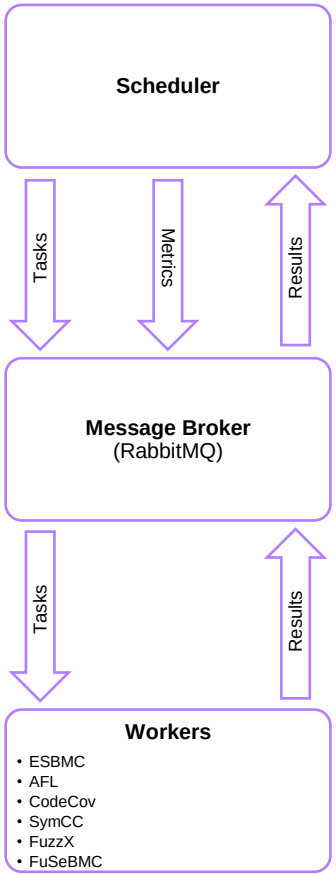
Architecture
Low Level Design

- VM-based isolation, via a hypervisor
- Built on top of a Kubernetes Cluster
- File sharing over NFS
- Intra-cluster communication via a distributed message queue



Architecture
Components

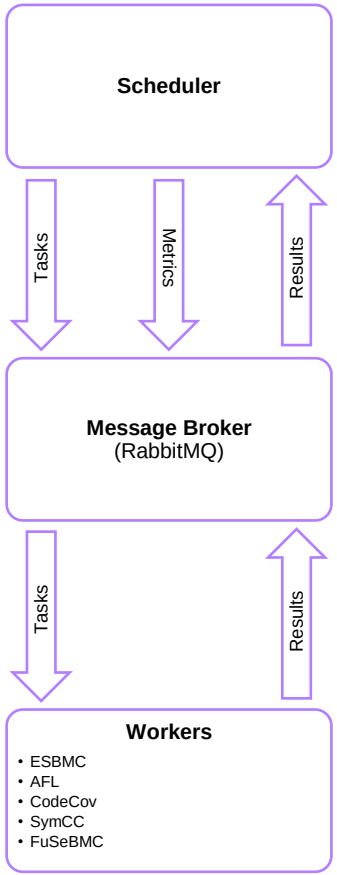
- Scheduler
  - Schedule fuzzing and analysis job
  - Aggregate the results
- Message Broker
  - Facilitates data exchange between scheduler and workers
  - Stores fuzzing metrics, artifacts and reports
  - Provides an event-queue between systems
- Workers
  - Analysis worker: perform static analysis of source code (ESBMC)
    Retrieve tasks from *Static Analysis Queue* and push to *Results Queue*
  - Fuzzing worker: fuzz compiled code (AFL, AFL+SymCC, AFL++, FuSeBMC)
    Retrieve tasks from *Fuzz Queue* and push to *Results Queue*
  - CodeCov



Architecture
Components

RabbitMQ

- Reliable communication system between different components of the system
- Easily deployable
- Easily add or remove workers as needed, and to balance the workload across them.



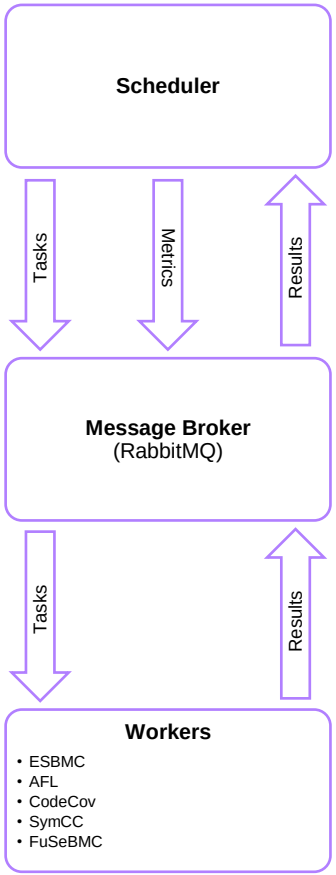
Architecture
Fuzzers

- **AFL** employs genetic algorithms to efficiently increase code coverage of the test cases. For many years after its release, AFL has been considered a "state of the art" fuzzer. [11]

- **SymCC** is a compiler wrapper which embeds symbolic execution into the program during compilation, and an associated run-time support library. In essence, the compiler inserts code that computes symbolic expressions for each value in the program. [10]

- **AFL++** *is a superior fork to Google's AFL - more speed, more and better mutations, more and better instrumentation, custom module support, etc. [9]*

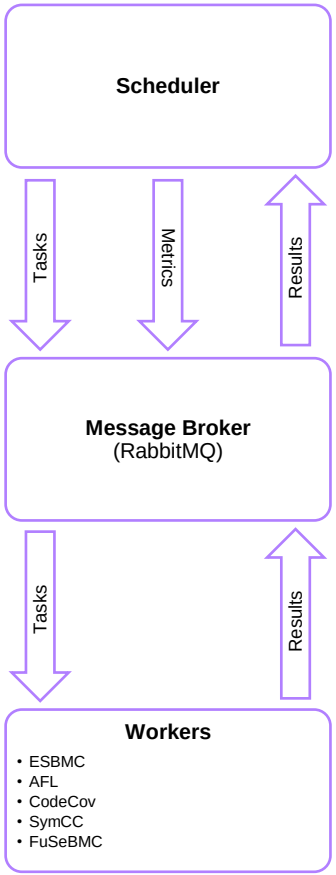
- **FuseBMC** is a test generator for finding security vulnerabilities in C programs. It incrementally injected labels to guide BMC and Evolutionary Fuzzing engines to produce test cases for code coverage and bug finding. [2]



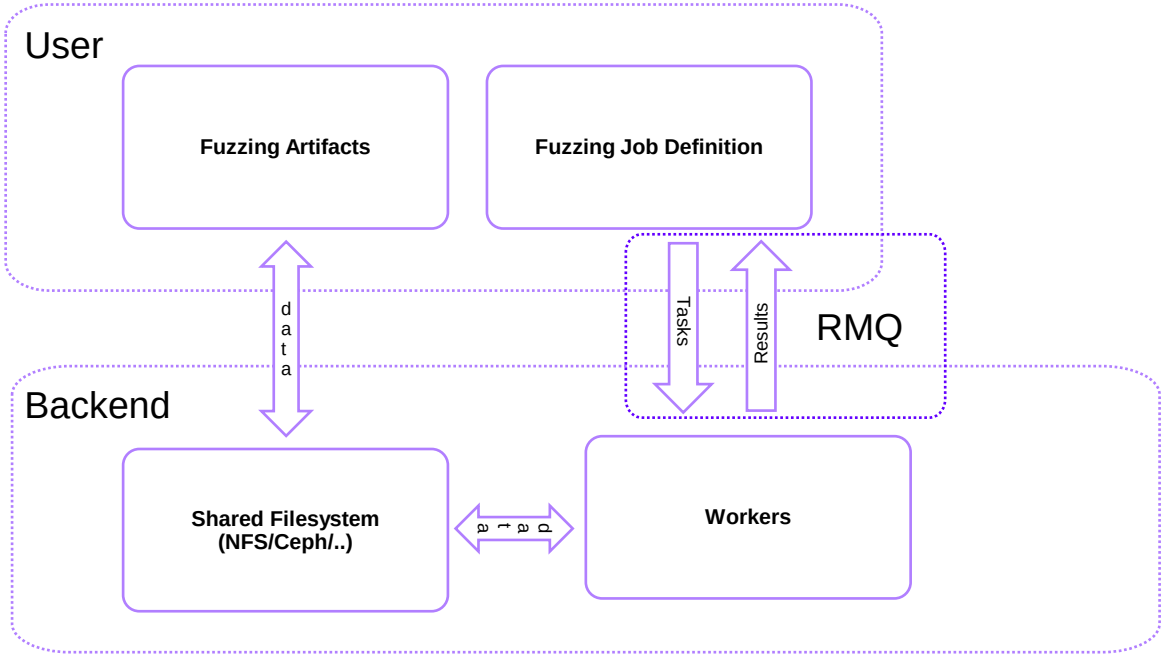
Architecture
Seed Generation

- **Strings from binary**
  Extract any string contained in the binary and leverage it as seed
  Practical method used in vulnerability hunting
  → does not need source code, fast

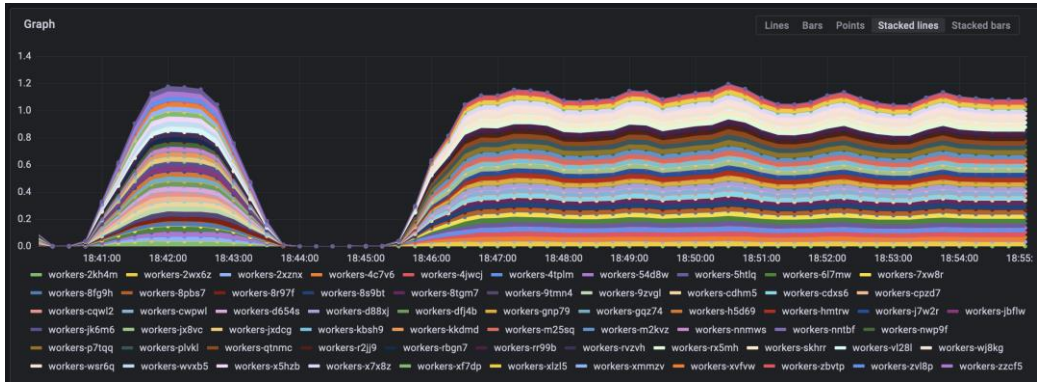
- **ESBMC**
  Leverages a BMC engine to generate relevant input for the source code
  → more precise seeds

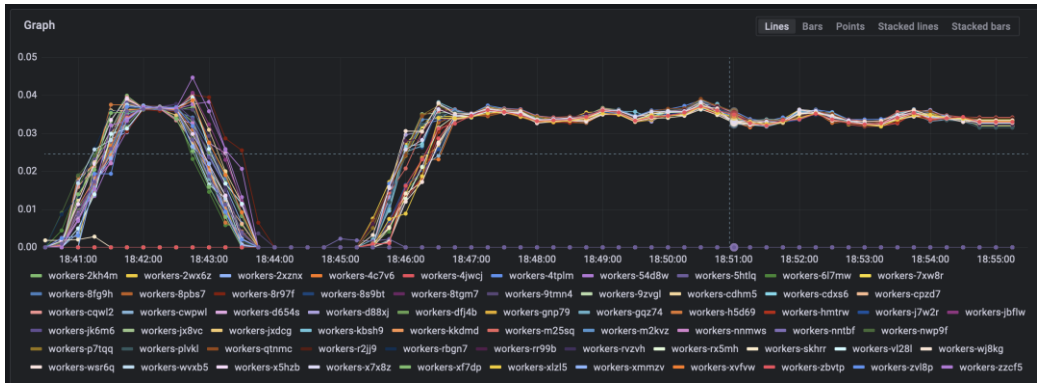


Current Workflow

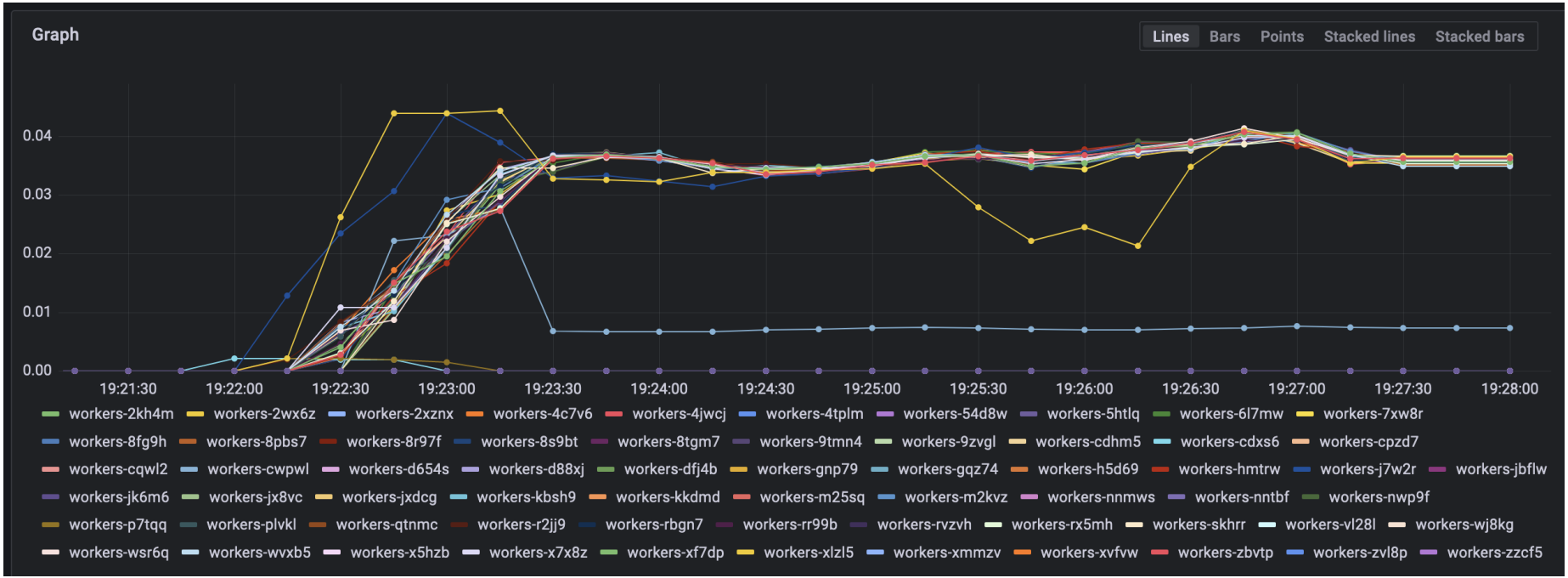


32 instances AFL fuzzing campaign

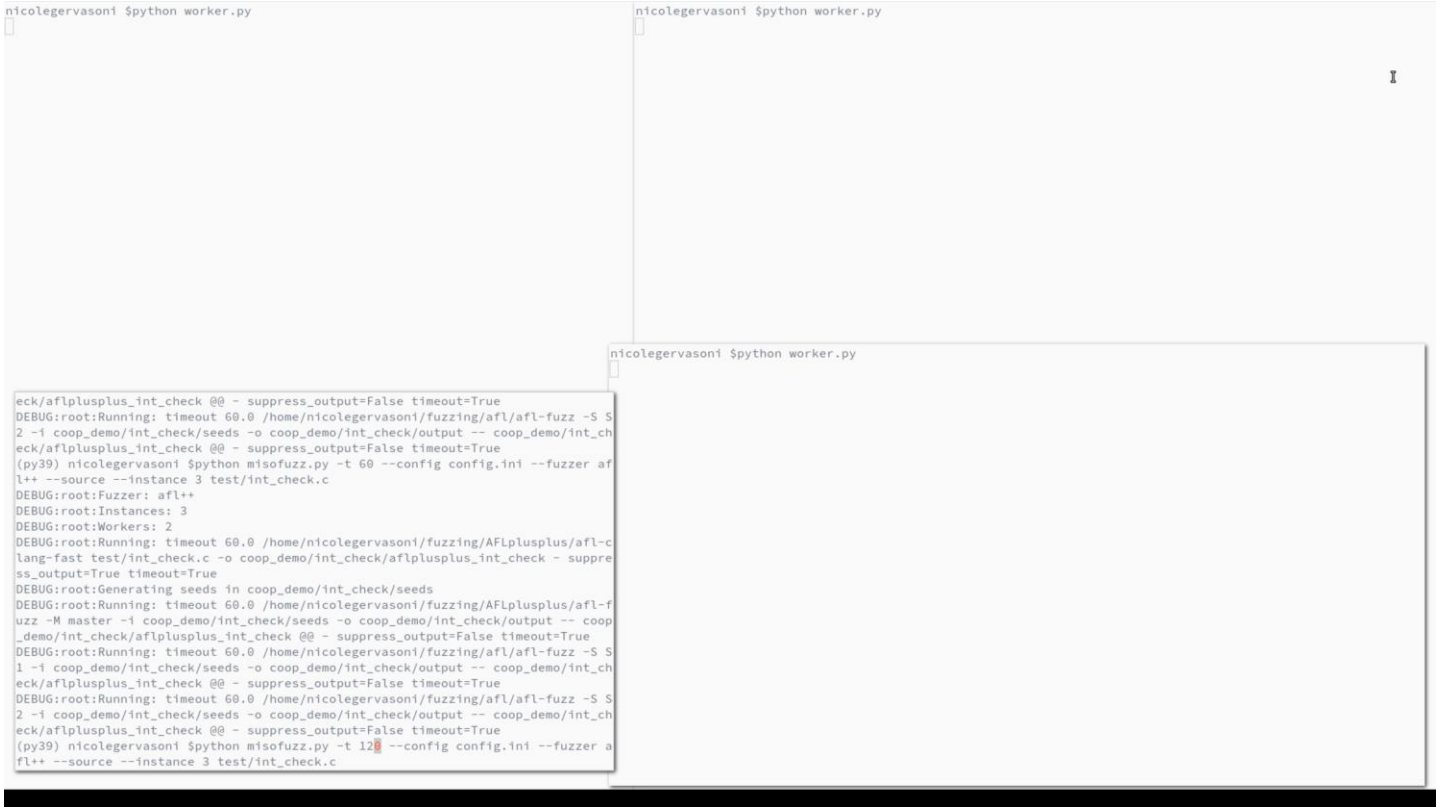




32 instances AFL+SymCC fuzzing campaign



DEMO


nicolegervasoni $python worker.py
nicolegervasoni $python worker.py
nicolegervasoni $python worker.py
eck/aflplusplus_int_check @@ - suppress_output=False timeout=True
DEBUG:root:Running: timeout 60.0 /home/nicolegervasoni/fuzzing/afl/afl-fuzz -S S
2 -i coop_demo/int_check/seeds -o coop_demo/int_check/output -- coop_demo/int_ch
eck/aflplusplus_int_check @@ - suppress_output=False timeout=True
(py39) nicolegervasoni $python misofuzz.py -t 60 --config config.ini --fuzzer af
l++ --source --instance 3 test/int_check.c
DEBUG:root:Fuzzer: afl++
DEBUG:root:Instances: 3
DEBUG:root:Workers: 2
DEBUG:root:Running: timeout 60.0 /home/nicolegervasoni/fuzzing/AFLplusplus/afl-c
lang-fast test/int_check.c -o coop_demo/int_check/aflplusplus_int_check - suppre
ss_output=True timeout=True
DEBUG:root:Generating seeds in coop_demo/int_check/seeds
DEBUG:root:Running: timeout 60.0 /home/nicolegervasoni/fuzzing/AFLplusplus/afl-f
uzz -M master -i coop_demo/int_check/seeds -o coop_demo/int_check/output -- coop
_demo/int_check/aflplusplus_int_check @@ - suppress_output=False timeout=True
DEBUG:root:Running: timeout 60.0 /home/nicolegervasoni/fuzzing/afl/afl-fuzz -S S
1 -i coop_demo/int_check/seeds -o coop_demo/int_check/output -- coop_demo/int_ch
eck/aflplusplus_int_check @@ - suppress_output=False timeout=True
DEBUG:root:Running: timeout 60.0 /home/nicolegervasoni/fuzzing/afl/afl-fuzz -S S
2 -i coop_demo/int_check/seeds -o coop_demo/int_check/output -- coop_demo/int_ch
eck/aflplusplus_int_check @@ - suppress_output=False timeout=True
(py39) nicolegervasoni $python misofuzz.py -t 120 --config config.ini --fuzzer a
fl++ --source --instance 3 test/int_check.c

DEMO


nicolegervasoni $python worker.py
nicolegervasoni $python worker.py
nicolegervasoni $python worker.py
(py39) nicolegervasoni $

Further development
Components

- **Implement a distributed fuzzing framework, with ease of integration and modularity as priority goals**
    - This approach can improve the performance of the fuzzing process and increase the likelihood of discovering bugs [5].
    - Workload is distributed across multiple instances and the discovered seeds are shared
    - Different fuzzers can adopt various strategies to increase coverage and decrease computation time.

- **GUI / WebUI**
- **Telemetry-driven scheduling**
    - Based on job performances
- **On-Demand Scaling**
    - depending on the available hardware resources and the job backlog pressure.

Further development
Tool comparison

MISOFuzz

- Cooperative fuzzing
- Distributed scheduler leveraging FuSeBMC
- Communication over RabbitMQ
- AFL, AFL++, SYMCC, FuSeBMC, FUZZX, extendable
- Leverages Bounded Model Checking

EnFuzz [5]

- Ensamble fuzzing
- Distributed scheduler
- Custom protocol
- AFL, AFLFast,FairFuzz, libFuzzer, Radamsa and QSYM

CollabFuzz [1]

- Cooperative fuzzing
- Centralized scheduler
- Communication over ZeroMQ
- AFL, AFLFast, FairFuzz, QSYM, radamsa, Honggfuzz, and libFuzzer, extendable

References

[1] S. ̈Osterlund et al. Collabfuzz: A framework for collaborative fuzzing. EuroSec 2021.

[2] K. Alshmrany et al. Fusebmc: A white-box fuzzer for finding security vulnerabilities in c programs. FASE 2021.

[3] G. Klees et al. Evaluating fuzz testing. CCS 2018.

[4] M. Bohme et al. Fuzzing: On the exponential cost of vulnerability discovery. ESEC/FSE 2020.

[5] Y. Chen et al. Enfuzz: Ensemble fuzzing with seed synchronization among diverse fuzzers. In USENIX, 2019.

[6] M. R. Gadelhaet al. Esbmc 5.0: An industrial-strength c model checker. ASE 2018.

[7] Y. Li et al. A large-scale parallel fuzzing system. ICAIP 2018.

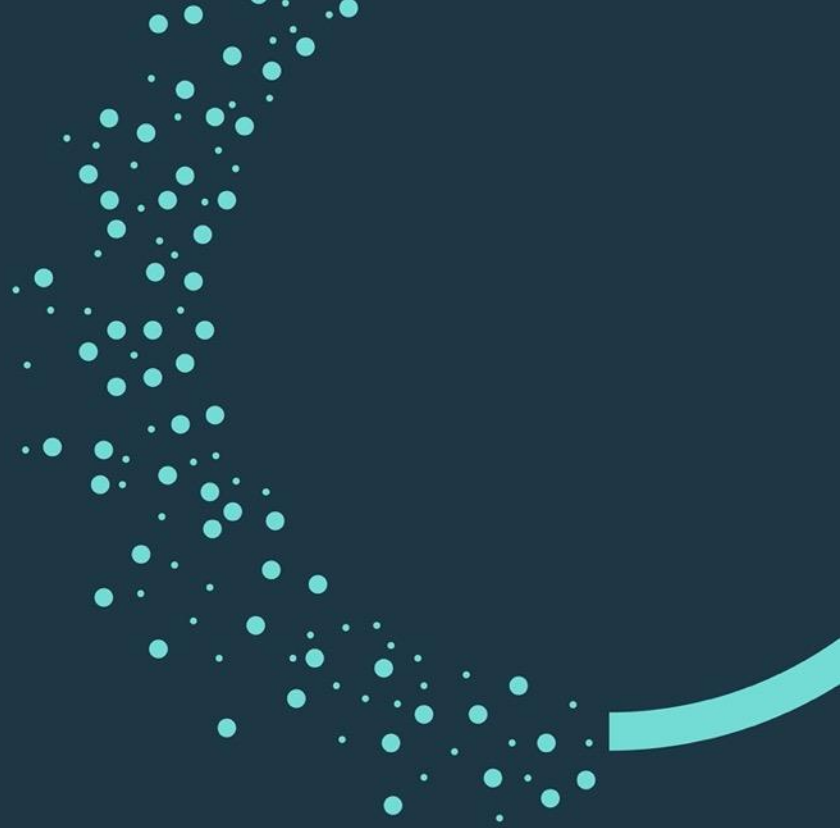
[8] J. Liang et al. Pafl: Extend fuzzing optimizations of single mode to industrial parallel mode. ESEC/FSE 2018.

[9] A. Fioraldi et al. AFL++: Combining incremental steps of fuzzing research. USENIX (WOOT 20), 2020.

[10] S. Poeplau and A. Francillon. Symbolic execution with SymCC: Don’t interpret, compile! USENIX, 2020.

[11] M. Zalewski, American Fuzzy Lop whitepaper. https://lcamtuf.coredump.cx/afl/

Technology Innovation Institute



tii.ae