Forkever: A framework for testing and exploiting programs

Bachelor Kolloquium

Jasper Rühl

Technical University of Munich
Department of Informatics
Chair of IT Security

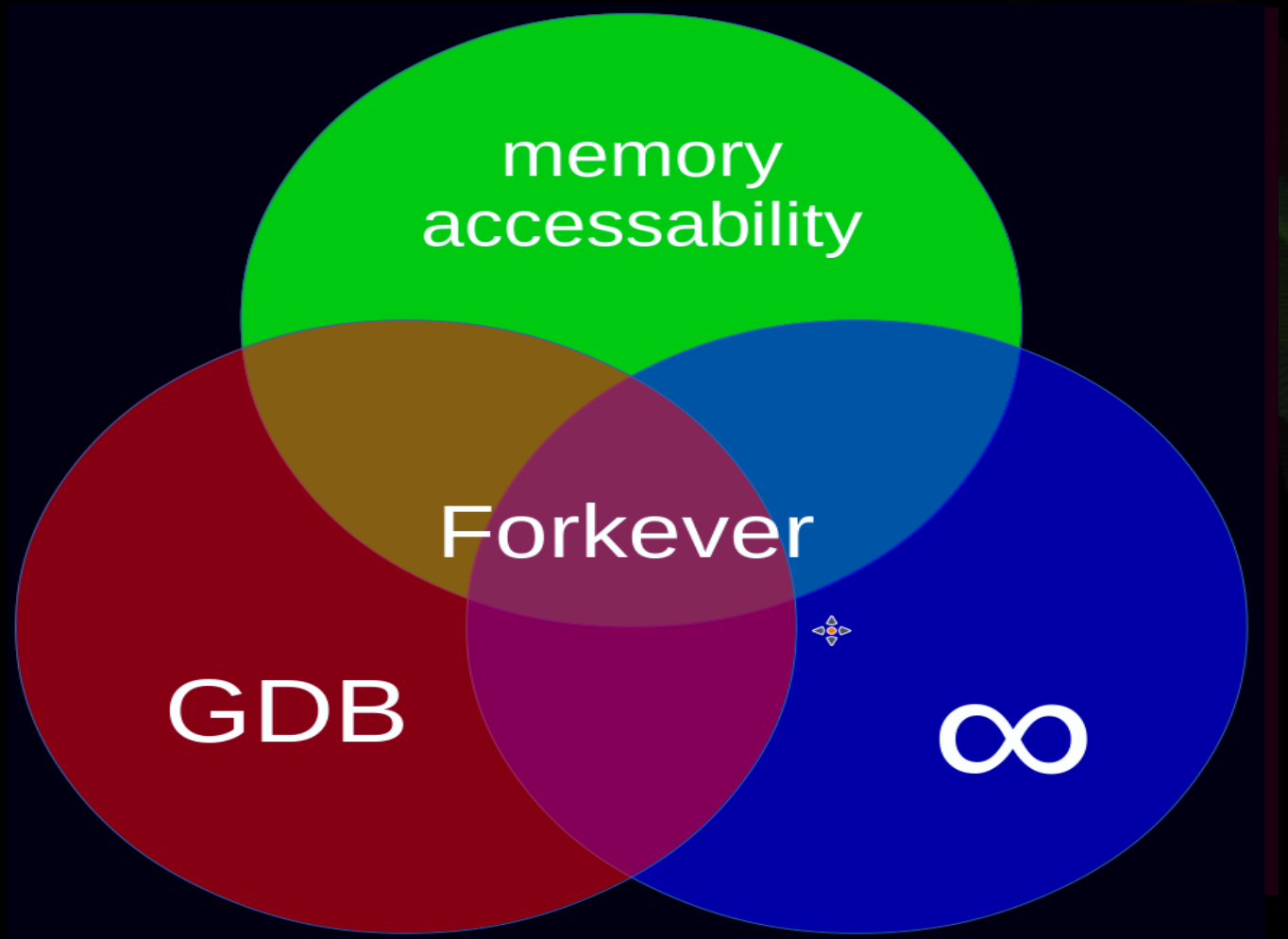
Background

- Ptrace
  - Set registers
  - Change memory
- Fork
  - Create copy of process

Motivation

- BX::heap_exploitation
- GDB is awesome
- Lots of script restarting
- x/50b 0x5555555240

Features



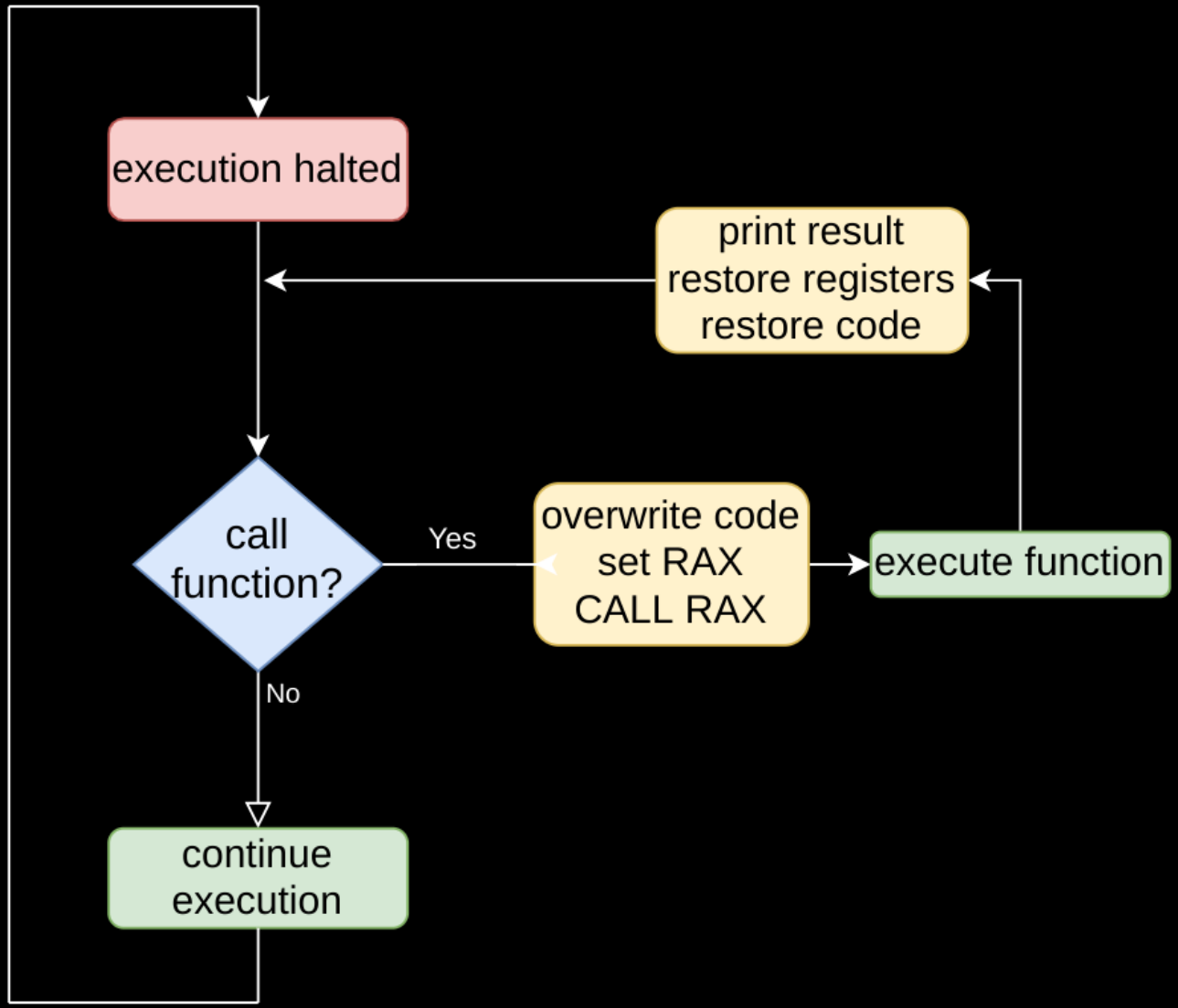
Features

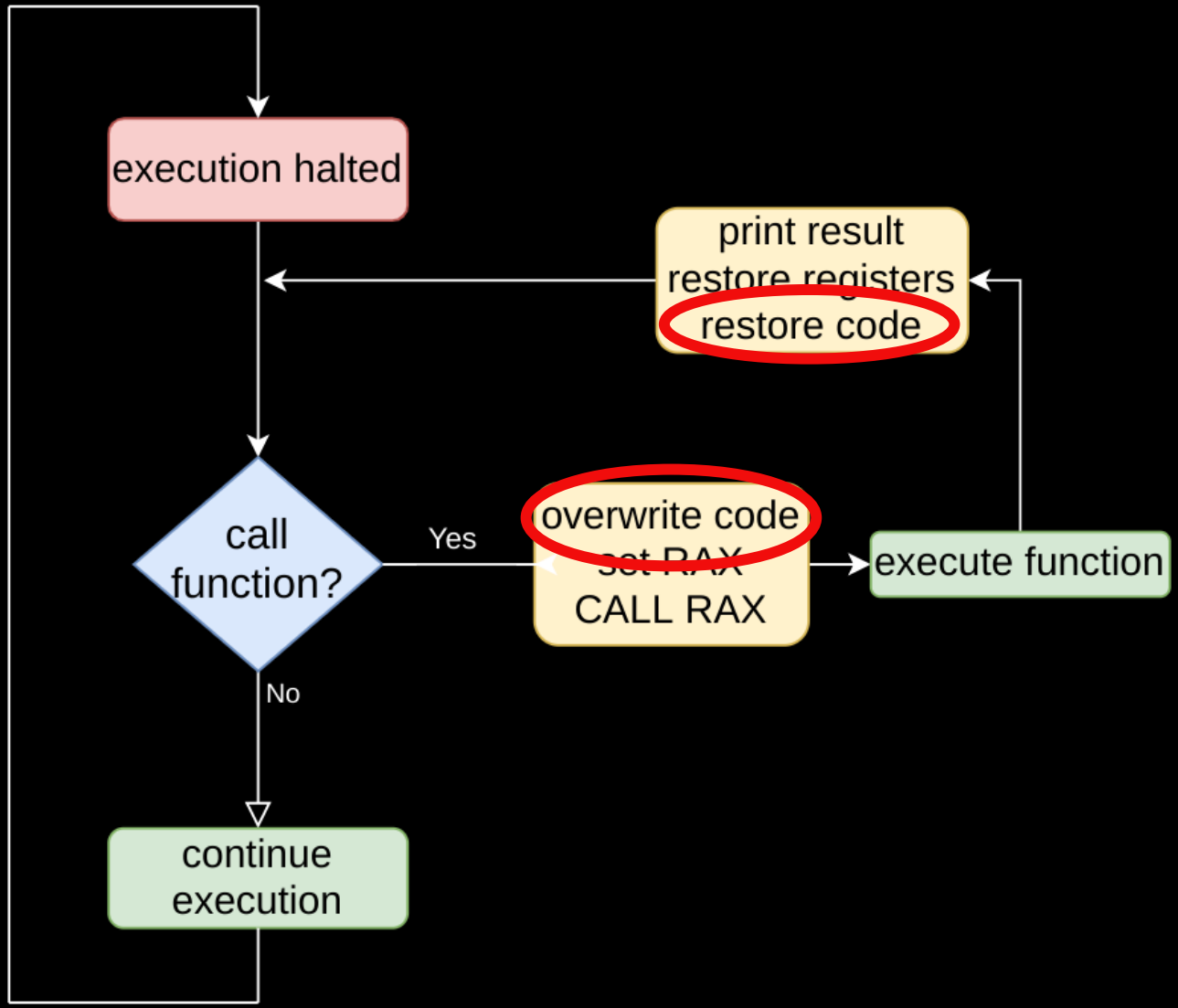
- GDB-like debugging
- Read and change memory with hexeditor
- Call arbitrary functions whenever
- Fork to backup process whenever

Call function

- Function address is known
- Execute function call
- Resume normally upon return
  - Return address on the stack?
  - Registers?

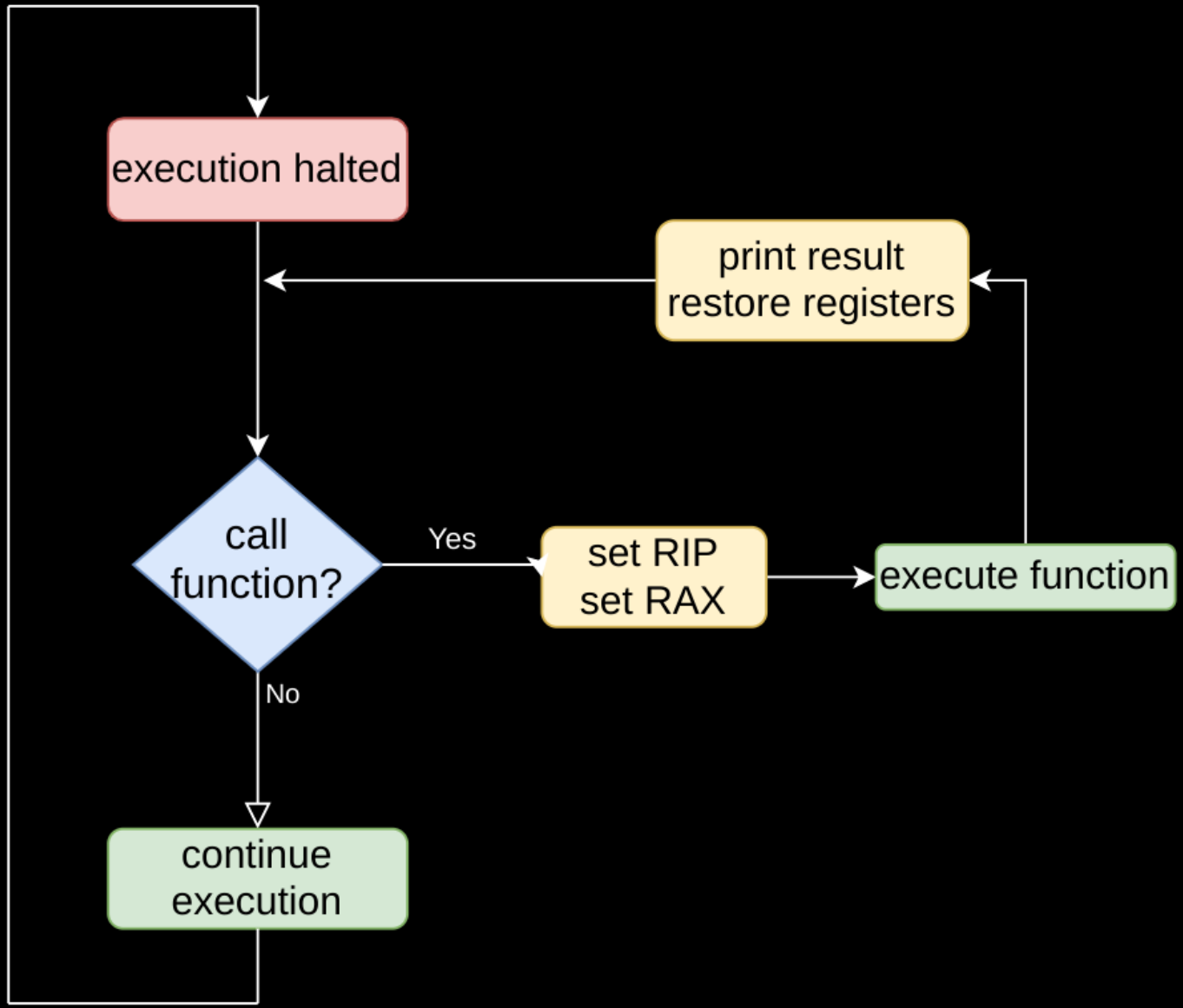
➔ CALL RAX; BREAK;

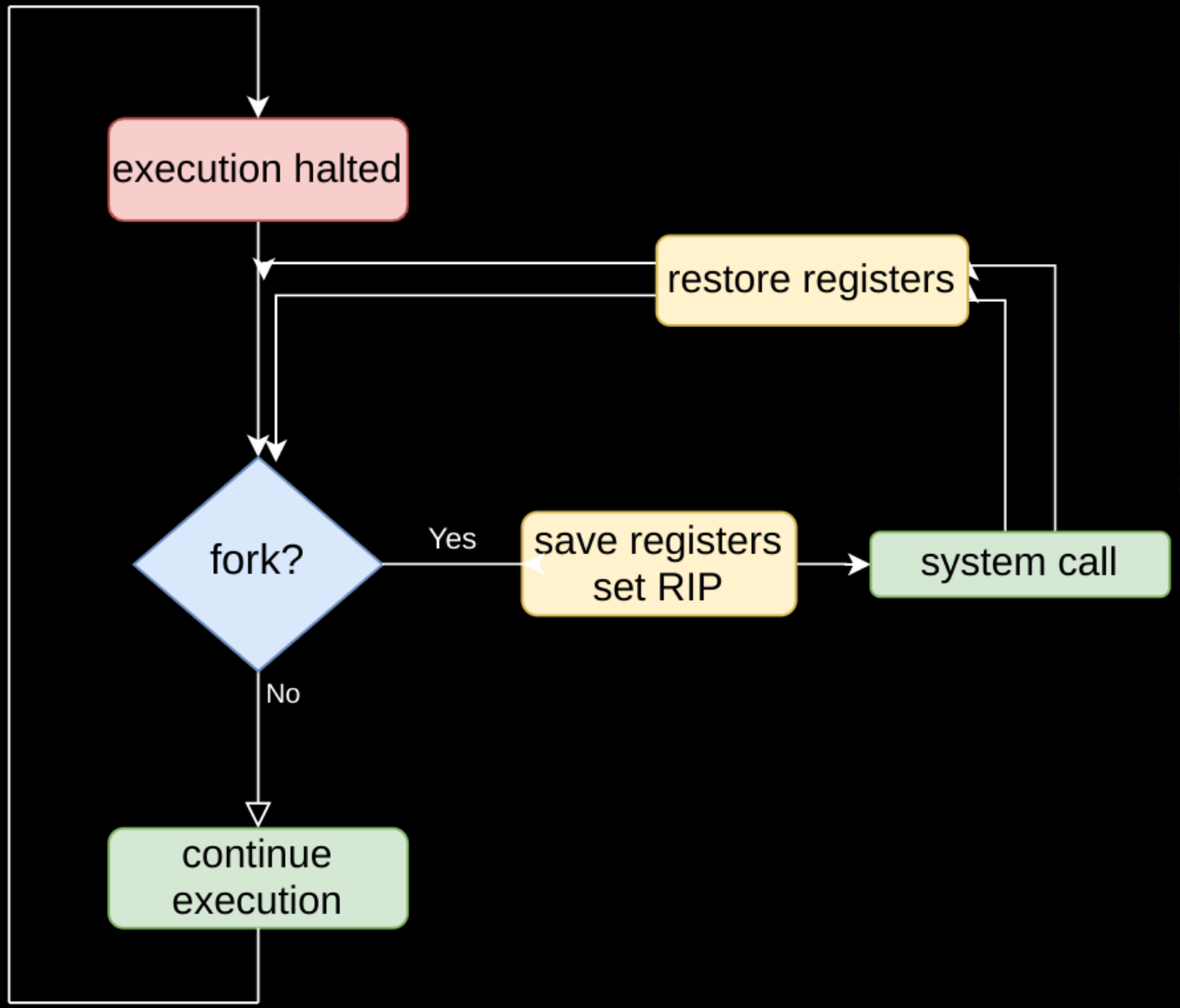

execution halted
print result
restore registers
restore code
call function?
Yes
overwrite code
set RAX
CALL RAX
execute function
No
continue execution


execution halted
print result
restore registers
restore code
call function?
Yes
overwrite code
set RAX
CALL RAX
execute function
No
continue execution

new memory page

- Executable
- Inserted instructions go here
- No memory changes necessary
- Just alter registers


execution halted
print result
restore registers
call
function?
Yes
set RIP
set RAX
execute function
No
continue
execution


execution halted
restore registers
fork?
Yes
save registers
set RIP
system call
No
continue
execution

Ignore Signal

- Upon termination, processes notify their parents
- Signal is only delivered when process is running
  - How to receive signal without changing state?
- Infinite loop

Demo

Thank you

STDIN Pipe

- Filedescriptors are shared
- Write once → consume once
  - Consume twice?

- Cache STDIN per process, forward upon request
  - Trace all system calls

Fuzzing

- Test program with random input repetitively
- Modification: find accepting word for automaton

- Start with 1 letter, mutate/extend input, evaluate

- 2 different evaluation strategies

ForkFuzzer

- Save state after each transition
- Dictionary: Prefixes → Processes

- Fast transitions → very slow

- Transitions are not repeated
- Slow transitions → better than normal approach

Calling arbitrary functions

- Save state
- Put address of the function into RAX
- Call RAX
- Restore state when finished

- Where to write this code?

Fork

- Same as calling a function
- Edge case: process is about to perform system call
- Ptrace attaches new process
- Forkever switches to new process