AFL*: a simple approach to fuzzing stateful systems

PhD Workgroup - 30th of April, Leiden University

Cristian Daniele, Radboud University - Netherlands

About me

- Third-year PhD at Radboud University, Netherlands
- Doing research in fuzzing (mostly stateful fuzzing)
- Interested in state-model learning

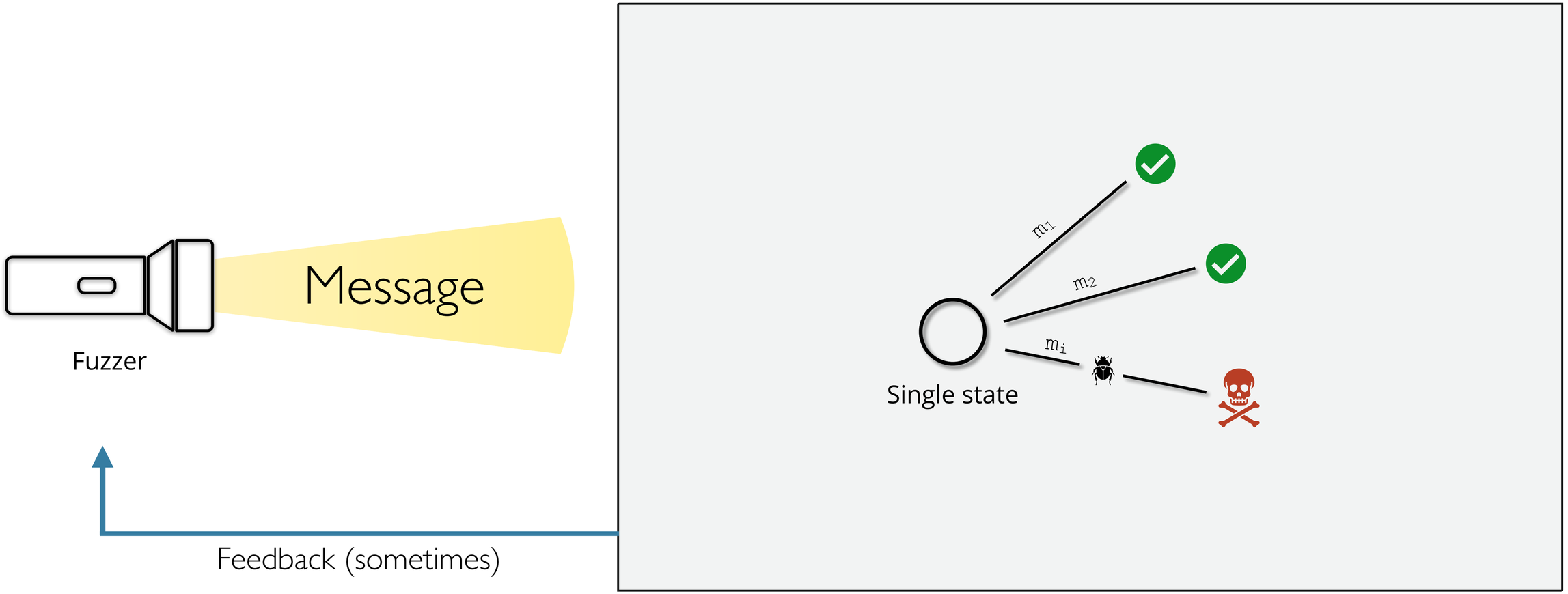
Contact me! :)

Conclusions

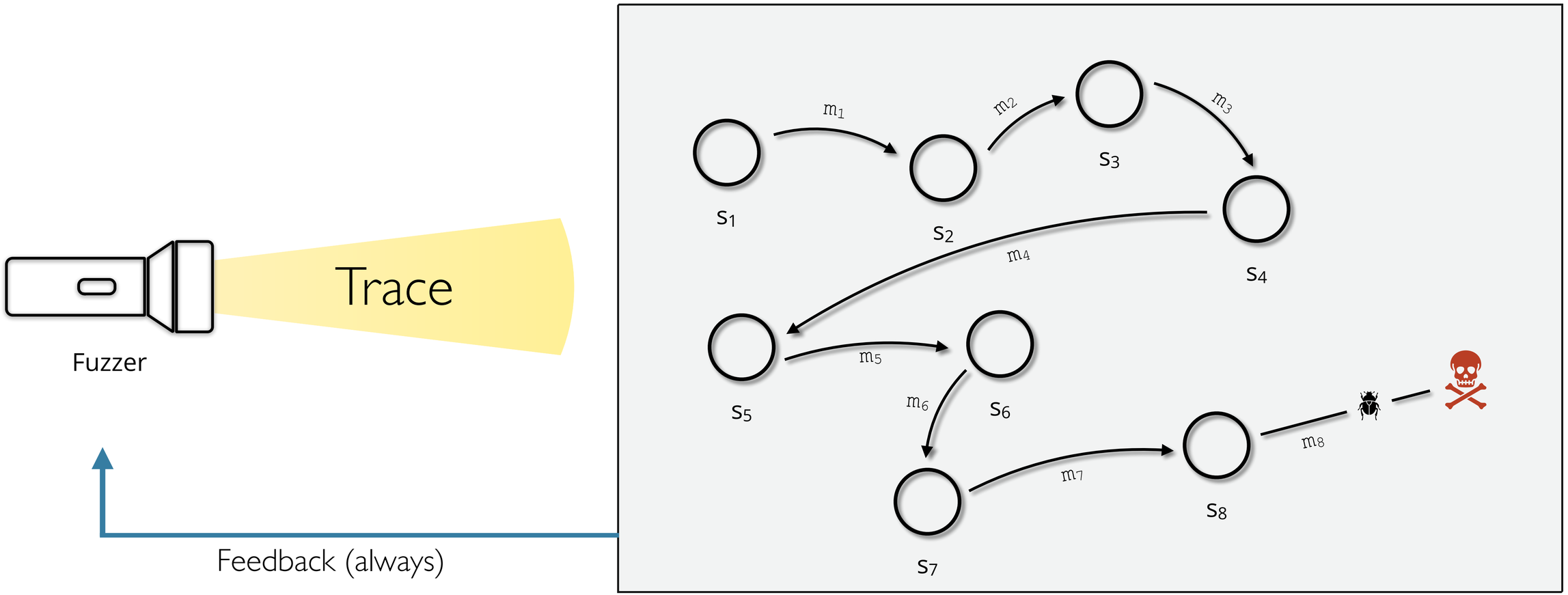
	LightFTP				Pure-ftdb			
	Edge coverage	State coverage	Execution per second	Messages sent	Code coverage	State coverage	Execution per second	Messages sent
AFL*	*25,28%*	*100%*	*~ 63k*	*25k*	*18,72%*	*100%*	*~ 31k*	*1.5 million*
AFLNet	*0,44%*	*80%%*	*~ 9*	*400*	*0,78%*	*100%*	*~ 27*	*1.700*

Comparison between AFLNet and AFL*

Stateless fuzzing



Stateful fuzzing



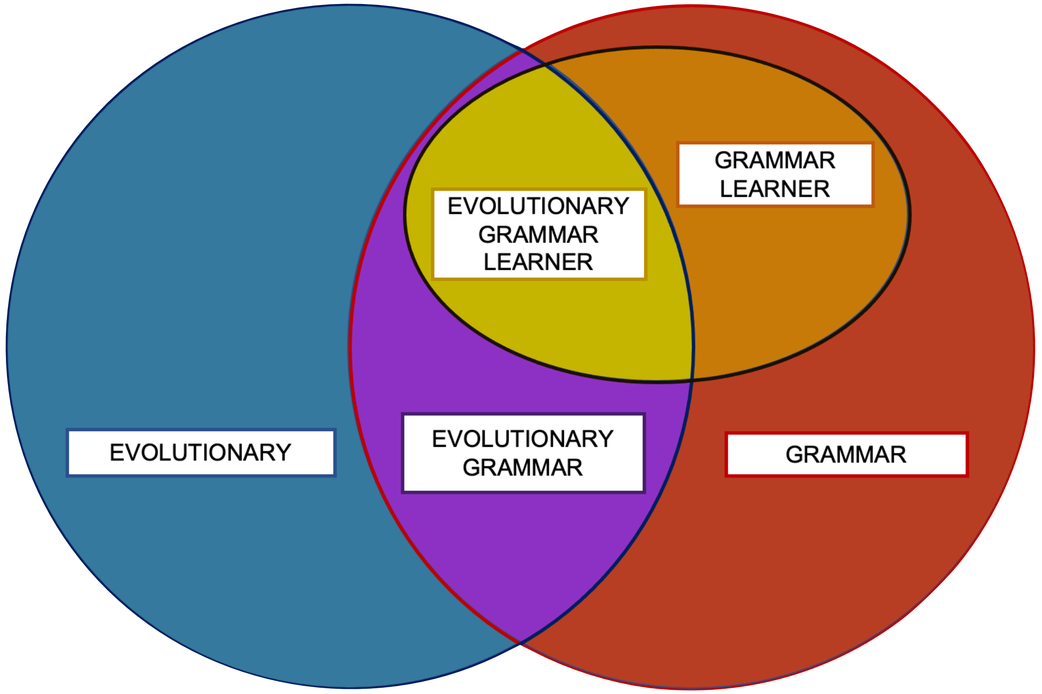
System Under Test (e.g., FTP protocol)

A few stateful fuzzers (not too many)

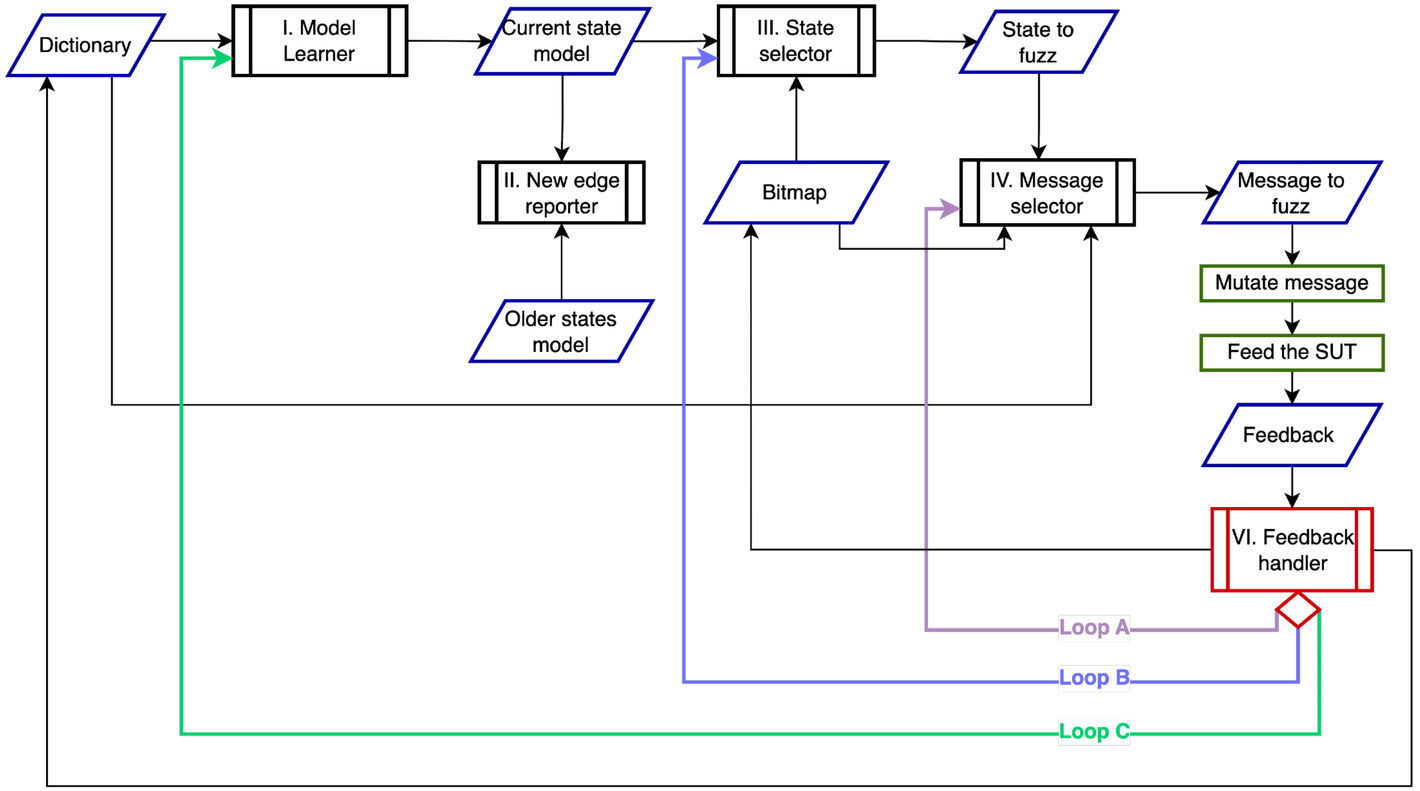
Seven relevant categories for fuzzer of stateful systems:

1. Evolutionary fuzzers
2. Grammar-Based fuzzers
3. Evolutionary Grammar-Based Fuzzers
4. Grammar Learner Fuzzers
5. Evolutionary Grammar-Learner Fuzzers
6. Machine Learning-Based Fuzzers
7. Man-in-the-middle Based Fuzzers

Fuzzers for Stateful Systems:
Survey and Research Directions



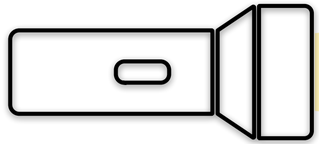
Initial idea: LearnFuzz




```
Start:  dict = init_dictionary()
Loop A: currentStateModel= StateLearner(dict)
        async(alert = NewEdgeReporter(stateModel,olderStateModels[]))
Loop B: stateToFuzz = StateSelector(currentStateModel,bitmap)
Loop C: messageToFuzz = MessageSelector(stateToFuzz,dict)
        systemFeedback=StatelessFuzzer(messageToFuzz)
        if(systemFeedback.newBranch==true){
                updateGeneralBitmap()
        }
        if(systemFeedback.newResponse==true){
                AddMessageDictionary()
        }
        nextStep=FeedbackHandler(systemFeedback)
        switch(nextStep) {
                case "improve state model":
                        goto Loop C
                        break
                case "pick new state":
                        goto Loop B
                        break
                case "pick new message":
                        goto Loop A
                        break
        }
```

Baseline: AFL*!

List of commands



AFL*

Multiple ‘single’ messages

```
int main(){
  int state=0;
  …
  START_SPECIAL_LOOP {
    command=read_command()
    …
    output=execute(command)
  } END_SPECIAL_LOOP
  …
}
```

Stateful system

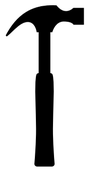
Devised only for stateless systems!

Easy but effective!

C.1: Exploring the state model in depth

AFL++’s persistent mode

C.2: Mutating single messages and traces

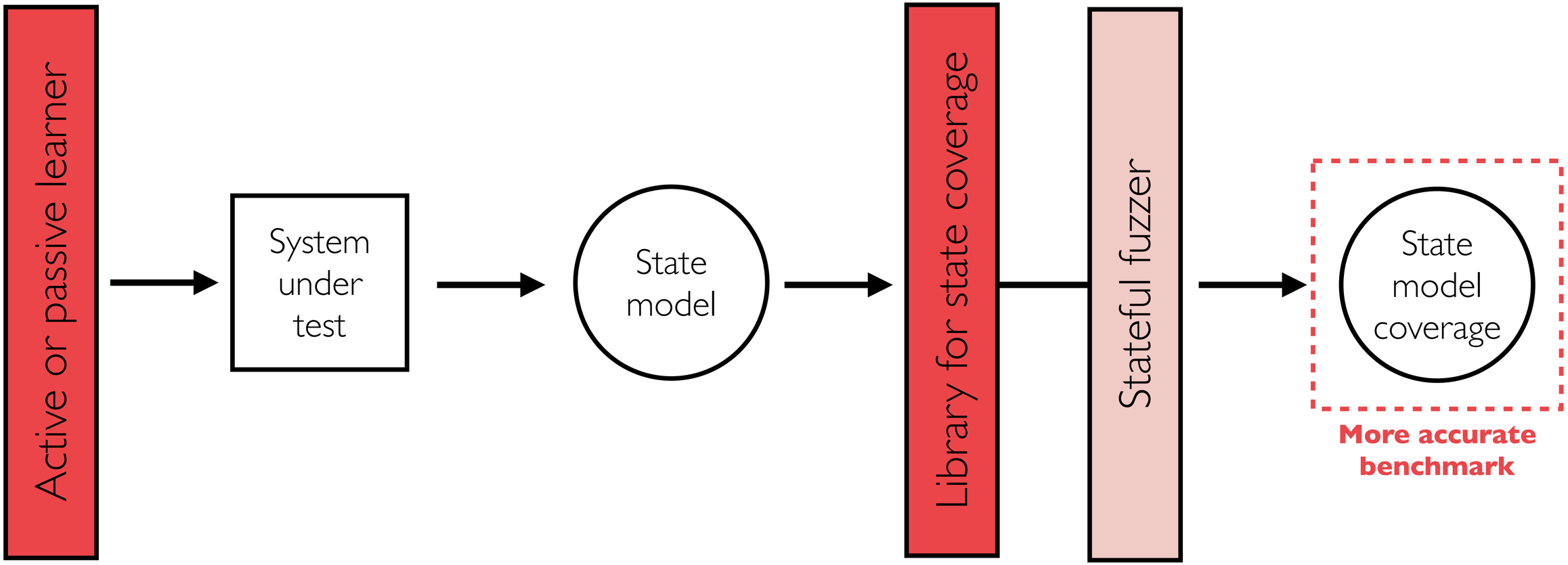


Custom mutator

C.3: Sending mutated messages over a TCP/IP socket

Preeny

But how many states can AFL* cover...?



All of them!

	LightFTP		Pure-ftdb	
	AFL*	AFLNet	AFL*	AFLNet
State 1	*14903*	*167*	*7390*	*304*
State 2	*5099*	*84*	*2460*	*210*
State 3	*3161*	*160*	*1328*	*463*
State 4	*2227*	*0*	*1031*	*150*

Comparison state coverage between AFLNet and AFL* on LightFTP

Biggest impact on performance?

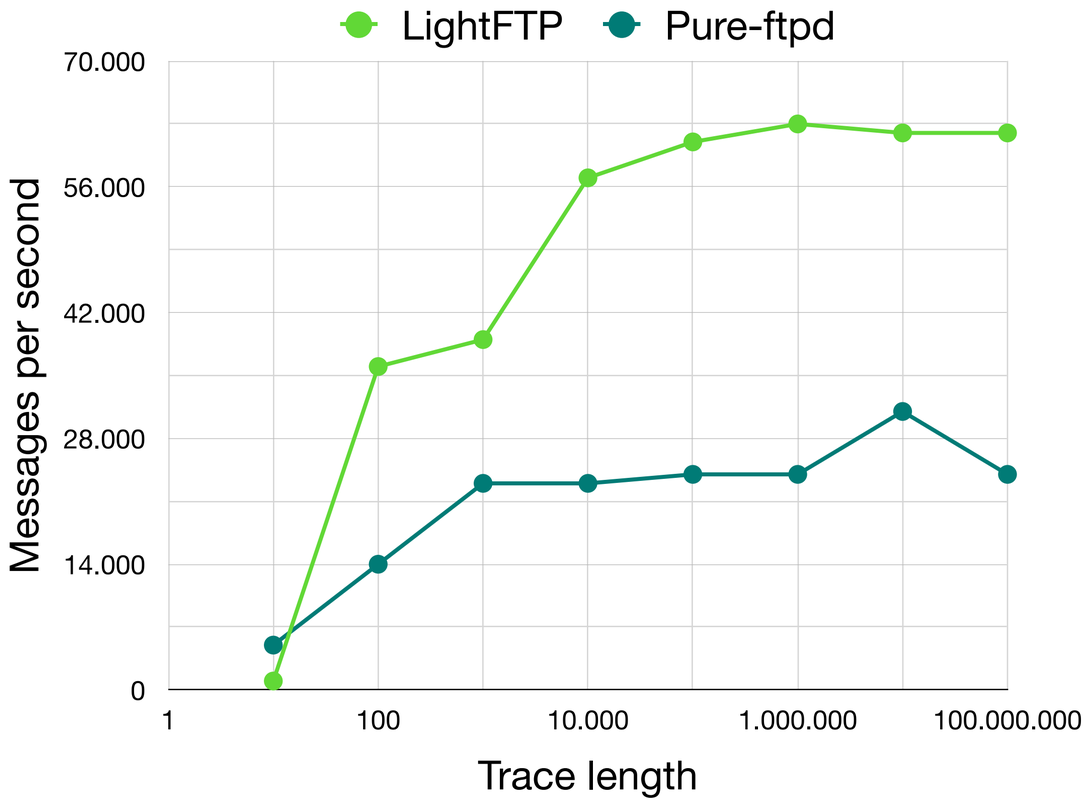
Overriding network calls?

AFL* mode	Execution per seconds
With network calls	*20*
Without network calls	*193*
With long traces	*44k*

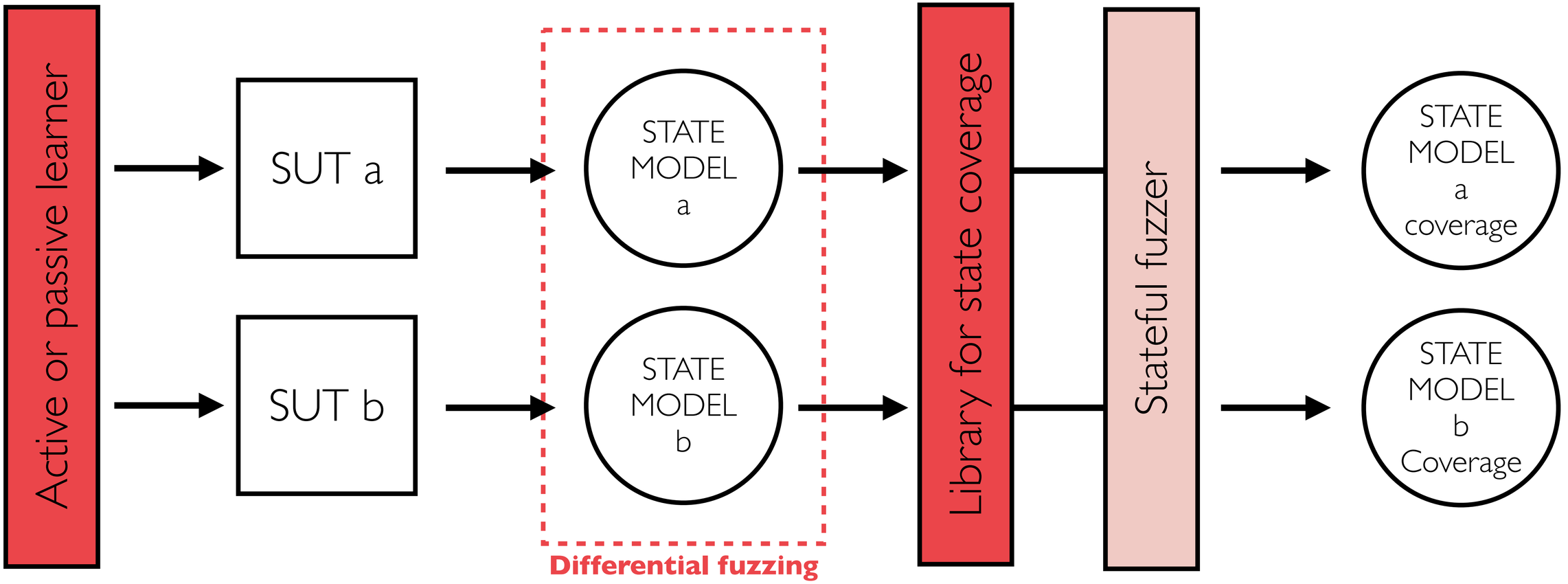
Comparison AFL* modes

On the (in)Efficiency of Fuzzing Network Protocols

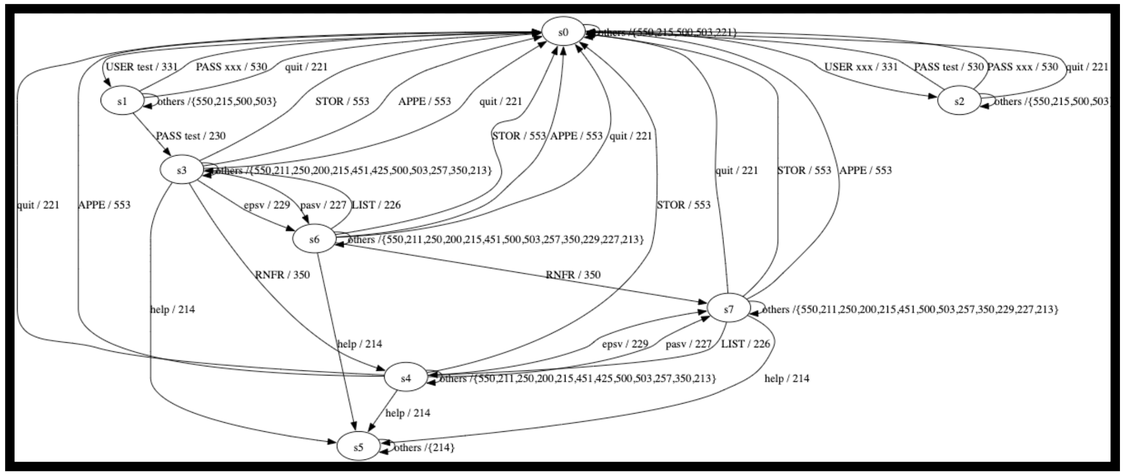
Sending long traces?



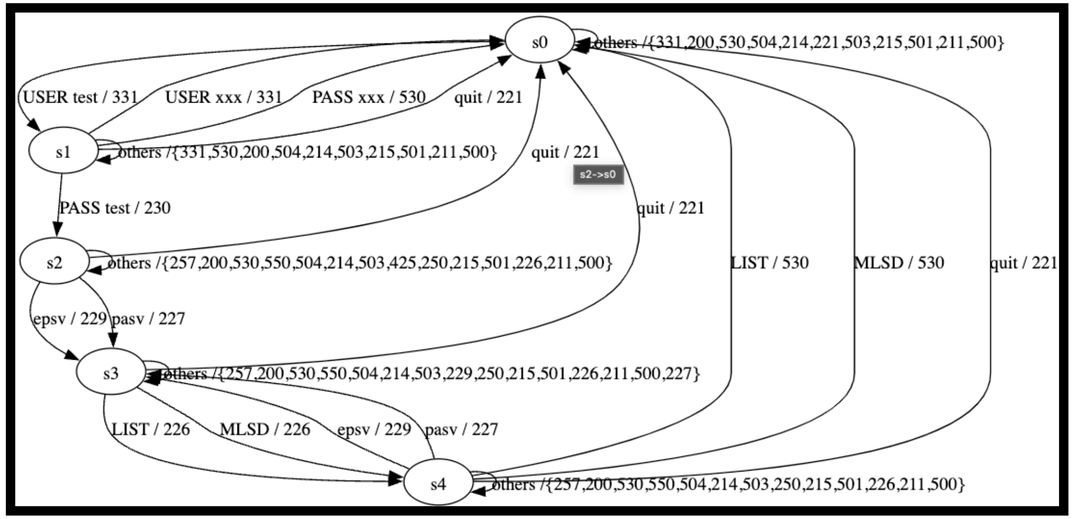
Since I already have the state models...



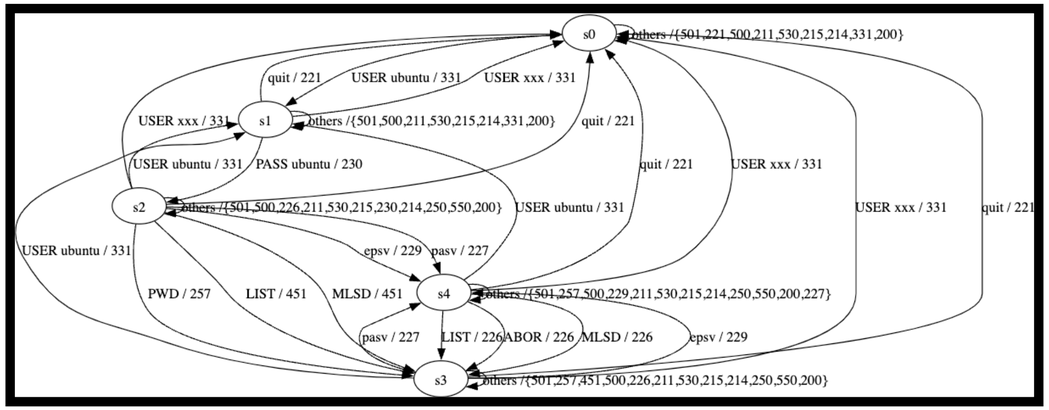
Differential testing!



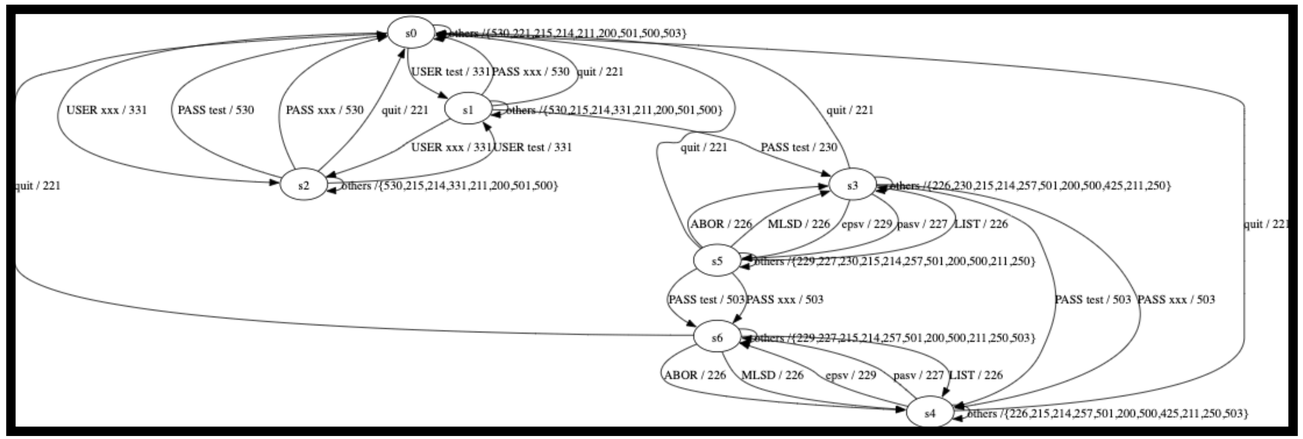
bftpd



Pure-FTPd

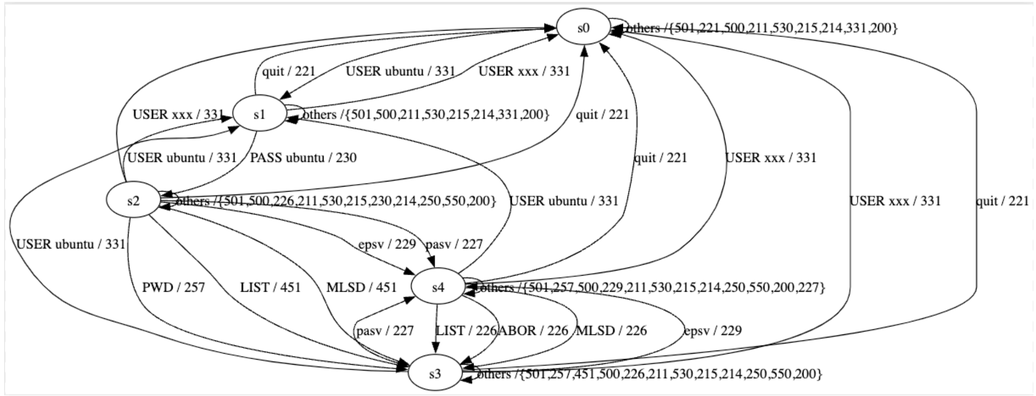


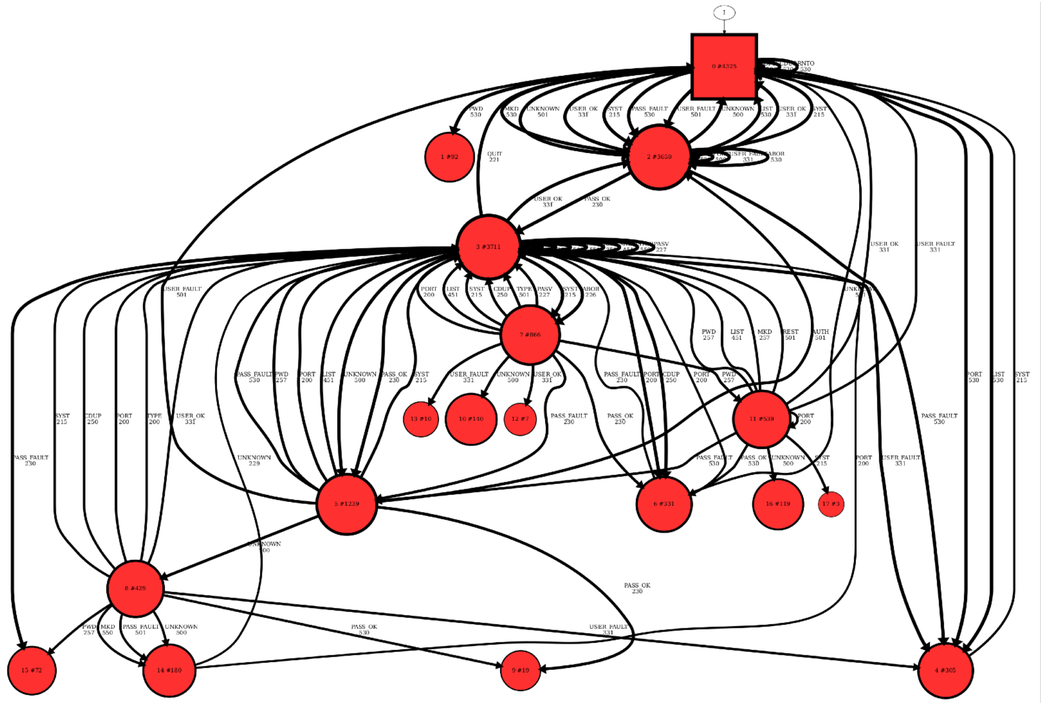
LightFTP



ProFTPd

Active or passive learning?





LightFTP state model inferred via passive learning*

* Jermo Vanoort's work

Take away slide

1. Stateful fuzzing is challenging!
2. Persistent mode — originally devised to fuzz state**less** systems — is extremely useful to the fuzz stateful ones
3. Different implementations of the same protocol might have different state models
4. Not clear why active and passive learners generate different state models

For questions, ideas or suggestions, contact me!

Contact me! :)