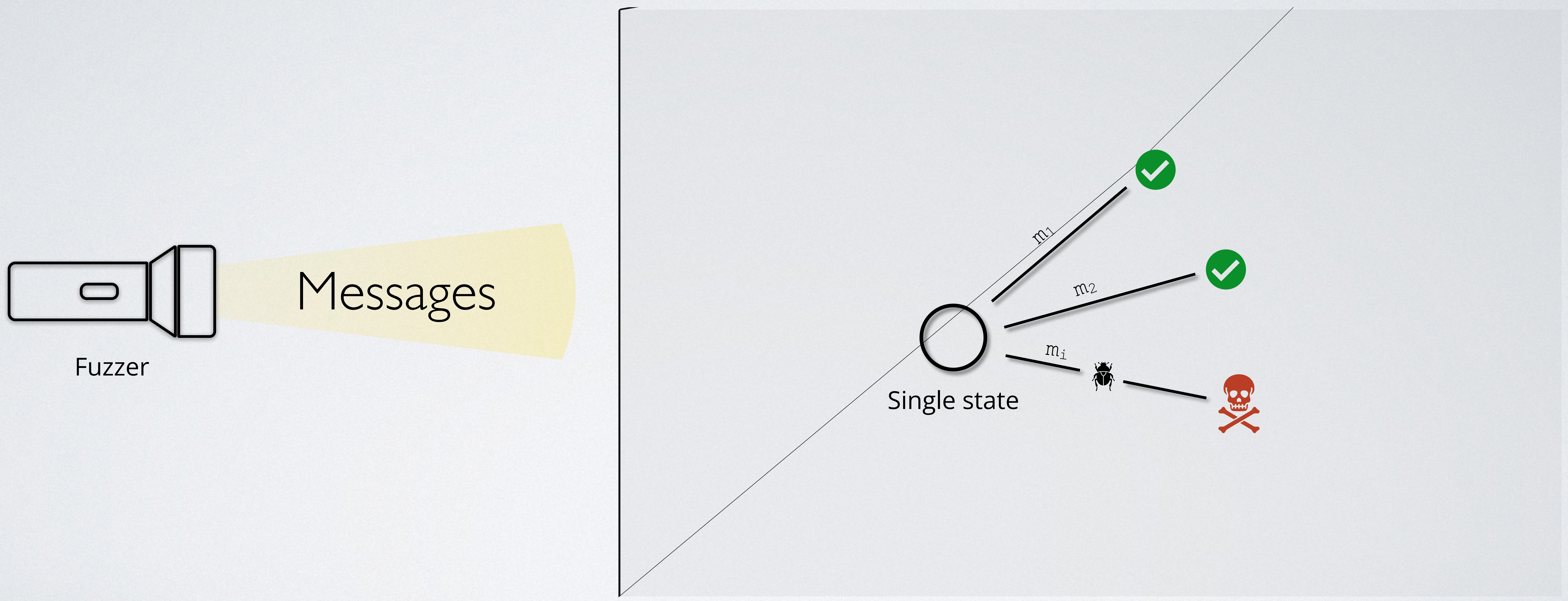
Stateful fuzzing: challenges, new approaches and future directions

INTERSCT Conference - 23rd of May, The Hague

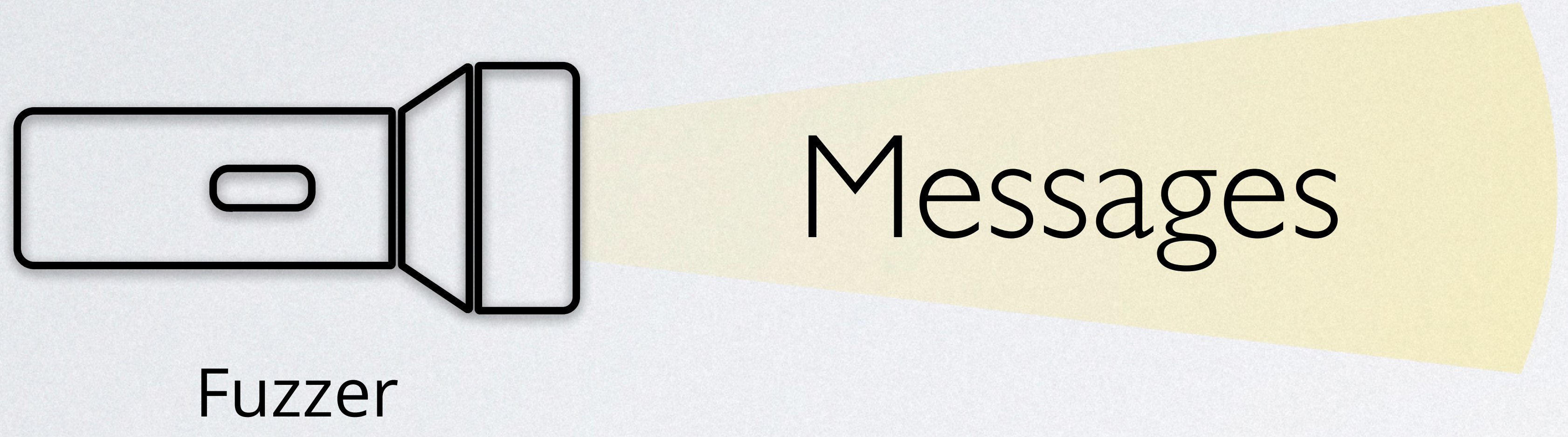
Cristian Daniele, Radboud University

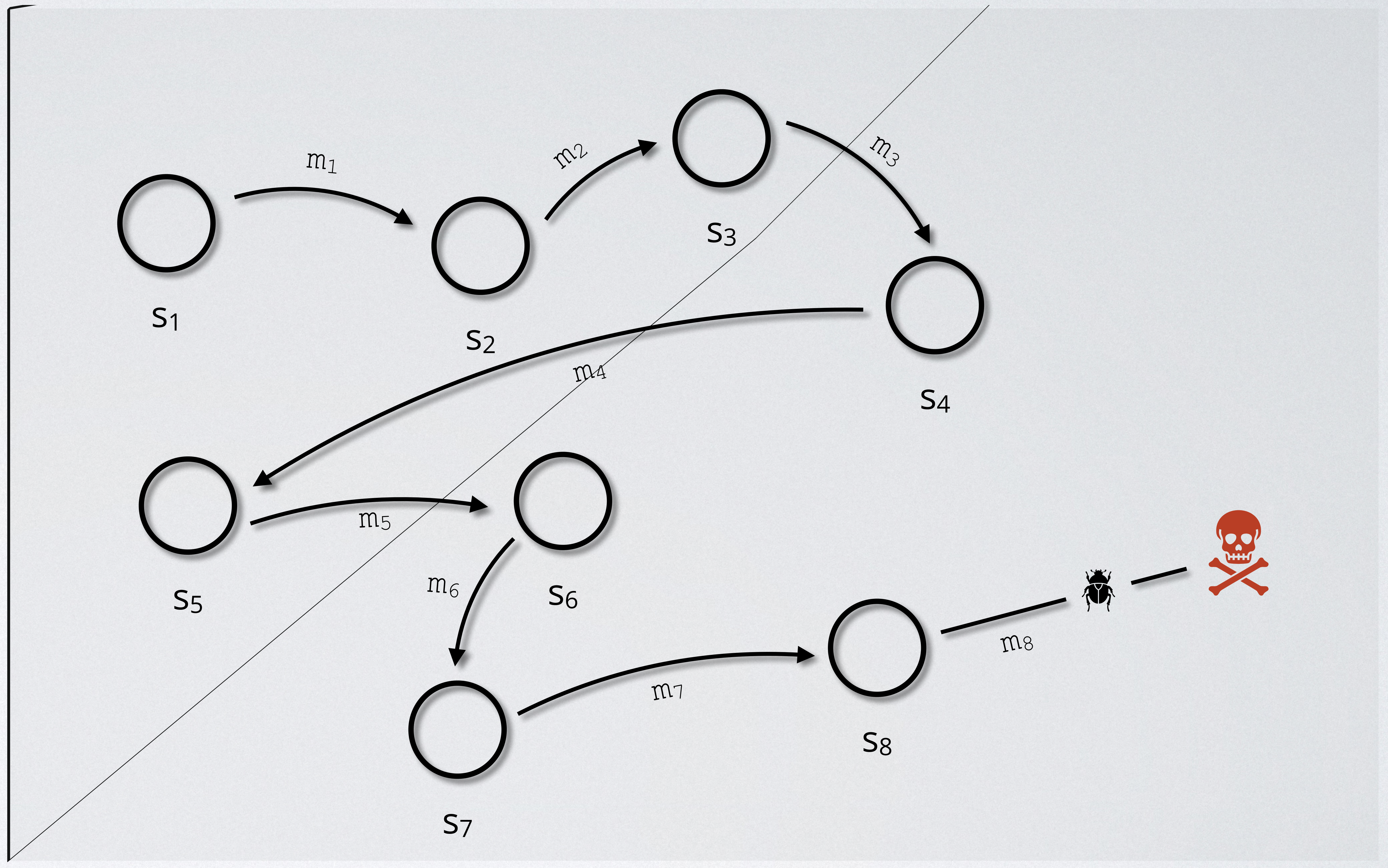
How does stateless fuzzing work?



System Under Test (e.g. image processing software)

How does stateful fuzzing work?



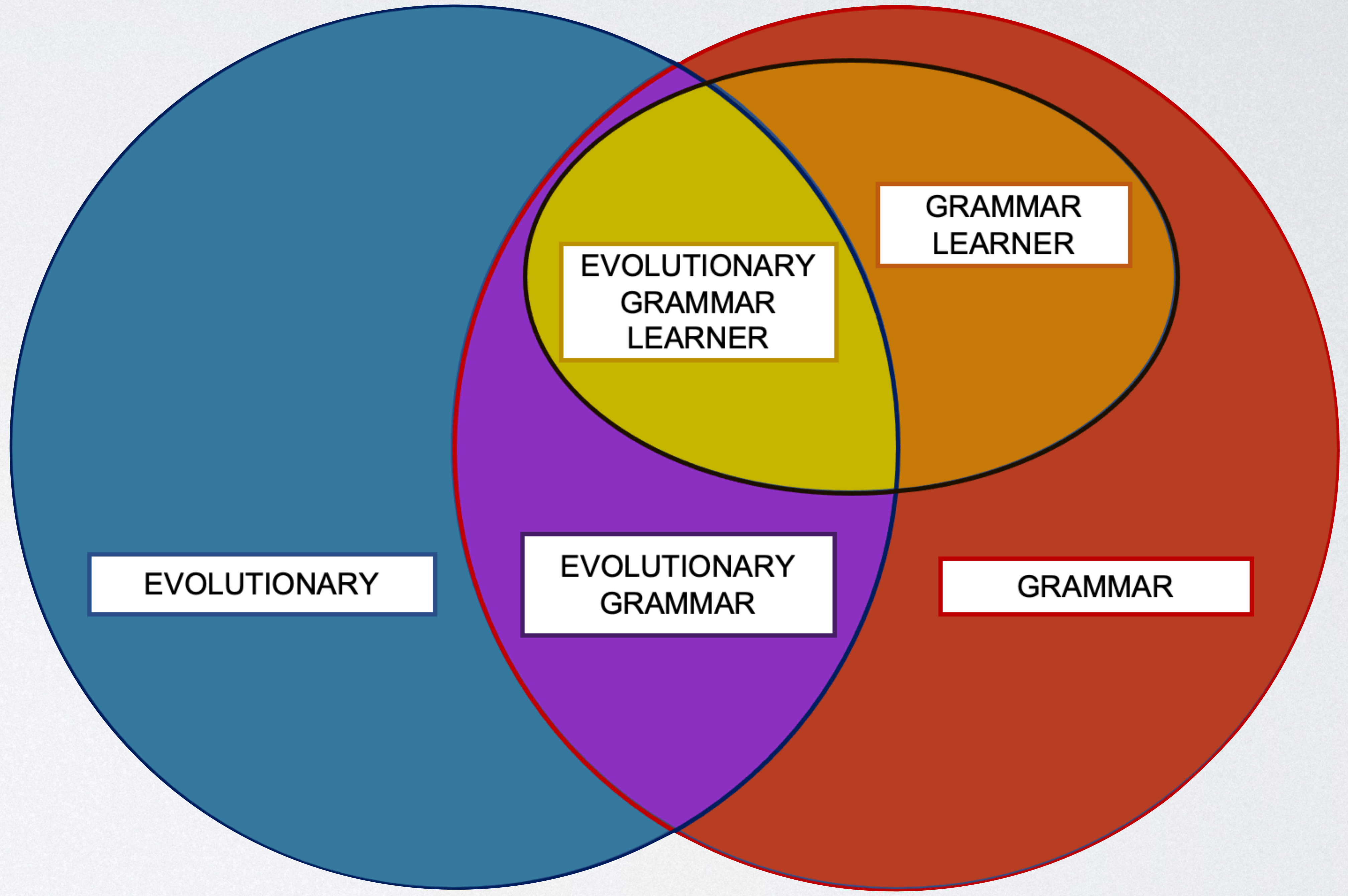


System Under Test (e.g. FTP)

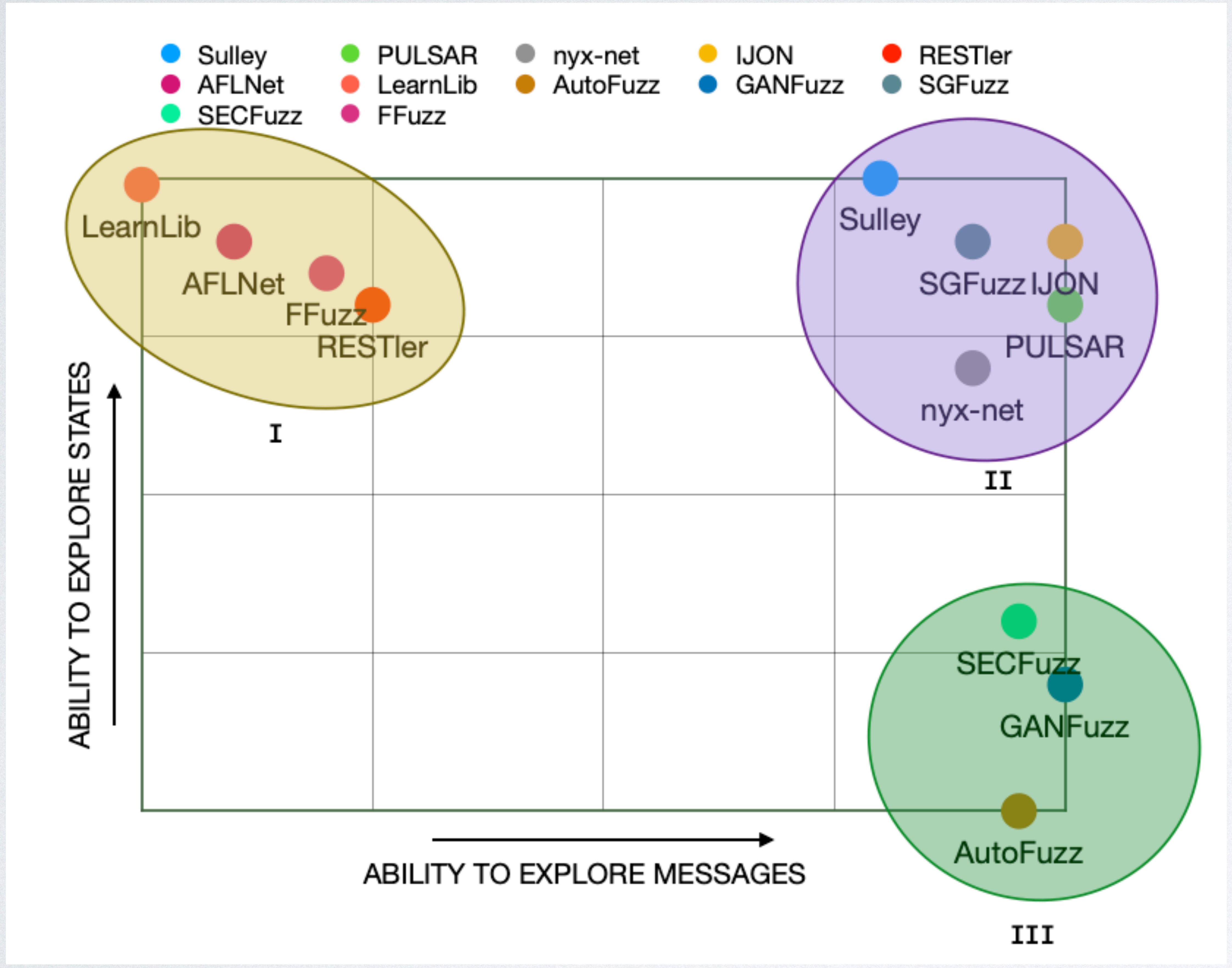
Families of stateful fuzzers

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**



Many approaches to deal with the statefulness



How fuzzers deal with the statefulness of a system:

1. Active Learning
2. Passive learning
3. Grammar
4. White-box approach

Overview of stateful fuzzers

FUZZER	Type of the fuzzer	Feedback system (i)	Feedback system (ii)	Input needed	Based on
nyx-net [16]	Evolutionary	Coverage	N/A	1.Target binary 2.Protocol specification 3.Seed inputs (optional)	AFL
FitM fuzzer [1]	Evolutionary	Coverage	N/A	1.Client binary 2.Server binary 3.Seed inputs	AFL
SNPS fuzzer [2]	Evolutionary	Coverage	N/A	1.Target binary 2.Seed input	AFL
Chen et al. [3]	Evolutionary	1.Coverage 2.Branches	N/A	1.Target binary 2.Seed input	1.AFL 2.Manual code annotation
SGFuzz [30]	Evolutionary	1.Coverage 2.Variables	N/A	1.Target binary 2.Seed input	1.AFL 2.Automatic code annotation
IJon [1]	Evolutionary	1.Coverage 2.Variables	N/A	1.Target source code	1.AFL 2.Manual code annotation
Peach [5]	Grammar based	N/A	N/A	1.Grammar	-
SNOOZE [6]	Grammar based	N/A	N/A	1.Grammar	-
PROTOS [7]	Grammar based	N/A	N/A	1.Grammar	-
Sulley [8]	Grammar based	N/A	N/A	1.Grammar	-
BooFuzz [9]	Grammar based	N/A	N/A	1.Grammar	Sulley
Fuzzowksi [10]	Grammar based	N/A	N/A	1.Grammar	BooFuzz
Asp Fuzz [31]	Grammar based	N/A	N/A	1.Grammar	-
RESTler [11]	Evolutionary Grammar Based	Response	N/A	Grammar	-
SPFuzz [12]	Evolutionary Grammar Based	Coverage	N/A	Grammar	AFL
EPF [20]	Evolutionary Grammar Based	Coverage	N/A	Grammar	1.AFL 2.Fuzzowski
Hsu et al. [14]	Grammar Learner	N/A	N/A	1.Message grammar	Passive learning
Pulsar [13]	Grammar Learner	N/A	N/A	1.Trace	Passive learning
Glade [32]	Grammar Learner	N/A	N/A	1.Trace	Active learning
AFLnet [17]	Evolutionary Grammar Learner	Coverage	Response	1.Target binary 2.Seed traces	AFL
FFUZZ [4]	Evolutionary Grammar Learner	Coverage	Response	1.Target binary 2.Seed input	AFL, AFLNet
StateAFL [18]	Evolutionary Grammar Learner	Coverage	Memory	1.Target binary 2.Seed input	AFLNet
SGPFuzzer [15]	Evolutionary Grammar Learner	Coverage	Response	1.Target binary 2.PCAP file	AFL
LearnLib [19]	Evolutionary Grammar Learner	N/A	Response	Set of messages	L*
Doupé et al. [21]	Evolutionary Grammar Learner	N/A	Response	None	Web application crawling
GANFuzz [24]	ML based	N/A	N/A	Traces	seq2seq model
Fuzzing of Network Protocols [25]	ML based	N/A	N/A	Traces	seq2seq model
SeqFuzzer [23]	ML based	N/A	N/A	Traces	seq-gan model
AutoFuzz [26]	Man in the middle based	N/A	N/A	Live traffic	Passive learning
Live Protocol Fuzzing [28]	Man in the middle based	N/A	N/A	Live traffic	-
SECFUZZ [27]	Man in the middle based	N/A	N/A	Live traffic	-

Fuzzers for stateful systems:
Survey and Research Directions

Challenges in fuzzing network protocols

C.1: Exploring the state model in depth

C.2: Mutating single messages and traces

C.3: Saving the entire trace

C.4: Sending mutated messages over a TCP/IP socket

AFL*

C.1: Exploring the state model in depth	C.2: Mutating single messages and traces	C.3: Saving the entire trace	C.4: Sending mutated messages over a TCP/IP socket
AFL++'s persistent mode	Custom mutator	Modification on AFL++	Preeny

AFL* results

Fuzzer	Execution speed	Time to find a bug in the command parser	Time to find a bug in the argument parser
AFLNet	*9 mess / sec*	*>24h*	*>24h*
AFL*	*34k mess / sec*	*1m50s*	*15m27s*

Comparison between AFLNet and AFL* on LightFTP

Future works

(I) Investigate the relationship between the code coverage and the state coverage

(II) Use a mathematical model learning approach (like LearnLib) to improve the AFL* state awareness and focus on the most ”promising” states.

(III) Test AFL* on other protocols (like OPCUA and 5G)

Take away slide!

1. Stateful fuzzing is challenging!
2. There are many ways to deal with the statefulness of a system
3. Persistent mode — originally devised to fuzz state**less** systems — is extremely useful to the fuzz stateful ones
4. It is important to implement protocols fuzzer-friendly.

For questions, ideas or suggestions, contact me!

cristian.daniele@ru.nl