THE UNIVERSITY OF MELBOURNE

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

AFLNet Five Years Later: On Coverage-Guided Protocol Fuzzing

Ruijie Meng

PhD student at NUS

Incoming tenure-track faculty at CISPA

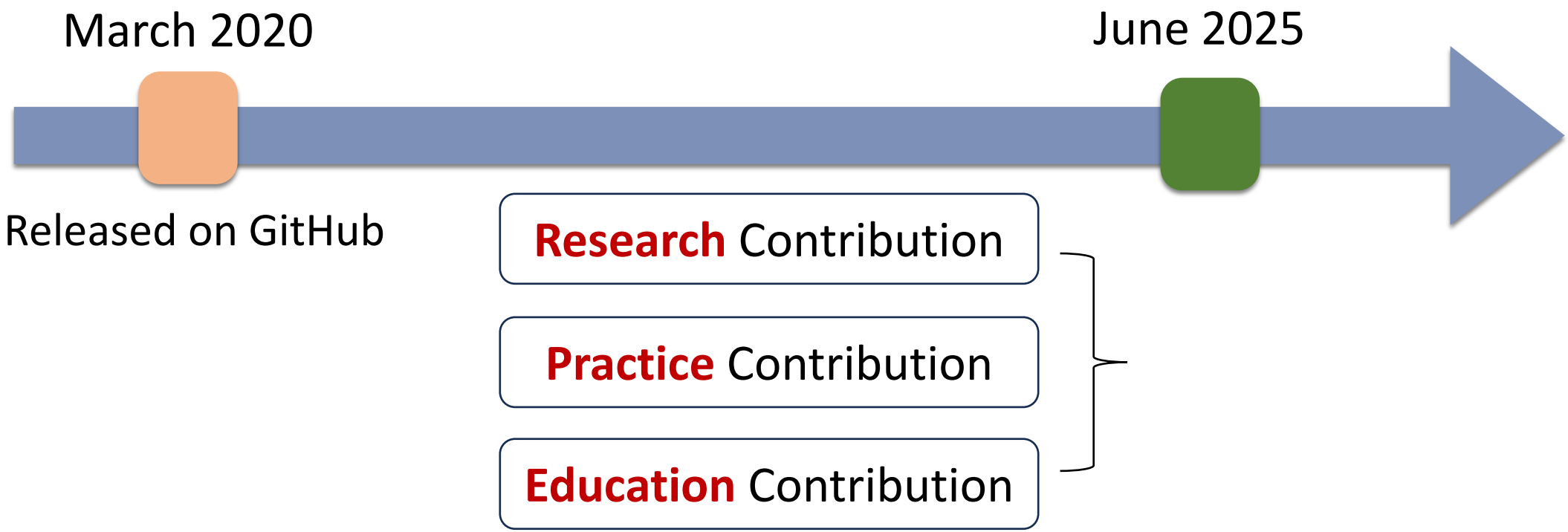
Co-authors: Van-Thuan Pham, Marcel Böhme and Abhik Roychoudhury

What is AFLNet?

First code- and state-coverage guided *protocol* fuzzer (ICST'20, Tool)

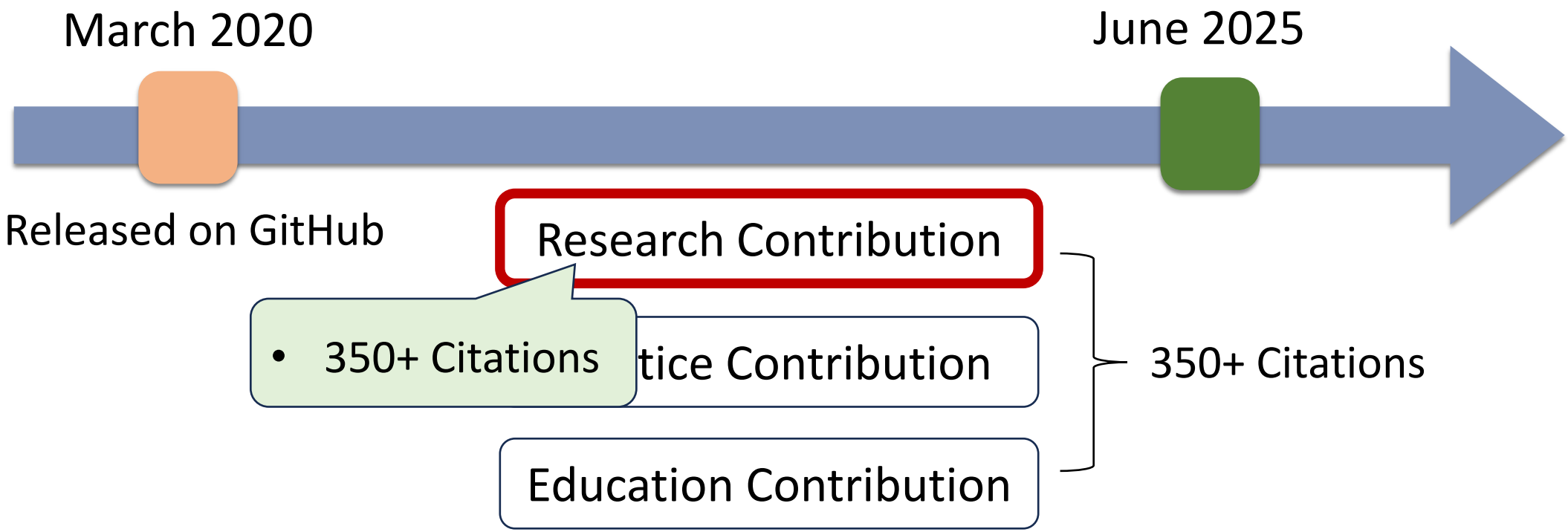
Contributions to Community

First code- and state-coverage guided *protocol* fuzzer (ICST'20, Tool)



Contributions to Community

First code- and state-coverage guided *protocol* fuzzer (ICST'20, Tool)



Contributions to Community

First code- and state-coverage guided *protocol* fuzzer (ICST'20, Tool)

Released on GitHub
RTSP, FTP, SSH, TLS, SMTP

Research Contribution

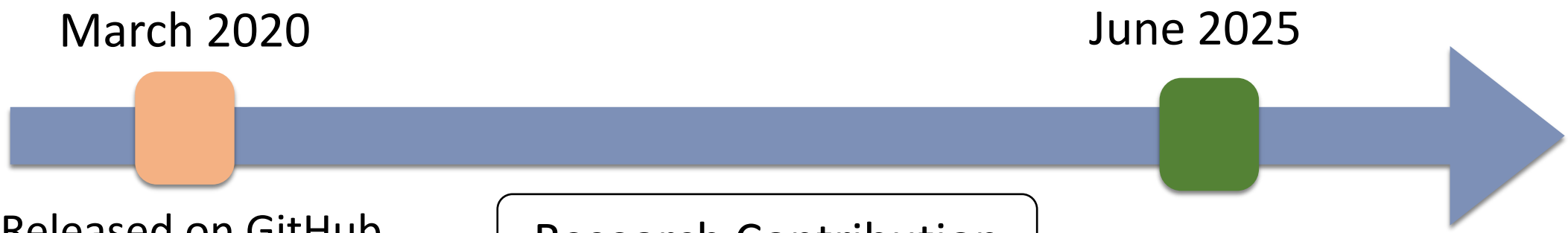
Practice Contribution

350+ Citations, 870+ Stars, 17 protocols, more targets

- 870+ stars on GitHub and 17 protocols supported
- Used for challenging targets, e.g., 5G protocols, smart home ecosystem, medical imaging applications and automotive systems

Contributions to Community

First code- and state-coverage guided *protocol* fuzzer (ICST'20, Tool)



Released on GitHub

Research Contribution

Practice Contribution

Education Contribution

350+ Citations, 870+ Stars, 17 protocols, more targets, course materials

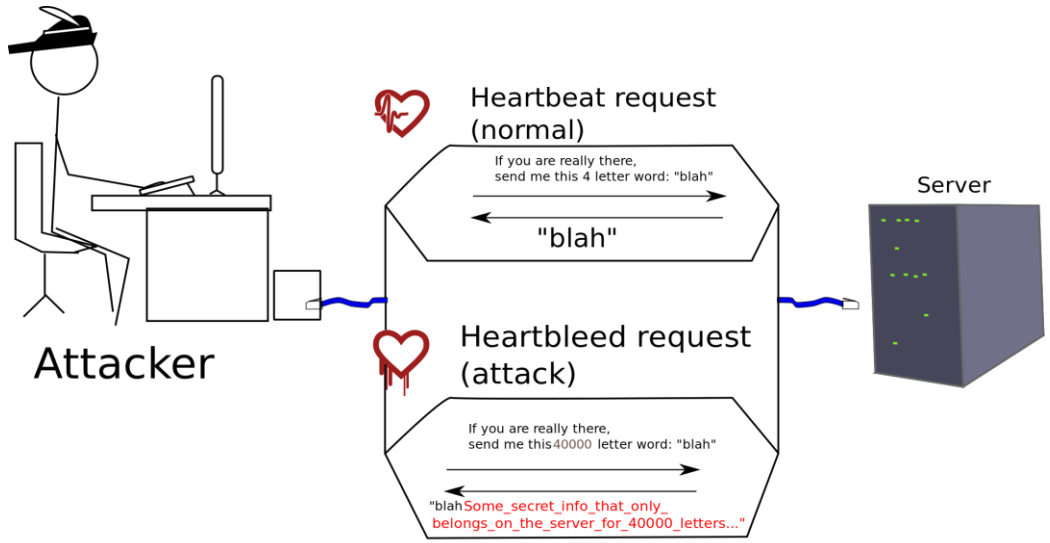
- Introduced in graduate courses, e.g., at University of Melbourne and Carnegie Mellon University

Why has AFLNet generated such impact in a short period?

Why do we need stateful protocol fuzzing?

Testing Protocols is Important

- Network protocols are backbone of critical infrastructure
- Bugs in network protocols damage a lot



Heartbleed Vulnerability in OpenSSL in 2014

Network protocols must be automatically and continuously tested for security vulnerabilities

But…Testing Protocols is Challenging

- **Statefulness** and **sequences of inputs** pose challenges

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
LIST
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

FTP Example of message exchange between client (red) and server (black)

But…Testing Protocols is Challenging

- **Statefulness** and **sequences of inputs** pose challenges

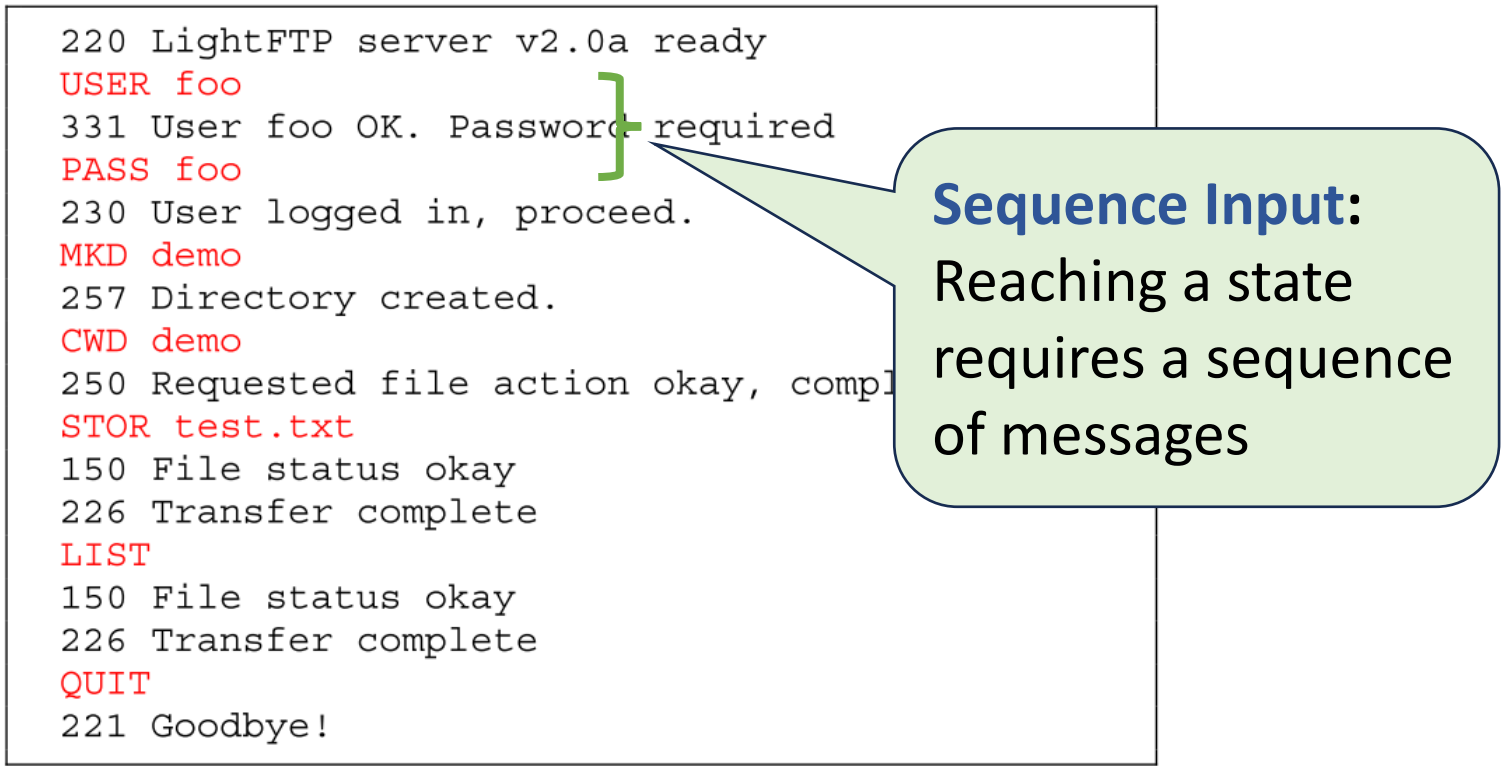
```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proce
MKD demo
257 Directory created.
CWD demo
250 Requested file action
STOR test.txt
150 File status okay
226 Transfer complete
LIST
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Statefulness: sending the ***same*** input message twice might yield a ***different*** response every time based on ***states***

FTP Example of message exchange between client (red) and server (black)

But…Testing Protocols is Challenging

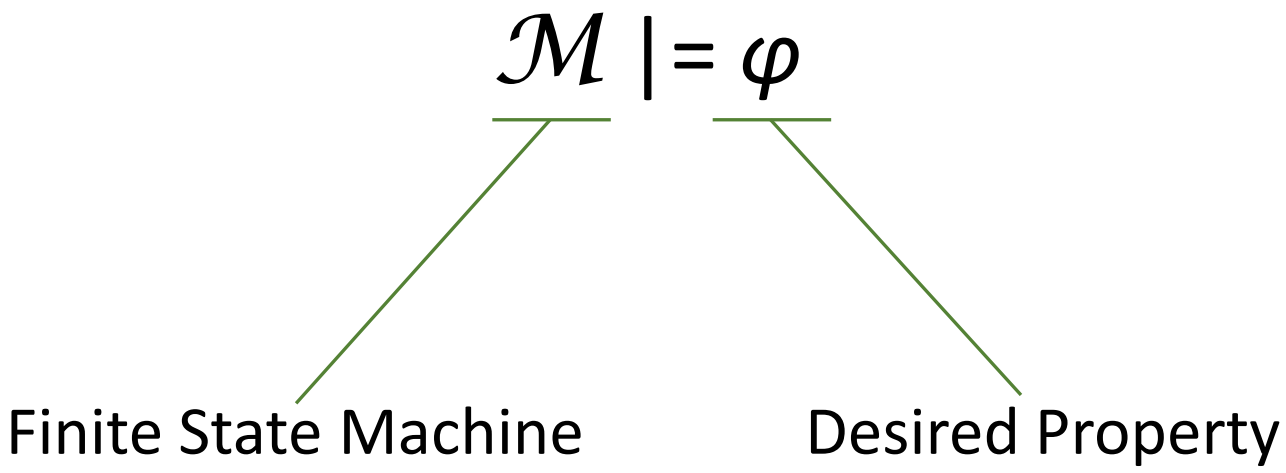
- **Statefulness** and **sequences of inputs** pose challenges



FTP Example of message exchange between client (red) and server (black)

Existing Technique – Model Checking

- A verification technique, but in practice for “bug finding”



Existing Technique – Model Checking

- A verification technique, but in practice for “bug finding”
- $\mathcal{M} \models \varphi$
- Effective but with Limitations:
  - A temporal logic property needs to be provided

Existing Technique – Model Checking

- A verification technique, but in practice for “bug finding”
- $\mathcal{M} \models \varphi$
- Effective but with Limitations:
  - A temporal logic property needs to be provided
  - System modeling is not trivial

inconsistently with each other [12, 18]. Most approaches abstract away the environment behind a model [2, 11], but writing abstract models is labor-intensive (taking in some cases multiple person-years [2]), models are rarely 100% accurate, and they tend to lose

“S2E: A Platform for In-Vivo Multi-Path Analysis of Software Systems”, ASPLOS’11

and the CompCert compiler [120]. Yet, there are, unfortunately, two main inter-related limitations of formally-verified systems: (1) they require years of PhD-level expertise, with the specification often larger than the verified code itself, and (2) the resulting systems lack many of the features and/or performance of their non-verified counterparts. In

“Software Security Analysis in 2030 and Beyond: A Research Roadmap”, TOSEM’25

Existing Technique – Model Checking

- A verification technique, but in practice for “bug finding”
- $\mathcal{M} \models \varphi$
- Effective but with Limitations:
  - A temporal logic property needs to be provided
  - System modeling is not trivial
  - Bugs are reported in a counter-example trace not inputs

Existing Technique – Fuzzing

- Stateful blackbox fuzzing (e.g., Peach)
  - Writing models still involves much manual effort and expertise
  - Learn nothing from past execution
- Stateless greybox fuzzing (e.g., AFL)
  - Almost no manual effort and expertise required
  - Neither know states or message sequences

> *"One of the things that I struggle with is the limitation AFL seems to have, in that it only performs fuzzing with one input (a file). For many systems, such as network protocols, it would be useful if fuzzing could be done on a sequence of inputs. This sequence of inputs might be for example messages necessary to complete a handshake in TLS/TCP."*
>
> - Paul (a member of the AFL's user group) [5]

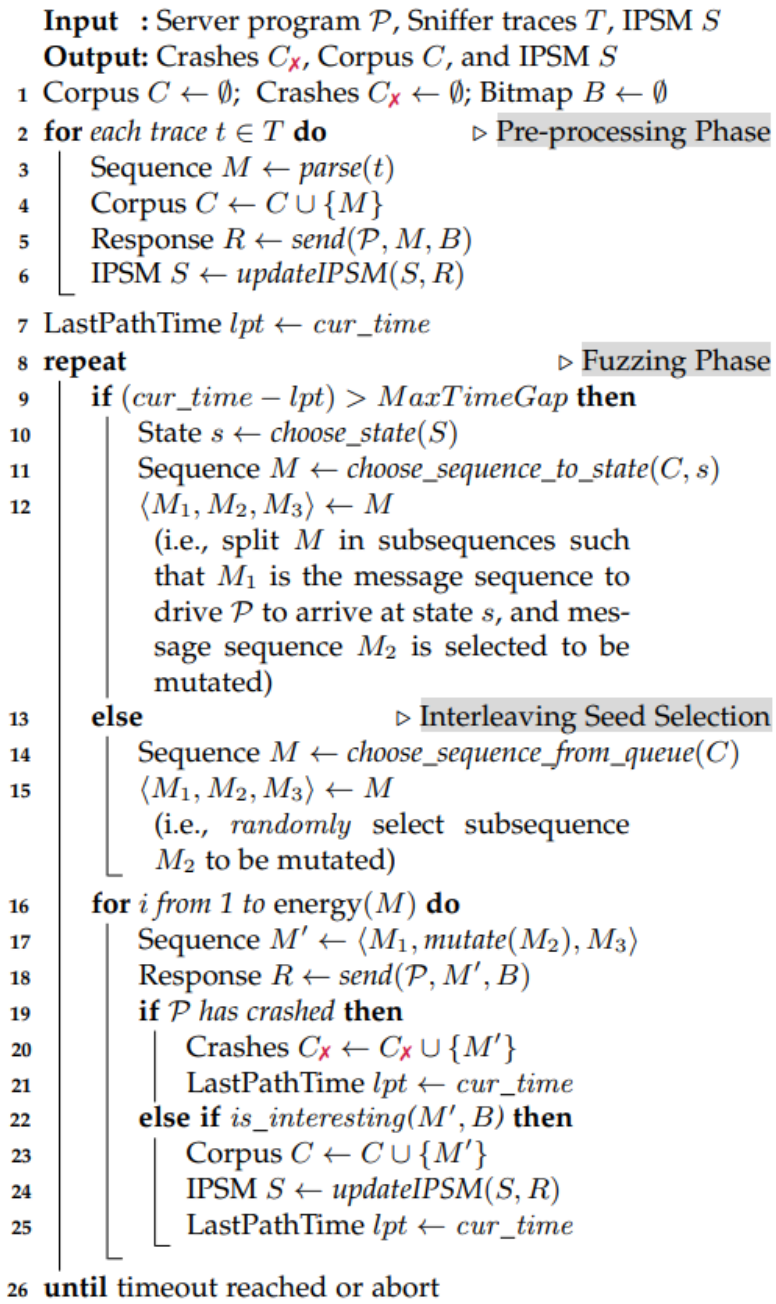
> *"I'm interested in doing something fairly non-traditional and definitely not currently supported by AFL. I would like to perform fuzzing of a large and complex external server that cannot easily be stripped down into small test cases."*
>
> - Tim Newsham (a member of the AFL's user group) [5]

Requests from AFL's users asking for stateful fuzzing support

Why has AFLNet generated such impact in a short period?

- Providing a practical solution for this long-standing problem

AFLNet Approach



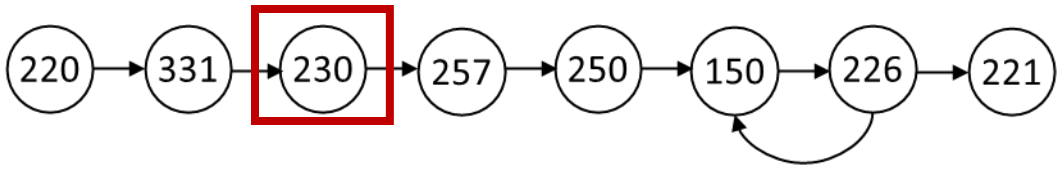
Input : Server program $\mathcal{P}$, Sniffer traces T, IPSM S
Output: Crashes $C_{\text{✗}}$, Corpus C, and IPSM S

1 Corpus $C \leftarrow \emptyset$; Crashes $C_{\text{✗}} \leftarrow \emptyset$; Bitmap $B \leftarrow \emptyset$
2 **for** *each trace* $t \in T$ **do** ▷ Pre-processing Phase
3 Sequence $M \leftarrow parse(t)$
4 Corpus $C \leftarrow C \cup \{M\}$
5 Response $R \leftarrow send(\mathcal{P}, M, B)$
6 IPSM $S \leftarrow updateIPSM(S, R)$
7 LastPathTime $lpt \leftarrow cur\_time$
8 **repeat** ▷ Fuzzing Phase
9 **if** $(cur\_time - lpt) > MaxTimeGap$ **then**
10 State $s \leftarrow choose\_state(S)$
11 Sequence $M \leftarrow choose\_sequence\_to\_state(C, s)$
12 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
 (i.e., split M in subsequences such that M_1 is the message sequence to drive $\mathcal{P}$ to arrive at state s, and message sequence M_2 is selected to be mutated)
13 **else** ▷ Interleaving Seed Selection
14 Sequence $M \leftarrow choose\_sequence\_from\_queue(C)$
15 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
 (i.e., *randomly* select subsequence M_2 to be mutated)
16 **for** *i from 1 to* energy(M) **do**
17 Sequence $M' \leftarrow \langle M_1, mutate(M_2), M_3 \rangle$
18 Response $R \leftarrow send(\mathcal{P}, M', B)$
19 **if** $\mathcal{P}$ *has crashed* **then**
20 Crashes $C_{\text{✗}} \leftarrow C_{\text{✗}} \cup \{M'\}$
21 LastPathTime $lpt \leftarrow cur\_time$
22 **else if** $is\_interesting(M', B)$ **then**
23 Corpus $C \leftarrow C \cup \{M'\}$
24 IPSM $S \leftarrow updateIPSM(S, R)$
25 LastPathTime $lpt \leftarrow cur\_time$
26 **until** timeout reached or abort

Recording and replay for fuzzing

AFLNet Approach

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
LIST
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

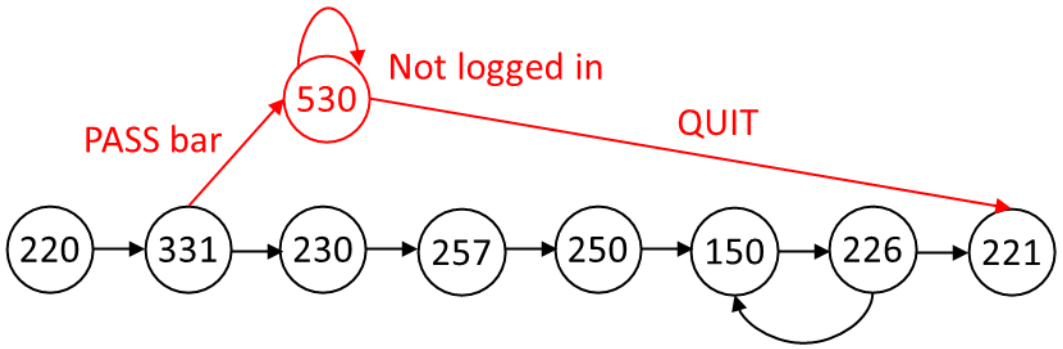


Input : Server program $\mathcal{P}$, Sniffer traces T, IPSM S
Output: Crashes $C_{\text{✗}}$, Corpus C, and IPSM S

1 Corpus $C \leftarrow \emptyset$; Crashes $C_{\text{✗}} \leftarrow \emptyset$; Bitmap $B \leftarrow \emptyset$
2 **for** *each trace* $t \in T$ **do** ▷ Pre-processing Phase
3 Sequence $M \leftarrow parse(t)$
4 Corpus $C \leftarrow C \cup \{M\}$
5 Response $R \leftarrow send(\mathcal{P}, M, B)$
6 IPSM $S \leftarrow updateIPSM(S, R)$
7 LastPathTime $lpt \leftarrow cur\_time$
8 **repeat** ▷ Fuzzing Phase
9 **if** $(cur\_time - lpt) > MaxTimeGap$ **then**
10 State $s \leftarrow choose\_state(S)$
11 Sequence $M \leftarrow choose\_sequence\_to\_state(C, s)$
12 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
 (i.e., split M in subsequences such that M_1 is the message sequence to drive $\mathcal{P}$ to arrive at state s, and message sequence M_2 is selected to be mutated)
13 **else** ▷ Interleaving Seed Selection
14 Sequence $M \leftarrow choose\_sequence\_from\_queue(C)$
15 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
 (i.e., *randomly* select subsequence M_2 to be mutated)
16 **for** i *from 1 to* energy(M) **do**
17 Sequence $M' \leftarrow \langle M_1, mutate(M_2), M_3 \rangle$
18 Response $R \leftarrow send(\mathcal{P}, M', B)$
19 **if** $\mathcal{P}$ *has crashed* **then**
20 Crashes $C_{\text{✗}} \leftarrow C_{\text{✗}} \cup \{M'\}$
21 LastPathTime $lpt \leftarrow cur\_time$
22 **else if** $is\_interesting(M', B)$ **then**
23 Corpus $C \leftarrow C \cup \{M'\}$
24 IPSM $S \leftarrow updateIPSM(S, R)$
25 LastPathTime $lpt \leftarrow cur\_time$
26 **until** timeout reached or abort

AFLNet Approach

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
LIST
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

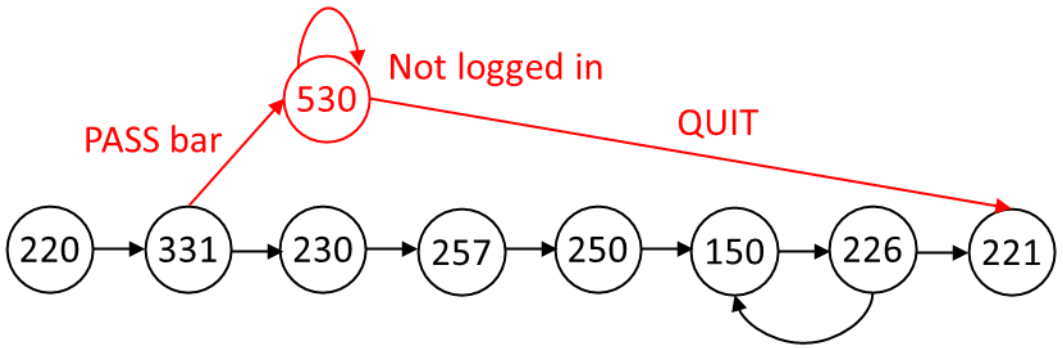


Input : Server program $\mathcal{P}$, Sniffer traces T, IPSM S
Output: Crashes $C_{\text{✗}}$, Corpus C, and IPSM S

1 Corpus $C \leftarrow \emptyset$; Crashes $C_{\text{✗}} \leftarrow \emptyset$; Bitmap $B \leftarrow \emptyset$
2 **for** *each trace* $t \in T$ **do** ▷ Pre-processing Phase
3 Sequence $M \leftarrow parse(t)$
4 Corpus $C \leftarrow C \cup \{M\}$
5 Response $R \leftarrow send(\mathcal{P}, M, B)$
6 IPSM $S \leftarrow updateIPSM(S, R)$
7 LastPathTime $lpt \leftarrow cur\_time$
8 **repeat** ▷ Fuzzing Phase
9 **if** $(cur\_time - lpt) > MaxTimeGap$ **then**
10 State $s \leftarrow choose\_state(S)$
11 Sequence $M \leftarrow choose\_sequence\_to\_state(C, s)$
12 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
(i.e., split M in subsequences such that M_1 is the message sequence to drive $\mathcal{P}$ to arrive at state s, and message sequence M_2 is selected to be mutated)
13 **else** ▷ Interleaving Seed Selection
14 Sequence $M \leftarrow choose\_sequence\_from\_queue(C)$
15 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
(i.e., *randomly* select subsequence M_2 to be mutated)
16 **for** i *from 1 to* energy(M) **do**
17 Sequence $M' \leftarrow \langle M_1, mutate(M_2), M_3 \rangle$
18 Response $R \leftarrow send(\mathcal{P}, M', B)$
19 **if** $\mathcal{P}$ *has crashed* **then**
20 Crashes $C_{\text{✗}} \leftarrow C_{\text{✗}} \cup \{M'\}$
21 LastPathTime $lpt \leftarrow cur\_time$
22 **else if** $is\_interesting(M', B)$ **then**
23 Corpus $C \leftarrow C \cup \{M'\}$
24 IPSM $S \leftarrow updateIPSM(S, R)$
25 LastPathTime $lpt \leftarrow cur\_time$
26 **until** timeout reached or abort

AFLNet Approach

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
LIST
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

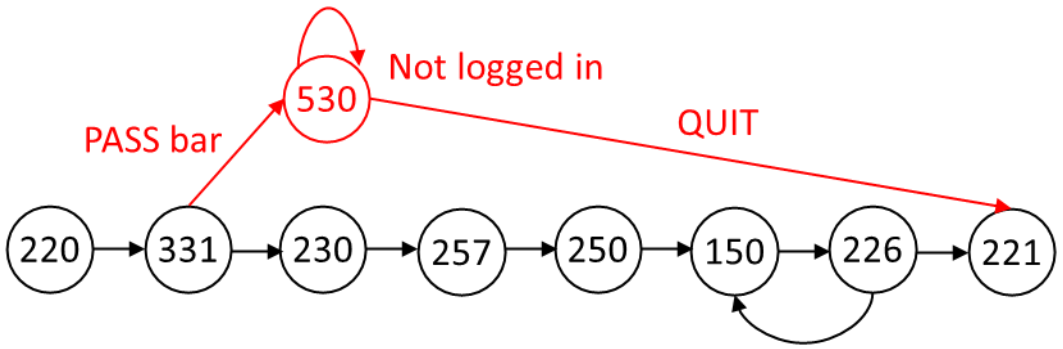


Input : Server program $\mathcal{P}$, Sniffer traces T, IPSM S
Output: Crashes $C_{\text{✗}}$, Corpus C, and IPSM S

1 Corpus $C \leftarrow \emptyset$; Crashes $C_{\text{✗}} \leftarrow \emptyset$; Bitmap $B \leftarrow \emptyset$
2 **for** *each trace* $t \in T$ **do** ▷ Pre-processing Phase
3 Sequence $M \leftarrow parse(t)$
4 Corpus $C \leftarrow C \cup \{M\}$
5 Response $R \leftarrow send(\mathcal{P}, M, B)$
6 IPSM $S \leftarrow updateIPSM(S, R)$
7 LastPathTime $lpt \leftarrow cur\_time$
8 **repeat** ▷ Fuzzing Phase
9 **if** $(cur\_time - lpt) > MaxTimeGap$ **then**
10 State $s \leftarrow choose\_state(S)$
11 Sequence $M \leftarrow choose\_sequence\_to\_state(C, s)$
12 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
(i.e., split M in subsequences such that M_1 is the message sequence to drive $\mathcal{P}$ to arrive at state s, and message sequence M_2 is selected to be mutated)
13 **else** ▷ Interleaving Seed Selection
14 Sequence $M \leftarrow choose\_sequence\_from\_queue(C)$
15 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
(i.e., *randomly* select subsequence M_2 to be mutated)
16 **for** *i from 1 to* energy(M) **do**
17 Sequence $M' \leftarrow \langle M_1, mutate(M_2), M_3 \rangle$
18 Response $R \leftarrow send(\mathcal{P}, M', B)$
19 **if** $\mathcal{P}$ *has crashed* **then**
20 Crashes $C_{\text{✗}} \leftarrow C_{\text{✗}} \cup \{M'\}$
21 LastPathTime $lpt \leftarrow cur\_time$
22 **else if** $is\_interesting(M', B)$ **then**
23 Corpus $C \leftarrow C \cup \{M'\}$
24 IPSM $S \leftarrow updateIPSM(S, R)$
25 LastPathTime $lpt \leftarrow cur\_time$
26 **until** timeout reached or abort

AFLNet Approach

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
LIST
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

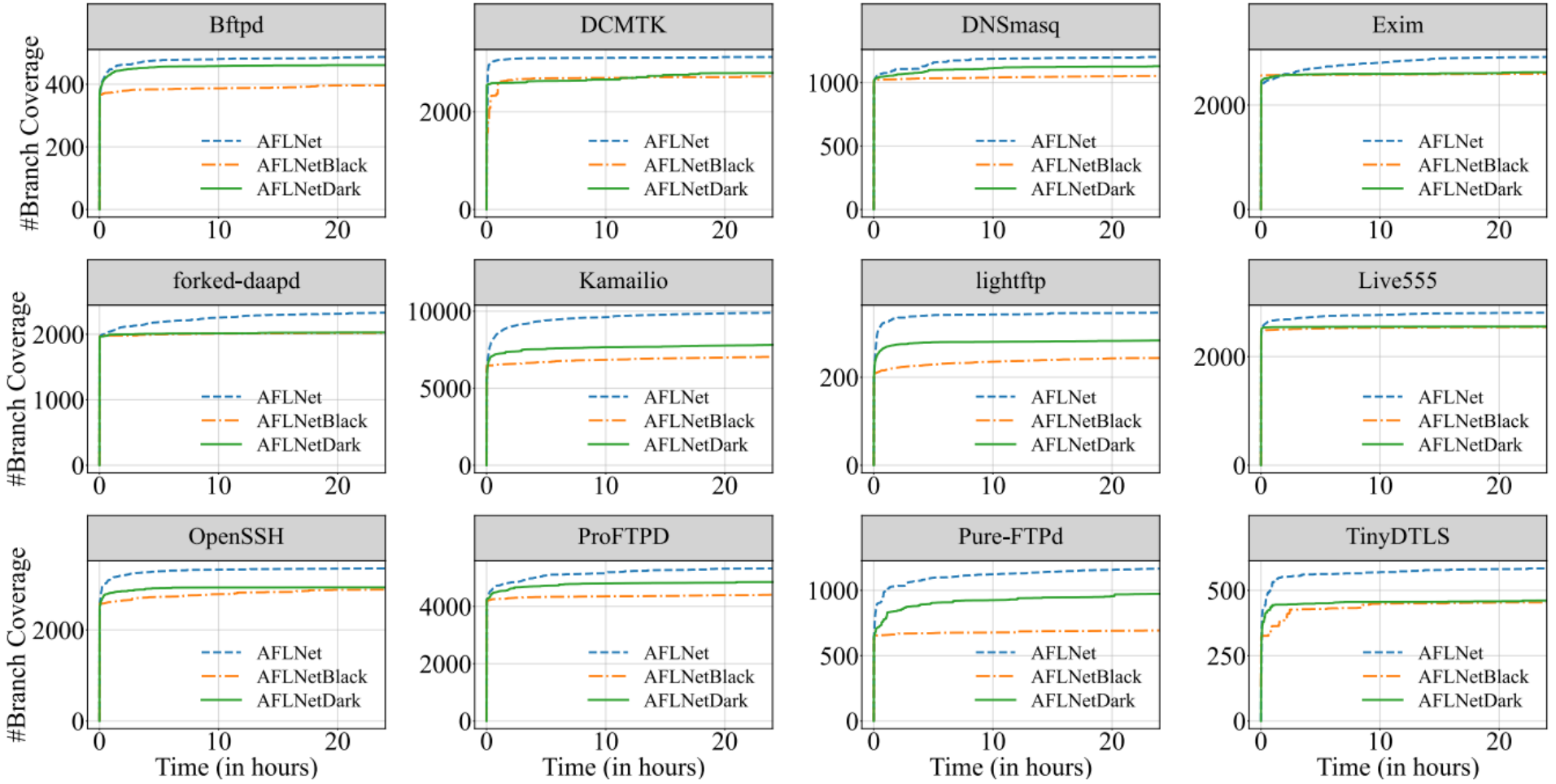


Input : Server program $\mathcal{P}$, Sniffer traces T, IPSM S
Output: Crashes $C_{\text{✗}}$, Corpus C, and IPSM S

1 Corpus $C \leftarrow \emptyset$; Crashes $C_{\text{✗}} \leftarrow \emptyset$; Bitmap $B \leftarrow \emptyset$
2 **for** *each trace* $t \in T$ **do** ▷ Pre-processing Phase
3 Sequence $M \leftarrow parse(t)$
4 Corpus $C \leftarrow C \cup \{M\}$
5 Response $R \leftarrow send(\mathcal{P}, M, B)$
6 IPSM $S \leftarrow updateIPSM(S, R)$
7 LastPathTime $lpt \leftarrow cur\_time$
8 **repeat** ▷ Fuzzing Phase
9 **if** $(cur\_time - lpt) > MaxTimeGap$ **then**
10 State $s \leftarrow choose\_state(S)$
11 Sequence $M \leftarrow choose\_sequence\_to\_state(C, s)$
12 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
(i.e., split M in subsequences such that M_1 is the message sequence to drive $\mathcal{P}$ to arrive at state s, and message sequence M_2 is selected to be mutated)
13 **else** ▷ Interleaving Seed Selection
14 Sequence $M \leftarrow choose\_sequence\_from\_queue(C)$
15 $\langle M_1, M_2, M_3 \rangle \leftarrow M$
(i.e., *randomly* select subsequence M_2 to be mutated)
16 **for** i *from 1 to* energy(M) **do**
17 Sequence $M' \leftarrow \langle M_1, mutate(M_2), M_3 \rangle$
18 Response $R \leftarrow send(\mathcal{P}, M', B)$
19 **if** $\mathcal{P}$ *has crashed* **then**
20 Crashes $C_{\text{✗}} \leftarrow C_{\text{✗}} \cup \{M'\}$
21 LastPathTime $lpt \leftarrow cur\_time$
22 **else if** $is\_interesting(M', B)$ **then**
23 Corpus $C \leftarrow C \cup \{M'\}$
24 IPSM $S \leftarrow updateIPSM(S, R)$
25 LastPathTime $lpt \leftarrow cur\_time$
26 **until** timeout reached or abort

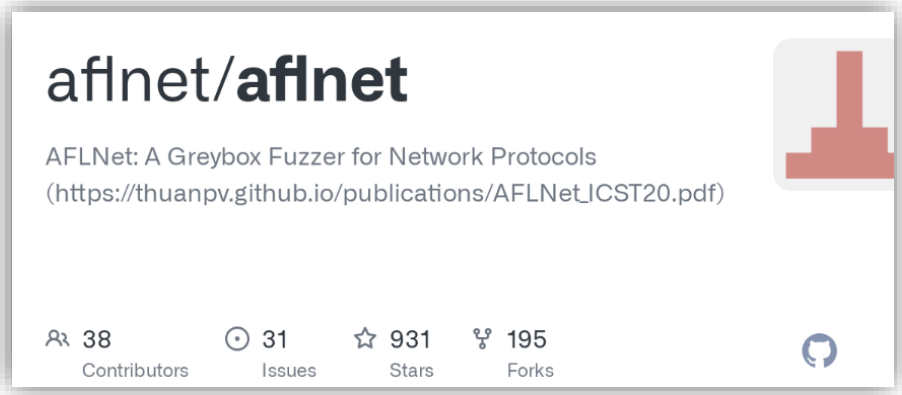
Multiple Feedback Modes

- Coverage in black-, dark-, and grey-modes



Open-Sourced Tool

- AFLNet acts as the client
  - Follow real-world architectures of network protocols
  - Reduce manual effort in understanding protocols or modifying source code



however, a better option is AFLnet (https://github.com/aflnet/aflnet) which allows you to define network state with different type of data packets.

best practice in fuzzing network services from the AFL++ document

Why has AFLNet generated such impact in a short period?

- Providing a practical solution for this long-standing problem
- **Open Science Approach**

More Research is In Progress

Recent Progress in Stateful Fuzzing

What is a state?

- SGFuzz (Usenix Sec'22)
- StateAFL (EMSE'22)
- ChatAFL (NDSS'24)

…

How to maximize syntactic validity of each message?

- ChatAFL (NDSS'24)

…

Protocol Environment Fuzzing

- ChaosAFL (ArXiv'23)
- EnvFuzz (CCS'23)

…

How to maximize fuzzing throughput?

- Nyx-Net (EuroSys'22)
- SnapFuzz (ISSTA'22)

…

Path Forward

Network protocol

Autonomous vehicle

Distributed System

More stateful software being routinely checked

AI agent

Kernel Drivers

Industrial control system

Cyber-Physical System

Internet of Things

What is AFLNet?

First code- and state-coverage guided *protocol* fuzzer (ICST'20, Tool)

March 2020 — Released on GitHub

June 2025

- Research Contribution
- Practice Contribution
- Education Contribution

350+ Citations, 870+ Stars, 17 protocols, more targets, course materials

AFLNet Approach

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
LIST
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Input : Server program $\mathcal{P}$, Sniffer traces T, IPSM S
Output: Crashes $C_\mathcal{X}$, Corpus C, and IPSM S
Corpus $C \leftarrow \emptyset$; Crashes $C_\mathcal{X} \leftarrow \emptyset$; Bitmap $B \leftarrow \emptyset$
for *each trace* $t \in T$ **do** ▷ Pre-processing Phase
  Sequence $M \leftarrow parse(t)$
  Corpus $C \leftarrow C \cup \{M\}$
  Response $R \leftarrow send(\mathcal{P}, M, B)$
  IPSM $S \leftarrow updateIPSM(S, R)$
LastPathTime $lpt \leftarrow cur\_time$
repeat ▷ Fuzzing Phase
  if $(cur\_time - lpt) > MaxTimeGap$ **then**
    State $s \leftarrow choose\_state(S)$
    Sequence $M \leftarrow choose\_sequence\_to\_state(C, s)$
    $(M_1, M_2, M_3) \leftarrow M$
    (i.e., split M in subsequences such that M_1 is the message sequence to drive $\mathcal{P}$ to arrive at state s, and message sequence M_2 is selected to be mutated)
  else ▷ Interleaving Seed Selection
    Sequence $M \leftarrow choose\_sequence\_from\_queue(C)$
    $(M_1, M_2, M_3) \leftarrow M$
    (i.e., *randomly* select subsequence M_2 to be mutated)
  for *i from 1 to* energy(M) **do**
    Sequence $M' \leftarrow (M_1, mutate(M_2), M_3)$
    Response $R \leftarrow send(\mathcal{P}, M', B)$
    if $\mathcal{P}$ *has crashed* **then**
      Crashes $C_\mathcal{X} \leftarrow C_\mathcal{X} \cup \{M'\}$
      LastPathTime $lpt \leftarrow cur\_time$
    else if $is\_interesting(M', B)$ **then**
      Corpus $C \leftarrow C \cup \{M'\}$
      IPSM $S \leftarrow updateIPSM(S, R)$
      LastPathTime $lpt \leftarrow cur\_time$
until timeout reached or abort

Why has AFLNet generated such impact in a short period?

- Providing a practical solution for this long-standing problem
- Open Science Approach

Recent Progress in Stateful Fuzzing

What is a state?
- SGFuzz (Usenix Sec'22)
- StateAFL (EMSE'22)
- ChatAFL (NDSS'24)
- ...

How to maximize syntactic validity of each message?
- ChatAFL (NDSS'24)
- ...

Protocol Environment Fuzzing
- ChaosAFL (ArXiv'23)
- EnvFuzz (CCS'23)
- ...

How to maximize fuzzing throughput?
- Nyx-Net (EuroSys'22)
- SnapFuzz (ISSTA'22)
- ...

Thanks!