FitM

Dominik Maier
Otto Bittner
Julian Beier
Marc Munier

Binary-Only Coverage-Guided Fuzzing for Stateful Network Protocols

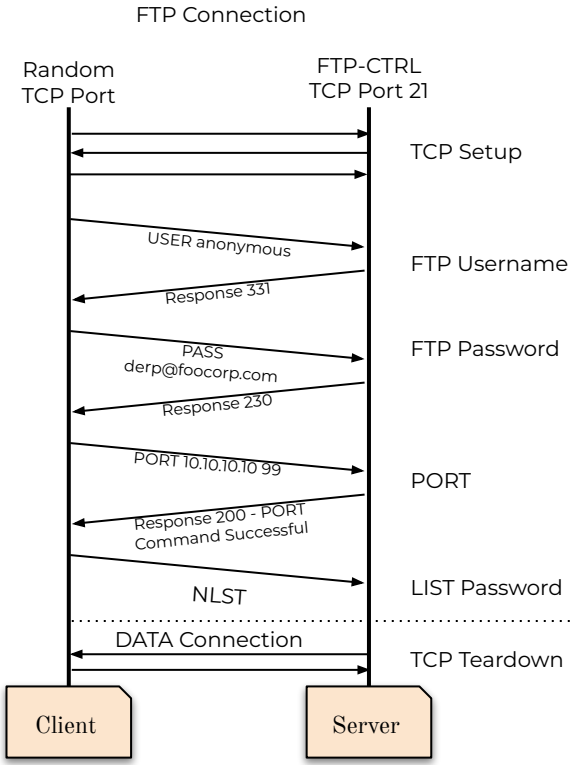
Binary Analysis Research (BAR) Workshop 2022

Fuzzing

TL;DR: Throw corner-case input at a program until it breaks.

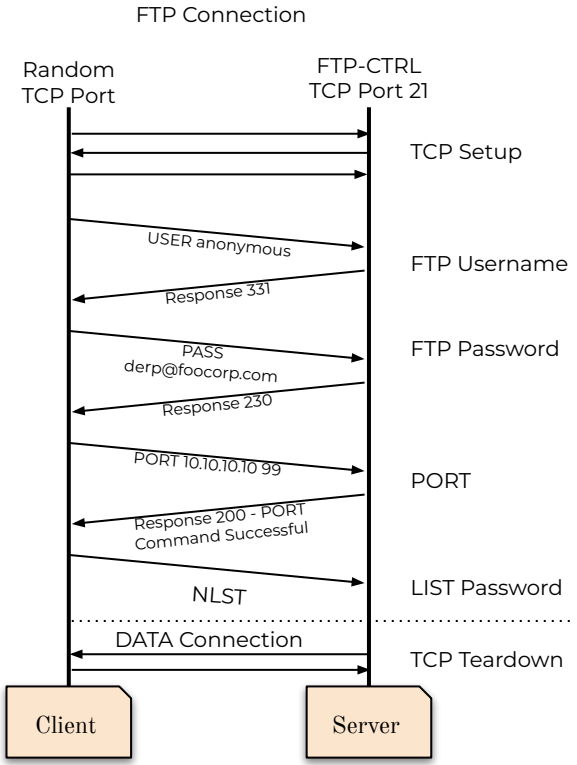
Protocols are hard.

Motivation - Protocol Exploration



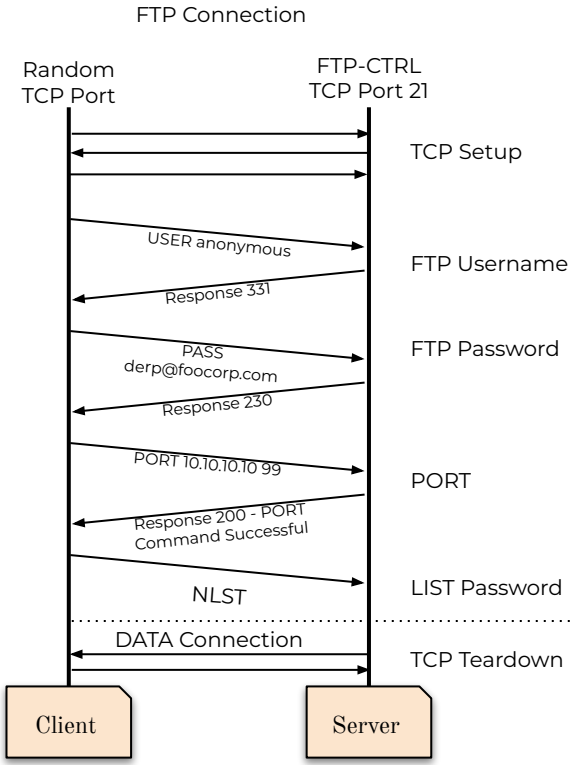
USER anonymous\n

Motivation - Protocol Exploration



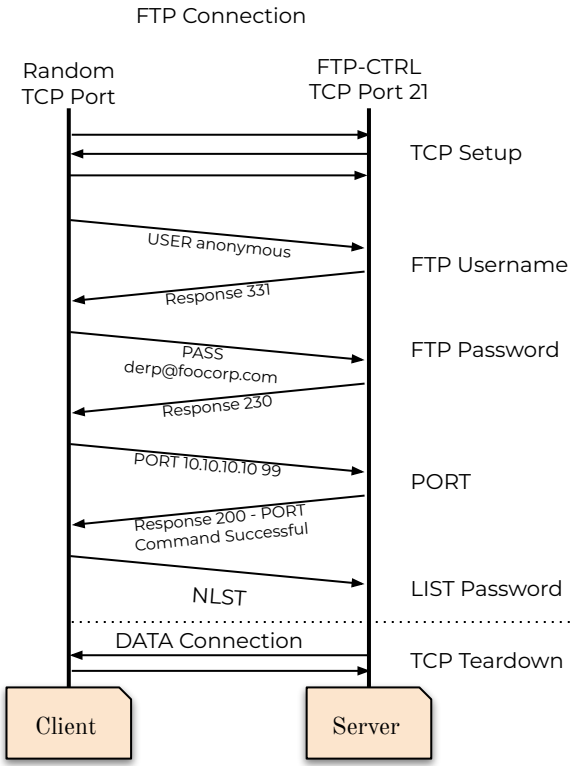
USER anonymous\nPASS anonymous\n

Motivation - Protocol Exploration



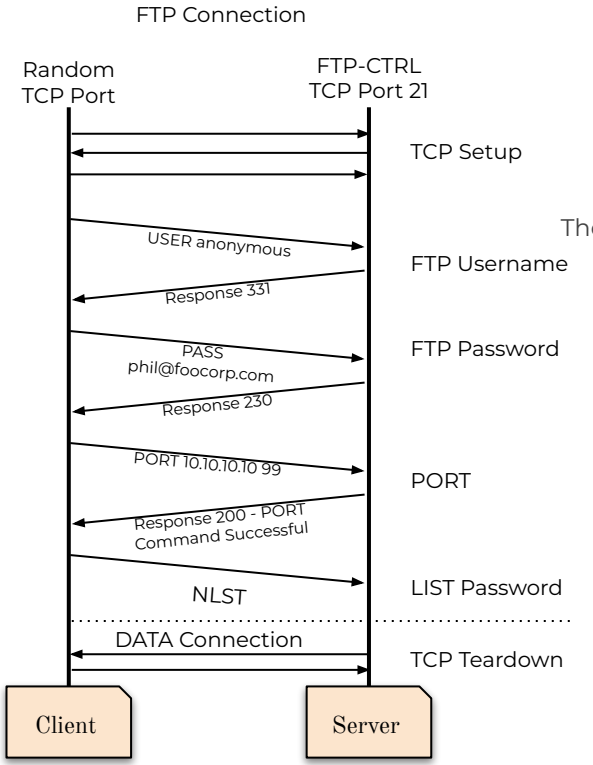
USER anonymous\nPASS anonymous\nPORT 1337\n

Motivation - Protocol Exploration



USER anonymous\nPASS anonymous\nPORT 1337\nNLST\n ...

Motivation - Structured Inputs



The password depends on the username

Typical problems:

- State dependent inputs required
- Highly structured input
- Checksums, length fields, etc.

Motivation - Exec Speed

Empty harness

```
$ cat libafl.c
volatile int test;
int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
 test += 1;
}
```

```
dmnk@dmnk ~/tmp/LibAFL/fuzzers/fuzzbench main*
❯ ./fuzzer -i in -o out
Workdir: "/usr/local/google/home/dmnk/tmp/LibAFL/fuzzers/fuzzbench"
Out dir at "out" already exists.
Spawning next client (id 0)
First run. Let's set it all up
Let's fuzz :)
Loading file "in/a" ...
[Stats #0] run time: 0h-0m-0s, clients: 1, corpus: 0, objectives: 0, executions: 0, exec/sec: 0
[Testcase #0] run time: 0h-0m-0s, clients: 1, corpus: 1, objectives: 0, executions: 1, exec/sec: 0
[LOG Debug]: Loaded 1 initial testcases.
We imported 1 inputs from disk.
[Stats #0] run time: 0h-0m-15s, clients: 1, corpus: 1, objectives: 0, executions: 3246583, exec/sec: 216418
[Stats #0] run time: 0h-0m-30s, clients: 1, corpus: 1, objectives: 0, executions: 6508710, exec/sec: 216944
[Stats #0] run time: 0h-0m-45s, clients: 1, corpus: 1, objectives: 0, executions: 9774893, exec/sec: 217204
[Stats #0] run time: 0h-1m-0s, clients: 1, corpus: 1, objectives: 0, executions: 13043834, exec/sec: 217384
```

Recv-in-a-loop

```
$ cat server.c
for (int i = 0; i < 1000000; i++){
  connfd = accept(sockfd, (SA*)&cli, &len);
  recv(connfd, buf, 4, 0);
  close(connfd);
}
```

```
❯ time ./server
Socket successfully created..
Socket successfully binded..
Server listening..
./server  0.08s user 3.15s system 99% cpu 3.252 total
```

Motivation - Exec Speed

Empty harness

- > 200.000 inputs/sec delivered
- Dynamically produced input

Recv-in-a-loop

- ~30.000 inputs/sec delivered
- Static input

Input generation outperforms socket interactions almost by an order of magnitude
⇒ Potentially huge performance losses in fuzzing
⇒ Bad scaling due to kernel interactions

Motivation

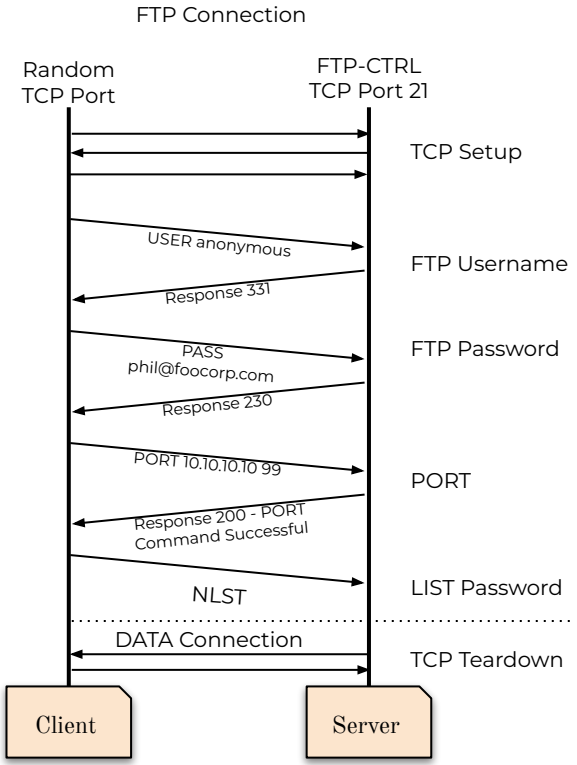
Recap:

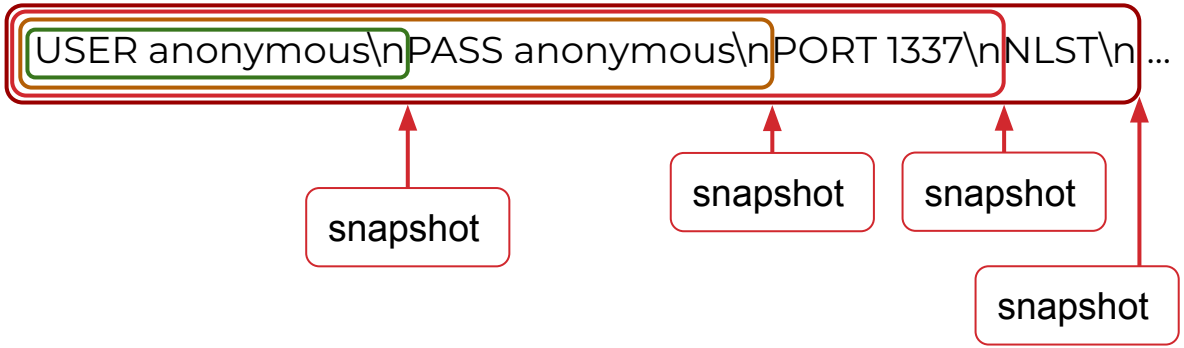
- Exploring state space gets harder with depth
- Structured input generation needs additional insight
- OS network stack is slow

FitM

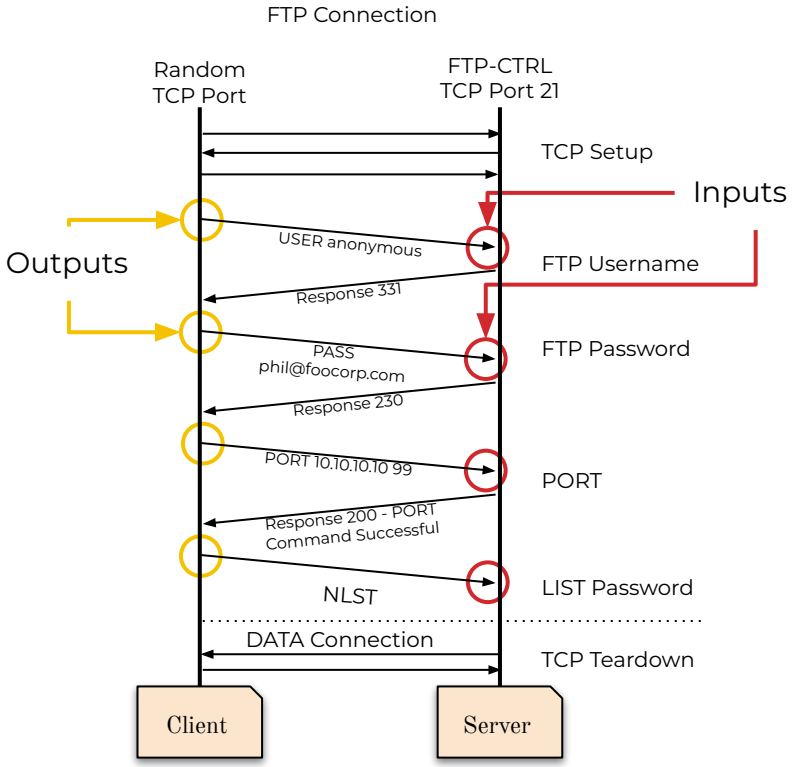
Fuzzer in the Middle

Solution - Coverage Guided Fuzzers





Solution - Input Generation



Available commands of an FTP server (HELP)

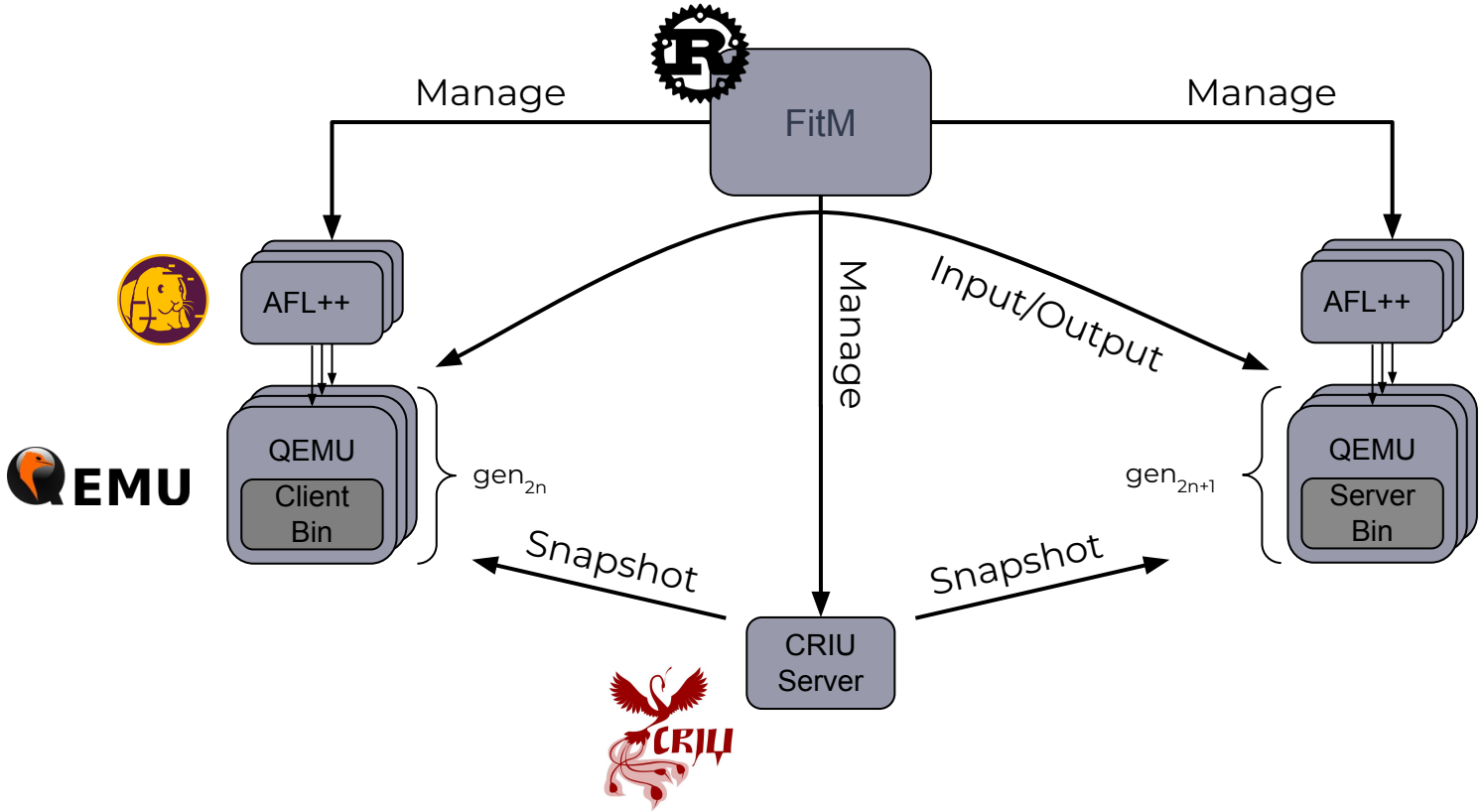
```
ABOR ACCT ALLO APPE CDUP CWD DELE EPRT
EPSV FEAT HELP LIST MDTM MKD MODE NLST
NOOP OPTS PASS PASV PORT PWD QUIT REIN
REST RETR RMD RNFR RNTO SITE SIZE SMNT
STAT STOR STOU STRU SYST TYPE USER XCUP
XCWD XMKD XPWD XRMD
```

Solution - Slow Network Stack

QEMU

[1]

Overview - Technologies



[1], [2], [3]

Protocols

```
while ( ctx.ControlSocket != INVALID_SOCKET ) {
    if ( !recvcmd(&ctx, rcvbuf, sizeof(rcvbuf)) )
        break;

    i = 0;
    while ((rcvbuf[i] != 0) && (isalpha(rcvbuf[i]) == 0))
        ++i;

    cmd = &rcvbuf[i];
    while ((rcvbuf[i] != 0) && (rcvbuf[i] != ' '))
        ++i;

    cmdlen = &rcvbuf[i] - cmd;
    while (rcvbuf[i] == ' ')
        ++i;

    if (rcvbuf[i] == 0)
        params = NULL;
    else
        params = &rcvbuf[i];

    cmdno = -1;
    rv = 1;
    for (c=0; c<MAX_CMDS; c++)
        if (strncasecmp(cmd, ftpprocs[c].Name, cmdlen) == 0)
        {
            cmdno = c;
            rv = ftpprocs[c].Proc(&ctx, params);
            break;
        }

    if ( cmdno != FTP_PASSCMD_INDEX )
        writelogentry(&ctx, " @@ CMD: ", rcvbuf);
    else
        writelogentry(&ctx, " @@ CMD: ", "PASS ***");

    if ( cmdno == -1 )
        sendstring(&ctx, error500);
```

< Recv

< Send

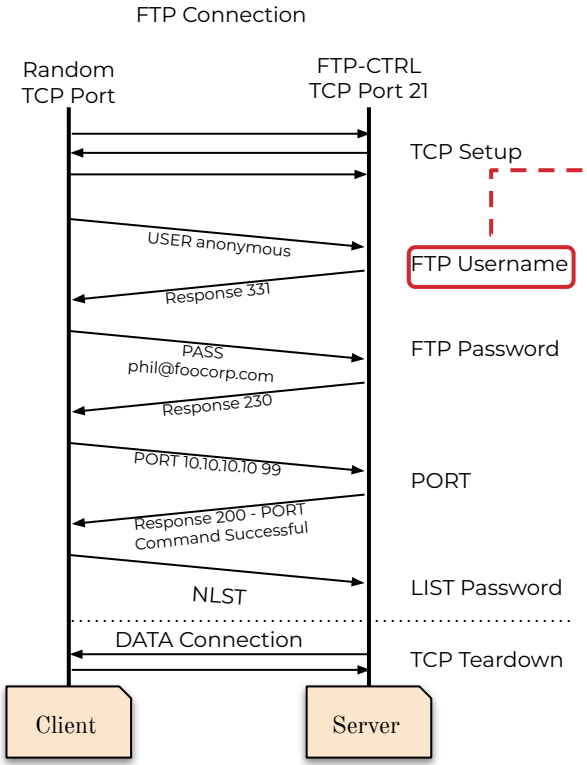
Assumption:

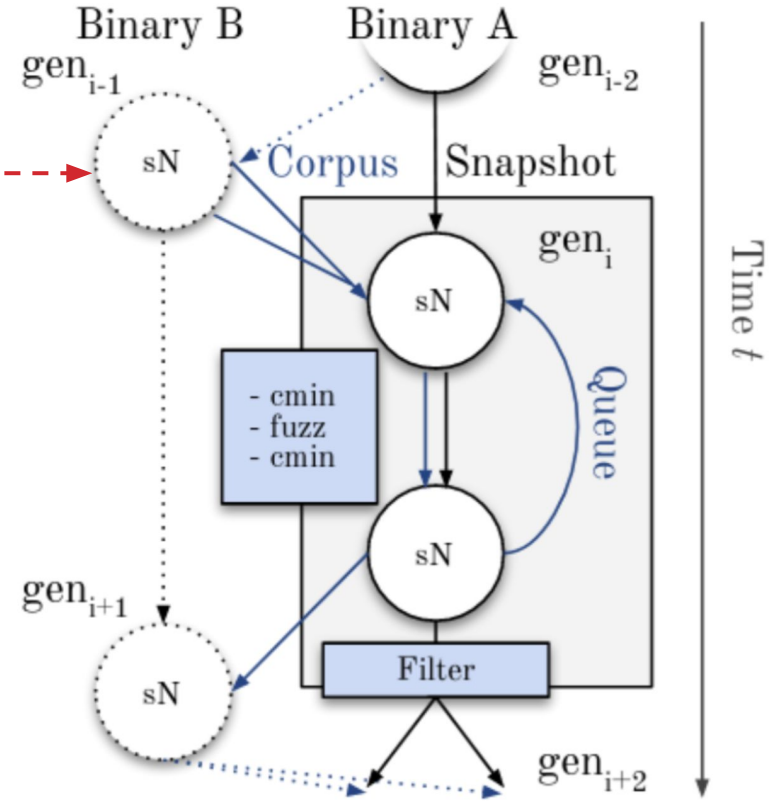
Network-apps

==

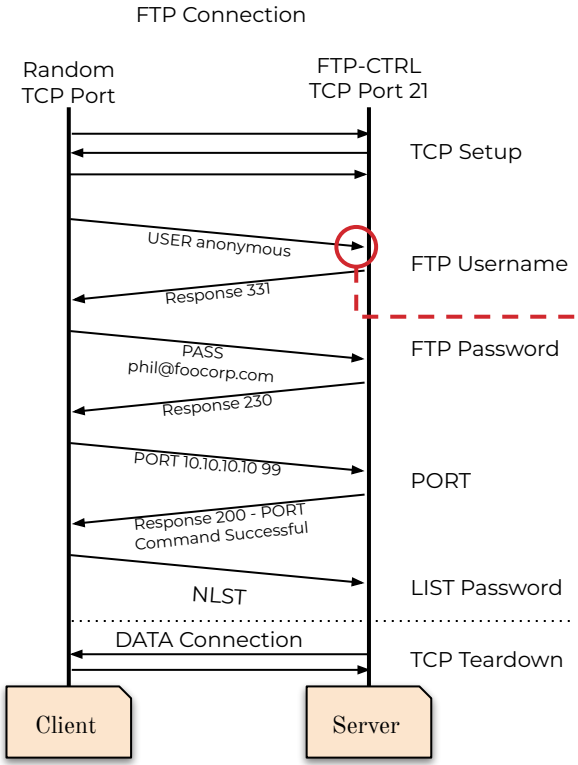
"recv-send-recv"-loops

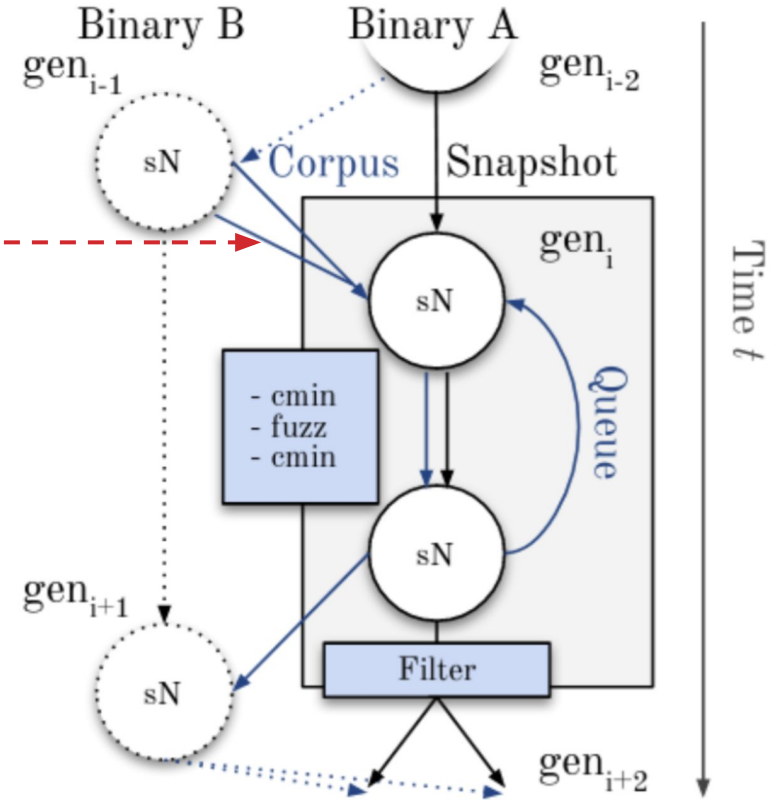
Implementation Model



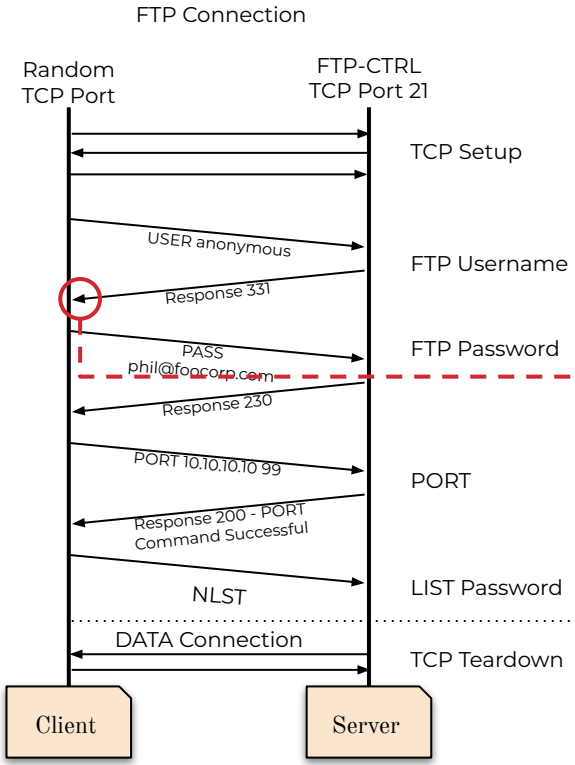


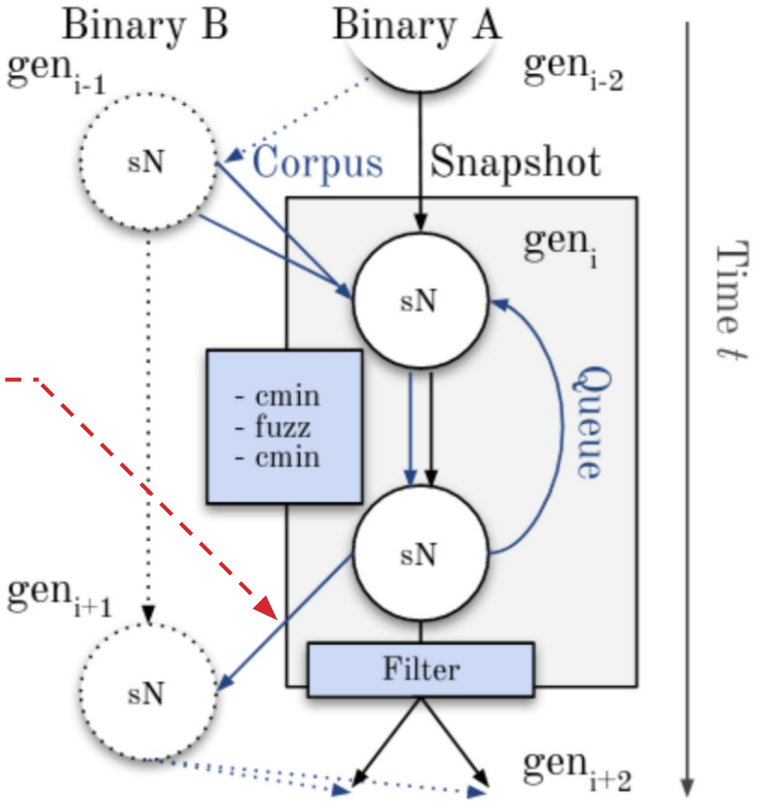
Implementation Model



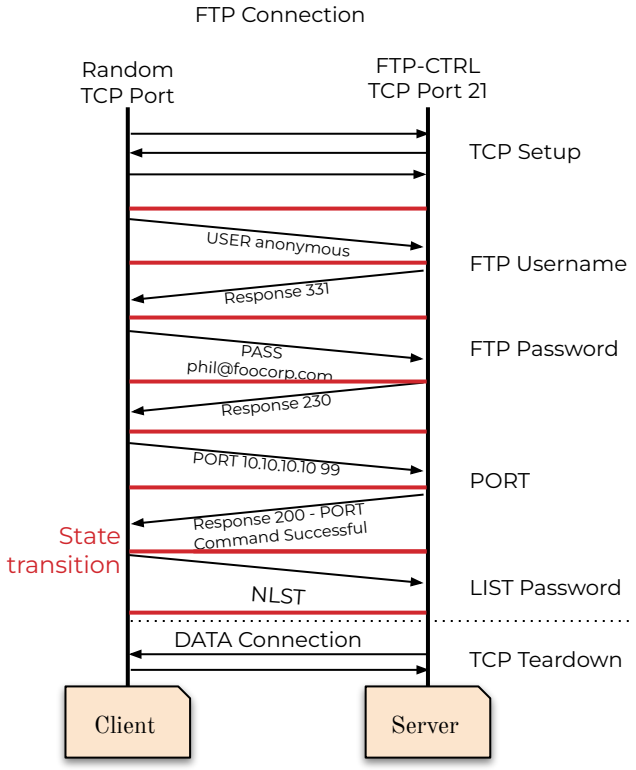


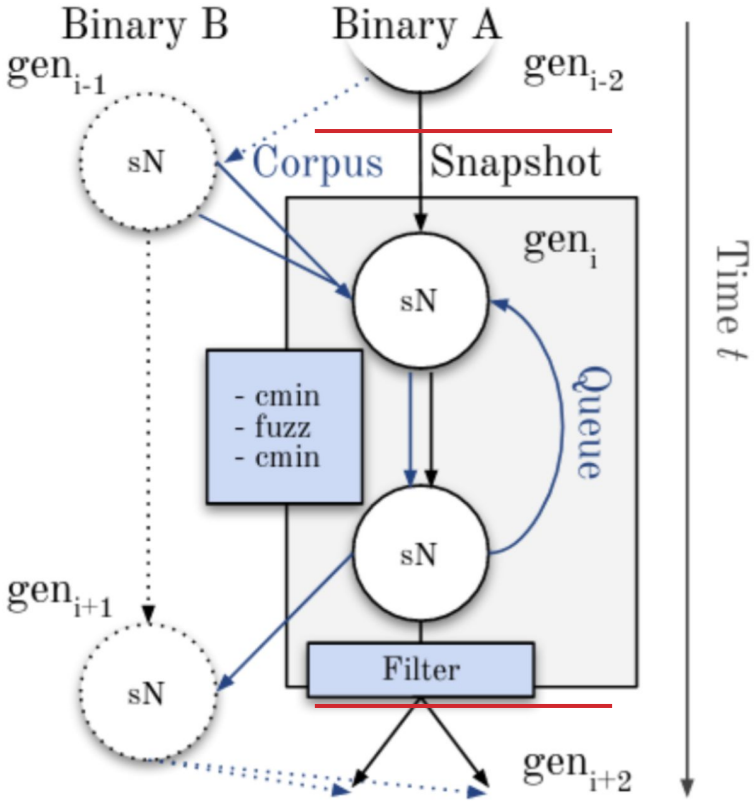
Implementation Model





Implementation Model





Network Emulation with QEMU

- Deliver input via shmem
- Remove kernel code as much as possible
- Supports sync & async

⇒ Big speedup

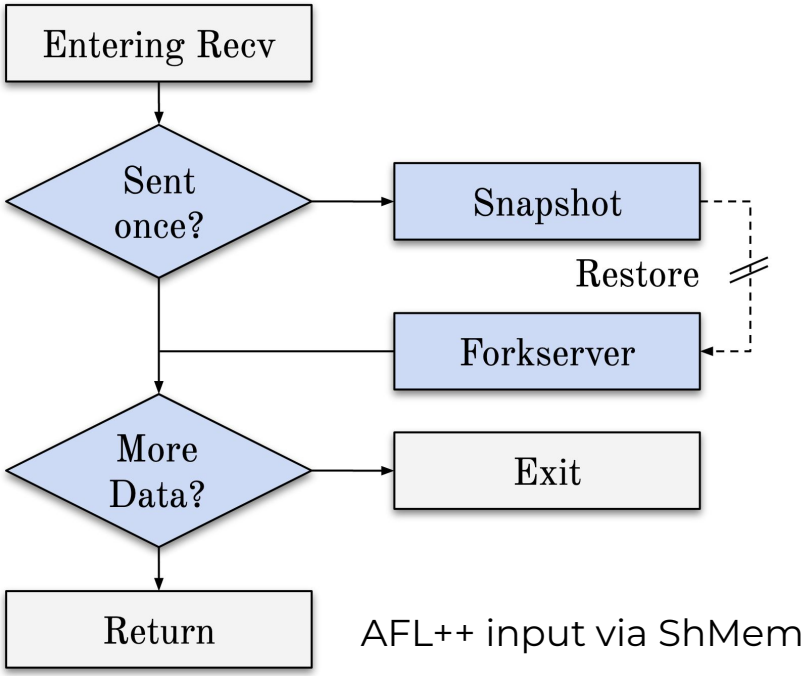
`accept(4)`	`fnctl`	`read`	`sendmsg`
`bind`	`epoll`	`recv`	`write`
`clone/fork`	`(p)poll`	`recvfrom`	`socket`
`connect`	`(p)select`	`recvmsg`	`...`
`dup(2)`	`exit`	`send(to)`	

FITM_FD

- Designated FD (999) returned by `socket()`
- Special handling in hooked functions

FITM_FD - recv

Hook recv if `fd == FITM_FD`



AFL++ input via ShMem

Evaluation

Number of Executions

Fuzzer	Traces[a]	BBs[b]	Hangs	Crashes	Depth	Total Execs
AFLNet	17	5880	22	0	5	424.701
FitM	146	6158	0	0[c]	10	113.683.192[d]

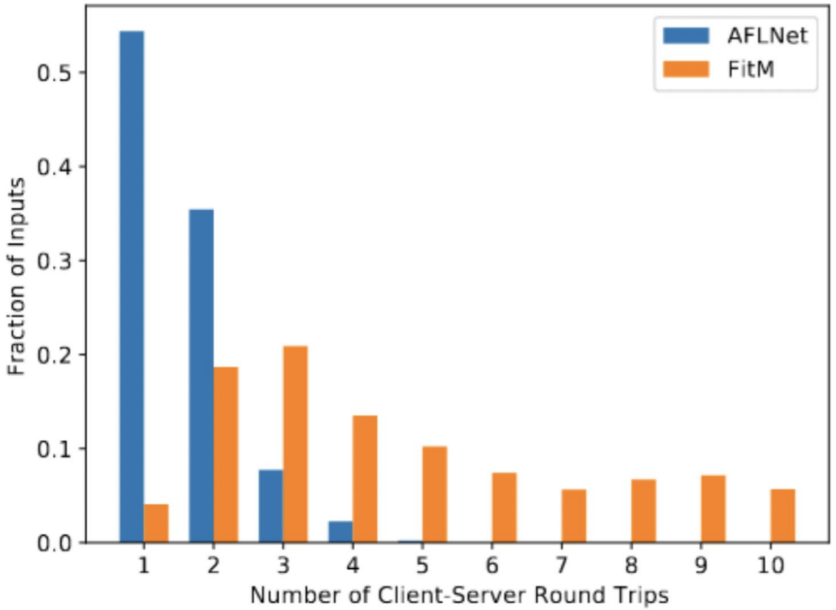
TABLE II: Fuzzing LightFTP for 15 hours in AFLNet & FitM

~267x faster exec speed from network emulation.

More protocol levels through snapshotting

- 1 core
- Intel i7-6850K
- 128GB Ram
- LightFTP v2.1 FTP server
- GNU Inteutils v1.9.4 FTP client
- No target patches / harness

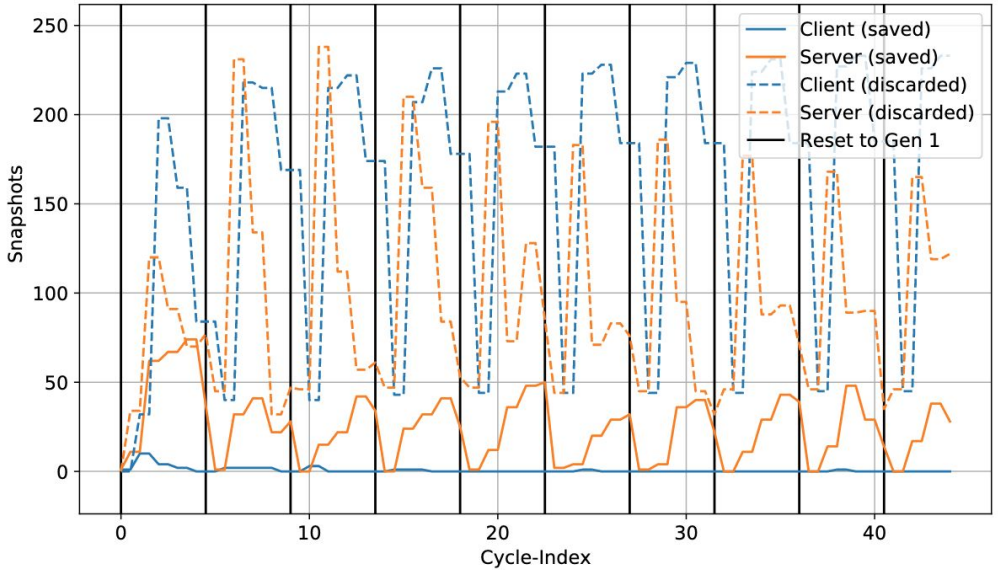
Fuzzed Protocol Stages



- Bigger fraction of inputs ends up in later stages of the protocol
- More state exploration

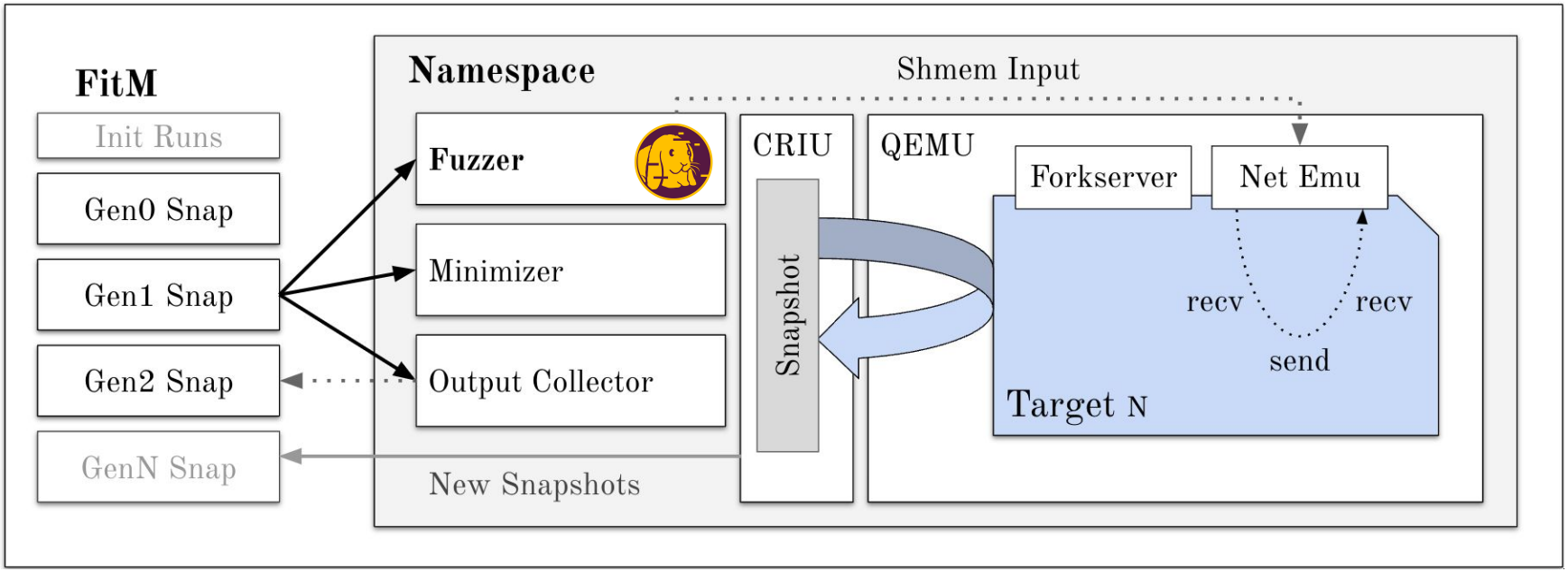
Filtering

- Retained snapshots per cycle trend downwards
- Filtering seems to work



Recap

FitM Building blocks



Conclusion

- We created FitM, a fuzzer for network interactions
- We fuzz client and server at the same time
- FitM emulates binaries and the network layer
- It uses snapshotting to reach deeper protocol states
- **Open Source** at https://github.com/fgsect/fitm

```
while (questions());

char buf[16];
strncpy(buf, ""
    "Thank you for your attention."
    "\n", sizeof(buf));
printf("%s", buf);
```

[1] https://upload.wikimedia.org/wikipedia/commons/thumb/4/45/Qemu_logo.svg/230px-Qemu_logo.svg.png
[2] https://static.openvz.org/artwork/CRIU-560px.png
[3] https://upload.wikimedia.org/wikipedia/commons/thumb/d/d5/Rust_programming_language_black_logo.svg/220px-Rust_programming_language_black_logo.svg.png

FITM_FD - socket

- Check if TCP socket
- Check if initial socket calls should be skipped
- Return FITM_FD

FITM_FD - send

Only if fd is FITM_FD

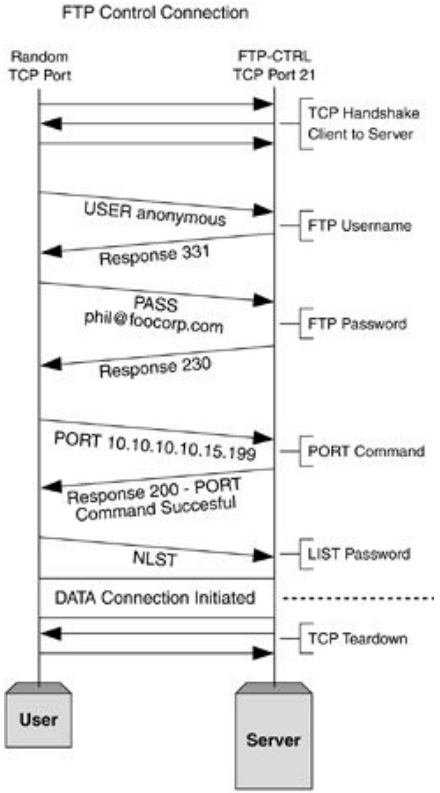
- Touch AFL_MAP: Emphasize send paths
- CREATE_OUTPUTS:
  - Env variable to control output generation
  - If true, write output to snapshot-specific file
  - If false, return len from args

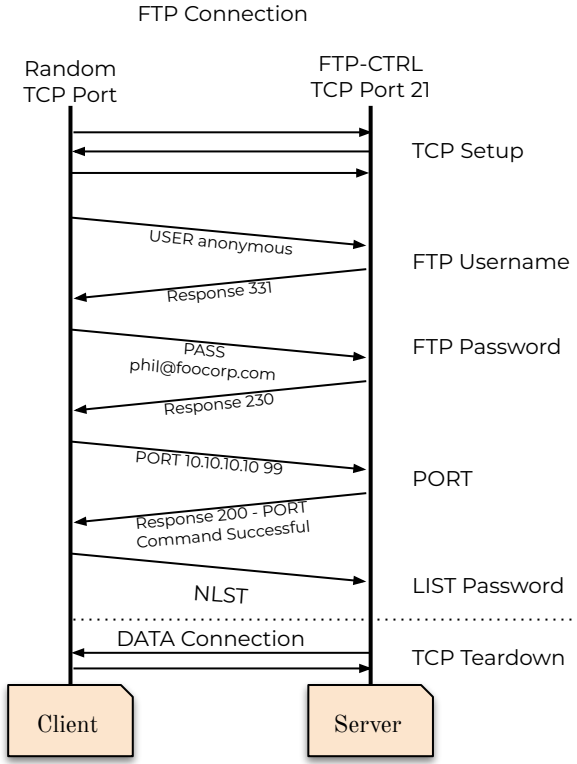
FITM_FD - recv

Only if fd is FITM_FD

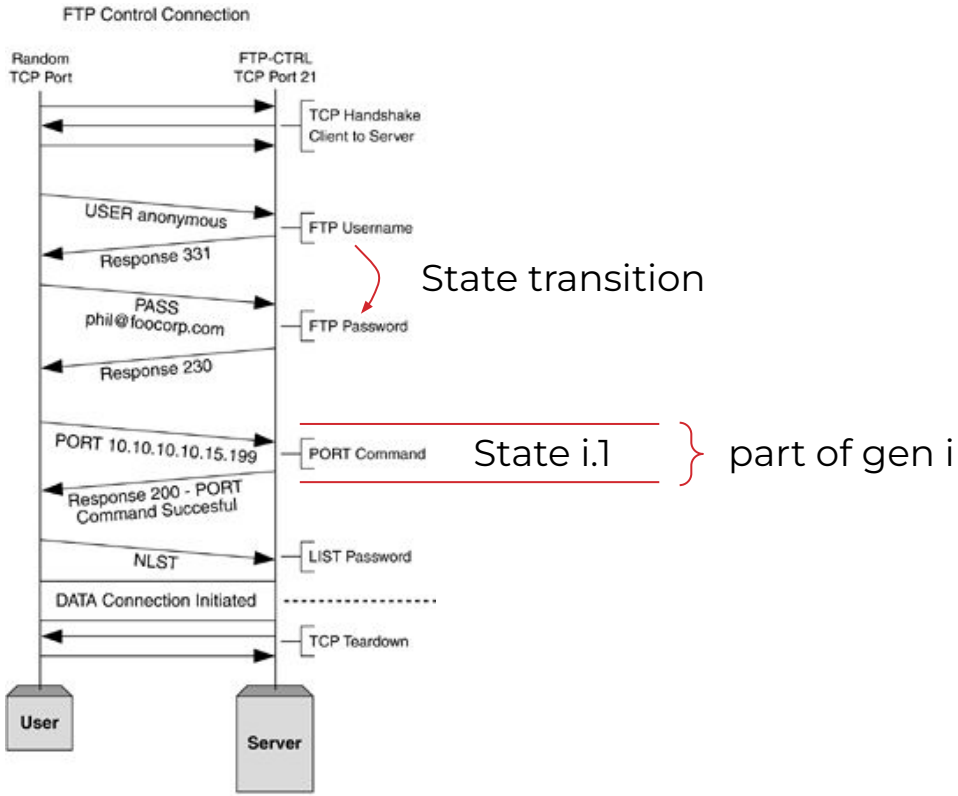
- Touch AFL_MAP: Emphasize read paths
- Check if previously called `send()`
- TIMEWARP_MODE:
  - Env variable to control snapshot generation
  - If true, snapshot and exit
  - If false, exit
- Start AFL forkserver
- Read input from file provided by FitM

Motivation - Coverage Guided Fuzzers





Protocols



State transition

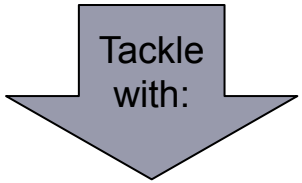
State i.1 } part of gen i

Available commands of an FTP server (HELP)

```
ABOR ACCT ALLO APPE CDUP CWD DELE EPRT
EPSV FEAT HELP LIST MDTM MKD MODE NLST
NOOP OPTS PASS PASV PORT PWD QUIT REIN
REST RETR RMD RNFR RNTO SITE SIZE SMNT
STAT STOR STOU STRU SYST TYPE USER XCUP
XCWD XMKD XPWD XRMD
```

Challenge #1 - State Explosion

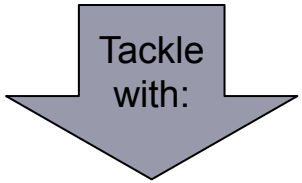
State Explosion: Each state has >1 children



- AFL-cmin
- Random select
- String distance filter on "produced output"

Challenge #2 - Dead Ends

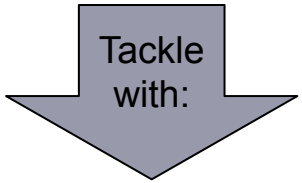
Dead Ends: error states, setup/teardown states



Random Restarts

Challenge #3 - Desync

Desync between generations



Cross generational input