HERMES

Do Minh Tuan - 20144861

Supervisor: Dr. Tran Quang Duc

Advisor: Dr. Nguyen Anh Quynh

Agenda

- Background
- Problems of current network fuzzer
- Hermes
- Result of experiments
- Conclusion

BACKGROUND

Motivation

- "IoT" connects machines and systems together
  - Increases the risk in security over network
- Protocols: MQTT, CoAP, XMPP, …
  - Huge attack surface for hacker to compromise the system
- Need a testing system to discover vulnerabilities

Security Vulnerability

- Weakness which can be exploited by cyber attack
- Window of vulnerability: 0-day, N-day
- IoT common's security bug:
  - Authentication
  - Insecure service
  - Privacy concern
- Memory corruption in protocol implementation

Fuzzing techniques

- Generate inputs to a program in order to
  - Exercise different parts of program
  - Induce undefined or undesired behaviors
- Fuzzing types: *black-box*, *white-box*, *grey-box*
- Evaluate the efficiency of a fuzzer:
  - Amount of discovered vulnerabilities
  - Efficiency with which vulnerabilities are discovered
  - Code coverage of the program under test

PROBLEMS

State-of-the-art

	AFL + Preeny	Fuzzy For Worm	Pulsar
Description	- Extend from AFL - De-socket app	- Replay traffics - Most advanced, full featured	- Stateful blackbox fuzzing - Use intelligent model to infer msg
Pros	Powerful features from AFL	- Simple to use - Smart mutation	Effective state exploration
Cons	- Reset session carefully - Not maintained anymore	- Slow speed - Unrelated components	Less accuracy in code coverage

Hermes

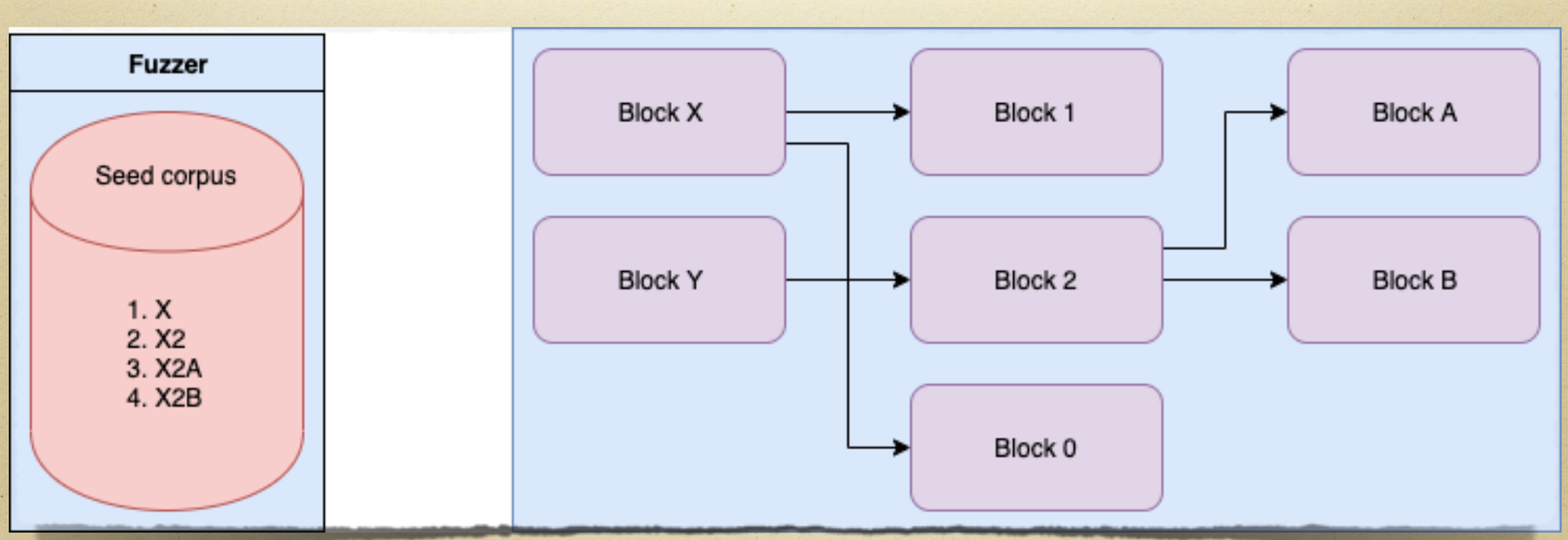
The next generation of protocol fuzzing

Hermes Fuzzer

- Use to fuzz protocol over network
- Crazy speed (100x faster than state-of-the-art)
- Context awareness
- Feedback driven fuzzing

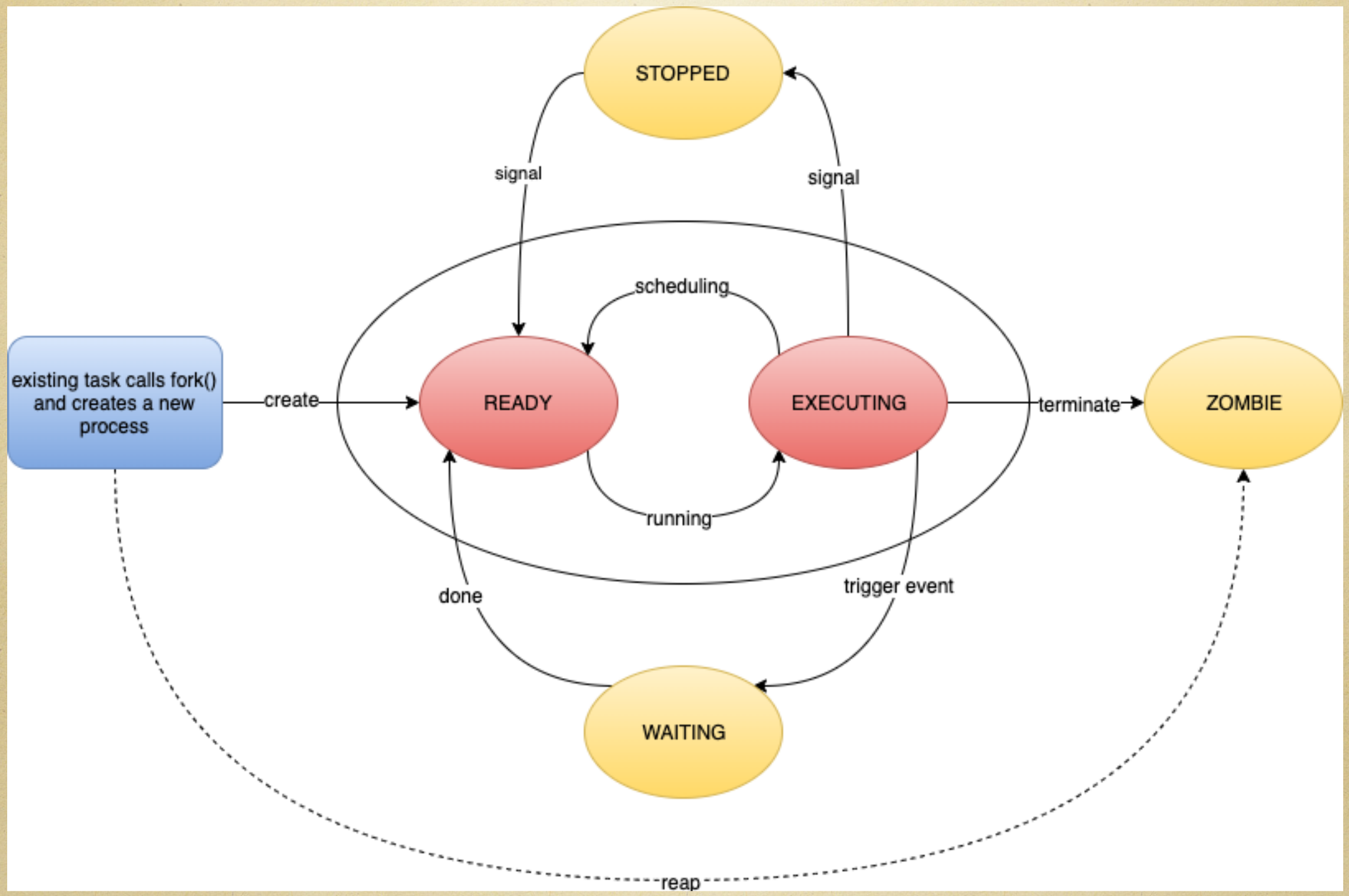
Feedback driven fuzzing

- Mutate input based on the feedback
  - It can find out “X2B” in just 4 steps



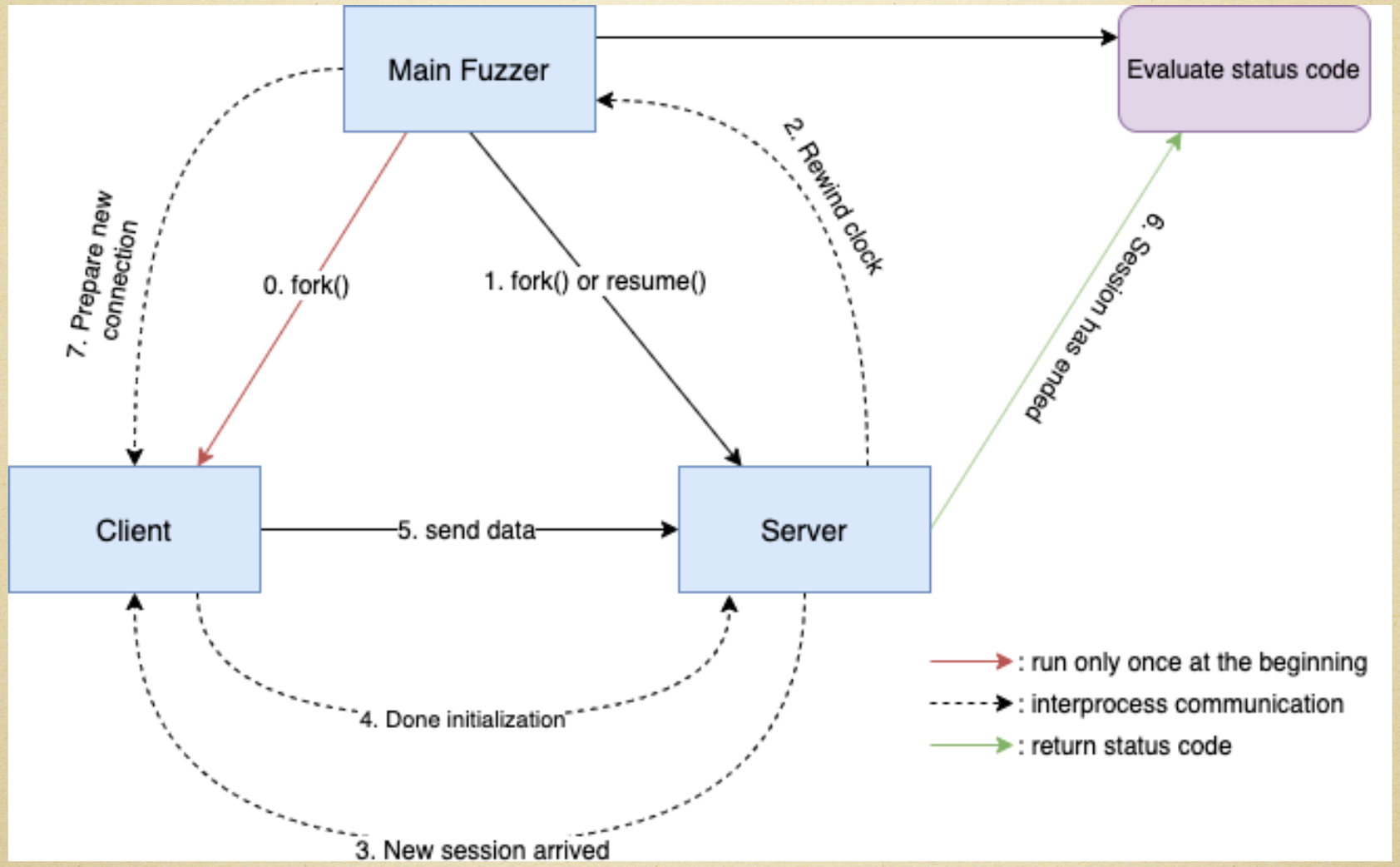
Technologies

- Pause / Resume a process in Linux
- Hooking system APIs
- Options for multiple kinds of protocol's format
- Synchronization among processes


STOPPED
signal
signal
scheduling
existing task calls fork() and creates a new process
create
READY
EXECUTING
terminate
ZOMBIE
running
done
trigger event
WAITING
reap

System Design

- 3 main components:
  - Main Fuzzer: sync processes, randomize input, conduct results
  - Fake client: take mutated input and send to target server
  - Target server: protocol implementation need to be tested
- Extend from American Fuzzy Lop
  - Inherited powerful features


Main Fuzzer
Evaluate status code
7. Prepare new connection
0. fork()
1. fork() or resume()
2. Rewind clock
6. Session has ended
Client
5. send data
Server
4. Done initialization
3. New session arrived
: run only once at the beginning
: interprocess communication
: return status code

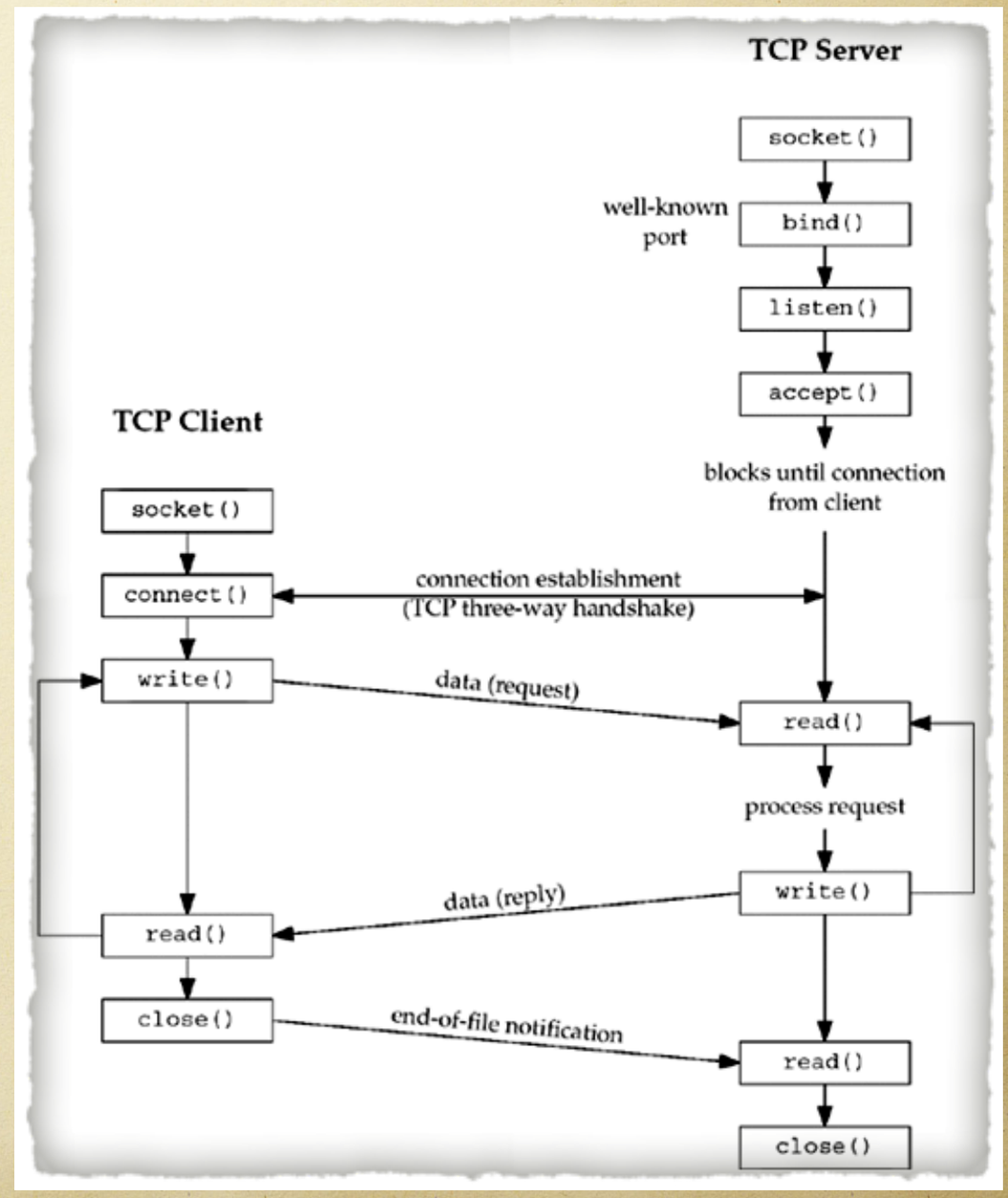
Limitations

- Custom network stack
- Session awareness
- Complete stateful fuzzing
- Encrypted protocols

Proof of Concepts

Echo server

- Server - client model
- Simple mechanism
- Multiple sessions
- Setup a trap to evaluate fuzzers



```
                     Hermes 1.0 (server)

– process timing ─────────────────────────────────────┬─ overall results ──
       run time : 0 days, 0 hrs, 0 min, 7 sec         │  cycles done : 44
  last new path : 0 days, 0 hrs, 0 min, 7 sec         │  total paths : 4
last uniq crash : 0 days, 0 hrs, 0 min, 7 sec         │ uniq crashes : 1
 last uniq hang : none seen yet                       │   uniq hangs : 0
– cycle progress ──────────────────┬─ map coverage ───┴────────────────────
 now processing : 3 (75.00%)       │    map density : 0.01% / 0.02%
paths timed out : 0 (0.00%)        │ count coverage : 1.00 bits/tuple
– stage progress ──────────────────┼─ findings in depth ───────────────────
 now trying : havoc                │ favored paths : 4 (100.00%)
stage execs : 468/512 (91.41%)     │  new edges on : 4 (100.00%)
total execs : 100k                 │ total crashes : 3 (1 unique)
 exec speed : 11.8k/sec            │  total tmouts : 0 (0 unique)
– fuzzing strategy yields ─────────┴───────────┬─ path geometry ──────────
  bit flips : 0/128, 1/124, 0/116              │    levels : 4
 byte flips : 0/16, 0/12, 0/4                  │   pending : 0
arithmetics : 3/1208, 0/0, 0/0                 │  pend fav : 0
 known ints : 0/79, 0/336, 0/176               │ own finds : 3
 dictionary : 0/0, 0/0, 0/0                    │  imported : n/a
      havoc : 0/58.1k, 0/39.6k                 │ stability : 100.00%
       trim : 33.33%/1, 0.00%                  ├──────────────────────────
───────────────────────────────────────────────┘          [cpu000: 16%]
```

```
ubuntu@ip-172-31-36-122:~/Tools/ffw/vulnserver$ ../ffw.py --fuzz
Basedir: /home/ubuntu/Tools/ffw
Config file: /home/ubuntu/Tools/ffw/vulnserver/config.py
mount: only root can use "--types" option
Start child: 0
Thread#  Fuzz/s   Count   Crashes
0 Start fuzzing...
0: 107.76       539      0  000
0: 112.99      1134      0  000
0: 117.10      1761      0  000
0: 116.74      2340      0  000
0: 117.54      2944      0  000
0: 118.18      3551      0  000
0: 118.76      4163      0  000
0: 118.32      4740      0  000
0: 118.11      5322      0  000
0: 119.44      5980      0  000
0: 119.85      6600      0  000
0: 119.87      7201      0  000
0: 119.41      7771      0  000
```

```
Reading data...
Splitting ngrams...
Calc indices...
Setup matrix...
to check: 2

                 _
  _ __  _   _| |___  __ _ _ __
 | '_ \| | | | / __|/ _` | '__|
 | |_) | |_| | \__ \ (_| | |
 | .__/ \__,_|_|___/\__,_|_|  v0.1-dev
 |_|

>>> Extracting DERRICK files from tcp-echo-server
>>> Generating PRISMA input files from tcp-echo-server
>>> Clustering data...
Error during clustering (not enough data?)
Cluster file not generated: /home/ubuntu/Tools/pulsar/models/tcp-echo-server/tcp-echo-server.cluster
Exiting learning module...
```

Chat-room

- Cloned from public repository
- Client uses command to register and send msgs
- *Hermes quickly found an buffer overflow vulnerability*

	Hermes	FFW	Pulsar
Testcases	180M	8M	345k
Paths	83	20	50
Bugs	2	1	1

MQTT broker

- Embedded web server library from Cesanta
- Implement MQTT protocol
- *Hermes has found* CVE-2019-19307

	Hermes	FFW	Pulsar
Testcases	140M	7M	200k
Paths	412	112	97
Bugs	4	1	0

Conclusion

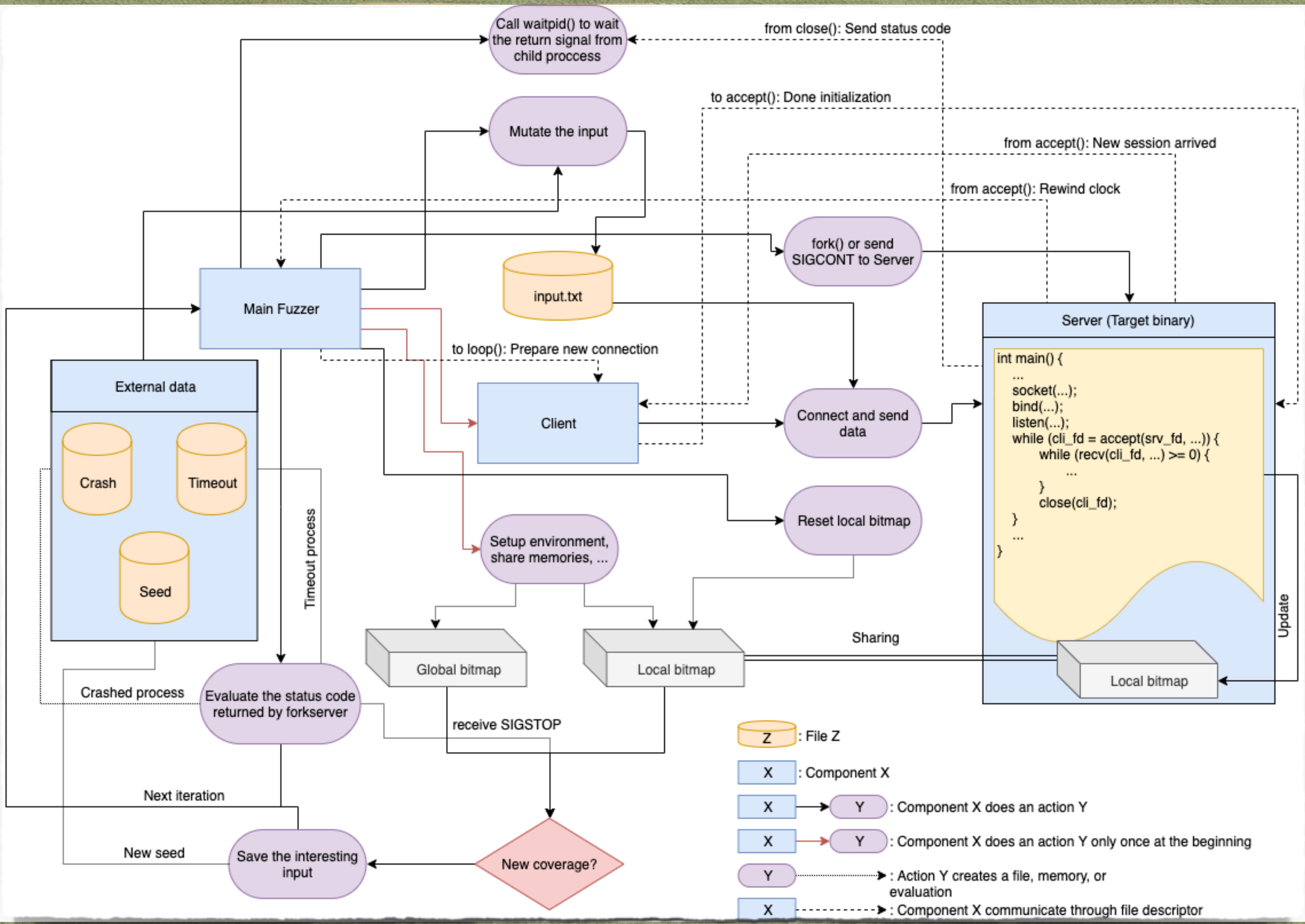
- Actual situation in IoT era
- Problems of current network fuzzing
- The solution: Hermes
- Comparison among fuzzers

Publications

- XCon - Beijing, China 2019
- T2 Conference - Helsinki, Finland 2019
- Insomni'hack - Geneva, Switzerland, 2020
- BlackHat Asia - Singapore, 2020

Q&A

Hermes Fuzzer's workflow


Call waitpid() to wait the return signal from child proccess
from close(): Send status code
to accept(): Done initialization
Mutate the input
from accept(): New session arrived
from accept(): Rewind clock
fork() or send SIGCONT to Server
input.txt
Main Fuzzer
Server (Target binary)
int main() {
...
socket(...);
bind(...);
listen(...);
while (cli_fd = accept(srv_fd, ...)) {
while (recv(cli_fd, ...) >= 0) {
...
}
close(cli_fd);
}
...
}
to loop(): Prepare new connection
External data
Client
Connect and send data
Crash
Timeout
Seed
Timeout process
Reset local bitmap
Setup environment, share memories, ...
Global bitmap
Local bitmap
Sharing
Local bitmap
Update
Crashed process
Evaluate the status code returned by forkserver
receive SIGSTOP
Next iteration
New seed
Save the interesting input
New coverage?
Z : File Z
X : Component X
X Y : Component X does an action Y
X Y : Component X does an action Y only once at the beginning
Y : Action Y creates a file, memory, or evaluation
X : Component X communicate through file descriptor