Christian Hopps
LabN Consulting, LLC

“chopps’ dev”

My Linux Kernel Dev Environment

A demo

- Automated Virtual (Munet) Topologies and VMs (qemu)
- Interactive
- Packet Captures
- Debugging with GDB
- Coverage
- Profiling - Flamegraphs
- Tracing

The Dev Source Code (IPTFS)

- CONFIG_XFRM_IPTFS=y ☺

- net/xfrm/xfrm_iptfs.c
  - non-static functions: xfrm_iptfs_*
  - static functions iptfs_*
  - 3 sections:
    - State
    - Ingress (output)
    - Egress (input)

The Code

- Linux:
  - Dev Env: https://github.com/LabNConsulting/iptfs-dev
    - kernel: https://github.com/LabNConsulting/iptfs-linux
    - iproute2: https://github.com/LabNConsulting/iptfs-iproute2
  - Dev Env using github actions to run regression/unit tests! ☺
- Testing/Tools:
  - munet: https://github.com/LabNConsulting/munet
  - wireshark: https://github.com/LabNConsulting/wireshark
  - cov.el: https://github.com/choppsv1/cov/commits/chopps/add-bg-overlay-tint

Tools

- Hardware/Topology: munet
- Testing Framework: pytest
- Used by above
  - Interactive: tmux (best) or SCREEN or X11
  - Profiling: perf and FlameGraph
  - Packet Captures: tshark and wireshark
  - Debugging: gdb
  - Coverage: native and cov.el

munet – network and hardware emulation

- Config file defines:
  - Topology (node and networks)
  - Kinds (common attributes for nodes)
  - Additional CLI commands
  - yaml/toml/json format
- Launches and configures nodes of the following types:
  - Linux Namespace
  - Container (docker/podman)
  - QemuVM
- Connects nodes together with LAN or P2P networks
  - Normally veth’s and bridges
  - Physical Interfaces also supported
    - Mapped into namespaces
    - SRIOV for QemuVM

munet cont.

- Uses TMUX, SCREEN or X11 to open:
  - Extendible Munet CLI
  - Shells in the namespace or containers
  - Consoles to Qemu VMs
  - Tail –f's of stdout/stderr for logs
- Capture packets on any network
- Profile running code (uses perf)
- Converage

```
#                                192.168.0.0/24
#   --+--------------------+------ mgmt0 ------+--------------------+---
#     | .1                 | .2                | .3                 | .4
#   +----+               +----+              +----+               +----+
#   | h1 | --- net0 --- | r1 | --- net1 --- | r2 | --- net2 --- | r1 |
#   +----+ .1        .2 +----+ .2        .3 +----+ .3        .4 +----+
#          10.0.0.0/24          10.0.1.0/24          10.0.2.0/24
```

```
topology:
  networks-autonumber: true
  networks:
    - name: mgmt0
      ip: 192.168.0.254/24
    - name: net0
      ip: 10.0.0.0/24
    - name: net1
      ip: 10.0.1.0/24
    - name: net2
      ip: 10.0.2.0/24
```

```
nodes:
   - name: h1
     kind: host
     connections: [ mgmt0, net0 ]
   - name: r1
     kind: linux
     connections: [ mgmt0, net0, net1 ]
   - name: r2
     kind: linux
     connections: [ mgmt0, net2, net1 ]
   - name: h2
     kind: host
     connections: [ mgmt0, net2 ]
```

```
kinds:
  - name: linux
    merge: [ "qemu" ]
    gdb-cmd: "/usr/bin/sudo -E gdb %CONFIGDIR%/../../output-linux/vmlinux"
    gdb-target-cmds: ["target remote %RUNDIR%/s/gdbserver"]
    gdb-run-cmds: ["c"]
    gdb-run-cmd: ["c"]
    qemu:
      kernel: "%CONFIGDIR%/../../output-linux/arch/x86/boot/bzImage"
      initrd: "%CONFIGDIR%/../../output-buildroot/images/rootfs.cpio.gz"
      sshkey: "%CONFIGDIR%/../../root-key"
      #cmdline-extra: "acpi=off idle=poll nokaslr"
      #cmdline-extra: "idle=poll nokaslr trace_buf_size=1024M"
      cmdline-extra: "idle=poll nokaslr"
      memory: "2048"
      kvm: true
      ncpu: 1
      console:
        timeout: 180
```

Setup the Environment

```
$ git clone git@github.com:LabNConsulting/iptfs-dev.git
$ cd iptfs-dev/

$ make setup
[... git clones linux, buildroot, and iproute2]
$ git clone https://github.com/brendangregg/FlameGraph

$ # initialize a python virtual environment
$ python3 -m venv venv
$ . venv/bin/activate
$ pip install -r python-requirements.txt
```

Run interactive

```
$ cd iptfs-dev/
$ sudo venv/bin/pytest tests/simplenet --pause
…
== PAUSING: before step 0: Before test network up ==
PAUSED, "cli" for CLI, "pdb" to debug, "Enter" to continue: cli

--- Munet CLI Starting ---


munet> con r1
…
```

Enable coverage and build

```
$ cd iptfs-dev/

$ # change to build coverage kernel
$ sed -i -e 's/^LINUXCONFIG/# &/;/linux-cov.config/s/^# L/L/' Makefile

$ # Enable profiling in linux/net/xfrm
$ echo "GCOV_PROFILE := y" >> linux/net/xfrm/Makefile

$ make
[... builds kernel, buildroot w/ custom iproute]
```

Run test generating coverage

```
$ cd iptfs-dev/
$ sudo venv/bin/pytest tests/simplenet --coverage
…
tests/simplenet/test_simplenet.py::test_policy_tun_up
-----------------------------------
2023-11-01 17:05:34,235 INFO: root: STEP 0: first ping
2023-11-01 17:05:34,242 INFO: root: STEP 1: second ping
2023-11-01 17:05:34,247 INFO: root: STEP 2: third ping
PASSED
…
2023-11-01 16:48:15,865 INFO: munet.l3qemuvm.r1: Saved coverage data in VM at /gcov-data.tgz
2023-11-01 16:48:15,940 INFO: munet.l3qemuvm.r1: Saved coverage data on host at /tmp/unet-
test/tests.simplenet.test_simplenet/r1/gcov-data.tgz
…
```

Coverage: Extract

```
$ cd iptfs-dev/
$ scripts/extract-cov.sh

extracting resutls from /tmp/unet-test/tests.simplenet.test_simplenet/r1/gcov-data.tgz…
…
Reading tracefile coverage.info
Extracting /home/chopps/w/demo/iptfs-dev/linux/net/xfrm/xfrm_iptfs.c
Extracted 1 files
Writing data to iptfs.info
Summary coverage rate:
  lines......: 31.7% (350 of 1104 lines)
  functions..: 43.8% (21 of 48 functions)
  branches...: no data found

$ # used by emacs cov.el
$ ln -s $(pwd)/iptfs.info linux/net/xfrm/coverage.info
```

Profiling: Flamegraphs

```
$ cd iptfs-dev/
$ # enable fast config
$ sed -i -e 's/^LINUXCONFIG/# &/;/linux-fast.config/s/^# L/L/' Makefile
$ make flame-clean
$ make flame
[ runs test, extracts perf data, creates flamegraph and downloads it to my mac ]
```

Debugging

```
$ cd iptfs-dev/
$ sudo venv/bin/pytest -v tests/simplenet --pause-at-end \
    --gdb=r1 --gdb-breakpoints=iptfs_enqueue
…
[ launches test, opens window with gdb on r1 kernel and sets a breakpoint ]
```

pytest customizations

```
--profile             Enable profiling if supported by test
--tracing             Enable tracing if supported by test
--cli-on-error        CLI on test failure
--coverage            Enable coverage gathering if supported
--gdb=HOST[,HOST...]  Comma-separated list of nodes to launch gdb on, or 'all'
--gdb-breakpoints=BREAKPOINT[,BREAKPOINT...]
                      Comma-separated list of breakpoints
--gdb-use-emacs       Use emacsclient to run gdb instead of a shell
--pcap=NET[,NET...]   Comma-separated list of networks to capture packets on, or 'all'
--pause               Pause after each test
--pause-at-end        Pause before taking munet down
--shell=NODE[,NODE...]
                      Comma-separated list of nodes to spawn shell on, or 'all'
--stdout=NODE[,NODE...]
                      Comma-separated list of nodes to open tail-f stdout window on, or 'all'
--stderr=NODE[,NODE...]
                      Comma-separated list of nodes to open tail-f stderr window on, or 'all'
```

Questions and Comments