Tracing packets in the kernel OVS datapath with Retis

Adrián Moreno
Paolo Valerio
Agenda

1. Introduction
2. What is retis? Feature Overview
3. Demonstration
4. Future plans
5. Contribution & contact
Introduction
The 3-D network tracing problem

Many places & components: need modularity

Packet mutates: need tracking

Many different packets: need filtering
Existing tools

- tcpdump: the original BPF use case!
- Dropwatch: laser focused on drops
- pwru: super useful
- bpftrace / perf / stap: super-flexible but a bit complex
What is retis?
What is retis?

A comprehensive network visibility and tracing tool that provides contextual information from several control and data paths.

It supports OVS’ kernel datapath from day 1 including upcall tracking using USDT probes.
Basic usage

$> retis collect --collectors skb,ovs --probe tp:net:netif_rx

520726797911702 [curl] 1096276 [tp] net:netif_rx
if 8 (p0) 10.244.2.4.47444 > 10.96.167.138.80 len 60 proto TCP (6) flags [S] seq 3114861420 win 65280

520726797925931 [curl] 1096276 [tp] openvswitch:ovs_dp_upcall
if 8 (p0) 10.244.2.4.47444 > 10.96.167.138.80 len 60 proto TCP (6) flags [S] seq 3114861420 win 65280
upcall (miss) port 2579844760 cpu 6

**Collectors**: select *what* data to extract

**Probes**: select *where* to look for packets

- Some are *explicit*
- Some are *automatic*
What is retis?

Existing collectors: skb

$> retis collect -c skb --skb-sections=eth,ip ...


0a:58:0a:f4:02:03 > 0a:58:0a:f4:02:01 ethertype IPv4 (0x0800) 10.244.2.4 >
10.244.2.3 ttl 63 tos 0x0 id 53570 off 0 [DF] len 52 proto TCP (6)

▪ Does not add any automatic probe
▪ Extracts skb & packet information
What is retis?

Existing collectors: ovs

$> retis collect -c ovs --ovs-track

163005033290036 [tp] openvswitch:ovs_dp_upcall
  upcall (miss) port 4036369011 cpu 3

- Automatically adds probe to several places in the kernel datapath
  - openvswitch:ovs_dp_upcall, openvswitch:ovs_do_execute_action, etc
- If “--ovs-track” is set, it also sets USDT probes in OVS
  - Requires OVS compiled with –enable-usdt-probes
- Extracts OVS information and performs upcall tracking
What is retis?

Existing collectors: ovs

- ovs-vswitchd
- handler threads
- recv_upcall
- OpenFlow Classification
- dpif_netlink_operate
- flow_exec
- flow_put
- queue_userspace_packet
- ovs_execute_actions
- ovs_do_execute_action
- pkt
- tp
- kp
- USDT
- nl sockets

Openvswitch
Kmod
### Existing collectors: **ovs III**

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>How tracking is done</th>
<th>Assumptions made</th>
</tr>
</thead>
<tbody>
<tr>
<td>(kp) queue_userspace_packet</td>
<td>(USDT) upcall_recv</td>
<td>Hash of first N bytes of packet</td>
<td>In the same netlink socket queue, packets are not duplicated</td>
</tr>
<tr>
<td>(USDT) upcall_recv</td>
<td>(USDT) flow_exec / flow_put</td>
<td>Order of events per thread on each upcall batch</td>
<td>Current ovs behavior: put/exec operation is done once per upcall</td>
</tr>
<tr>
<td>(USDT) flow_exec</td>
<td>(kp) ovs_execute_actions</td>
<td>Hash of first N bytes of packet</td>
<td>In the same netlink socket queue, packets are not duplicated</td>
</tr>
</tbody>
</table>
Existing collectors: **skb-tracking**

$> retis collect -c skb-tracking -p kprobe:ip_rcv$

162394033150650 [k] ip_rcv #93b24ea8dabaffff938401f80200 (skb 18446624793922523392)

- Does not add any automatic probe
- Internally tracks `struct sk_buff *` and adds unique packet identifiers
  - Detects clones and packet modifications (e.g. NAT)
What is retis?

Existing collectors: **skb-drop**

```bash
$> retis collect -c skb-drop
162035545368105 [tp] skb:kfree_skb drop (NETFILTER_DROP)
```

- Automatically adds probe to `raw_tracepoint: kfree_skb`
- Extracts drop_reason from any compatible function
What is retis?

Existing collectors: nft

```
$> retis collect -c nft --nft-verdicts=drop --allow-system-changes

1162724441463506 [k] __nft_trace_packet
   table global (101) chain egress (1) drop
```

- Automatically adds probe to __nft_trace_packet
- Needs to add a small chain to nft (hence “--allow-system-changes”)
- Extracts nft table, chain and verdict
  - It can be configured to only select events based on verdicts.
What is retis?

Existing collectors: ct

$> retis collect -c ct -p kprobe:tcp_v4_rcv

523071806115911 [handler3] 286541/284995 [k] tcp_v4_rcv

  ct_state NEW tcp (SYN_SENT) orig [10.244.2.4.39028 > 10.244.1.6.8080] reply
[10.244.1.6.8080 > 10.244.2.4.39028] zone 0

- Does not add any automatic probe
- Extracts conntrack entry information
<table>
<thead>
<tr>
<th>Collector name</th>
<th>Data it collects</th>
<th>Extra features</th>
<th>Automatic probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>skb</td>
<td>Packet information</td>
<td>Configurable fields</td>
<td>-</td>
</tr>
<tr>
<td>nft</td>
<td>Nftables context</td>
<td>Filter on verdict</td>
<td>__nft_trace_packet</td>
</tr>
<tr>
<td>skb-drop</td>
<td>Drop reason</td>
<td>-</td>
<td>skb:kfree_skb drop</td>
</tr>
<tr>
<td>skb-tracking</td>
<td>Unique packet tracking ID</td>
<td>Tracks packets</td>
<td>-</td>
</tr>
<tr>
<td>ovs</td>
<td>OpenvSwitch information</td>
<td>Tracks upcalls</td>
<td>Many (kernel and USDT)</td>
</tr>
<tr>
<td>ct</td>
<td>Conntrack entry</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Profiles

$> \text{retis collect} \ --\text{collectors=skb,skb-drop} \ -p \text{kprobe:udp_rcv} \ \\
-\text{p kprobe:ip_rcv} \ -\text{p kprobe:ip\_finish\_output2} \ \\
-\text{p kprobe:napi\_gro\_receive} \ -\text{p kprobe:inet\_gro\_receive} \ \\
-\text{p kprobe:udp\_gro\_receive} \ ...$

- We can end up with long command lines
- Kernel knowledge needed

:(
What is retis?

version: 1.0
name: udp
about: Probe the UDP stack
collect:
  - args:
  collectors: skb, skb-drop
  probe:
    - kprobe:udp_rcv
    - kprobe:ip_rcv
    - kprobe:ip_finish_output2
    - tp:net:napi_gro_receive_entry
    - kprobe:inet_gro_receive
    - kprobe:udp_gro_receive

Profiles

$> retis --profile udp collect

- Easy to share yaml format
- Can express system dependencies
  - Kernel version requirements
  - Presence of symbols
- It translates into CLI args which are merged with the ones provided by user
What is retis?

Pcap filtering

```bash
$> retis collect -f "host example.com and tcp[tcpflags] & tcp-fin != 0 and port 80"
```

- Essentially filters that work with tcpdump work with retis
  - cBPF filters are generated using libpcap and then converted to eBPF
  - The resulting raw program is then injected into the program to be attached.
  - MAC header is required (although the limitation will be lifted)
- Integrated with skb-tracking:
  - If a packet matches the filter it gets tracked, meaning that if the packet gets manipulated the tool is still able to collect events for it.
- Integrated with ovs-tracking
What is retis?

Events and post-processing

$> \text{retis collect -c nft --out retis.data}

- Events are just json!
What is retis?

sort

First event of this packet

Subsequent events of the same packet

$> retis sort retis.data

155943457666870 [k] ip_rcv #8dd46a663736ffff9383209cae00 (skb 18446624794873888768) n 0
  192.168.100.2 > 172.100.100.100 len 84 proto ICMP (1) type 8 code 0
  + 155943457677661 [k] nf_conntrack_in #8dd46a663736ffff9383209cae00 (skb 18446624794873888768) n 1
    192.168.100.2 > 172.100.100.100 len 84 proto ICMP (1) type 8 code 0
    + 155943457682062 [k] nf_conntrack_icmp_packet #8dd46a663736ffff9383209cae00 (skb 18446624794873888768) n 2
      192.168.100.2 > 172.100.100.100 len 84 proto ICMP (1) type 8 code 0
    + 155943457685500 [k] nf_nat_ipv4_pre_routing #8dd46a663736ffff9383209cae00 (skb 18446624794873888768) n 3
      192.168.100.2 > 172.100.100.100 len 84 proto ICMP (1) type 8 code 0
Demo time!
What’s next?
Future plans

▸ Planned for next release (eoy):
  • pcap-ng module
  • BTF-based metadata filtering
▸ Containers
▸ Python integration!
  • Stable python API to access event data
▸ TUI?
▸ OVS datapath actions
▸ Improve ovs-vswitchd tracking?
▸ https://github.com/retis-org/retis/issues
Contributions welcomed!

- Contact us!
  - https://github.com/retis-org/retis
  - IRC: #retis (Libera.Chat)

- New collectors, profiles, sub-commands, use-case suggestions, etc...
Thank you!