Tracing packets in OVS: an update on Retis

Paolo Valerio Adrián Moreno

Red Hat

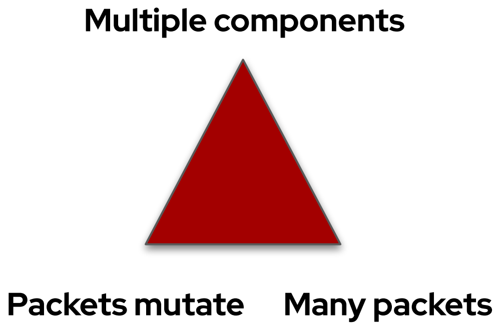
Table of Contents

Introduction

Problem statement

Existing tools are:

- Very specific in what they do (e.g. tcpdump, dropwatch).
- Powerful but sometimes a bit complex (e.g. stap, bpftrace).
- Focused on kernel only (e.g. no direct ovs)



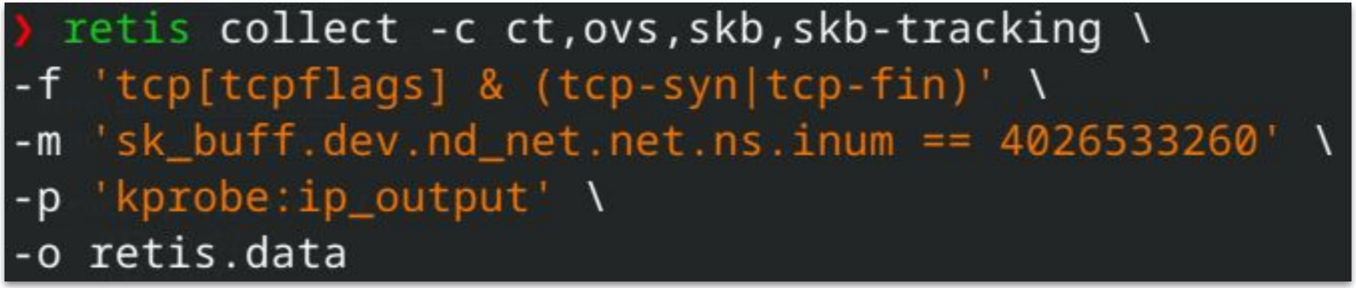
Red Hat

Retis in a nutshell

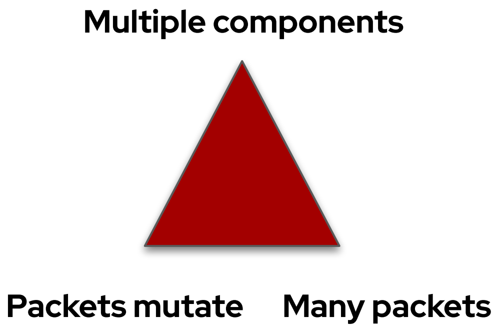
Retis is a network tracing tool designed to enhance visibility in the Linux networking stack and different control and/or datapaths.

- Includes advanced filtering capabilities.
- Dumps packets from functions and tracepoints consuming sk_buff.
- Tracks packets.
- **Supports OvS** kernel datapath with upcall tracking.
- And much more (ct, nft, profiles, stack traces, pcap, …)

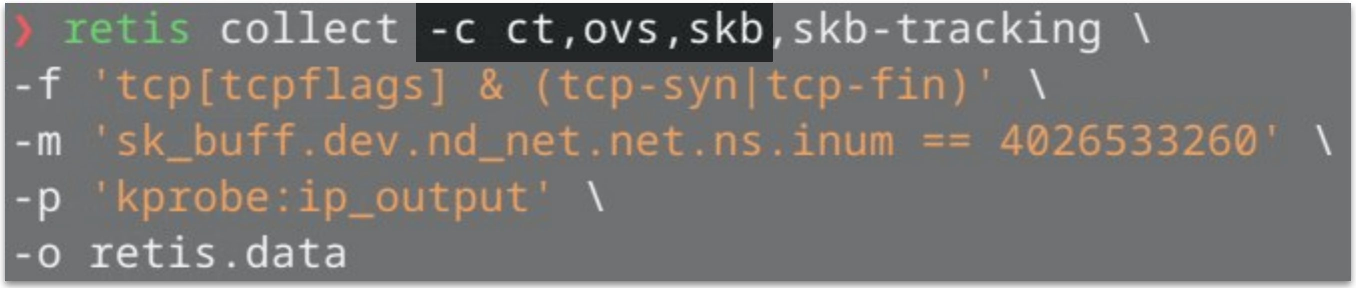
Usage (collect)



```
❯ retis collect -c ct,ovs,skb,skb-tracking \
-f 'tcp[tcpflags] & (tcp-syn|tcp-fin)' \
-m 'sk_buff.dev.nd_net.net.ns.inum == 4026533260' \
-p 'kprobe:ip_output' \
-o retis.data
```

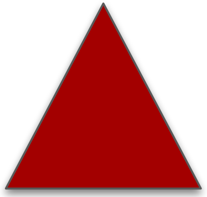


Usage (collect)



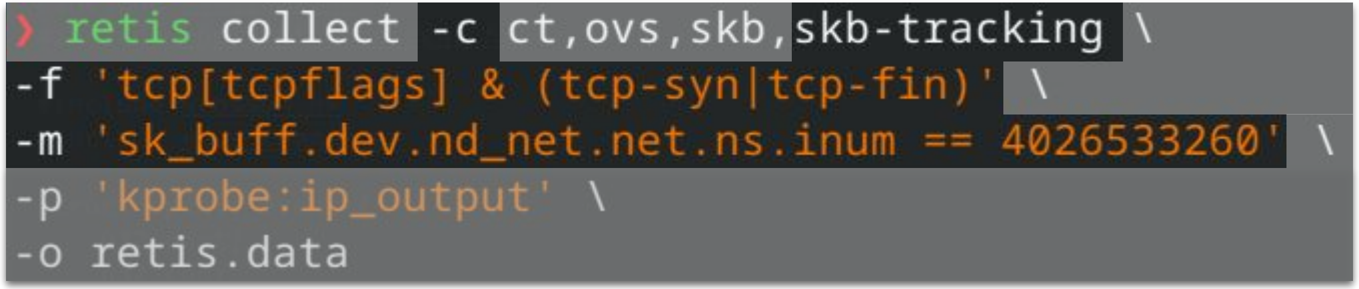
```
❯ retis collect -c ct,ovs,skb,skb-tracking \
-f 'tcp[tcpflags] & (tcp-syn|tcp-fin)' \
-m 'sk_buff.dev.nd_net.net.ns.inum == 4026533260' \
-p 'kprobe:ip_output' \
-o retis.data
```

Multiple components

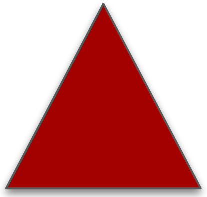


- Every collector retrieves specific informations
- skb, ct, ovs, nft, skb-drop
- Some collectors instrument probes

Usage (collect)



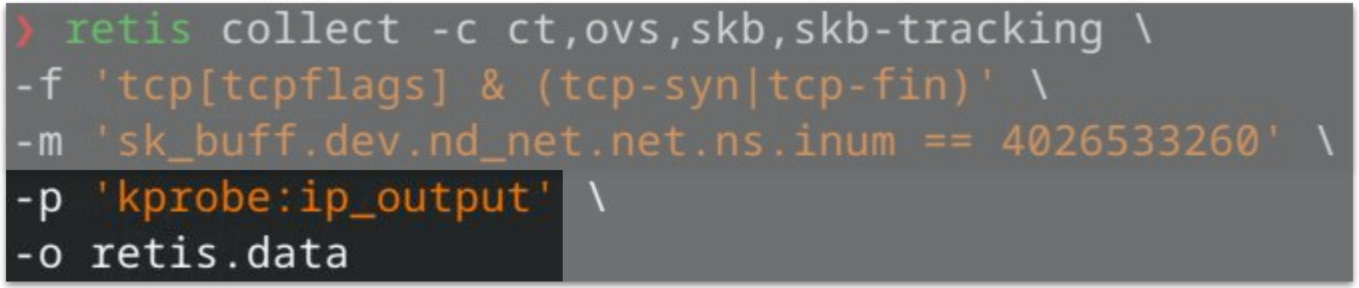
```
❯ retis collect -c ct,ovs,skb,skb-tracking \
-f 'tcp[tcpflags] & (tcp-syn|tcp-fin)' \
-m 'sk_buff.dev.nd_net.net.ns.inum == 4026533260' \
-p 'kprobe:ip_output' \
-o retis.data
```



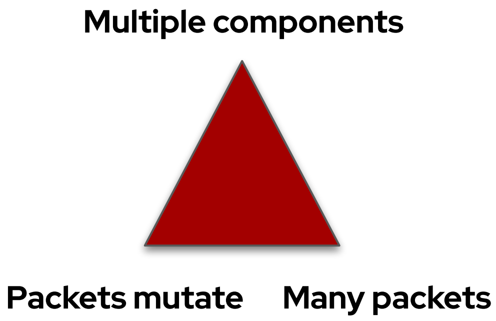
Packets mutate **Many packets**

- Filtering
  - Packet filtering (pcap-filter)
  - Meta-data filtering
    - ***'sk_buff._nfct:~0x7:nf_conn.mark'***
    - ***(nf_conn *)(skb->_nfct & NFCT_PTRMASK)->mark != 0***
- Tracking allows to follow packets across the probes

Usage (collect)

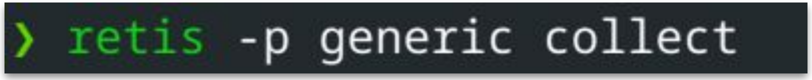


```
❯ retis collect -c ct,ovs,skb,skb-tracking \
-f 'tcp[tcpflags] & (tcp-syn|tcp-fin)' \
-m 'sk_buff.dev.nd_net.net.ns.inum == 4026533260' \
-p 'kprobe:ip_output' \
-o retis.data
```

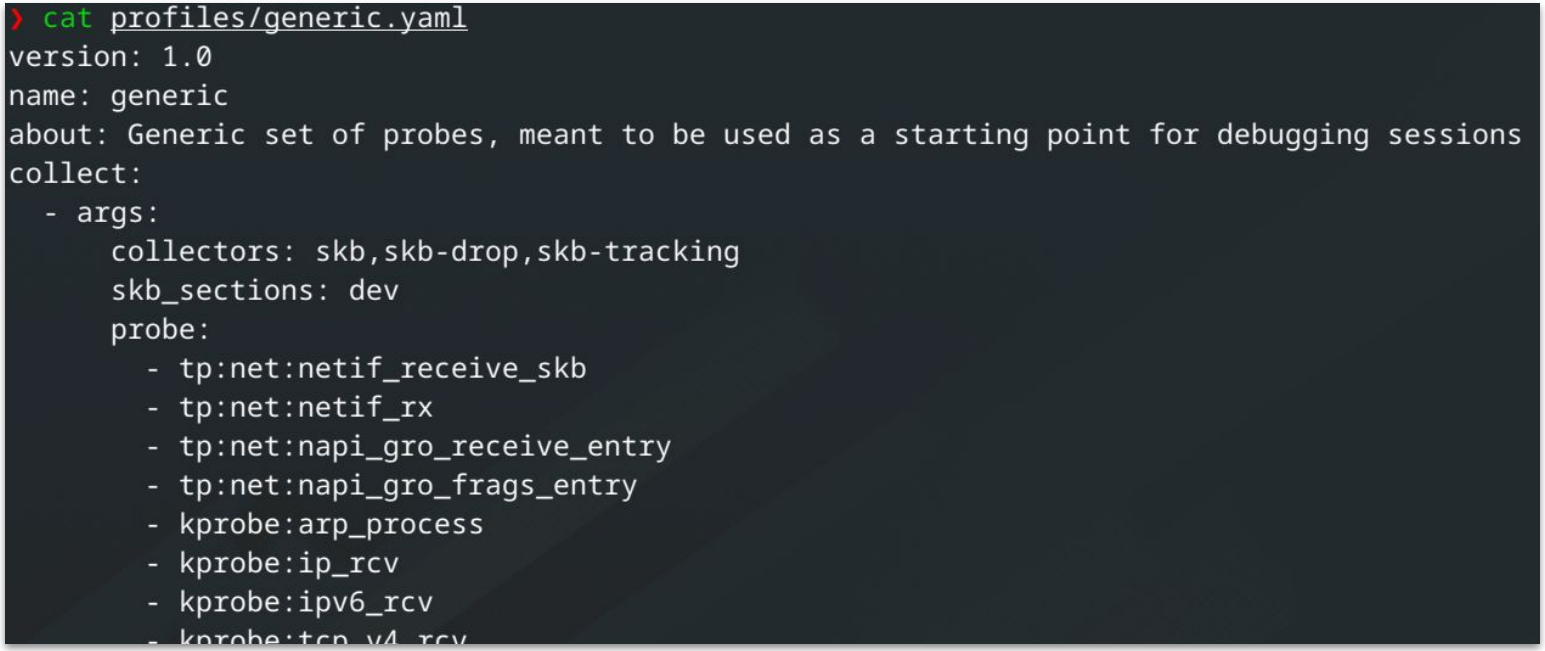


- From where data should be collected
- Wildcard support (e.g. *-p kprobe:tcp_**)
- Store events in json

Usage (profiles)

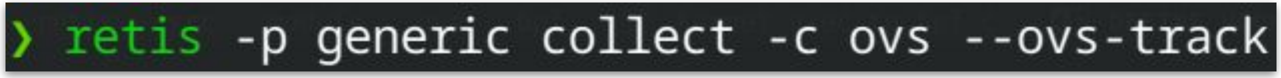


```
❯ retis -p generic collect
```



```
❯ cat profiles/generic.yaml
version: 1.0
name: generic
about: Generic set of probes, meant to be used as a starting point for debugging sessions
collect:
  - args:
      collectors: skb,skb-drop,skb-tracking
      skb_sections: dev
      probe:
        - tp:net:netif_receive_skb
        - tp:net:netif_rx
        - tp:net:napi_gro_receive_entry
        - tp:net:napi_gro_frags_entry
        - kprobe:arp_process
        - kprobe:ip_rcv
        - kprobe:ipv6_rcv
        - kprobe:tcp_v4_rcv
```

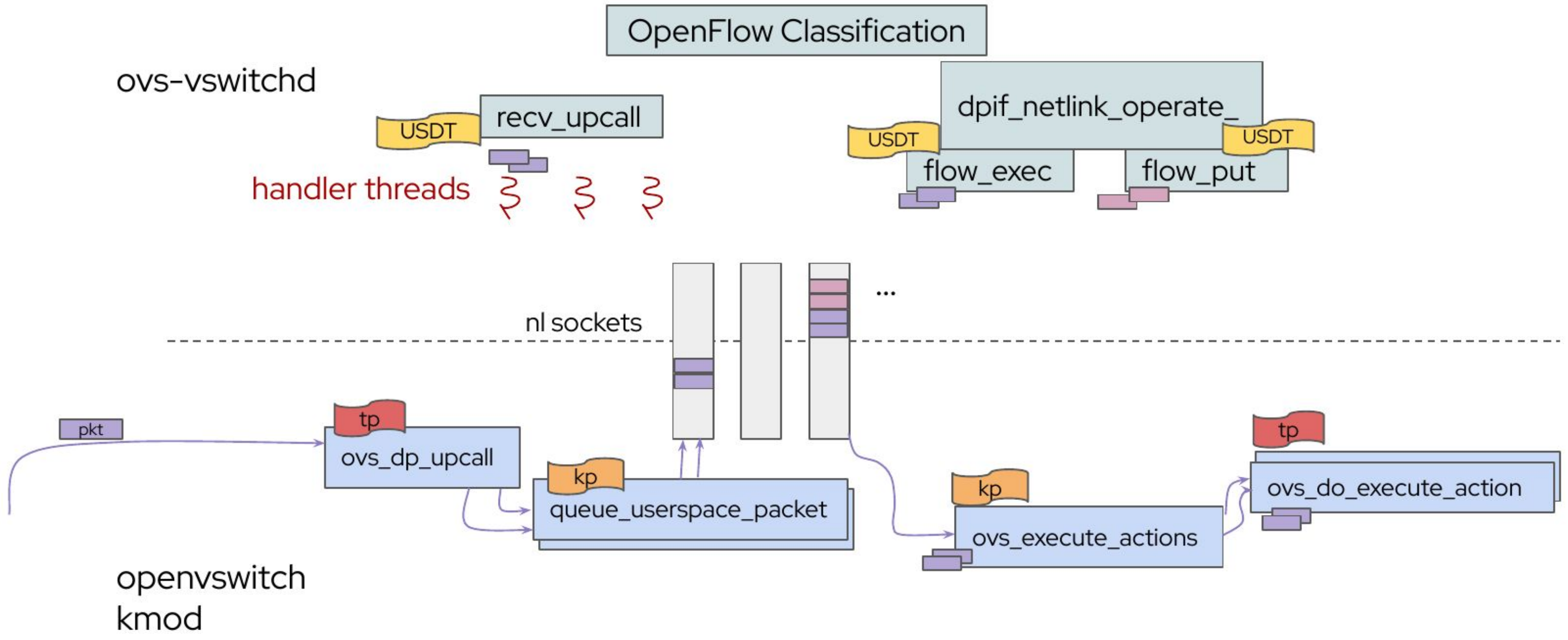
Usage (ovs)



```
❯ retis -p generic collect -c ovs --ovs-track
```

- eBPF programs (tp, kprobes and kretprobes) are automatically attached to specific places in the datapath
- Also eBPF programs are attached to User-space Static Defined Tracepoints
  - Requires vswitchd to be compiled with *--enable-usdt-probes*
- Tracks packets both in the **datapath and upcalls**!

Usage (ovs)

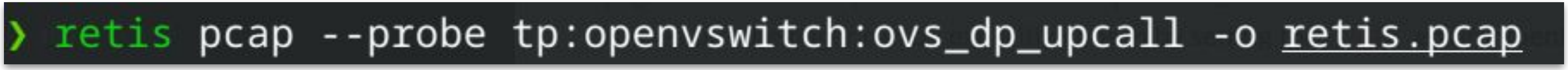

OpenFlow Classification
ovs-vswitchd
USDT
recv_upcall
handler threads
dpif_netlink_operate_
USDT
flow_exec
USDT
flow_put
nl sockets
...
pkt
tp
ovs_dp_upcall
kp
queue_userspace_packet
kp
ovs_execute_actions
tp
ovs_do_execute_action
openvswitch
kmod

Usage (sort)

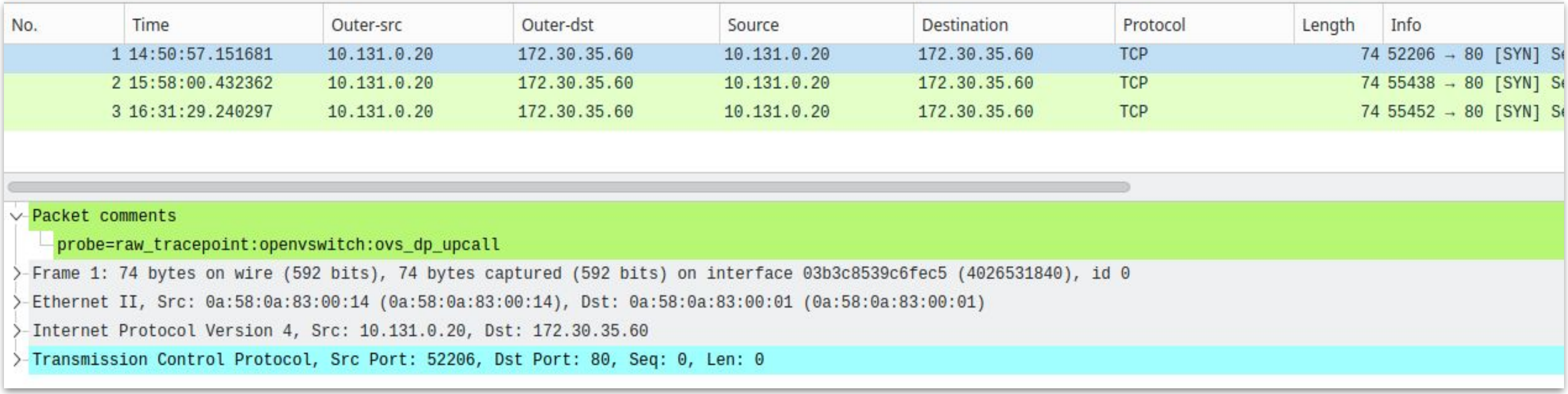
```
❯ retis sort
```

```
[...]
  ↳ 7588668917790 (2) [curl] 62932 [tp] openvswitch:ovs_dp_upcall #6e6dfe80260ffff9cc44eec1800 (skb
ffff9cc44d7f74f0) n 12
     ns 4026531840 if 24 (4db551c149eb83b) rxif 24 0a:58:0a:80:02:12 > 0a:58:0a:80:02:01 ethertype IPv4 (0x0800)
10.128.2.18.48010 > 172.30.104.182.80 ttl 64 tos 0x0 id 56102 off 0 [DF] len 60 proto TCP (6) flags
[...]

  ↳ 7588669082491 (0) [handler14] 1069/1874 [u] dpif_recv:recv_upcall (ovs-vswitchd) #6e6dfe80260ffff9cc44eec1800
(skb ffff9cc44d7f74f0) n 15
     upcall_recv q 1721070243 pkt_size 74
[...]

  ↳ 7588669222251 (0) [handler14] 1069/1874 [tp] openvswitch:ovs_do_execute_action #6e6dfe80260ffff9cc44eec1800
(skb ffff9cc44d7f74f0) n 18
     [...]
     exec ct zone 31 nat(dst=10.128.2.17:8080) commit,hash q 1721070243
```

Usage (pcap)



```
❯ retis pcap --probe tp:openvswitch:ovs_dp_upcall -o retis.pcap
```



```
No.   Time              Outer-src    Outer-dst     Source       Destination   Protocol  Length Info
    1 14:50:57.151681   10.131.0.20  172.30.35.60  10.131.0.20  172.30.35.60  TCP           74 52206 → 80 [SYN] Se
    2 15:58:00.432362   10.131.0.20  172.30.35.60  10.131.0.20  172.30.35.60  TCP           74 55438 → 80 [SYN] Se
    3 16:31:29.240297   10.131.0.20  172.30.35.60  10.131.0.20  172.30.35.60  TCP           74 55452 → 80 [SYN] Se

∨ Packet comments
    probe=raw_tracepoint:openvswitch:ovs_dp_upcall
> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 03b3c8539c6fec5 (4026531840), id 0
> Ethernet II, Src: 0a:58:0a:83:00:14 (0a:58:0a:83:00:14), Dst: 0a:58:0a:83:00:01 (0a:58:0a:83:00:01)
> Internet Protocol Version 4, Src: 10.131.0.20, Dst: 172.30.35.60
> Transmission Control Protocol, Src Port: 52206, Dst Port: 80, Seq: 0, Len: 0
```

Features coming soon (1.5)

python support

- python bindings: using pyo3.rs
- Built-in shell: for quick exploration
- python library: available in PyPi

Red Hat

python bindings

- Feels like a dict(string, *EventSection*)
- **sections()** lists available sections
- **show()** helper returns the string representation

```
>>> event.__class__
<class 'builtins.Event'>
>>> "skb" in event
True
>>> skb = event["skb"]
>>> skb.sections()
['skb', 'common', 'kernel', 'skb-tracking']
>>> event.show()
'488378974516873 (7) [curl] 3947832 [k] ip_output #1bc2d974ab689ffff99dd8193b480
(skb ffff99dd2ce8ace8)\n  10.244.1.3.43878 > 10.96.66.141.80 ttl 64 tos 0x0 id
64803 off 0 [DF] len 60 proto TCP (6) flags [S] seq 547727953 win 65280'
```

python bindings: EvenSections

- Access fields directly
- **show()** helper formats the section

```
>>> skb.__class__
SkbEvent
>>> skb
SkbEvent { eth: None, arp: None, ip: Some(SkbIpEvent { saddr: "10.244.1.3",... }
>>> skb.ip.saddr
"10.244.1.3"
>>> skb.show()
'10.244.1.3.43878 > 10.96.66.141.80 ttl 64 tos 0x0 id 64803 off 0 [DF] len 60 proto
TCP (6) flags [S] seq 547727953 win 65280'
```

python bindings: File readers

- Read both unsorted or sorted event files
- **EventFile**: reads files and determines type (sorted or unsorted)
  - **EventReader:** Iterate through events
  - **Series Reader:** Iterate through event series (sorted)

```
>>> file = EventFile("retis.data")
>>> file.__class__
EventFile
>>> events = file.events()
>>> events.__class__
EventReader
>>> for e in events:
   print(e.show())
```

```
>>> file = EventFile("sorted_retis.data")
>>> file.__class__
EventFile
>>> series = file.series()
>>> series.__class__
SeriesReader
>>> for s in series:
    print("A new series starts")
    for e in s:
       print(e.show())
```

Embedded python executor

- A global variable called **reader** of type *EventFile* is available:

```
$ retis python myscript.py
Event from: 10.244.2.5
Event from: 10.244.2.5
Event from: 10.244.2.5
...
```

```
for event in reader.events():
    if "skb" in event:
        saddr = event["skb"].ip.saddr
        print(f"Event from: {saddr}")
```

Embedded python shell

- If no script provided, it drops to an interactive shell

```
$ retis python
Python 3.12.7 (main, Oct  1 2024, 00:00:00) [GCC 14.2.1 20240912 (Red Hat
14.2.1-3)] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> for e in reader.events():
...   print(e.show())
```

Python library

- Available in PyPi (*pip install retis*)

```python
from retis import EventFile

reader = EventFile("retis.data")
for event in reader.events():
      if "skb" in event:
          saddr = event["skb"].ip.saddr
          print(f"Event from: {saddr}")
```

OVS flow enrichment

- ovs module now (kret)probes **ovs_flow_tbl_lookup_stats**
  - Retrieves, ufid, flow and acts pointers

```
488378974556131 (7) [curl] 3947832 [kr] ovs_flow_tbl_lookup_stats
  ufid 1150e2a5-518c-4170-8519-07d7800d2cbc hit (mask/cache) 4/0 flow ffff99dd4f348e38
sf_acts ffff99e20c783600
```

OVS flow enrichment

- **--ovs-enrich-flows** option
- At runtime, connect to OVS via unixctl
  - "dpctl/get-flow"
  - "ofproto/detrace" (if available)
- Connections are throttled and results cached
- A new event is generated, only one per (UFID, flow *, sf_acts *)

OVS flow enrichment

```
488378977879200
  ufid:1150e2a5-518c-4170-8519-07d7800d2cbc
recirc_id(0),in_port(6),eth(src=0a:58:0a:f4:01:03,dst=0a:58:a9:fe:01:01),eth_type(0x0800),ipv4(src=
10.244.1.3,dst=10.96.66.141,proto=6,frag=no),tcp(dst=80), packets:22, bytes:1776, used:0.001s,
flags:SFP., actions:ct(zone=17,nat),recirc(0x14)
  openflow:
      cookie=0x876a89c8, table_id=8, duration=16982s, n_packets=72, n_bytes=6209,
n_offload_packets=0, n_offload_bytes=0,
priority=50,metadata=0x5,actions=set_field:0/0x1000->reg10,resubmit(,73),move:NXM_NX_REG10[12]->NXM
_NX_XXREG0[111],resubmit(,9)
      …
```

OVS flow enrichment + sort

```
488378974516873 (7) [curl] 3947832 [k] ip_output ..
  10.244.1.3.43878 > 10.96.66.141.80 ttl 64 tos 0x0 id ...
  ↳ 488378974524485 (7) [curl] 3947832 [tp] net:net_dev_queue ..
      if 2 (eth0) 10.244.1.3.43878 > 10.96.66.141.80 ttl 64 tos 0x0 id
                                          [...]
  ↳ 488378974556131 (7) [curl] 3947832 [kr] ovs_flow_tbl_lookup_stats .. #
      ufid 1150e2a5-518c-4170-8519-07d7800d2cbc hit (mask/cache) 4/0 flow ffff99dd4f348e38 sf_acts
ffff99e20c783600
      odpflow
recirc_id(0),in_port(6),eth(src=0a:58:0a:f4:01:03,dst=0a:58:a9:fe:01:01),eth_type(0x0800),ipv4(src=10
.244.1.3,dst=10.96.66.141,proto=6,frag=no),tcp(dst=80), packets:22, bytes:1776, used:0.001s,
flags:SFP., actions:ct(zone=17,nat),recirc(0x14)
      openflow
            cookie=0x3966198f, duration=16982s, n_packets=50, n_bytes=3829, n_offload_packets=0,
n_offload_bytes=0,
priority=100,in_port=5,actions=set_field:0x11/0xffff->reg13,set_field:0x1->reg11,set_field:0x3->reg12
,set_field:0x5->metadata,set_field:0x3->reg14,set_field:0/0xffff0000->reg13,resubmit(,8)
```

Limitations

- We don’t have a good way of tracking deletions and updates
  - We use (*struct sw_flow **) and (*struct sf_acts **) to detect when the datapath flow has changed
- We cannot show the datapath flow for upcalls because of:

```
OVS_USDT_PROBE(dpif_netlink_operate__, op_flow_put,
               dpif, put, &flow, &aux->request);
```

- We might miss some flows

Demo

https://asciinema.org/a/690359

Thank you

IRC: retis @liberachat

https://github.com/retis-org/retis/

https://retis.readthedocs.io/en/stable/

https://quay.io/repository/retis/retis?tab=tags

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat