RFC 4301 “Populate From Packet” (PFP) in Linux

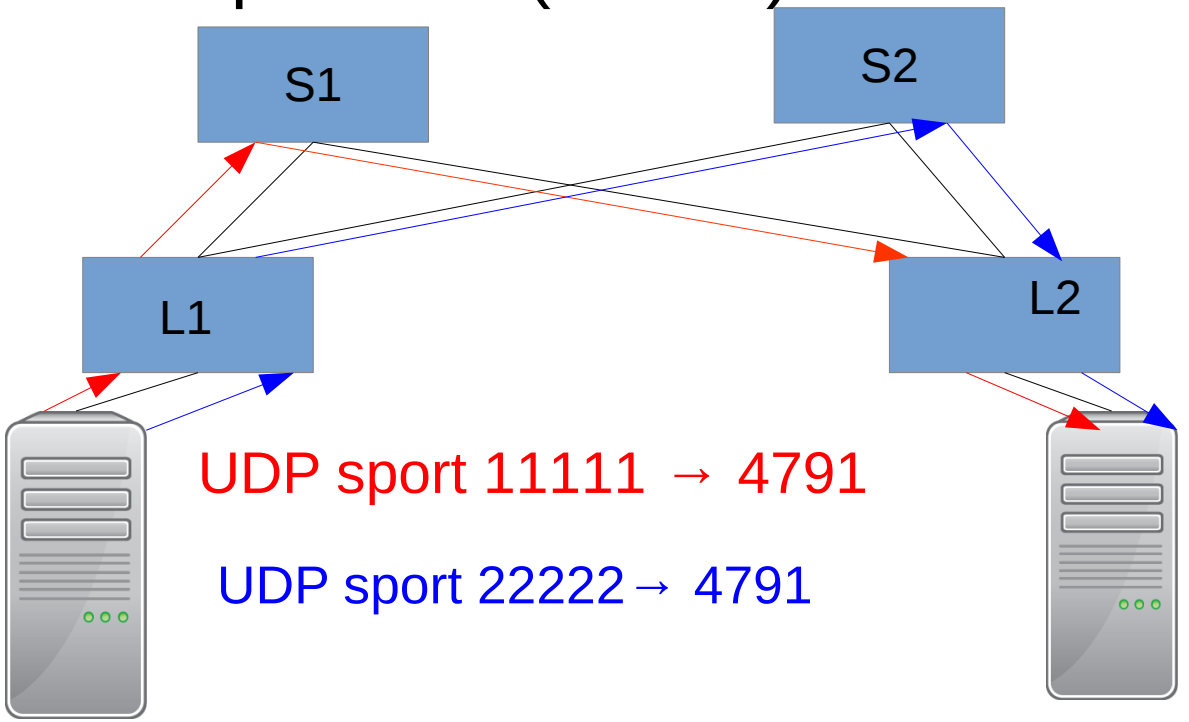
Sowmini Varadhan (sowmini.varadhan@oracle.com)

Agenda

- Problem description: what is this and why do we need it?
  - Follows up on discussion at http://swan.libreswan.narkive.com/Tjgazg3z/ipsec-pfp-support-on-linux
  - Entropy and RSS for performance
  - How to retain entropy after IPsec
- Use-cases: RDS-TCP, VXLAN, RoCEv2
- PFP deep-dive
  - Current kernel code
  - Proposed changes needed to get this
- Discussion: DoS vector threats, sysctl knobs..

ECMP: Equal Cost Multipathing

- For efficient network usage
  - We want to avoid re-ordering of packets for a given flow at the routers
  - We want packets of different flows to exploit parallel paths where possible (ECMP)

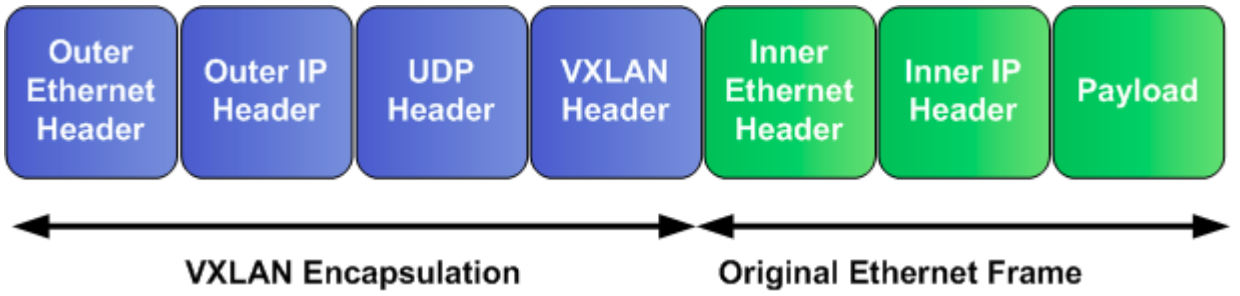


What is a “flow”?

- One definition “all ipv4 packets between host A to host B”, or “all UDP packets from A to B”
- But this has a very large granularity- using TCP or UDP 4-tuple (src addr, src port, dst addr, dst port) gives us better flow hashing.
- Same principles apply to RSS at the host as well- use well-defined fields in the packet to define a “flow”
- Existing hardware at routers and ASICs already knows to look for IP addresses and (for TCP and UDP) the port numbers for flow-hashing

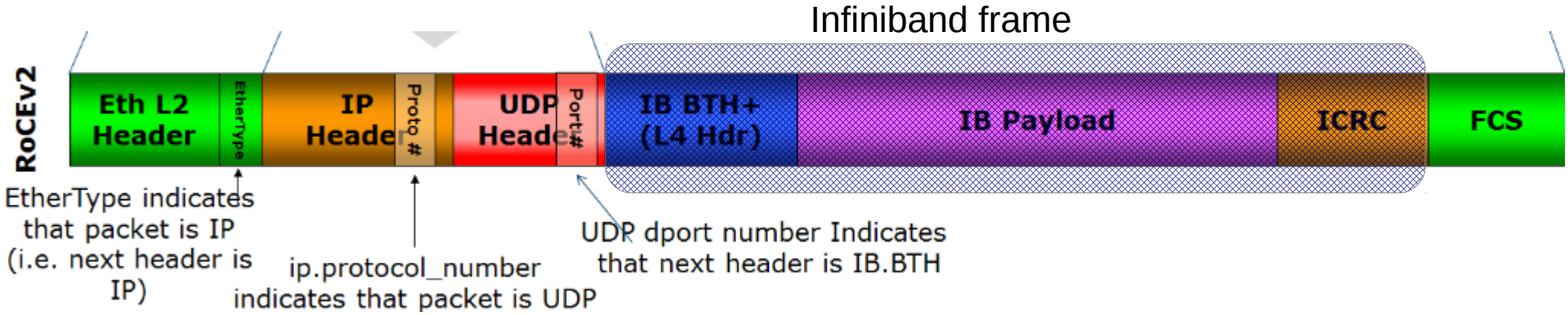
Entropy and UDP encapsulations like VXLAN

- We have a number of L3 tunneling protocols that encapsulate with TCP or UDP
- E.g., VXLAN tunnels a tenant L2 frame from a VM in UDP.
  - UDP destinaton port 4789 (IANA designated port# for VXLAN)
  - UDP sport is a hash of fields in the tenant frame so that client flows can be hashed across available ECMP paths at the router, UDP sport is randomized to provide “entropy”
  - VXLAN frame format (Ref: https://docs.citrix.com/zh-cn/netscaler/11/networking/vxlans.html):
  -



Other UDP encapsulations: RoCE

- RoCEv2 allows applications to tunnel IB frames over UDP/IP, and obtain benefits of RDMA over commodity ethernet
  - Tunneling is done in the NIC firmware



TCP encapsulations: RDS-TCP

- Socket-over-socket model:
  - Application opens a PF_RDS socket and sees datagram semantics.
  - RDS Datagram is sent over a kernel TCP socket; TCP provides reliable, ordered delivery
- TCP flows are between IANA registered RDS-TCP port (16385) and a client transient port.
- Multipath RDS: RDS socket is hashed to one of 8 TCP flows between any pair of peers. Client transient port provides RSS/ECMP entropy.
  - Why 8 sockets? Because the rule-of-thumb for best performance using microbenchmarks like iperf says “use 8-10 socket because the typical NIC has about 8 hardware rings”

Securing kernel manage UDP/TCP tunneling sockets

- IPsec to provide privacy/encryption for data sent through kernel managed TCP/UDP sockets
- IPsec operates at the IP layer, encrypts TCP/UDP header.
- Software/hardware IPsec offloads help improve the single-stream performance profile
- But have we compromised the entropy?

IPsec/ESP hides the port numbers

- TCP and UDP headers may get encrypted with IPsec
- Fortunately, the SPI (32 bits) gets inserted at the exact same byte offset as TCP/UDP port numbers (sport/dport are each 16 bits)
- Unique SPI per IPsec tunnel – so it can be used to define a flow!
  - Existing hardware can look at the same byte offset to find the flow hash for TCP, UDP and ESP.

Getting an SPI per TCP/UDP 4-tuple

- We typically do not know the client transient port number a priori- most client sockets will use a kernel-assigned port number (implicit bind during connect(), or bind to port 0)
- That means we cannot set up the SADB/SPD entry for the exact 4-tuple, we end up setting two tunnels
- Example for iperf testing, the server listens by default at port 5001. To test Ipsec + iperf for 8 TCP streams using a transient client port (typical iperf test) we would only be able to set up the *swan tunnel "`leftprotoport=tcp/5001`"
- 8 TCP 4-tuples → 2 SPIs; Entropy is lost

But does it really matter?

- Oracle DB testing: cluster based appliance that can do parallel queries using RDS-TCP to the database
- Three test cases
  - Clear traffic, with load spread across 8 TCP connections
  - IPsec with a single tunnel for *.16385 (single-SPI)
  - Force the client ports via explicit bind(), and set up 8 IPsec tunnels (8-SPI)
- Elapsed time for the test, peak throughput and number of CPUs processing softirq at peak throughput were instrumented

Results for data appliance parallel query test

	Elapsed time (s)	Peak throughput(Gbps)	Number of cpus processing softirq
Clear traffic over 8 TCP paths	30	8.2	4
IPsec with 1 SPI	606	0.281	1
IPsec with 8 SPIs	204	1.1	4

- Note that the throughput improves by a factor of 5X by ensuring that each TCP connection gets a unique SPI
- The kernel used for the table above was a UEK 4.1 kernel that did not have support for IPsec software or hardware offload. We expect the gap between clear and IPsec traffic to be reduced with newer kernels that have IPsec offload support (ongoing work to instrument this case)
- Having an SPI per TCP connection helps performance, but how to achieve this without being constrained to requiring an explicit bind() for each TCP/UDP socket?

RFC 4301: Populate From Packet

- RFC 4301 has a remedy for this

  “If IPsec processing is specified for an entry, a "populate from packet" (PFP) flag may be asserted for one or more of the selectors in the SPD entry (Local IP address; Remote IP address; Next Layer Protocol; and, depending on Next Layer Protocol, Local port and Remote port, or ICMP type/code, or Mobility Header type). If asserted for a given selector X, the flag indicates that the SA to be created should take its value for X from the value in the packet. Otherwise, the SA should take its value(s) for X from the value(s) in the SPD entry.”

- For TCP/UDP if the SPD is marked “PFP” send an upcall to pluto and create an SA for the exact 4-tuple for the packet that triggered the SPD lookup.

Current kernel behavior

- With "`auto = ondemand`" for *swan tunnel setup, when data is sent (via `connect()` or `send()/sendto()`) an SADB_ACQUIRE upcall is sent from the kernel to the `pluto` daemon.
- SADB_ACQUIRE is sent from `xfrm_state_find()` if `xfrm_state_lookat()` tells us that there is no acquire currently in progress.
  - In the iperf example, today we'd trigger one ACQUIRE for the first 4-tuple that matched *.5001 and subsequent packet triggers that matched *.5001 would all find `acq_in_progress`

Proposed modification

- Add a new XFRM_POLICY_PFP flag, and provide the netlink hooks to allow userspace to set it on a specific SPD
- In `xfrm_state_lookat()`, if we find an xfrm_state in XFRM_STATE_ACQ

```
if (pol->flags & XFRM_POLICY_PFP) {
    // return acq_in_progress only if xfrm_selector_match() succeeds
    if ((x->sel.family &&
        !xfrm_selector_match(&x->sel, fl, x->sel.family)) ||
        !security_xfrm_state_pol_flow_match(x, pol, fl))
            return;  // no acquire in progress.
}
*acq_in_progress = 1; // don't send another ACQUIRE
```

Changes to `pluto`

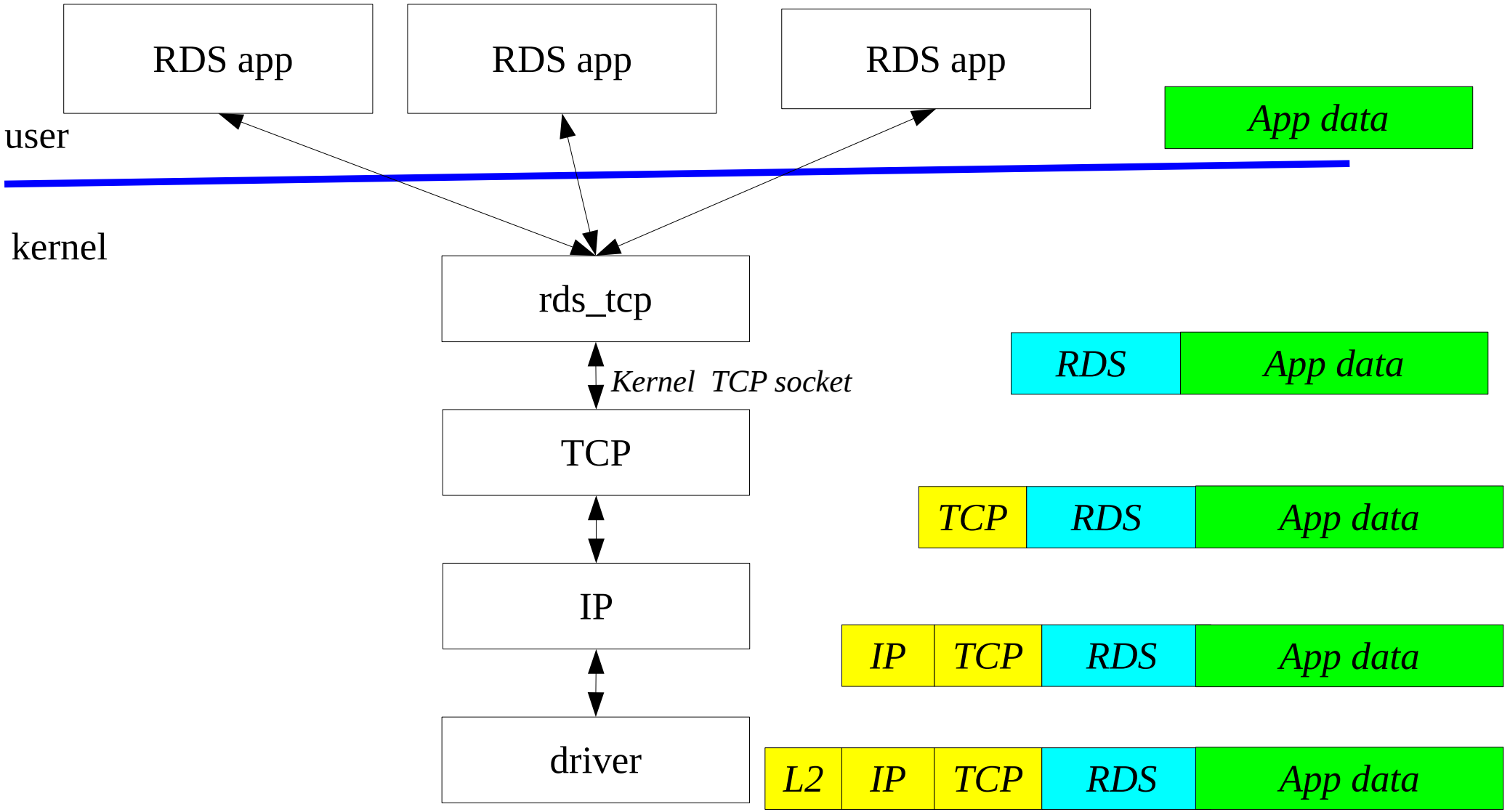
- Need to be able to set the new XFRM_POLICY_PFP flag
- And much more… Paul Wouters will describe

Is there a DoS threat with PFP?

- Clear TCP/UDP traffic gets entropy from the 4-tuple- kernel has to manage some state per socket, routers use the entropy to find the hash needed for ECMP
- If we have an SPI per 4-tuple, we end up creating IPsec tunnel state merely for entropy
- we may end up with many more tunnels than the available multipathing
  - e.g., 1000 UDP sockets, but only 8-way multipathing => 1000 IPsec tunnels.
- Sysctl tunables to place upper limits on this? e.g., generate a max of *k* ACQUIRE's for a PFP SPD by using a mask for port matching?

Backup slides

RDS-TCP Architectural Overview



RSS and entropy

- Shannon Nelson pointed out to me that Niantic does the RSS hashing based on the TCP/UDP 4-tuple after decrypt (when offload has been enabled)
- All drivers should follow that model for offloaded IPsec (this will give the desired steering even when PFP has not been requested)
- Even when IPsec has not been (or cannot be) offloaded, NICs should use SPI for the RSS hash.