Fastpath for IPSec gateways using the flowtable infrastructure

Pablo Neira Ayuso
<pablo@netfilter.org>
IPSec Workshop
Prague, Czechia
Flowtable bypass

Fig. 1 Netfilter hooks and flowtable interactions
Flowtable bypass (2)

- For each packet, extract tuple and perform look up at the flowtable.
  - Miss: Let the packet follow the classic forwarding path.
  - Hit:
    - Attach route from flowtable entry (… flowtable is acting as a cache).
    - NAT mangling, if any.
    - Decrement TTL.
    - Send packet via neigh_xmit(…).
  - Exceptions (any of them, forces slow path):
    - If packet is over MTU, pass it up to classic forwarding path.
    - Secpath info is available.
    - IP Options available.

- Garbage collector:
  - Expire flows if we see no more packets after N seconds.
  - TCP reset and fin packets are passed up to slow path.
Flowtable bypass (3)

• Configure flow bypass through **one single rule:**

```plaintext
table ip x {
  flowtable f {
    hook ingress priority 0; devices = { eth0, eth1};
  }
  chain y {
    type filter hook forward priority 0;
    ip protocol tcp flow add @f
  }
}
```

• Conntrack entries are owned by the flowtable:

```
# cat /proc/net/nf_conntrack
ipv4 2 tcp 6 src=10.141.10.2 dst=147.75.205.195 sport=36392 dport=443 src=147.75.205.195 dst=192.168.2.195 sport=443 dport=36392 [OFFLOAD] mark=0 zone=0 use=2
```
Flowtable bypass (4)

- Flow offload forward PoC in software is ~2.75 faster in software:
  - pktgen_bench_xmit_mode_netif_receive.sh to dummy device to exercise the forwarding path
    - One single CPU
    - Smallest packet size (worst case)

- Performance numbers:
  - Classic forwarding path (baseline): 1848888pps
  - Flow offload forwarding: 5155382pps
Flowtable bypass (5)

• Upstream since 4.16 (January 2018).

• Recent patches:
  – Tear down feature: send flows back to slow path
    • RST and FIN packets.
    • Limited pickup time.
    • Only for TCP and UDP by now.
  – Fix offloading of SNAT+DNAT flows
  – Fix: Don’t remove offload when other netns's interface is down.
  – Fix interaction with VRF.
  – Attach dst to skbuff.
Flowtable bypass (6)

- Hardware offload infrastructure (~200 LOC) available.
- Not yet upstream, waiting for a driver :-(
- User enables explicitly “offload” flag to enable hardware offload.
- New ndo hook for offloads or generalise existing ndo for this purpose.
Earlier flowtable bypass + GRO

- [PATCH net-next,RFC 00/13] New fast forwarding path on Thu, 14 Jun 2018 16:19:34 +0200 (Joint work with Steffen).

- Idea:
  - Do flowtable lookup earlier than ingress (before taps)
  - Avoid reiterative routing lookups
  - Combine it with GRO batching
    - Build a chain of skbuffs with same flowtable entry
    - Pass them in on go to neigh_xmit
  - Otherwise, slow path (pass it to generic GRO handlers)

- Feedback:
  - GRO not the right place for batching? Use sublists?
  - Aaron Conole’s patchset: No IPSec integration though
Earlier flowtable bypass + GRO (2)

table x {
  flowtable f {
    hook {early_ingress} priority 0; devices = { eth0, eth1 }
  }
  chain y {
    type filter hook forward priority 0;
    ip protocol tcp flow add @f
  }
}

• Numbers:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Current Rate</th>
<th>Fast Forward Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP TSO</td>
<td>32.5 Gbps</td>
<td>35.6 Gbps</td>
</tr>
<tr>
<td>UDP</td>
<td>17.6 Gbps</td>
<td>35.6 Gbps</td>
</tr>
<tr>
<td>ESP</td>
<td>6 Gbps</td>
<td>7.5 Gbps</td>
</tr>
</tbody>
</table>
Ongoing work

• Patch to add IPSec support (not tested):
  – [https://patchwork.ozlabs.org/patch/982747/](https://patchwork.ozlabs.org/patch/982747/)

• Setup entry in flowtable from first packet.
  – Needs explicit configuration from user.

• Empty devices in flowtable?
  
  ```
  table x {
    flowtable f {
      hook ingress priority 0; devices = {}
    }
  }
  chain y {
    type filter hook forward priority 0;
    ip protocol tcp flow add @f
  }
  ```