Stateful Connection Tracking & Stateful NAT

Justin Pettit
VMware

Thomas Graf
Noiro Networks, Cisco
Agenda

- Connection Tracking
- NAT
- Integration of other stateful services
We had a performance problem

• With some traffic patterns, performance of OVS could be quite bad

• Last week, we added a post to Network Heresy describing the changes we made to improve performance and the results

• Focus of past two years was on performance. Now that we feel good about the performance, we're back to looking at features

• Any addition to OVS must consider its implication on performance
Implementing a Firewall

• OVS has traditionally only supported stateless matches
• As an example, currently, two ways to implement a firewall in OVS
  – Match on TCP flags (Enforce policy on SYN, allow ACK|RST)
    • Pro: Fast
    • Con: Allows non-established flow through with ACK or RST set, only TCP
  – Use “learn” action to setup new flow in reverse direction
    • Pro: More “correct”
    • Con: Forces every new flow to OVS userspace, reducing flow setup by orders of magnitude
  – Neither approach supports “related” flows or TCP window enforcement
Connection Tracking

- We are adding the ability to use the conntrack module from Linux
  - Stateful tracking of flows
  - Supports ALGs to punch holes for related “data” channels
    - FTP, TFTP, SIP
- Implement a distributed firewall with enforcement at the edge
  - Better performance
  - Better visibility
- Introduce new OpenFlow extensions:
  - Action to send to conntrack
  - Match fields on state of connection
Netfilter Conntrack Integration

1. OVS Flow Table
2. `conntrack()`
3. Connection State (conn_state=)
4. Recirculation

Userspace Netlink API

Netfilter Connection Tracker

Create & Update CT entries

CT Table
## Conntrack Example

Conntrack example that only allows port 2 to respond to TCP traffic initiated from port 1:

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>in_port(1), tcp, conn_state=-tracked</code></td>
<td><code>conntrack(zone=10), normal</code></td>
</tr>
<tr>
<td><code>in_port(2), tcp, conn_state=-tracked</code></td>
<td><code>conntrack(zone=10, flags=recirc)</code></td>
</tr>
<tr>
<td><code>in_port(2), tcp, conn_state=+established</code></td>
<td><code>output:1</code></td>
</tr>
<tr>
<td><code>in_port(2), tcp, conn_state=+new</code></td>
<td><code>drop</code></td>
</tr>
</tbody>
</table>
Connection Tracking Zones
Flow Caching Works

- Preliminary results are quite promising
  - OVS+conntrack uses a nearly consistent rate regardless of number of rules
  - Bridge+iptables uses more CPU as rule count increases

Number of gigacycles per second required to saturate a 10Gbps link with netperf TCP_STREAM. (Lower is better)
Design Goals

- Performance implications must be thought through
- Thought needs to be put into API, since OVS and OpenFlow have traditionally been flow-based and stateless
- While this will only be supported on Linux initially, the API shouldn’t be Linux-specific
- New stateful features being discussed and leveraging kernel:
  - NAT
  - Load-balancing through IPVS
  - DPI
Release Plan

• Will ship with OVS 2.4
• Will include:
  – connmark
  – IP fragment reassembly
  – Hide break in pipeline from userspace
• Available to try now:
  – https://github.com/justinpettit/ovs/tree/conntrack
Stateful NAT Overview

• SNAT and DNAT
• Based on connection tracking work
• Leverages stateful NAT mechanism of Netfilter
• Able to do port range and address range mappings to masquerade multiple IPs
• Mapping Modes
  – Persistent (across reboots)
  – Hash based
  – Fully random
Stateful NAT Flow

1. OVS Flow Table

2. conntrack()

3. nat()

4. Recirculation

Create & Update CT entries

Netfilter Connection Tracker

Netfilter NAT

CT Table
## NAT Example

SNAT all TCP packets on port 1 to the IP range 10.0.0.1/24 with reverse SNAT on port 2:

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>in_port(1), tcp</code></td>
<td><code>conntrack(zone=10), nat(type=src, min=10.0.0.1, max=10.0.0.255, type=hash)</code></td>
</tr>
<tr>
<td><code>in_port(2), tcp, conn_state=-tracked</code></td>
<td><code>conntrack(zone=10, flags=recirc)</code></td>
</tr>
<tr>
<td><code>in_port(2), tcp, conn_state=+established</code></td>
<td><code>nat(reverse)</code></td>
</tr>
<tr>
<td><code>in_port(2), tcp, conn_state=+new</code></td>
<td><code>drop</code></td>
</tr>
</tbody>
</table>
Stateful services integration: NFQUEUE action

Userspace

Kernel

L2 inject
skb->mark = 0x10

{sync|async}
nfqueue(queue=2)

conn_state=+tracked,
conn_mark=0x20
NFQUEUE action

- Can reuse existing Netfilter queueing mechanism and libnfnfnetlink + libctctnetlink
- Operational Modes:
  1. Steal/Drop – Async verdict via CT template
  2. Reinjection – Sync verdict via NFQA_MARK
- Needed netfilter modifications:
  - Reinjection routine for NFPROTO_BRIDGE back into netif_rx()
- OVS modifications
  - New nfqueue action
Q&A

• More Information:
  – http://openvswitch.org/

• Code:
  – Conntrack (WIP):
    • https://github.com/justinpettit/ovs/tree/conntrack
  – Stateful NAT (WIP):
    • https://github.com/tgraf/ovs/tree/nat