Open vSwitch with conntrack

Jesse Gross
Netfilter workshop 2014
Brief Background on Open vSwitch

• The goal of OVS is to be a programmable switch.
• Implements both traditional switching functionality as well programmability through OpenFlow.
• Frequently used to build larger networking applications for OpenStack, network virtualization, etc.
• Most functionality is implemented/evaluated in userspace and a set of flows are programmed into the kernel with matches and actions.

Example:
Traditionally switch pipelines perform L2 switching based on VLAN ID. With OVS, GRE key can be substituted with no changes or mapping.
L2 Learning Switch – Example Flow

Userspace maintains MAC tables and generates flows with matches and actions for traffic that is flowing through the switch.

Match:

\[
\text{recirc}_\text{id}(0), \text{skb}_\text{priority}(0), \text{in}_\text{port}(1), \text{eth}(\text{src}=\text{b2:1a:43:d5:fa:4c}, \text{dst}=\text{2a:00:8f:31:f6:49}), \text{eth}_\text{type}(0x0800), \text{ipv4}(\text{src}=30.0.0.1/0.0.0.0, \text{dst}=30.0.0.2/0.0.0.0, \text{proto}=1/0, \text{tos}=0/0x3, \text{ttl}=64/0, \text{frag}=\text{no}/0xff)
\]

Action:

Output: port 2
Problem: Stateful or Non-Flow Based Services

- OVS is good for stateless, flow based networking.
- In many cases, this provides the means to compose a network:
  - Switching
  - Routing
  - Build your own network processing pipeline

Largely leaves out “services” that might be inserted into that network. **Firewalls are an obvious gap.**

How to retain the benefits of OVS in these situations?
Implementing a Firewall

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
Integrating conntrack

- OVS can call into the kernel connection tracker:
  - Large library of protocol support and ALGs for related flows.
  - Automatically get benefit of performance optimizations.

- A stateless flow can select a class of traffic that needs to be tracked, passing matching packets off to the conntracker for further evaluation.

- Result is the state of the flow that can be matched by the flow table:
  - NEW
  - ESTABLISHED
  - RELATED
  - INVALID

- Conntrack zones provide isolation between OVS flows (if desired).
What is a flow?

OVS and conntrack may have different ideas of what a flow is.

Open vSwitch
- OVS supports a large number of fairly fine-grained fields, including those not traditionally considered to be part of a “flow” (i.e. IP TTL).
- User supplies wildcards so actual match can range from very broad to very narrow.

Connection tracker
- Flow is defined by the protocol definition and related ALGs (such as FTP).
- Typically something like a 5-tuple

OK if these differ: OVS is just trying to select when and how connection tracking is applied.
Stateful Firewall – User’s View

Goal:
- All outbound traffic is allowed, added to state table.
- Inbound traffic is allowed if part of established connection.

OVS Flows:

Outbound:
Match: in_port=inside
Action: conntrack(zone=tennant), output:outside

Inbound
Match: in_port=outside
Action: conntrack(zone=tennant), recirculate
Match: conntrack_state=established
Action: output:inside
Going Further

Problem extends to a variety of other possible situations besides firewalls.

Other examples that have come up:

- NAT
- Deep packet inspection
- Load balancing
- QoS (largely solved already though)

Connection tracking is probably the best place to start and could serve as a template. See Thomas’ presentation for more.