FWaaS V2 Driver Implementation

Mickey Spiegel

(emspiege@us.ibm.com)

Chandan Dutta Chowdhury

(chandanc@juniper.net)

Sarath Chandra Mekala

(sarathcm@juniper.net / sarathmekala.openstack@gmail.com)

Table of contents

Introduction

This document will capture the discussion and notes around FWaaS V2 driver implementation.

Planned Enhancements:

- Granularity -- Neutron Ports, Service Function Chaining Ports, Router ports
- Filter East-West intra/inter subnet traffic, North-South Traffic
- Ability to create different FW policies with different rules for ingress/egress directions
- Firewall Group construct -- binds policies to ports
- Address group, Service groups provide re-usability.
- Multiple FW group association to the same neutron port.

Service Group -- Group of L4 ports. Ex: All Web Server Ports SG = 80, 8080, 443
Address Group -- Group of IP addresses

Firewall Stratum (Defense in depth)

- When both FWaaS and Security Groups are associated with the same Neutron port, a packet must be allowed by both features, i.e. “deny” wins between FWaaS and Security Groups. --- Clear. Done.
- When there are multiple firewall groups associated with a specific Neutron port, packets will be allowed if any one of the firewall groups associated with that Neutron port allows the packet.This behavior is similar to the case of multiple Security Groups associated with the same VM port. -- Complex. Currently the latest FG rule wins.

To illustrate the above points with an example:

FG --> Firewall Group

FWaaS Stratum : Outer Ring
==========================
FG1 : Ports 70, 80
Rule1 : Deny rule for FG1

SG Stratum : Inner Ring
========================
SG1 : 60
SG2 : 60, 70
Rule2 : Deny SG1
Rule3 : Allow SG2

1. When filtering a packet on Port 60 – Rule3 allow rule wins over Rule2 deny rule.
   - (needs clarification, what about rule ordering ?)
     - Good point. We need to think this through. I had a discussion with Mickey on this earlier. The following is the gist of the discussion:

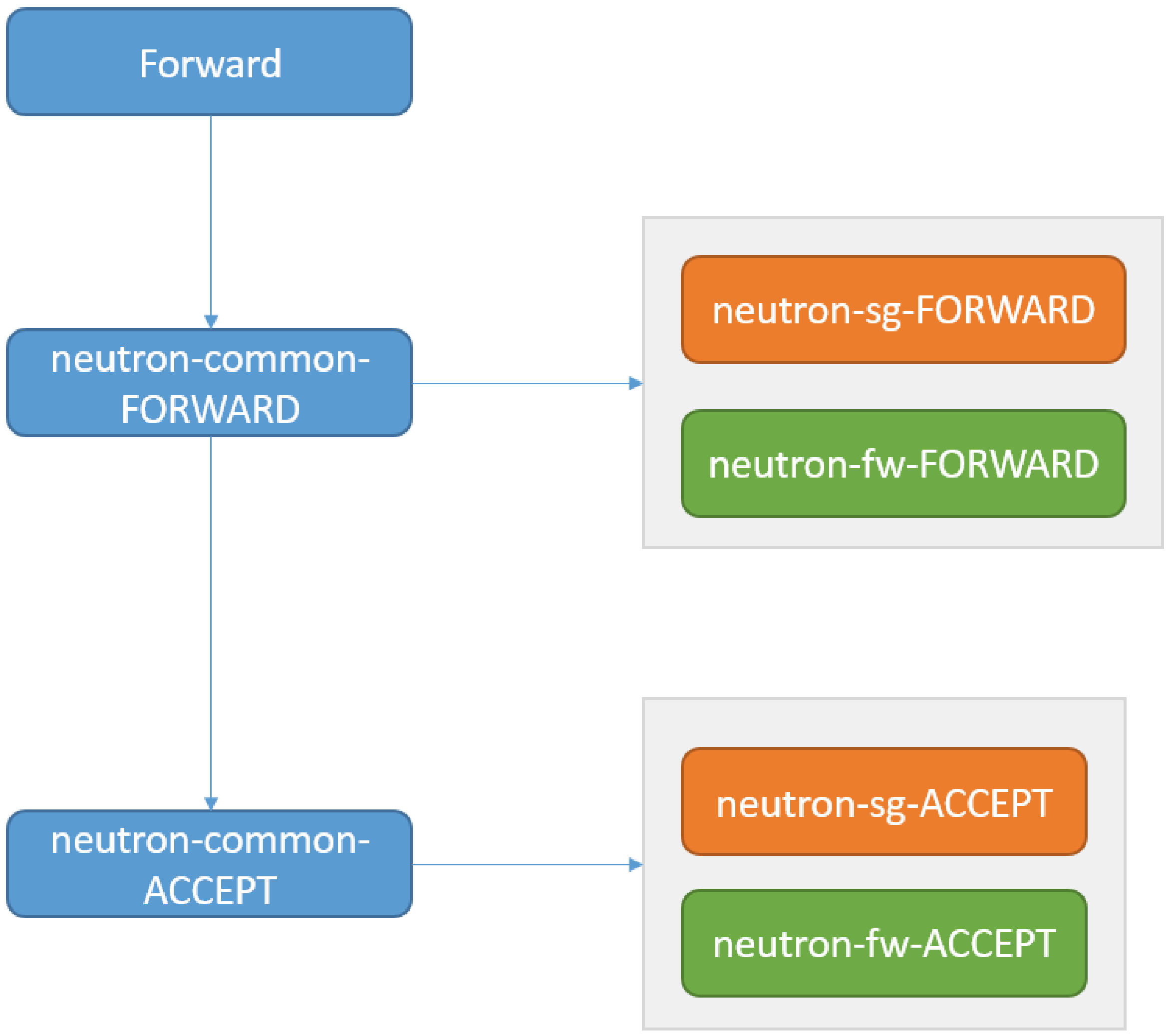
"When there are multiple firewall groups associated with a specific Neutron port, there is no position or priority between the different firewall groups. "
Can we introduce the concept of depth/priority to a FW group so that this behavior can be made more deterministic.

There are a couple of reasons why I don't think a position or priority on a firewall group would work well:
- This is roughly equivalent to one giant set of firewall rules with position or priority
- How would you normalize values between different firewall groups?
- Different ports may use different firewall groups in different combinations, with preferred placement varying depending on the combination

2. When filtering a packet on port 70 -- The packet is denied at the Outer ring itself by Rule1.

Requirements:
1. Rules in the same stratum need to be ordered.
2. ACCEPT rule clash between SG & FW rules need to be resolved.

1. Changes to Security Group IPTables driver implementation



As per this new proposal, **IptablesManager** will pre-create the following chains in the fixed order:
Forward

- neutron-common-FORWARD
  - neutron-sg-FORWARD
  - neutron-fw-FORWARD
- neutron-common-ACCEPT
  - neutron-sg-ACCEPT
  - neutron-fw-ACCEPT

The SG iptables driver will insert its chains under neutron-sg-FORWARD instead of FORWARD chain.
FwaaS will insert its chains under neutron-fw-FORWARD chain.

The SG driver will add its ACCEPT rule to neutron-sg-ACCEPT chain while FwaaS will add its ACCEPT rules to neutron-fw-ACCEPT chain. This way the stratum ordering will be preserved.

The rule evaluation flow would be as follows, when a packet needs to be filtered it enters the neutron-common-FORWARD chain. It’s sent first to neutron-sg-FORWARD chain where SG rules are evaluated. If it matches a deny then the packet is dropped. In the case it’s allowed the chain returns and the control comes back to neutron-common-FORWARD chain. The packet is then sent to neutron-fw-FORWARD where it will get evaluated. The same logic as SG applies here as well. If the packet is allowed then the control is returned back to neutron-common-FORWARD chain, which then returns back to the FORWARD chain.

Next, the FORWARD chain will forward the packet to neutron-common-ACCEPT chain to be evaluated for ACCEPT condition. Its first sent to neutron-sg-ACCEPT where SG ACCEPT rules are evaluated. If there is no match, the control returns back to neutron-common-ACCEPT which in turn sends the packet to neutron-fw-ACCEPT chain. If the packet finds a match then it gets accepted if not the control returns back to neutron-common-ACCEPT chain. The packet is then returned back to the FORWARD chain and it proceeds to evaluate the other chains present below.

NOTE : The same logic is applied to input and output chains as well.

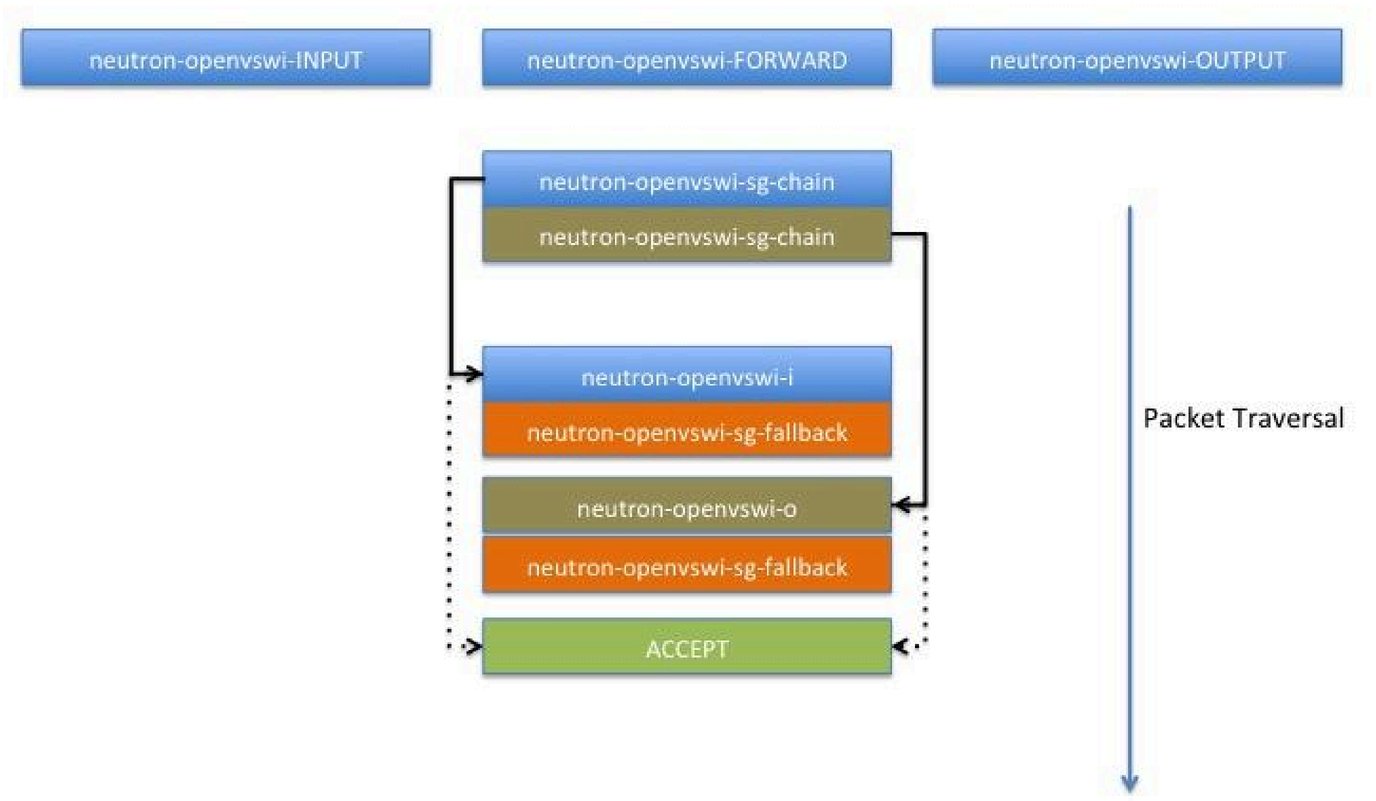
IPTables diff after the latest changes : http://paste.openstack.org/show/544987/
Review posted @ https://review.openstack.org/#/c/348177/

2. Changes to Firewall IPTables Implementation

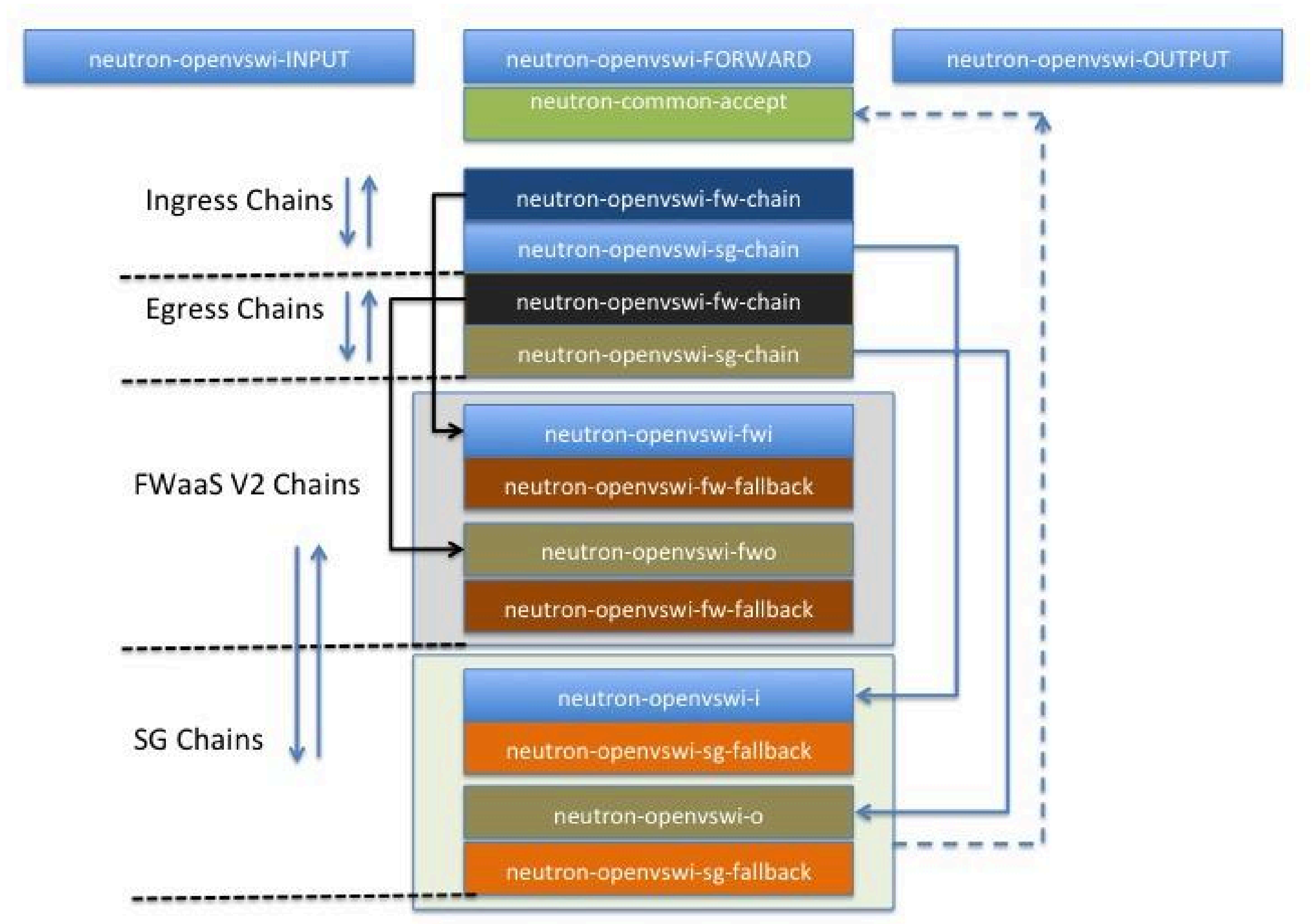
(TBD)

Old Discussions (Preserved for posterity)

Security Groups Current Implementation



Proposed changes to rule chaining between SG and FW



Current Discussion

I don't think fwaas should use "neutron-openvswi" as the wrap name. The thought was to have two independent instances of IptablesManager, one for SG and one for FWaaS. They need to have different wrap names so that they don't clobber each other's rules. Since SG uses the default wrap name, FWaaS has to use a non-default wrap name, specifying something like "neutron-fwaas", at least for L2.

My initial thought was to ask SG to provide a hook whereby we could ask for a jump to an FWaaS chain. I have since come to think that that might be too ugly, and they might not go for that. We cannot put the jump directly in their chains, or they will wipe it out every time they rebuild the iptables rules. The WIP patch is along these lines, and would have to be changed if we abandon this approach.

I don't think you can control the order of rules from different IptablesManager instances in the same chain. It looks to me like it swaps to the last IptablesManager instance's rules first. That means your suggestion of a jump to FW before jumping to SG will not work, at least with the existing iptables_manager.py code.

Instead, I think we should introduce a new "neutron-security-common" unwrapped chain. From "neutron-security-common" it can jump to sg-chain and to fwaas-chain. The ACCEPT would move to "neutron-security-common", instead of its current position in "sg-chain". I had not yet thought about how to get the ACCEPT at the end "neutron-security-common", even after rules from a different IptablesManager instance.

My initial thought was to jump from "neutron-openvswi-FORWARD" and "neutron-fwaas-FORWARD" to "neutron-security-common". It seems a little weird to jump from an unwrapped chain (FORWARD) to a wrapped chain (neutron-openvswi-FORWARD) to an unwrapped chain (neutron-security-common), but this approach might avoid the need to change iptables_manager.py. I am not sure how big a deal that is.

If we want to jump to "neutron-security-common" directly from FORWARD, then that is a change to iptables_manager.py. That would still require some thought, I guess two levels of unwrapped chains, because the ACCEPT should only apply if at least one of the physdev rules is matched.

Change proposal

The SG firewall driver is implemented in iptables_firewall.py module, it uses the generic iptables configuration module called iptables_manager.py. I see that the iptables_manager.py initializes the iptables configuration with some common infrastructure at the start. Let's take the example of **Filter** table. The iptables_manager module(IpTablesManager class actually) creates common jump chains for each of the default chains(INPUT, FORWARD, OUTPUT) in the Filter table (neutron-openvswi-INPUT, neutron-openvswi-FORWARD, neutron-openvswi-OUTPUT).

These chains are called wrapped chains as they are prefixed by the name of the executable binary that created the instance of IpTablesManager. In our case it is the **neutron-openvswitch-agent**.

All the above is generic code and is not part of SG firewall module (iptables_firewall.py).

Next SG firewall creates its own chain **neutron-openvswi-sg-chain** and attaches it to the generic wrapped chains (e.g. neutron-openvswi-FORWARD) created by iptables_manager in the previous step using a jump rules. E.g.

```
Chain neutron-openvswi-FORWARD (1 references)
 pkts bytes target  prot opt in   out    source              destination
      9  1205 neutron-openvswi-sg-chain  all  --  *         *      0.0.0.0/0
0.0.0.0/0           PHYSDEV match --physdev-out tap6fe09f52-99 --physdev-is-bridged /* Direct
traffic from the VM interface to the security group chain. */
```

```
      9  1152 neutron-openvswi-sg-chain  all  --  *        *      0.0.0.0/0
0.0.0.0/0           PHYSDEV match --physdev-in tap6fe09f52-99 --physdev-is-bridged /* Direct
traffic from the VM interface to the security group chain. */
```

The neutron-openvswi-sg-chain chain pushes the packets through interface and direction specific chain like **neutron-openvswi-i6fe09f52-9** or **neutron-openvswi-o6fe09f52-9.**

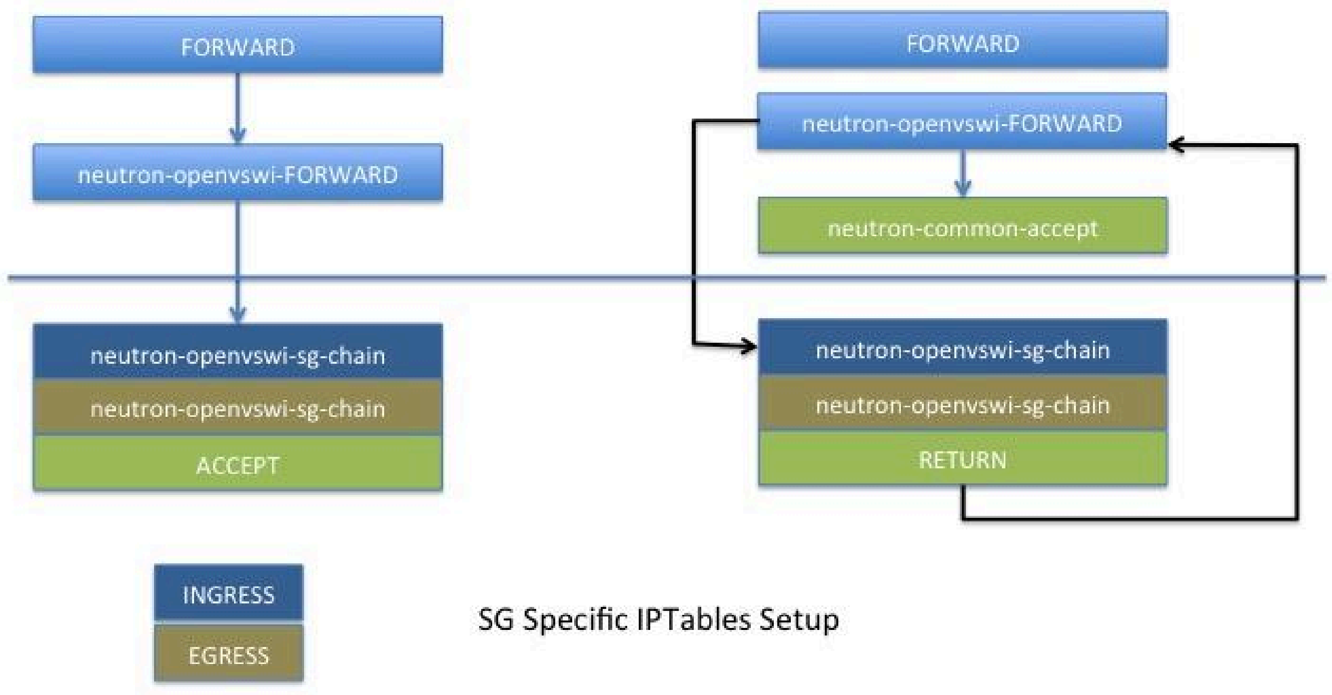
In the current implementation ACCEPT rule is part of the SG chain.

```
Chain neutron-openvswi-sg-chain (2 references)
 pkts bytes target  prot opt in   out    source              destination
      9  1205 neutron-openvswi-i6fe09f52-9  all  --  *      *      0.0.0.0/0
0.0.0.0/0           PHYSDEV match --physdev-out tap6fe09f52-99 --physdev-is-bridged /* Jump to
the VM specific chain. */
      9  1152 neutron-openvswi-o6fe09f52-9  all  --  *      *      0.0.0.0/0
0.0.0.0/0           PHYSDEV match --physdev-in tap6fe09f52-99 --physdev-is-bridged /* Jump to
the VM specific chain. */
   18  2357 ACCEPT  all  --  *   *      0.0.0.0/0           0.0.0.0/0
```

In my proposal patch I have moved the ACCEPT rule to be executed after the generic wrapped chain.
Patch: http://paste.openstack.org/show/526005/

Following is a simplified picture of the change.

Generic IPTables Chain Setup

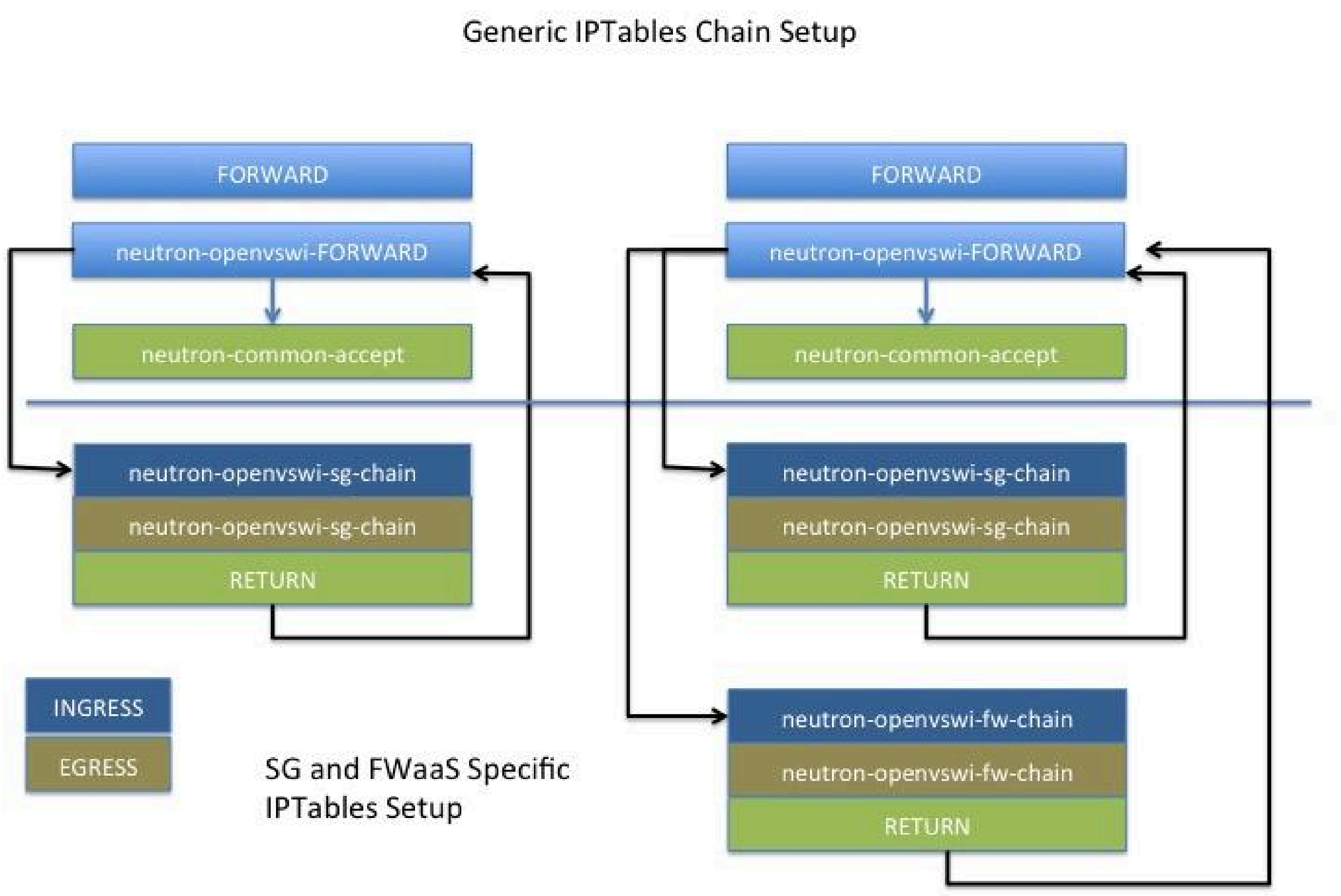


SG Specific IPTables Setup

Multiple Drivers

Here is how it will look with multiple drivers. **The order of jumps does not matter, as the individual driver chains will return control to the generic chain to take the final decision of Accepting the packet**.

The drop by default rule is the last rule of the driver specific rule, it will be triggered if none of the driver specific rule match. This is the current behaviour of the SG driver too.



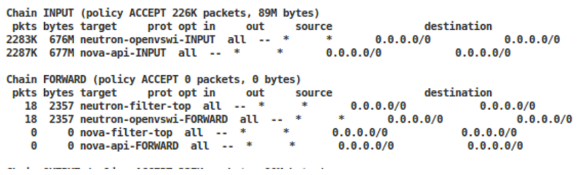
Generic IPTables Chain Setup
FORWARD
neutron-openvswi-FORWARD
neutron-common-accept
neutron-openvswi-sg-chain
neutron-openvswi-sg-chain
RETURN
INGRESS
EGRESS
SG and FWaaS Specific IPTables Setup
FORWARD
neutron-openvswi-FORWARD
neutron-common-accept
neutron-openvswi-sg-chain
neutron-openvswi-sg-chain
RETURN
neutron-openvswi-fw-chain
neutron-openvswi-fw-chain
RETURN

Here is an output of iptables -L -n -v

- Iptables dump before patch(not using the same setup):http://paste.openstack.org/show/526012/
- Iptables dump after patch: http://paste.openstack.org/show/526013/

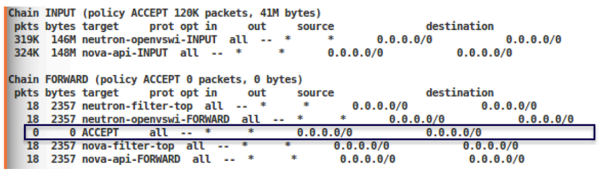
Change Summary

The following screenshot shows the summary of the change, but you can get the full context from the iptables dump captured above.



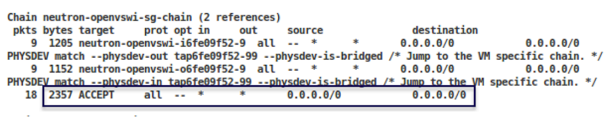
```
Chain INPUT (policy ACCEPT 226K packets, 89M bytes)
 pkts bytes target     prot opt in     out     source               destination
2283K  676M neutron-openvswi-INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
2287K  677M nova-api-INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   18  2357 neutron-filter-top  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   18  2357 neutron-openvswi-FORWARD  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 nova-filter-top  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 nova-api-FORWARD  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

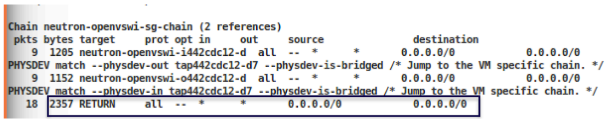


```
Chain INPUT (policy ACCEPT 120K packets, 41M bytes)
 pkts bytes target     prot opt in     out     source               destination
 319K  146M neutron-openvswi-INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0
 324K  148M nova-api-INPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   18  2357 neutron-filter-top  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   18  2357 neutron-openvswi-FORWARD  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
   18  2357 nova-filter-top  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   18  2357 nova-api-FORWARD  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```



```
Chain neutron-openvswi-sg-chain (2 references)
 pkts bytes target     prot opt in     out     source               destination
    9  1205 neutron-openvswi-i6fe09f52-9  all  --  *      *       0.0.0.0/0            0.0.0.0/0
PHYSDEV match --physdev-out tap6fe09f52-99 --physdev-is-bridged /* Jump to the VM specific chain. */
    9  1152 neutron-openvswi-o6fe09f52-9  all  --  *      *       0.0.0.0/0            0.0.0.0/0
PHYSDEV match --physdev-in tap6fe09f52-99 --physdev-is-bridged /* Jump to the VM specific chain. */
   18  2357 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```



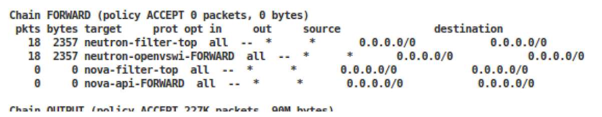
```
Chain neutron-openvswi-sg-chain (2 references)
 pkts bytes target     prot opt in     out     source               destination
    9  1205 neutron-openvswi-i442cdc12-d  all  --  *      *       0.0.0.0/0            0.0.0.0/0
PHYSDEV match --physdev-out tap442cdc12-d7 --physdev-is-bridged /* Jump to the VM specific chain. */
    9  1152 neutron-openvswi-o442cdc12-d  all  --  *      *       0.0.0.0/0            0.0.0.0/0
PHYSDEV match --physdev-in tap442cdc12-d7 --physdev-is-bridged /* Jump to the VM specific chain. */
   18  2357 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Update 1: In response to comments
I have updated the patch and the corresponding iptables output. I have changed the ACCEPT to match on the interface in a separate chain called **neutron-common-accept**
Here are the links to the update :

Patch: http://paste.openstack.org/show/526709/
Iptables output: http://paste.openstack.org/show/542471/

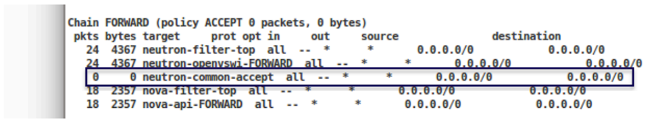
The corresponding packet flow images above has also been updated.

Following are the updated screenshots

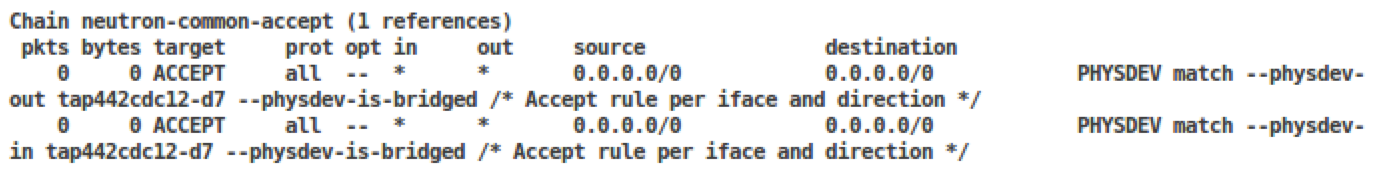


```
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   18  2357 neutron-filter-top  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   18  2357 neutron-openvswi-FORWARD  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 nova-filter-top  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 nova-api-FORWARD  all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 227K packets, 90M bytes)
```



```
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   24  4367 neutron-filter-top  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   24  4367 neutron-openvswi-FORWARD  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 neutron-common-accept  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   18  2357 nova-filter-top  all  --  *      *       0.0.0.0/0            0.0.0.0/0
   18  2357 nova-api-FORWARD  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```



```
Chain neutron-common-accept (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV match --physdev-
out tap442cdc12-d7 --physdev-is-bridged /* Accept rule per iface and direction */
    0     0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            PHYSDEV match --physdev-
in tap442cdc12-d7 --physdev-is-bridged /* Accept rule per iface and direction */
```

The ordering of rules will not impact the filtering as we are not inserting the rules to the **FORWARD** chain, instead we will be adding the FWaaS chain to **neutron-openvswi-FORWARD**

Patches

- https://review.openstack.org/#/c/269961/1/neutron_fwaas/services/firewall/drivers/v2/linux/iptables_fwaas.py.unified
- Connection Tracking patch : https://review.openstack.org/#/c/333338/
- Moving the ACCEPT rule to generic chain setup: http://paste.openstack.org/show/526005/
  - Update 1: http://paste.openstack.org/show/526709/
-