**[Neutron 理解 (9): OpenStack 是如何实现 Neutron 网络 和 Nova虚机 防火墙的 [How Nova Implements Security Group and How Neutron Implements Virtual Firewall]](http://www.cnblogs.com/sammyliu/p/4675991.html)**

学习 Neutron 系列文章：

（1）[Neutron 所实现的虚拟化网络](http://www.cnblogs.com/sammyliu/p/4622563.html%20)

（2）[Neutron OpenvSwitch + VLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4626419.html%20)

（3）[Neutron OpenvSwitch + GRE/VxLAN 虚拟网络](http://www.cnblogs.com/sammyliu/p/4627230.html%20)

（4）[Neutron OVS OpenFlow 流表 和 L2 Population](http://www.cnblogs.com/sammyliu/p/4633814.html%20)

（5）[Neutron DHCP Agent](http://www.cnblogs.com/sammyliu/p/4419195.html)

（6）[Neutron L3 Agent](http://www.cnblogs.com/sammyliu/p/4636091.html%20)

（7）[Neutron LBaas](http://www.cnblogs.com/sammyliu/p/4656176.html%20)

（8）[Neutron Security Group](http://www.cnblogs.com/sammyliu/p/4658746.html%20)

（9）[Neutron FWaas 和 Nova Security Group](http://www.cnblogs.com/sammyliu/p/4675991.html)

（10）[Neutron VPNaas](http://www.cnblogs.com/sammyliu/p/4677386.html%20)

（11）[Neutron DVR](http://www.cnblogs.com/sammyliu/p/4713562.html)

（12）[Neutron VRRP](http://www.cnblogs.com/sammyliu/p/4692081.html)

（13）[High Availability （HA）](http://www.cnblogs.com/sammyliu/p/4741967.html)

    本文的基础知识和所用到的技术和实现和 [Neutron 理解 (8): Neutron 是如何实现虚机防火墙的 [How Netruon Implements Security Group]](http://www.cnblogs.com/sammyliu/p/4658746.html) 非常类似，因此会省去相同的部分。

**1. Nova 安全组**

**1.1 配置**

|  |  |  |  |
| --- | --- | --- | --- |
| 节点 | 配置文件 | 配置项 | 说明 |
| **controller** | /etc/nova/nova.conf | security\_group\_api = nova | 是的 nova secgroup\* 命令使用的是 nova 安全组的 API |
| /etc/neutron/plugins/ml2/ml2\_conf.ini | enable\_security\_group = False | 禁止 Neutron 安全组 |
| **nova-compute** | /etc/nova/nova.conf  /etc/nova/nova-compute.conf | firewall\_driver = nova.virt.firewall.IptablesFirewallDriver | 指定 Nova 安全组的驱动，可以是IptablesFirewallDriver 或者 NWFilterFirewall。默认是 IptablesFirewallDriver。见下面的说明。 |
|  | /etc/neutron/plugins/ml2/ml2\_conf.ini | enable\_security\_group = False | 禁止 Neutron 安全组 |
| **network** | /etc/neutron/plugins/ml2/ml2\_conf.ini | enable\_security\_group = False | 禁止 Neutron 安全组 |

 nova 提供两种实现方式：使用 libvirt's nwfilter 的实现以及使用 linux iptables 的实现，默认的方式是使用 linux iptables。可以通过设置配置项  firewall\_driver 的值指定。需要注意的是，即使使用 iptables，依然使用了部分 nwfilter 功能。参见 <https://ask.openstack.org/en/question/19456/how-security-group-is-implemented/>

* firewall\_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
* firewall\_driver=nova.virt.libvirt.firewall.NWFilterFirewall

**1.2 CLI**

[复制代码](javascript:void(0);)

列表安全组：  
s1@controller:~$ nova secgroup-list-rules novasg1

+-------------+-----------+---------+-----------+--------------+

| IP Protocol | From Port | To Port | IP Range | Source Group |

+-------------+-----------+---------+-----------+--------------+

| tcp | 22 | 22 | 0.0.0.0/0 | |

+-------------+-----------+---------+-----------+--------------+  
创建一个安全组规则：

s1@controller:~$ nova secgroup-add-rule novasg1 udp 53 53 100.1.100.0/24  
+-------------+-----------+---------+----------------+--------------+  
| IP Protocol | From Port | To Port | IP Range | Source Group |  
+-------------+-----------+---------+----------------+--------------+  
| udp | 53 | 53 | 100.1.100.0/24 | |  
+-------------+-----------+---------+----------------+--------------+

删除虚机的安全组：

s1@controller:~$ nova remove-secgroup 2c59a875-bc23-4605-ad70-5315d7a3f8e2 novasg1

添加安全组到虚机：

s1@controller:~$ nova add-secgroup 2c59a875-bc23-4605-ad70-5315d7a3f8e2 novasg1

创建第二个安全组：

s1@controller:~$ nova secgroup-add-rule novasg2

添加规则：

s1@controller:~$ nova secgroup-add-rule novasg2 icmp -1 -1 0.0.0.0/0  
+-------------+-----------+---------+-----------+--------------+  
| IP Protocol | From Port | To Port | IP Range | Source Group |  
+-------------+-----------+---------+-----------+--------------+  
| icmp | -1 | -1 | 0.0.0.0/0 | |  
+-------------+-----------+---------+-----------+--------------+

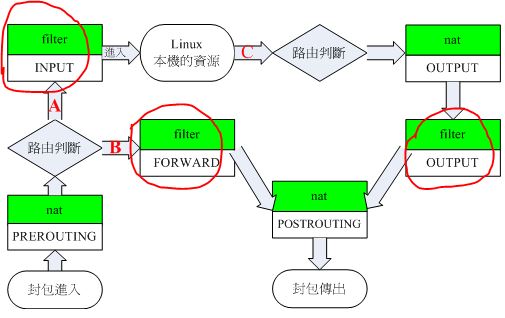
再添加安全组到虚机：

s1@controller:~$ nova add-secgroup 2c59a875-bc23-4605-ad70-5315d7a3f8e2 novasg2

[复制代码](javascript:void(0);)

**1.3 iptables 链**

Nova-compute 增加了 filter 表的 INPUT,OUTPUT 和 FORWARD 链：



[复制代码](javascript:void(0);)

-N nova-compute-FORWARD  
-N nova-compute-INPUT  
-N nova-compute-OUTPUT  
-N nova-compute-inst-122 #每个虚机一个链，命名规则是 ”inst“-<instance 在数据库中的 id>  
-N nova-compute-local  
-N nova-compute-provider  
-N nova-compute-sg-fallback  
-N nova-filter-top  
-A INPUT -j nova-compute-INPUT  
-A FORWARD -j nova-filter-top  
-A FORWARD -j nova-compute-FORWARD  
-A OUTPUT -j nova-filter-top  
-A OUTPUT -j nova-compute-OUTPUT  
-A nova-compute-FORWARD -s 0.0.0.0/32 -d 255.255.255.255/32 -p udp -m udp --sport 68 --dport 67 -j ACCEPT #允许本机上的虚机发出 DHCP 广播  
-A nova-compute-INPUT -s 0.0.0.0/32 -d 255.255.255.255/32 -p udp -m udp --sport 68 --dport 67 -j ACCEPT  #允许本机接受 DHCP 广播包  
-A nova-compute-inst-122 -m state --state INVALID -j DROP  
-A nova-compute-inst-122 -m state --state RELATED,ESTABLISHED -j ACCEPT  
-A nova-compute-inst-122 -j nova-compute-provider  
-A nova-compute-inst-122 -s 91.1.180.2/32 -p udp -m udp --sport 67 --dport 68 -j ACCEPT #接受该虚机所在子网的 DHCP Server 返回的包  
-A nova-compute-inst-122 -s 91.1.180.0/24 -j ACCEPT                                     #在配置项 allow\_same\_net\_traffic = true 的情况下接受同网段虚机的来访包  
-A nova-compute-inst-122 -p tcp -m tcp --dport 22 -j ACCEPT                             #用户安全组规则指定的来访包  
-A nova-compute-inst-122 -s 100.1.100.0/24 -p udp -m udp --dport 53 -j ACCEPT           #用户安全组规则指定的来访包  
-A nova-compute-inst-122 -p icmp -j ACCEPT                                              #用户安全组规则指定的来防爆  
-A nova-compute-inst-122 -j nova-compute-sg-fallback                                    #没被上面规则处理的其它来访包   
-A nova-compute-local -d 91.1.180.14/32 -j nova-compute-inst-122                        # “-d“ 决定了 nova 安全组只检查进入虚机的网络包  
-A nova-compute-sg-fallback -j DROP                                                     #丢弃其它包，只允许上述规则指定的网络访问  
-A nova-filter-top -j nova-compute-local

[复制代码](javascript:void(0);)

**2. FWaas**

**2.1 概念**

    从 Havana 版本开始，Neutron 提供一种基于 Neutron L3 Agent 的一种网络四层防火墙虚拟化参考实现 Firewall-as-a-service，简称 FWaas。本文的分析是基于 OpenStack Juno 版本进行的。Juno 版本中，FWaas 是分租户的，但是可以在多个租户之间共享。每个租户只允许一个防火墙。与物理的防火墙类似，FWaas 也有三个主要概念：

 （1）规则（Rule）：允许用户指定所要匹配的名称，描述，针对的协议（TCP, UDP, ICMP, ANY），行为（Allow，Deny），源/目的 IP 地址/子网 和 端口号/端口号区间。



与 neutron 安全组中的规则的区别是，这里需要指定被匹配到的数据包的处理行为是通过（ALLOW）和不通过（DENY），但是不能指定网络方向。FWaas 会将规则同时应用到进出网络的网络包上。

（2）策略（Policy）：规则的逻辑集合。Policy 可以是共享的 和 被审计的（Audited）。目前，FWaas 只是把 “audited” 保存到 DB 中，并没有对它做任何处理。



（3）防火墙（Firewall）：策略的逻辑集合。见上面右图。Juno 版本中，每个租户只能拥有最多一个 Firewall。防火墙可以是共享的。

    这里需要说明的是 FWaas 和 Security Group （安全组） 的区别。安全组规则在连接到一个实例的计算节点上的Linux桥 qbr 上实施，FWaaS 创建的防火墙规则在租户网络边缘实现的虚拟路由器上实施。 FWaaS 并不旨在取代安全组的功能，并且它提供更为补充安全组，特别是在其当前实现状态下。 FWaaS 目前缺乏安全组提供的一些功能，包括无法指定通信的方向等。与此相反，安全组，也因为他们缺乏创建特定规则拒绝所有流量的能力，因此需要 FWaas 作为补充。

**2.2 配置**

|  |  |
| --- | --- |
| 节点 | 配置和操作 |
| 控制节点上 | 修改 /etc/neutron/neutron.conf：  [default]  service\_plugins = router,lbaas,firewall  [service\_providers]  service\_provider = FIREWALL:Iptables:neutron.agent.linux.iptables\_firewall.OVSHybridIptablesFirewallDriver:default    service neutron-server restart    修改 /usr/share/openstack-dashboard/openstack\_dashboard/local/local\_settings.py：'enable\_firewall': True |
| 网络节点上 | 修改 /etc/neutron/fwaas\_driver.ini：  [fwaas] driver = neutron.services.firewall.drivers.linux.iptables\_fwaas.IptablesFwaasDriver enabled = True    service neutron-l3-agent restart |

**2.3 实现**

目前的实现中，FWaas 是通过在其所在 tenant 中的所有 Virtual Router 上添加 iptbales 规则来实现对进出数据网络的网络包进行控制的。

代码实现：

**控制节点上（class FirewallPlugin）：**

（1）create rule：纯数据库操作，将 rule 保存到数据中。

（2）create policy：纯数据库操作，将 policy 保存到数据中。

（3）如果将 rule 添加到一个已经添加到 firwall 的 policy，或者将一个 policy 加入一个存在的 firewall，在数据库操作后，通过 RPC fanout 到所有的 L3 Agent host：

{'args': {'firewall': {'status': 'PENDING\_UPDATE', 'name': u'fw-for-tcp', 'shared': None, 'firewall\_policy\_id': u'd14e23a3-2ee6-411d-b678-e6db3dac45f5', 'tenant\_id': u'74c8ada23a3449f888d9e19b76d13aab', 'admin\_state\_up': True, 'id': u'aa85bd66-dc4c-4d1b-909e-6f5736c279c7', 'firewall\_rule\_list': [{'protocol': u'icmp', 'description': u'', 'source\_port': None, 'source\_ip\_address': None, 'destination\_ip\_address': None, 'firewall\_policy\_id': u'd14e23a3-2ee6-411d-b678-e6db3dac45f5', 'position': 1, 'destination\_port': None, 'id': u'8658229d-6e34-4069-b091-e560f9e54dc9', 'name': u'rule-allow-icmp', 'tenant\_id': u'74c8ada23a3449f888d9e19b76d13aab', 'enabled': True, 'action': u'allow', 'ip\_version': 4L, 'shared': False}, {'protocol': u'tcp', 'description': u'', 'source\_port': None, 'source\_ip\_address': None, 'destination\_ip\_address': None, 'firewall\_policy\_id': u'd14e23a3-2ee6-411d-b678-e6db3dac45f5', 'position': 2, 'destination\_port': '80', 'id': u'00b5bad2-dd14-48d6-9a5c-7b65e6e8c480', 'name': u'fule-allow-tcp-80', 'tenant\_id': u'74c8ada23a3449f888d9e19b76d13aab', 'enabled': True, 'action': u'allow', 'ip\_version': 4L, 'shared': False}], 'description': u''}, 'host': 'controller'}, 'namespace': None, 'method': 'update\_firewall'}

* insert\_rule/remove\_rule/update\_firewall\_rule/update\_firewall\_policy -> Firewall\_db\_mixin.insert\_rule/remove\_rule/update\_firewall\_rule/update\_firewall\_policy -> \_rpc\_update\_firewall\_policy -> \_rpc\_update\_firewall -> (if policy has a firewall) FirewallAgentApi.update\_firewall
* create\_firewall -> Firewall\_db\_mixin.create\_firewall -> FirewallAgentApi.create\_firewall
* update\_firewall -> Firewall\_db\_mixin.update\_firewall -> FirewallAgentApi.update\_firewall
* delete\_firewall -> Firewall\_db\_mixin.delete\_firewall -> FirewallAgentApi.delete\_firewall
* FirewallAgentApi.create/update/delete\_firewall -> fanout\_cast ("create/update/delete\_firewall", topics.L3\_AGENT, "controller", firewall) -----> FWaaSL3AgentRpcCallback.create/update/delete\_firewall -> FWaaSL3AgentRpcCallback.\_invoke\_driver\_for\_plugin\_api

**网络节点上(class FWaaSL3AgentRpcCallback)：**

（1） 通过 RPC 获取所有的 router，在获取firewall 所在的 tenant 上的 routers

（2）调用 IptablesFwaasDriver.update\_firewall，依次更新每个 router 的 iptables 规则

（3）首先删除已有规则，然后根据配置的 rules 重新生成规则

[复制代码](javascript:void(0);)

root@network:/var/cache# ip netns exec qrouter-e438bebe-6795-4b68-a613-ec0df38d3064 iptables -t filter -S

-P INPUT ACCEPT

-P FORWARD ACCEPT

-P OUTPUT ACCEPT

-N neutron-filter-top

-N neutron-l3-agent-FORWARD

-N neutron-l3-agent-INPUT

-N neutron-l3-agent-OUTPUT

-N neutron-l3-agent-fwaas-defau #新增的 firewall chain

-N neutron-l3-agent-iv4aa85bd66 #新增的 firewall chain

-N neutron-l3-agent-local

-N neutron-l3-agent-ov4aa85bd66 #for firewall

-A INPUT -j neutron-l3-agent-INPUT

-A FORWARD -j neutron-filter-top

-A FORWARD -j neutron-l3-agent-FORWARD #将 forward 转到 neutron 的chain

-A OUTPUT -j neutron-filter-top

-A OUTPUT -j neutron-l3-agent-OUTPUT

-A neutron-filter-top -j neutron-l3-agent-local

-A neutron-l3-agent-FORWARD -o qr-+ -j neutron-l3-agent-iv4aa85bd66 #进数据网络的包

-A neutron-l3-agent-FORWARD -i qr-+ -j neutron-l3-agent-ov4aa85bd66 #出数据网络的包

-A neutron-l3-agent-FORWARD -o qr-+ -j neutron-l3-agent-fwaas-defau #进数据网络的包的默认处理 chain

-A neutron-l3-agent-FORWARD -i qr-+ -j neutron-l3-agent-fwaas-defau #出数据网络的包的默认处理 chain

-A neutron-l3-agent-INPUT -d 127.0.0.1/32 -p tcp -m tcp --dport 9697 -j ACCEPT

-A neutron-l3-agent-fwaas-defau -j DROP #默认丢弃没有被以上规则处理的所有包

-A neutron-l3-agent-iv4aa85bd66 -m state --state INVALID -j DROP

-A neutron-l3-agent-iv4aa85bd66 -m state --state RELATED,ESTABLISHED -j ACCEPT #接受状态为 RELATED, ESTABLISHED （已建立的连接）的包

-A neutron-l3-agent-iv4aa85bd66 -p tcp -m tcp --dport 80 -j ACCEPT #根据定义的 FWaas rule，接受目的端口为 80 的 tcp 包

-A neutron-l3-agent-ov4aa85bd66 -m state --state INVALID -j DROP

-A neutron-l3-agent-ov4aa85bd66 -m state --state RELATED,ESTABLISHED -j ACCEPT

-A neutron-l3-agent-ov4aa85bd66 -p tcp -m tcp --dport 80 -j ACCEPT #根据 FWaas rule，接收目的端口为 80 的 tcp 包

[复制代码](javascript:void(0);)