Section 1: In-Network Service Chaining

This section provides a step-by-step guide to setting up **In-Network Service Chaining** in OpenStack. The objective is to configure a **Service NAT VM** that allows communication between two networks (left-network and right-network).

1. Network Setup

I created two networks in OpenStack:

Left Network (10.10.10.0/24)

```
openstack network create left-network
openstack subnet create left-subnet \
    --network left-network \
    --subnet-range 10.10.10.0/24
```

Right Network (2.2.2.0/24)

```
openstack network create right-network
openstack subnet create right-subnet \
    --network right-network \
    --subnet-range 2.2.2.0/24
```

2. Creating the Service NAT VM

A Debian-based NAT VM is used to forward traffic between the two networks.

```
openstack server create \
    --image debian-10-openstack-amd64 \
    --flavor m1.small \
    --nic net-id=$(openstack network show left-network -c id -f value) \
    --nic net-id=$(openstack network show right-network -c id -f value) \
    --key-name my-key \
    --security-group default \
    service-nat-vm
```

3. Configuring NAT on the Service NAT VM

Once the Service NAT VM is running, SSH into it and configure IP forwarding and NAT.

3.1 Enable IP Forwarding

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Make it persistent:

```
echo "net.ipv4.ip_forward=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

3.2 Configure NAT for vm-right

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
sudo iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

3.3 Configure NAT for vm-left

```

sudo iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
sudo iptables -A FORWARD -i eth1 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

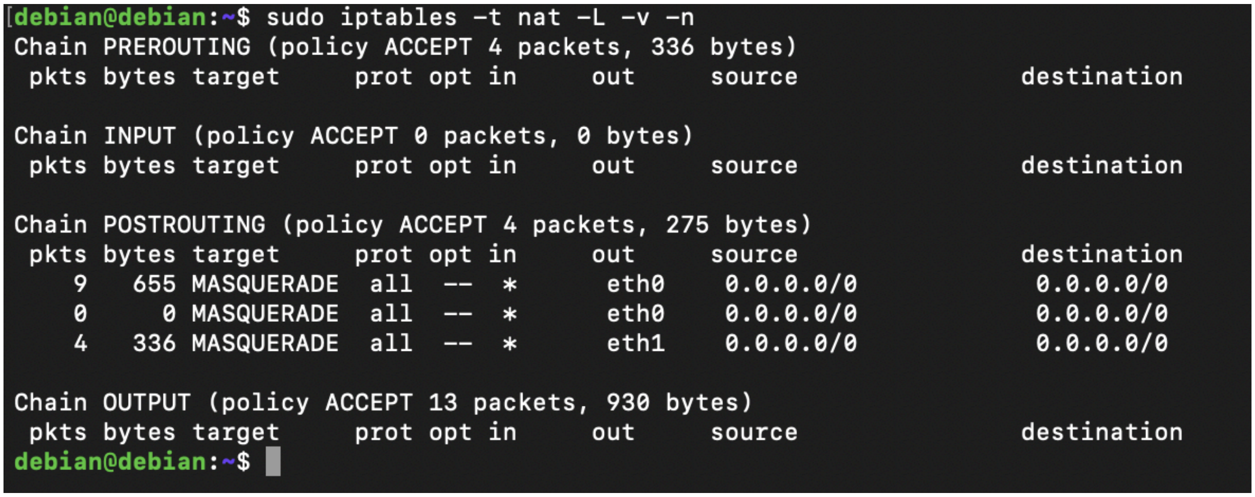
3.3 Make iptables Rules Persistent

```
sudo apt update
sudo apt install iptables-persistent -y
sudo netfilter-persistent save
sudo netfilter-persistent reload
```

4. Verification & Testing

4.1 Check NAT Rules

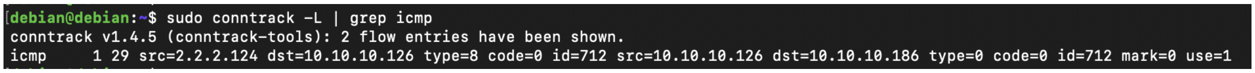
```
sudo iptables -t nat -L -v -n
```



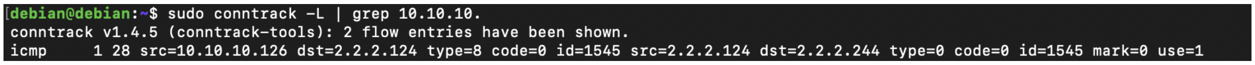
```
debian@debian:~$ sudo iptables -t nat -L -v -n
Chain PREROUTING (policy ACCEPT 4 packets, 336 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 4 packets, 275 bytes)
 pkts bytes target     prot opt in     out     source               destination
    9   655 MASQUERADE  all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
    0     0 MASQUERADE  all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
    4   336 MASQUERADE  all  --  *      eth1    0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT 13 packets, 930 bytes)
 pkts bytes target     prot opt in     out     source               destination
debian@debian:~$
```

4.2 Verify ICMP Traffic in NAT Translation

```
sudo conntrack -L | grep icmp
```



```
[debian@debian:~$ sudo conntrack -L | grep icmp
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
icmp     1 29 src=2.2.2.124 dst=10.10.10.126 type=8 code=0 id=712 src=10.10.10.126 dst=10.10.10.186 type=0 code=0 id=712 mark=0 use=1
```

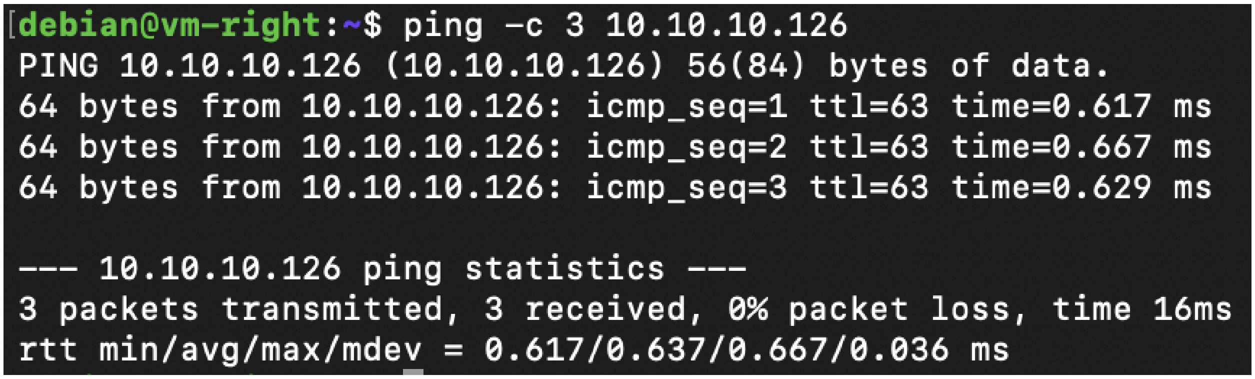


```
[debian@debian:~$ sudo conntrack -L | grep 10.10.10.
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
icmp     1 28 src=10.10.10.126 dst=2.2.2.124 type=8 code=0 id=1545 src=2.2.2.124 dst=2.2.2.244 type=0 code=0 id=1545 mark=0 use=1
```

✅**NAT is working** → The conntrack output shows that ICMP (ping) packets from vm-right (2.2.2.124) are being NAT-translated to vm-left (10.10.10.126) and vice versa.
✅**ICMP replies are being received** → vm-left (10.10.10.126) is replying back to 10.10.10.186 (the Service NAT VM) and vice versa.

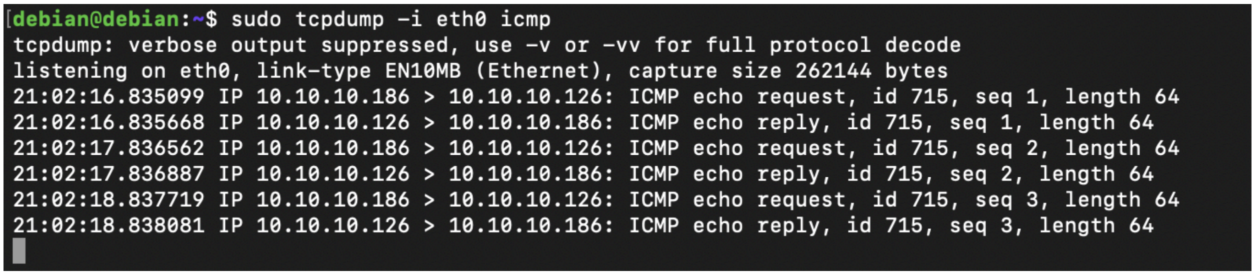
4.3 Test Connectivity

From vm-right to vm-left (*Should Work Through NAT*)



```
[debian@vm-right:~$ ping -c 3 10.10.10.126
PING 10.10.10.126 (10.10.10.126) 56(84) bytes of data.
64 bytes from 10.10.10.126: icmp_seq=1 ttl=63 time=0.617 ms
64 bytes from 10.10.10.126: icmp_seq=2 ttl=63 time=0.667 ms
64 bytes from 10.10.10.126: icmp_seq=3 ttl=63 time=0.629 ms

--- 10.10.10.126 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 16ms
rtt min/avg/max/mdev = 0.617/0.637/0.667/0.036 ms
```

Monitor Traffic on NAT VM



```
[debian@debian:~$ sudo tcpdump -i eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
21:02:16.835099 IP 10.10.10.186 > 10.10.10.126: ICMP echo request, id 715, seq 1, length 64
21:02:16.835668 IP 10.10.10.126 > 10.10.10.186: ICMP echo reply, id 715, seq 1, length 64
21:02:17.836562 IP 10.10.10.186 > 10.10.10.126: ICMP echo request, id 715, seq 2, length 64
21:02:17.836887 IP 10.10.10.126 > 10.10.10.186: ICMP echo reply, id 715, seq 2, length 64
21:02:18.837719 IP 10.10.10.186 > 10.10.10.126: ICMP echo request, id 715, seq 3, length 64
21:02:18.838081 IP 10.10.10.126 > 10.10.10.186: ICMP echo reply, id 715, seq 3, length 64
```

10.10.10.186 (eth0 of the NAT VM) is sending an ICMP echo request to 10.10.10.126 (vm-left).

10.10.10.126 (vm-left) responds with an ICMP echo reply to 10.10.10.186.

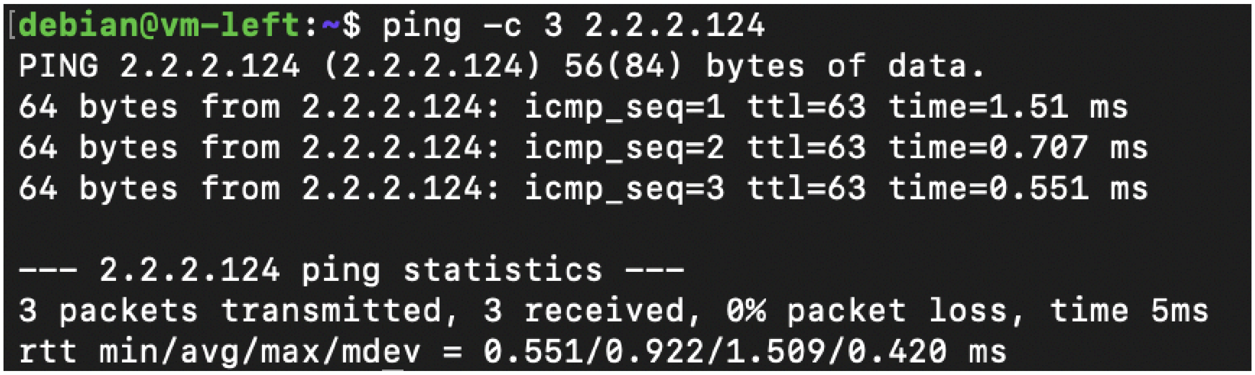
✅ **NAT is translating the source IP (2.2.2.124 → 10.10.10.186)**

- The original ping from vm-right (2.2.2.124) is **not directly visible** because it's **already translated** by NAT.
- The Service NAT VM **changes the source IP** to 10.10.10.186 (eth0's IP) before forwarding it to vm-left.

✅ **vm-left is replying correctly**

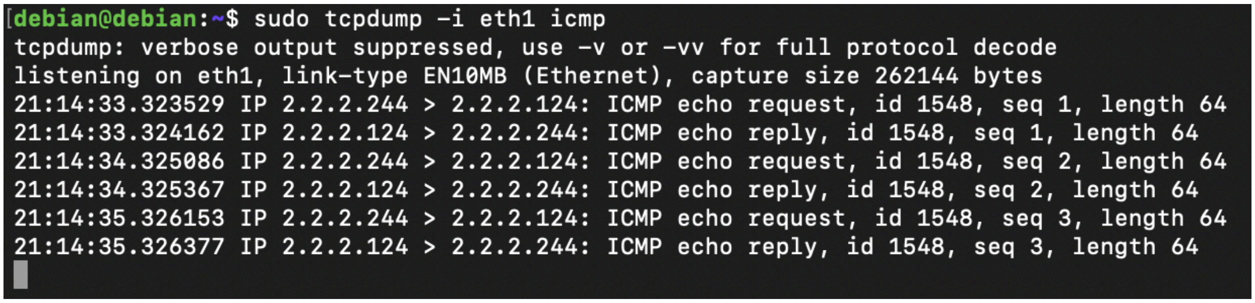
- The reply (10.10.10.126 → 10.10.10.186) shows that vm-left recognizes the NAT VM as the sender and responds to it.

From vm-left to vm-right (*Should Work Through NAT*)



```
[debian@vm-left:~$ ping -c 3 2.2.2.124
PING 2.2.2.124 (2.2.2.124) 56(84) bytes of data.
64 bytes from 2.2.2.124: icmp_seq=1 ttl=63 time=1.51 ms
64 bytes from 2.2.2.124: icmp_seq=2 ttl=63 time=0.707 ms
64 bytes from 2.2.2.124: icmp_seq=3 ttl=63 time=0.551 ms

--- 2.2.2.124 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 0.551/0.922/1.509/0.420 ms
```

Monitor Traffic on NAT VM



```
[debian@debian:~$ sudo tcpdump -i eth1 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
21:14:33.323529 IP 2.2.2.244 > 2.2.2.124: ICMP echo request, id 1548, seq 1, length 64
21:14:33.324162 IP 2.2.2.124 > 2.2.2.244: ICMP echo reply, id 1548, seq 1, length 64
21:14:34.325086 IP 2.2.2.244 > 2.2.2.124: ICMP echo request, id 1548, seq 2, length 64
21:14:34.325367 IP 2.2.2.124 > 2.2.2.244: ICMP echo reply, id 1548, seq 2, length 64
21:14:35.326153 IP 2.2.2.244 > 2.2.2.124: ICMP echo request, id 1548, seq 3, length 64
21:14:35.326377 IP 2.2.2.124 > 2.2.2.244: ICMP echo reply, id 1548, seq 3, length 64
```

2.2.2.244 (translated IP of vm-left on eth1 of the NAT VM) is sending an **ICMP echo request** to 2.2.2.124 (vm-right).

2.2.2.124 (vm-right) responds with an **ICMP echo reply** to 2.2.2.244.

✅ **NAT is translating the source IP (10.10.10.126 → 2.2.2.244)**

- The original ping from vm-left (10.10.10.126) is **not directly visible** because it has already been **NAT-ed** to 2.2.2.244.
- The Service NAT VM **changes the source IP** to 2.2.2.244 before forwarding it to vm-right.

✅ **vm-right is replying correctly**

- The reply (2.2.2.124 → 2.2.2.244) shows that vm-right **sees the request as coming from 2.2.2.244** and correctly responds.
- The NAT VM should now **translate the response back** to 10.10.10.126 and forward it to vm-left.

This completes **Section 1: In-Network Service Chaining**. **NAT is successfully implemented**, allowing vm-right to communicate with vm-left through the **Service NAT VM**.

Section 2: Transparent Service Chaining

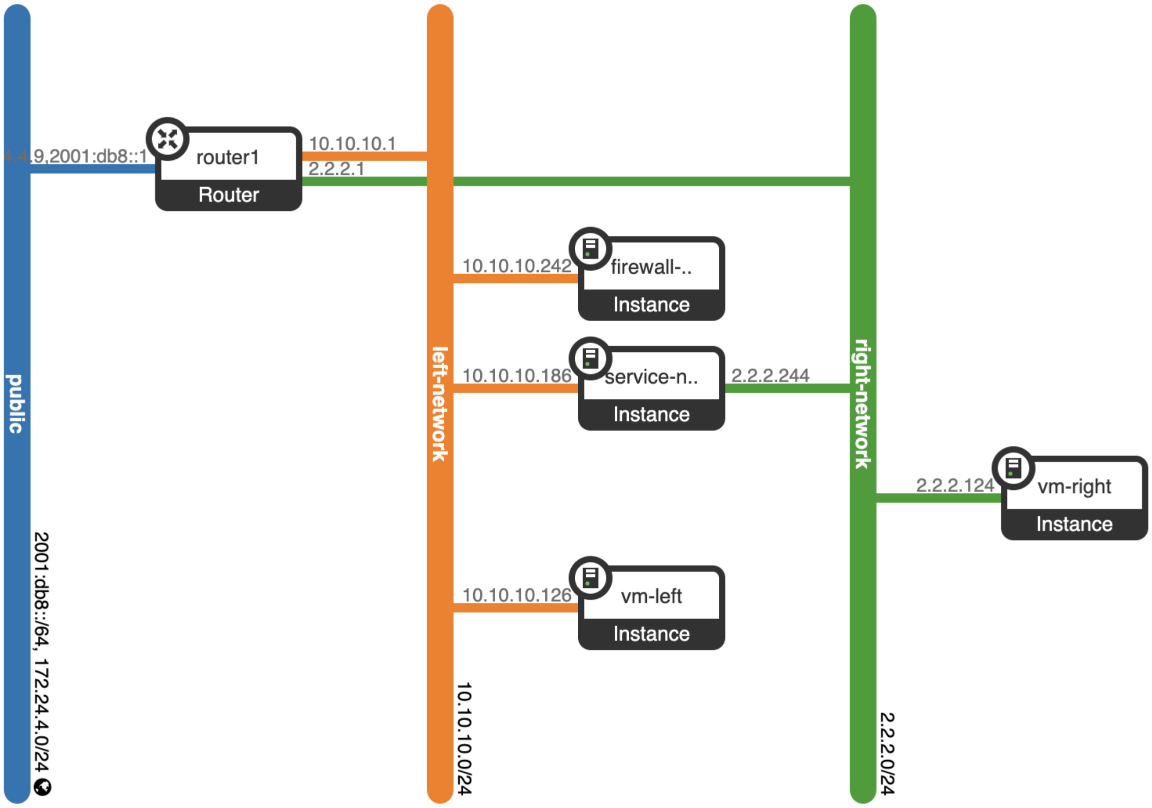
Objective

In this section, I introduced a **Layer 2 Firewall VM** into the service chaining setup. The goal is to enforce traffic flow through the firewall and block SSH traffic while allowing ICMP.

Network Setup

The updated network topology is as follows:

```
left-vm (10.10.10.126) ---> firewall-vm (10.10.10.242) ---> nat-vm
(10.10.10.186) ---> right-vm (2.2.2.124)
```



Component	Network	IP Address
left-vm	left-network	10.10.10.126
firewall-vm	left-network	10.10.10.242
service-nat-vm	left-network → right-network	10.10.10.186
right-vm	right-network	2.2.2.124

Traffic must pass through firewall-vm before reaching nat-vm.

1. Firewall VM Setup

Assign IP Address and Enable Forwarding

On **Firewall VM (`10.10.10.242`)**, run:

```
sudo ip addr add 10.10.10.242/24 dev eth0
sudo ip link set eth0 up
```

Enable IP forwarding to allow traffic through the firewall:

```
echo "net.ipv4.ip_forward = 1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

Now, packets can flow through the firewall.

2. Enforcing Traffic Flow Through Firewall

To force left-vm to send all traffic through firewall-vm, I set the default gateway:

On left-vm (*10.10.10.126*):

```
sudo ip route del default
sudo ip route add default via 10.10.10.242
```

Now, left-vm sends all traffic through firewall-vm and not directly to the nat-vm.

3. Firewall Rules (*iptables*) on firewall-vm

To enforce filtering, I applied the following rules on firewall-vm:

3.1. Block SSH Traffic (Port 22)

The following rules block SSH traffic from left-vm to right-vm and vice-versa.

```
sudo iptables -A FORWARD -s 10.10.10.126 -d 2.2.2.124 -p tcp --dport 22 -j DROP
sudo iptables -A FORWARD -s 2.2.2.124 -d 10.10.10.126 -p tcp --sport 22 -j DROP
```

SSH traffic is now blocked.

3.2. Allow ICMP (Ping) Traffic

The following rules allow ICMP traffic from left-vm to right-vm and vice-versa.

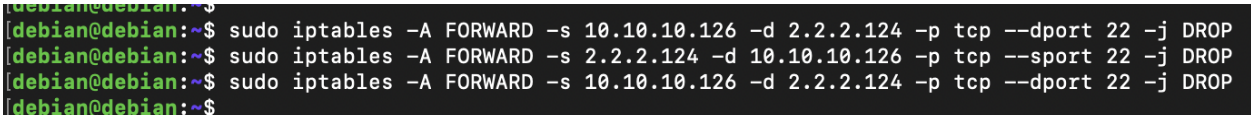
```
sudo iptables -A FORWARD -s 10.10.10.126 -d 10.10.10.186 -p icmp -j ACCEPT
sudo iptables -A FORWARD -s 10.10.10.186 -d 10.10.10.126 -p icmp -j ACCEPT
```

Ping should work between left-vm and nat-vm.

3.3. Allow All Other Traffic

```
sudo iptables -A FORWARD -j ACCEPT
```

All non-SSH traffic is allowed.

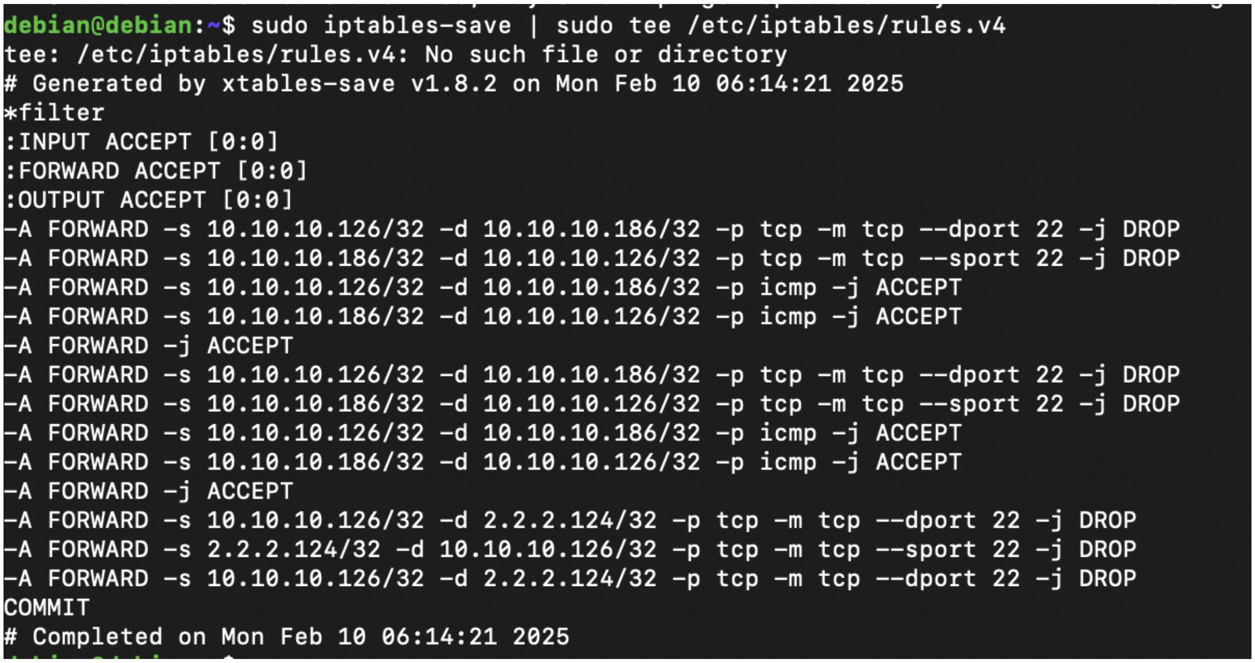


```
[debian@debian:~$
[debian@debian:~$ sudo iptables -A FORWARD -s 10.10.10.126 -d 2.2.2.124 -p tcp --dport 22 -j DROP
[debian@debian:~$ sudo iptables -A FORWARD -s 2.2.2.124 -d 10.10.10.126 -p tcp --sport 22 -j DROP
[debian@debian:~$ sudo iptables -A FORWARD -s 10.10.10.126 -d 2.2.2.124 -p tcp --dport 22 -j DROP
[debian@debian:~$
```

3.4. Make Rules Persistent

```
sudo apt install iptables-persistent -y
sudo iptables-save | sudo tee /etc/iptables/rules.v4
```

Firewall rules remain after reboot.

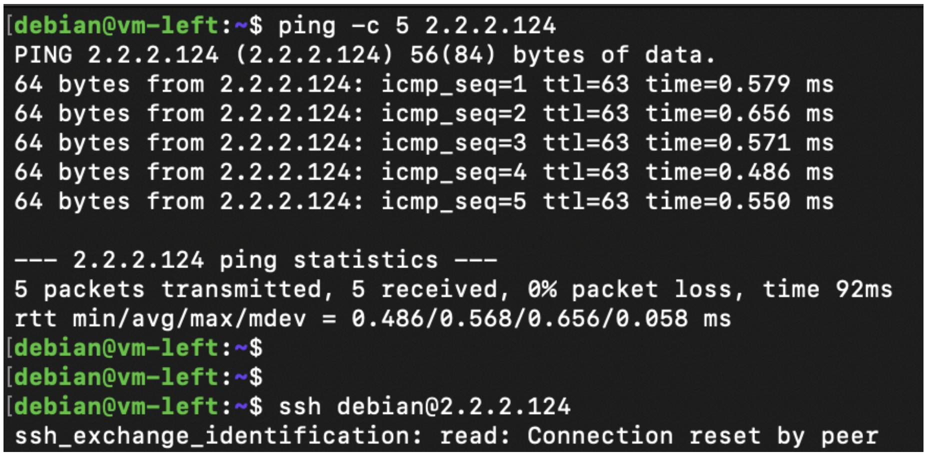


```
debian@debian:~$ sudo iptables-save | sudo tee /etc/iptables/rules.v4
tee: /etc/iptables/rules.v4: No such file or directory
# Generated by xtables-save v1.8.2 on Mon Feb 10 06:14:21 2025
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A FORWARD -s 10.10.10.126/32 -d 10.10.10.186/32 -p tcp -m tcp --dport 22 -j DROP
-A FORWARD -s 10.10.10.186/32 -d 10.10.10.126/32 -p tcp -m tcp --sport 22 -j DROP
-A FORWARD -s 10.10.10.126/32 -d 10.10.10.186/32 -p icmp -j ACCEPT
-A FORWARD -s 10.10.10.186/32 -d 10.10.10.126/32 -p icmp -j ACCEPT
-A FORWARD -j ACCEPT
-A FORWARD -s 10.10.10.126/32 -d 10.10.10.186/32 -p tcp -m tcp --dport 22 -j DROP
-A FORWARD -s 10.10.10.186/32 -d 10.10.10.126/32 -p tcp -m tcp --sport 22 -j DROP
-A FORWARD -s 10.10.10.126/32 -d 10.10.10.186/32 -p icmp -j ACCEPT
-A FORWARD -s 10.10.10.186/32 -d 10.10.10.126/32 -p icmp -j ACCEPT
-A FORWARD -j ACCEPT
-A FORWARD -s 10.10.10.126/32 -d 2.2.2.124/32 -p tcp -m tcp --dport 22 -j DROP
-A FORWARD -s 2.2.2.124/32 -d 10.10.10.126/32 -p tcp -m tcp --sport 22 -j DROP
-A FORWARD -s 10.10.10.126/32 -d 2.2.2.124/32 -p tcp -m tcp --dport 22 -j DROP
COMMIT
# Completed on Mon Feb 10 06:14:21 2025
```

4. Verification & Testing

✅ Test ICMP (*Ping Should Work*)

From left-vm (*10.10.10.126*):

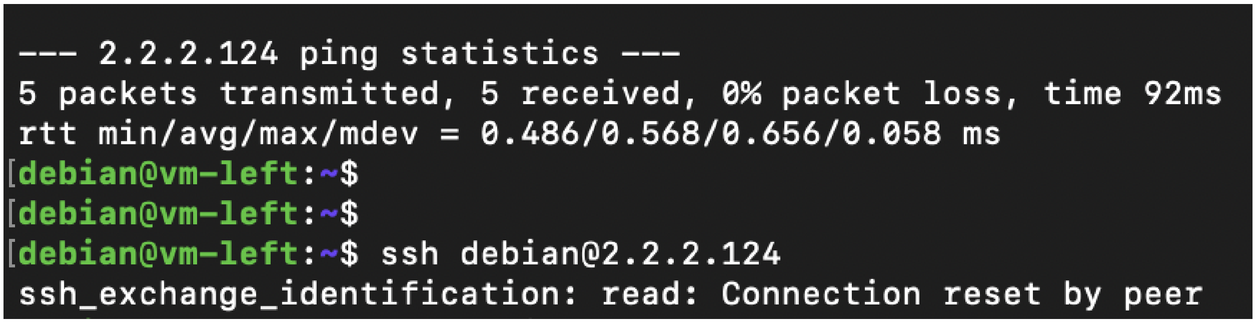


```
[debian@vm-left:~$ ping -c 5 2.2.2.124
PING 2.2.2.124 (2.2.2.124) 56(84) bytes of data.
64 bytes from 2.2.2.124: icmp_seq=1 ttl=63 time=0.579 ms
64 bytes from 2.2.2.124: icmp_seq=2 ttl=63 time=0.656 ms
64 bytes from 2.2.2.124: icmp_seq=3 ttl=63 time=0.571 ms
64 bytes from 2.2.2.124: icmp_seq=4 ttl=63 time=0.486 ms
64 bytes from 2.2.2.124: icmp_seq=5 ttl=63 time=0.550 ms

--- 2.2.2.124 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 92ms
rtt min/avg/max/mdev = 0.486/0.568/0.656/0.058 ms
[debian@vm-left:~$
[debian@vm-left:~$
[debian@vm-left:~$ ssh debian@2.2.2.124
ssh_exchange_identification: read: Connection reset by peer
```

Ping succeeded.

❌ Test SSH (*Should Be Blocked*)

From left-vm (*10.10.10.126)*:



```
--- 2.2.2.124 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 92ms
rtt min/avg/max/mdev = 0.486/0.568/0.656/0.058 ms
[debian@vm-left:~$
[debian@vm-left:~$
[debian@vm-left:~$ ssh debian@2.2.2.124
ssh_exchange_identification: read: Connection reset by peer
```

Now, firewall-vm successfully acts as a firewall, enforcing service chaining.

This completes **Section 2: Transparent Service Chaining**. Firewall VM is successfully implemented, blocking SSH from vm-right to vm-left through and vice-versa through the **Firewall VM**.