Network Firewall Lab

Contents

Lab Objectives and Setting

Overview

This lab explains firewall rules and related theory, and illustrates firewall configuration with examples. Students will play with iptables to get familiar with Linux built-in firewall software and common concepts about table, chain, and rules.

Objectives

Upon completion of this lab, students will be able to:

- Explain the basic concept of iptables
- Practice basic commands of iptables
- Manage basic firewall policy

Prerequisites

- Practical experience with SSH and basic Linux commands;
- Basic knowledge of network protocols and network devices.

EZSetup

EZSetup is a Web application capable of creating various user-defined cybersecurity practice environments (e.g., labs and competition scenarios) in one or more computing clouds (e.g., OpenStack or Amazon AWS). EZSetup provides a Web user interface for practice designers to visually create a practice scenario and easily deploy it in a computing cloud, which allows for customization and reduces overhead in creating and using practice environments. End users are shielded from the complexity of creating and maintaining practice environments and therefore can concentrate on cybersecurity practices. More information about EZSetup can be found at https://promise.nexus-lab.org/platform/.

Environment Setting

In this lab, students access three virtual machines (VM) in the cloud from EZSetup. They are server, firewall and client instances. The network topology for this lab is shown in Figure 1. The access information about the three VMs is provided in Table 1. Please refer to the EZSetup dashboard for the actual public IP addresses and passwords.

Table 1 VM properties and access information

Name	Image	RAM	VCPU	Disk	Login account
server	firewall	2GB	2	40GB	See EZSetup
firewall	firewall	2GB	2	40GB	See EZSetup
client	firewall	2GB	2	40GB	See EZSetup

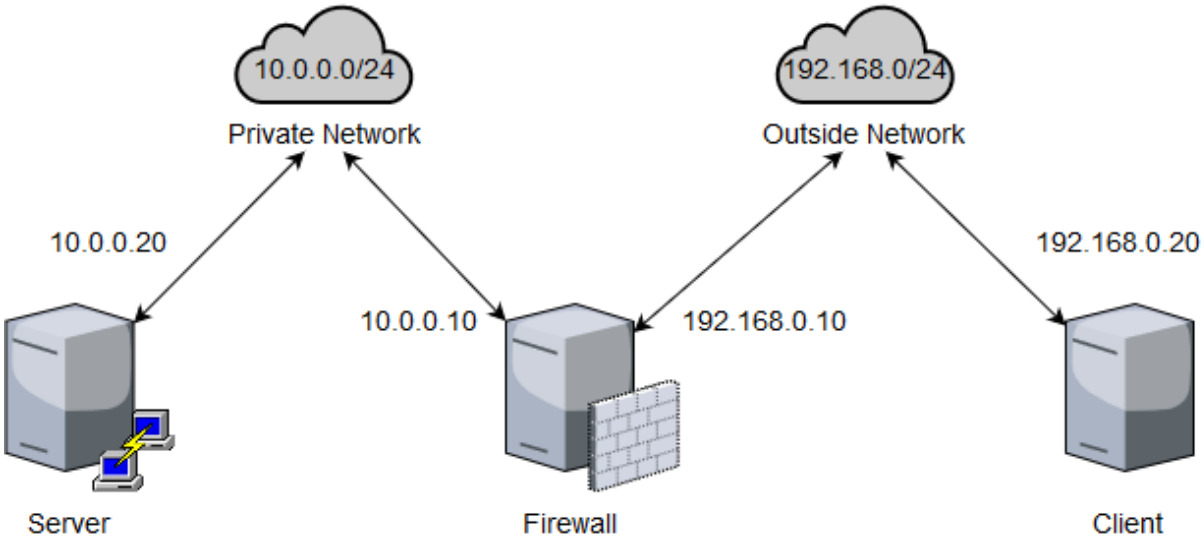

Figure 1 Lab network topology

License

In this lab, we understand the firewall concept and rules through the iptables. It is a firewall, installed by default on all official Ubuntu distributions. It is actually a front end to the kernel-level netfilter hooks that can manipulate the Linux network stack by configuring some specific rules. If the packet that crosses the

networking interface match the rules in the iptables, it will be processed depending on your predefined policy.

Also, in this lab, we create two networks: (1) private network (2) outside network. For the private network, it is behind the firewall and is connected to the Ethernet interface ens3 on the server and firewall instances. For the outside network, the Ethernet interface ens4 on firewall instances and the Ethernet interface ens3 on the client instance connects to it. In fact, the firewall instance connects both networks, which will play a role that manipulates the traffic in and out of the private network.

Each student will get a specific slice (a collection of resources that run in an isolated environment) of this lab, therefore many students can work on this lab each with his/her own unique practice environment. This ensures that each one can work separately without the need to worry about his/her own work being interfered by other users' operation.

1. Basic iptables Concepts

Before we use the iptables, we need to know the basic concepts of iptables. In the beginning, let's see the hierarchical structure of iptables: table, chain, and rules. The iptables uses tables to organize its rules. These tables classify rules depending on the type of decision they are used to make. For example, if the rule is used to decide whether to reject the packet to continue to its destination, it would be added to the filter table. There are several tables, each being used by some chains and specifying the main function of the rules as shown in the table below. In the most common use case, you will only use two of these: filter and nat. The other tables are aimed at complicated configuration involving multiple routers and routing decision. Thus, we will take some practices about the filter and nat table in task 3.

Table 2 iptables tables, chains and their functionalities

Table	**Chain**	**Functionality**
raw	PREROUTING OUTPUT	Configure packets so that they are exempt from connection tracking
mangle	PREROUTING INPUT FORWARD OUTPUT POSTROUTING	Packets content modification, such as TTL or TOS
nat	PREROUTING INPUT OUTPUT POSTROUTING	Network address translation (SNAT, DNAT, Masquerade)
filter	INPUT FORWARD OUTPUT	Packet filtering

Within each iptables table, rules are further organized within separate chains. The following table shows the corresponding the netfilter hook and function of each chain. Chains basically determine where rules are evaluated in the process flow of packets. Since each table has multiple chains, the table can be exerted at the different point in processing. By default, none of the chains contain any rules. It is up to you to append rules to the chains that you want to use. For example, if you want to block all incoming SSH traffic, we would add a rule to the INPUT chain of filter table.

Table 3 iptables chains, netfilter hooks and their functionalities

Chains	**Netfilter hook**	**Functionality**
PREROUTING	NF_IP_PRE_ROUTING	Implement the rules before routing
INPUT	NF_IP_LOCAL_IN	Traffic inbound from outside to the host
FORWARD	NF_IP_FORWARD	Traffic that uses the host as a router (source and destination are not the host)
OUTPUT	NF_IP_LOCAL_OUT	Traffic that the host sends out
POSTROUTING	NF_IP_POST_ROUTING	Implement the rules after routing

When a packet passes through a chain, it will match all the rules in this chain. It is certain that the matching process should follow the sequence. And, we have already known that the rules with similar purpose belong to a table. Thus, we need to consider the which table will be matched at first. Let's take an example when a packet walks through the PREROUTING chain. The packet looks as shown in Figure 2.

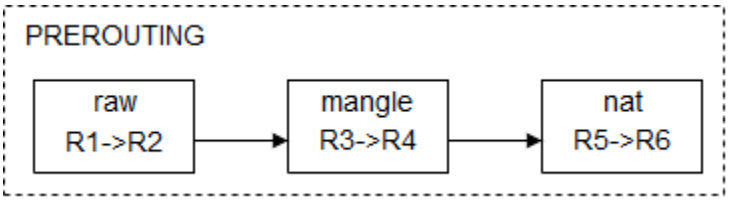

Figure 2 Packet walkthrough PREROUTING chain

As you can see, there are three tables in the PREROUTING chain and each table has two rules which are used R to express it. The matching sequence is

raw -> mangle -> nat

However, iptables serves four tables for us, the matching sequence is like this.

raw -> mangle -> nat -> filter

Now, let's see the simplified picture as shown below in Figure 3 that packet received on any interface traverses the iptables chains in the order.

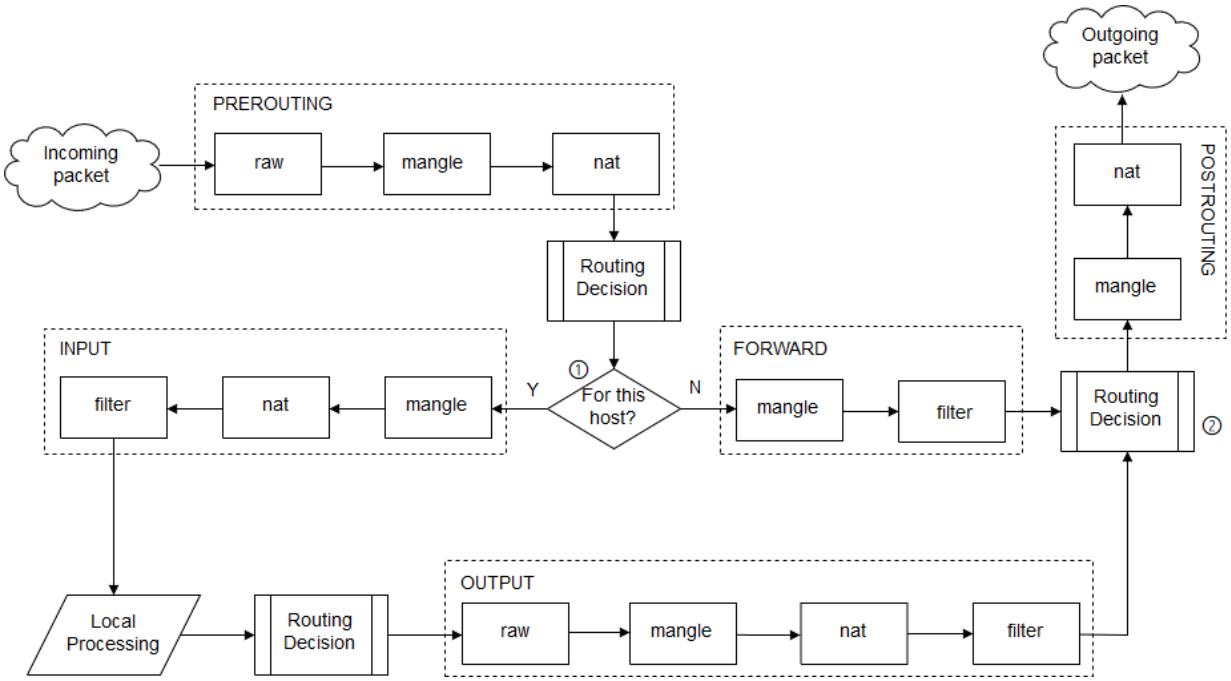

Figure 3 Packet traversing in iptables chains

The condition operation mark by ① involves deciding if the final destination of the packet is the local machine in which case the packet traverses through the INPUT chains or elsewhere in which case the packet traverses through the FORWARD chains. The routing decision marked by ② decides from which of the network interface to send out outgoing packets.

As a packet walks through each chain, rules on the chain are evaluated, one at a time, to see whether there is a match or not. If there is a match, the corresponding target action is executed. The three most commonl y used target actions are ACCEPT, DROP, REJECT.

2. Basic iptables Commands

Now, we have already learned the basic concept of iptables. After that, we will start to learn some iptables commands in task 2.

Task 2.1: List Rules

First, we need to know that iptables commands must be run with root privileges. This means that you need to login as root using “sudo su” command or precede all commands with “sudo”. In this lab, we are going to use “sudo” to implement the iptables commands.

To show the current rules that are configured for iptables. You need to run the iptables command with -L option like this:

$ sudo iptables -L

And you will see the output like this.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
```

```
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

For now, it is clear that there are three default chains (INPUT, OUTPUT, and FORWARD). Also, we can see each chain has ACCEPT as its default policy. We also see some column headers, but we do not see any actual rules. Now, let us add one sample rule in the firewall instance like this:

```
$ sudo iptables -A INPUT -i ens3 -s 10.0.0.20 -j DROP
```

Then sample rule above means we appended one rule to the end of the INPUT chain, which will drop all the incoming traffic from the IP address of server instance on the ens3 interface. The specific explanation of options will be introduced at the beginning of task 2.

Then If you want to limit the output to a specific chain, you can add the chain name directly after the -L option. For example, let's check the rules on INPUT chain by the following command.

```
$ sudo iptables -L INPUT
```

Because we have added a new rule, we will see the output like this.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  10.0.0.20           anywhere
```

The first line of output indicates the chain name (INPUT) followed by its default policy (ACCEPT). The next line is composed of the headers of each column in the tables, and is followed by the chain's rules. Let's go over what each header indicates from the Table 4 below.

Table 4 iptables chains column headers

Header	**Explanation**
target	If the packet matches the rule, the target specifies what should be done with it.
prot	The protocol, such as tcp, udp, icmp, or all
opt	IP options
source	The source of IP address
destination	The destination IP address

Now, let us ping the firewall instance from the server instance. You will see the output in the instance like this:

```
$ ping 10.0.0.10 -c3
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.

--- 10.0.0.10 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2016ms
```

It is obvious the incoming traffic from 10.0.0.20 was blocked by iptables on the firewall instance.

Task 2.2: Show Packet Counts and Aggregate Size

When we want to get a rough idea of which rules are matching against packets, we need to find a method to show the number of packets and the aggregate size of packets in bytes, that matched each specific rule. To do so, we use the -L and -v option together.

```
$ sudo iptables -L INPUT -v
```

You will see the output like this:

```
Chain INPUT (policy ACCEPT 104 packets, 7122 bytes)
 pkts bytes target     prot opt in     out     source     destination
    5   420 DROP       all  --  ens3   any     10.0.0.20     anywhere
```

If you want to clear the packet and byte counters for iptables rules, we can use the -Z option like the command line below. This is useful if you want to see if your server is receiving new traffic that matches your existing rules.

```
$ sudo iptables -Z
```

You can just clear the counter for rules in a specific chain. You just need to use the -Z option and specify the chain. For example, clear the INPUT chain counters following the command:

```
$ sudo iptables -Z INPUT
```

Now, let us check the number of packets match the sample rule in the INPUT chain again. You will find that the value of “pkts” is smaller than last screen dump.

```
$ sudo iptables -L INPUT –v
Chain INPUT (policy ACCEPT 8 packets, 416 bytes)
 pkts bytes target     prot opt in     out     source     destination
    0     0 DROP       all  --  ens3   any     10.0.0.20     anywhere
```

Task 2.3: Delete Rules

If you want to delete iptables rules by the specification, you can add -D option followed by the rule specification. For example, we can delete our sample rule like this:

```
$ sudo iptables -D INPUT -i ens3 -s 10.0.0.20 -j DROP
```

Then, you can check whether this sample rule is still on the INPUT chain. Sometimes, we need to delete all rules in a single chain, we will use -F option. For example, we can remove all the rules in the INPUT chain following the command line.

```
$ sudo iptables -F INPUT
```

If you want to flush all chains which will delete all the firewall rules, we still use -F option. The command line like this:

```
$ sudo iptables -F
```

3. Create Your Firewall Policy

In task 3, we will learn how to use iptables command line to create firewall policy depending on your requirement.

Task 3.1: Syntax Format

Firstly, two tables, Table 5, and Table 6, listed as below shows you how to write the iptables command lines. The Table 5 gives you a basic idea about syntax format.

Table 5 iptables syntax format

Table	Command	Chain	Parameter	Target
-t raw	-A	PREROUTING	-p	-j ACCEPT
-t mangle	-D	INPUT	-s	-j DROP
-t nat	-I	FORWARD	-d	-j REJECT
-t filter	-R	OUTPUT	-i	
	-L	POSTROUTING	-o	
	-F		--sport	
	-Z		--dport	
	-N			
	-X			
	-P			

Table 6 shows the options of command column in the first table. For example, we use -A option to append one or more rules to the end of the selected chain. Also, it introduces the functionality of different options of parameters.

Table 6 iptables command options and parameters

	Option	Functionality
Command	-A	Append one or more rules to the end of the selected chain.
	-D	Delete one or more rules from the selected chain.
	-I	Insert one or more rules in the selected chain as the given rule number.
	-R	Replace a rule in the selected chain.
	-L	List all rules in the selected chain.
	-F	Flush the selected chain
	-Z	Zero the packet and byte counters.
	-N	Create a new user-defined chain by the given name.
	-P	Set the policy for the chain to the given target.
	-X	Delete the optional user-defined chain specified.
Parameter	-p	The protocol of the rule or of the packet to check.

	Option	Functionality
	-s	Source specification.
	-d	Destination specification.
	-i	Name of an interface via which a packet was received.
	-o	Name of an interface via which a packet is going to be sent.

Task 3.2: Configure an Instance as Router

After we understand the basic syntax format, we can practice writing iptables rules. Before creating the iptables rule, let us bring the ens4 interface on firewall up following the command line below.

$ sudo ifconfig ens4 up

Then we are using “ifconfig” to assign an IP address to the ens4 interface.

$ sudo ifconfig ens4 192.168.0.10/24

Now, we see the information of ens4 as below when we use “ifconfig ens4” command to check.

```
ens4      Link encap:Ethernet  HWaddr fa:16:3e:24:82:75
          inet addr:192.168.0.10  Bcast:192.168.0.255
          Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe24:8275/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:405 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17398 (17.3 KB)  TX bytes:578 (578.0 B)
```

Now, let us set up firewall instance as a router by creating iptables rules and add a route on the server and client instance so that the firewall instance can connect the private network and outside network and manipulate the traffic between them.

Before adding iptables rules, we need to use the command below to uncomment the line that sets net.ipv4.ip_forward equal to 1, which means activate the packet forwarding function.

$ sudo sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/' /etc/sysctl.conf

Now, let us check whether we set the value of net.ipv4.ip_forward successfully following the command below.

$ sudo sysctl -p

Then we can add a rule to the FORWARD chain that incoming traffic from 10.0.0.0/24 subnet on the ens3 interface will be forwarded to the 192.168.0.0/24 subnet behind the ens4 interface just like the Figure 4 below.

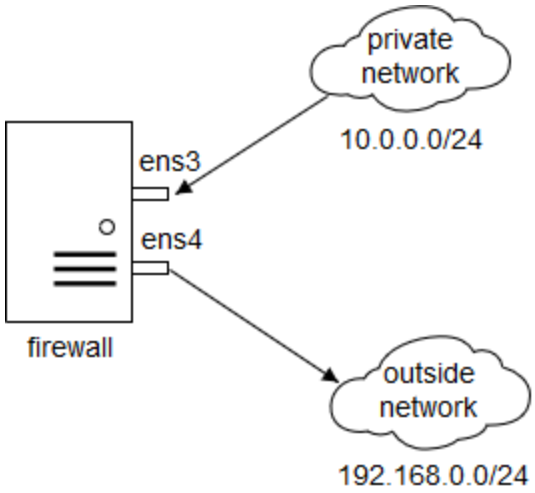

Figure 4 Packet forwarding from ens3 to ens4

$ sudo iptables -A FORWARD -i ens3 -o ens4 -s 10.0.0.0/24 -d 192.168.0.0/24 -j ACCEPT

Of course, you need the other rule that forwards the traffic from the 192.168.0.0/24 subnet on the ens4 to the 10.0.0.0/24 subnet on the ens3 so that instance on these two subnets can communicate with each other. Think about it, and there will be one of your exercises.

After you added the other rule, you need to add a route to the server and client instance. In the server instance, run the command line below.

$ sudo ip route add 192.168.0.0/24 via 10.0.0.10

In the client instance, execute the following command line.

$ sudo ip route add 10.0.0.0/24 via 192.168.0.10

After that, the firewall will play a role of a router that connects the private and outside network.

Task 3.3: NAT Table

In task 3.3, we will play with the NAT table. SNAT and DNAT will be introduced in this section.

Let us learn SNAT firstly. SNAT changes the source address of the packets passing through the Router. SNAT is typically used when an internal (private) host needs to initiate a session to an external (public) host. In this case, the firewall instance that is performing NAT changes the IP address of the source host in private network to the IP address of ens4 that connects the outside network, as shown in the following Figure 5.

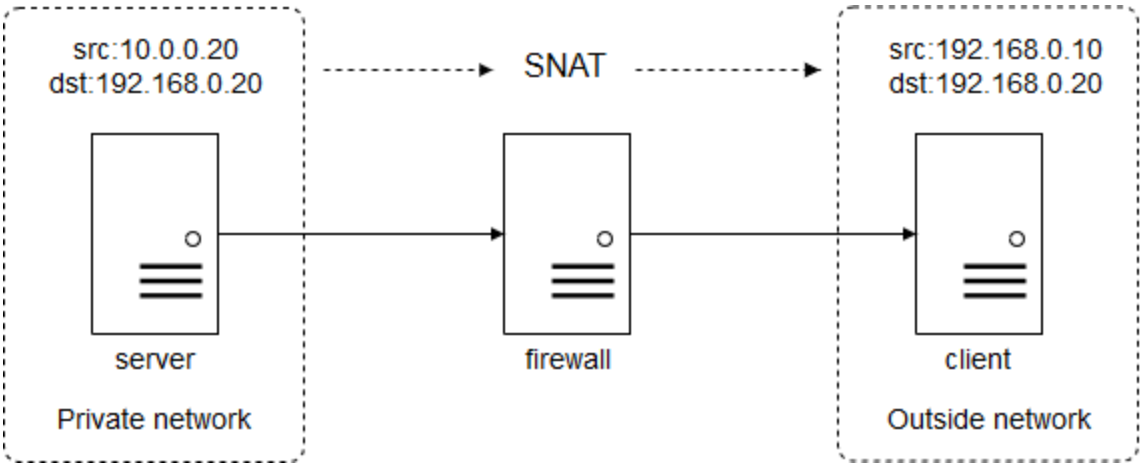

Figure 5 Source and destination IP addresses before and after SNAT

SNAT is performed **after** the routing decision is made. We use the command line like this:

$ sudo iptables -t nat -A POSTROUTING -s 10.0.0.20/24 -o ens4 -j SNAT --to-source 192.168.0.10

Afterward, you can use “tcpdump” command to listen to ICMP packets on the client instance. Then ping the client instance from server instance. You will see the output on the terminal of client instance like this:

$ sudo tcpdump icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens3, link-type EN10MB (Ethernet), capture size 262144 bytes
20:17:24.731865 IP 192.168.0.10 > 192.168.0.20: ICMP echo request, id 19109, seq 1, length 64
20:17:24.731915 IP 192.168.0.20 > 192.168.0.10: ICMP echo reply, id 19109, seq 1, length 64
20:17:25.732563 IP 192.168.0.10 > 192.168.0.20: ICMP echo request, id 19109, seq 2, length 64
20:17:25.732590 IP 192.168.0.20 > 192.168.0.10: ICMP echo reply, id 19109, seq 2, length 64

You can see these ICMP packets sent from the IP 192.168.0.10.

As we have already known the SNAT, let us see the DNAT. DNAT means that we translate the destination address of a packet to make it go somewhere else instead of where it was originally addressed. DNAT is commonly used when an external (public) host needs to initiate a session with an internal (private) host. In this case, while the traffic is coming from outside network to the private network, if the destination IP address is the ens3 interface, it will be changed to the IP address of the corresponding instance in private network, as shown in the following Figure 6.

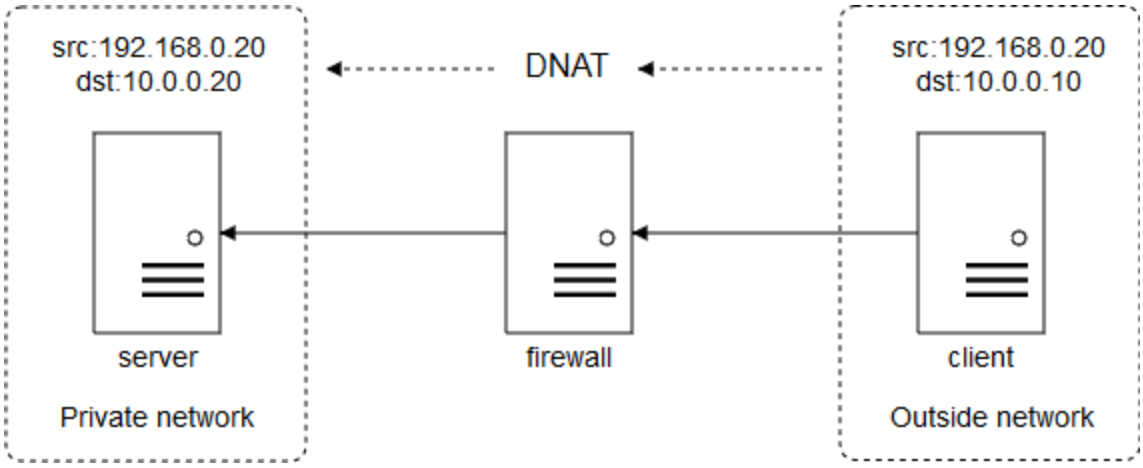

Figure 6 Source and destination IP addresses before and after DNAT

We need to know that DNAT is performed **before** the routing decision is made.

Task 3.4: Filter Table

The filter table is also frequently table when making the firewall policy. For example, if we want to drop all the ICMP traffic from client instance to the firewall instance, we need to add a rule on INPUT chain to filter the incoming traffic from client instance just like this:

$ sudo iptables -I INPUT -p icmp -s 192.168.0.20 -j DROP

Let the client instance ping the firewall instance. You will see that the client instance cannot receive any response from the firewall instance. The output is shown below.

$ ping 192.168.0.10 -c3
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.

--- 192.168.0.10 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2016ms

If we want to block the ICMP traffic forwarded to the server instance from the client instance, we would like to add a rule on FORWARD chain so that it will drop all the ICMP packet with destination IP 10.0.0.20 coming from the client instance. Run the following iptables command.

$ sudo iptables -I FORWARD -p icmp -s 192.168.0.20 -j DROP

Let the client instance ping the server instance. It is certain that the client instance cannot get the response. The output like this:

$ ping 10.0.0.20 -c3
PING 10.0.0.20 (10.0.0.20) 56(84) bytes of data.

--- 10.0.0.20 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 1999ms

Assignment

Task 3.2

1. Please add a rule that forwards the traffic from the 192.168.0.0/24 subnet on the ens4 to the 10.0.0.0/24 subnet on the ens3 so that instance on these two subnets can communicate with each other. After you finish it, please list the rules by iptables command.

2. After you added the route on server and client instances, can client instance ping server instance? What is the output? Please take a screenshot to show this result.

Task 3.3

1. Can you finish the DNAT function in the firewall instance using iptables command line? After you finish it, please list the rules.

2. Ping from client instance to the IP address of ens3, while turning on the tcpdump to inspect ICMP packets on the server instance. Please take a screenshot of the output of tcpdump.

Task 3.4

Before doing the exercises of task 3.4, you need to flush the rules in nat table by running “sudo iptables -t nat -F” command line.

1. Can you create an iptables rule on the firewall instance that blocks the client SSH (port 2222) and telnet (port 23) to the firewall instance? Please show the screenshot to prove that you cannot access the firewall through SSH and telnet.

2. Please add an iptables rule that block the client instance access the server instance through telnet (port 23). Take a screenshot of output when telnet the server instance from the client instance.

If we do not allow instance in the private network telnet to the firewall instance, can add a rule in the server instance? Take a screenshot of output when telnet the firewall instance from the server instance.

Complete all the tasks and save your answer (with screenshots) to each of the tasks into a PDF file. Submit the PDF file.