**CS4238 Lab: Linux Networking &**

**Firewall Configurations**

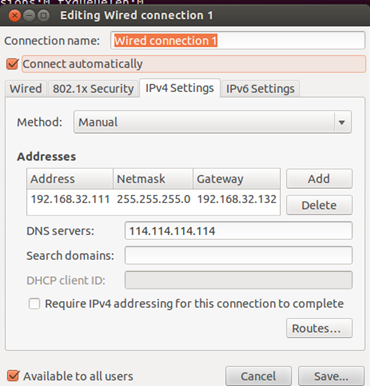
The **goal** of this lab is to get familiar with **Linux networking configuration** and its **firewall subsystem (iptables)**. You will also benefit from using tcpdump for observing traffic at selected network interfaces.

**Lab Set-up**

Imagine that you are building a corporate network with a *server* computer that is connected to the Internet through a *router* computer**.** In our setup, we will be using two VMs (running on our laptop) as the *server* and the *router.*

1. Create a VM using Virtualbox, let us call this the *router*. If you have already created a VM, you can skip this step and just rename your VM as the *router*.
2. Clone the aforementioned Ubuntu VM. You can name cloned VM as the *server*.
3. Remove any NAT adapter from the Server VM. Note that NAT is the default mode and it does not allow the two VMs to communicate with each other. So, add a host-only adapter to both VMs.

In the *server*, manually set the network connections. Note that this manual configuration will override the auto-config provided by the DHCP server. But before this, first, get the auto assigned IP addresses of the *router* and the *server*. You can use the *ip addr show* command for this at each VM.



1. Open the Wired Network Connection Editor at the *server.*
2. Set the IP address of the *server* to be the auto-assigned IP address got from the previous step.
3. Next, set the gateway at the server to be the *router*’s ip address.
4. Set the DNS server to 8.8.8.8.

In the *router*:

1. turn on **IP forwarding:** $ sudo sysctl -w net.ipv4.ip\_forward=1
2. Set the NAT rule with:

$ iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

This command assumes that your public/internet interface is enp0s3(eth0), and the local interface is enp0s8 (eth1). If not, please change them accordingly.

1. Accept traffic from local interface:

$ iptables -A INPUT -i enp0s8 -j ACCEPT

1. Allow established connections from the public interface:

$ iptables -A INPUT -i enp0s3 -m state --state ESTABLISHED,RELATED -j ACCEPT

1. Allow outgoing connections:

$ iptables -A OUTPUT -j ACCEPT

**Check Your Current Network Configuration**

On both VMs, run the following command:

$ ip addr show

How many interfaces can you see?

What types of interfaces are there?

What IP addresses are assigned to the interfaces?

Are you familiar with all of them?

**Note:**

You may know and prefer ifconfig tool. However, it is considered as obsolete now, since it does not support some features of a modern Linux kernel. A short comparison of ifconfig and ip command can be found at <https://p5r.uk/blog/2010/ifconfig-ip-comparison.html>.

Next, check your **routing table**:

$ ip route show

1. Can you explain the output?
2. Which interface will be used by packets heading to [www.comp.nus.edu.sg](http://www.comp.nus.edu.sg)?

If you are not sure, just ask your system:

$ ip route get *<target\_IP\_address>*

Finally, find out which DNS server is currently used for hostname resolution:

$ cat /etc/resolv.conf

**Linux Firewall, Wireshark/tcpdump**

1. Check status of firewall at your *server*:

$ iptables -L

What do you see? What is the purpose of the *chains*?

Would you expect such configuration in a real-world environment?

1. Now configure your firewall to **drop** packets from the host.
2. Run Wireshark. Enter icmp to the capture filter and start capturing traffic at enp0s8 interface.
3. Ping once IP address of your host machine:

$ ping <*host-ip*> -c 1

1. How many packets do you see in Wireshark?
2. Add a rule dropping all incoming packets from host machine:

$ iptables -A INPUT -s <*host-ip*> -j DROP

1. Ping once again.

What is the output of ping?

How many new packets do you see in Wireshark? Can you explain your observation?

1. List current firewall rules with line numbers and packet counters:

$ iptables -L -v --line-numbers

1. Remove the rule using its line number:

$ iptables -D INPUT 1

1. Check again the current iptables rules.

Stop the packet capture in Wireshark.

1. As an alternative to Wireshark, you can also try tcpdump:

$ sudo tcpdump –i enp0s8

**Firewall at Router**

In this part of the lab, you will need to check your firewall router at your *router* (the original VM).

**Initial Steps**

1. Print out current firewall rules at the *router*. What are the default policies?
2. On the *server*, choose any arbitrary unused port number, e.g. 10000, and run netcat server:

$ nc -l 10000

1. From the host, check you can send any string to the *server*:

$ nc <*Server-VM-IP*> 10000

Can you see the strings sent by the host on the *server*?

1. On the *router*, set the default policy of FORWARD chain to drop everything:

$ sudo iptables -P FORWARD DROP

1. Try to send strings to the *server*.

Can you ping it?

**First iptables Rules**

1. Discuss and set one firewall rule that allows incoming to the *server* only:

sudo iptables -A FORWARD ...

1. Test if your rule works.
2. Add new rules allowing SSH connections from the host to the *server*. Test it.

**Setting More Rules**

1. Run Wireshark at the *router*, and capture traffic at various interfaces. Try adding more firewall rules.

Set up more servers and generate relevant traffic from the host and   
see how traffic differs at the network level.

**Question**: for what purpose would you need to set firewall rules in the INPUT or OUTPUT chain?