Lab: Firewalls using IPTABLES

1 Overview

The objective of this lab is to (i) explore the functionalities of iptables, the Linux firewall and (ii) learn to create rules in iptables on the basis of ports/IPs. We will use SEED VM to do this lab and test our rules with the host machine. **You will be working in groups.**

2 Lab Environment

Virtual Machine: SEED can be downloaded from
https://drive.google.com/open?id=0BxtCKJljOE-NUnY0VGMwaDdyRTA

iptables: Iptables is a firewall, installed by default on all official Ubuntu distributions (Ubuntu, Kubuntu, Xubuntu). When you install Ubuntu, iptables is there, but it allows all traffic by default.You can read about it more here.

3 Lab Tasks

3.1 Task 1: Iptables basic commands

To enable and disable firewall:

```
> sudo ufw enable
> sudo ufw disable
```

To check if there are any current rules, type:

```
> sudo iptables -L
```

You should see:

Chain INPUT (policy DROP): This chain is used to filter the incoming connections, e.g. if a station tries to SSH your PC/server, iptables will check the rules and allow/reject the request.

Chain FORWARD (policy DROP): This chain is for incoming connections that are not actually destined to local machine, it's just for forwarding, iptables will check these rules and forward traffic accordingly.

Chain OUTPUT (policy ACCEPT): This chain is used to filter all outgoing connections, e.g. if you tries to ping an IP the firewall will check rules about it.

policy DROP means everything is dropped except specifically allowed.
policy ACCEPT means everything is accepted unless specified.

General command structure:

```
iptables <cmd> [<chain>] [<filter rules>] [-j <action>]
```

- commands: append rule (-A), insert rule (-I), delete rule (-D), replace rule (R), list rules (-L), flush rules (-F)
- chains: INPUT, OUTPUT, FORWARD
- filter rules: source/destination ip (-s/d <ip>), protocol (-p <tcp,udp,icmp,. . . >), interface (-i/-o <iface>), . . .
- actions: ACCEPT, DROP, RETURN, jump to user-defined chain, . . .

Let's create a rule which will block our ping request to a specific IP. First try this:

```
> ping monash.edu
```

Ping-able? Now block the ping request to `monash.edu` using following command:

```
> sudo iptables -A OUTPUT -s 0/0 -d monash-ip -j DROP
> sudo ufw enable
```

Try to ping monash-ip. Can you? Please check your firewall rules (hint: iptables -L) and find the drop rule. To delete all rules you can use:

```
> sudo iptables -F
```

3.2 Task 2: Creating Rules

To drop all ping traffic (icmp packets):

```
> sudo iptables -A INPUT -p ICMP --icmp-type 8 -j DROP
```

Open terminal in your 'host' system and ping VM IP (hint to check IPs: ifconfig). Is it Ping-able?
Now if you want to allow ICMP packets from your host but reject from others:

```
> sudo iptables -F
> sudo iptables -A INPUT -s host-ip -p ICMP --icmp-type 8 -j ACCEPT
> sudo iptables -A INPUT -p ICMP --icmp-type 8 -j DROP
> sudo iptables -L
```

You should be able to ping VM now.

What if you run above commands in reverse order, i.e DROP all ICMP packets first and then ACCEPT host-ip? Would it work? why or why not?

Let's drop traffic based on ports now. Please access your VM remotely from host using Telnet:

```
> telnet VM-IP
```

(username:seed password:dees)

As Telnet uses TCP port#23, we can block remote access of our VM using following command:

```
> sudo iptables -A INPUT -p tcp --dport 23 -j DROP
> sudo iptables -L
```

Try to remotely access your VM again. Is it accessible?

Please write rules for the following:

- Drop all incoming SSH connections
- Allow SSH connection from a particular machine
- Block all HTTP traffic and allow only HTTPS
- Block all FTP traffic from a particular machine