Installations

Installing VMWare Player

Download the latest version (v.5 or v.6) of VMware Player for your Operating System from https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/5_0

Installing Ubuntu OS

1. Download Ubuntu OS http://www.ubuntu.com/download/desktop and save it somewhere on your computer
2. Open up VMWare Player
3. Click on **Create a New Virtual Machine**
4. Select Installer disc image file (iso): browse for your Ubuntu .iso file and click **Next**
5. Type in your full name in the space provided. Use your J-number as Username (with a lowercase j). In my case, I use **natarajan** as the username. For your password, Select a password of your choice (easy to remember; but, difficult to find out by others). Click **Next** after entering the information.
6. Next, type in a name for your virtual machine (use your J-number again). Click **Next**.
7. On the next page, select **Store virtual disk as a single file**, and click **Next**.

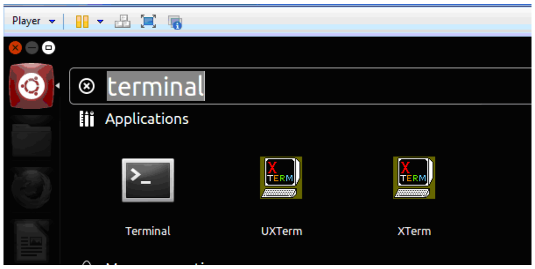
8. Click **Finish** on the next page and wait for the OS to be installed.

9. Next, log into Ubuntu OS with your password and press **Enter**.

10. Click the **Player** menu, and go to **Manage** then **Virtual Machine settings.**

11. When the settings come up, make sure that the **Network Adapter** is set to **NAT**, and click **OK**.

12. Launch a terminal by clicking the **Dash Home** (indicated in the picture below) and typing **terminal** in the box provided. Then click the **Terminal** icon.



Installing CentOS

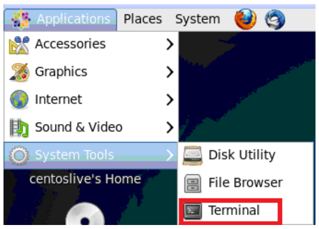
1. Download CentOS (CentOS-6.4-i386-LiveCD.iso) http://centos.icyboards.com/6.4/isos/i386/ and save it somewhere on your computer
2. Open up VMWare Player
3. Click on **Create a New Virtual Machine**
4. Select Installer disc image file (iso): browse for your CentOS .iso file and click **Next**
5. For Guest Operating System, choose Linux --> CentOS (**do not choose** CentOS 64-bit): we are using x86 version. Click **Next**. Give the VM - the name you want.
5. On the next page, select **Store virtual disk as a single file**, and click **Next**.

6. Click **Finish** on the next page.
7. Now Select CentOS from the VM Player menu and click **Play Virtual Machine.** Go through the OS installation process.
8. You can setup automatic login without requiring a password. If you wish to setup a password, you could also do so. You should be now logged into the CentOS system.

9. Click the **Player** menu, and go to **Manage** then **Virtual Machine settings.**

10. When the settings come up, make sure that the **Network Adapter** is set to **NAT**, and click **OK**.

11. Launch a terminal from the Applications --> System --> Terminal menu.



IPtables Tutorial

IPtables is a packet filter-based implementation of the Linux kernel firewall (netfilter). It defines tables that contain a chain of rules that specify how packets should be treated. The hierarchy is iptables --> tables --> chains --> rules. There may be built-in tables and chains as well as user-defined ones.

There are three independent tables (the presence of a table depends on the kernel configuration options): ***filter***, ***nat*** and ***mangle***. We specify the table to be used through the **-t** option.

- The ***filter*** table is the default table (if no -t option is used) and it has three built-in chains:
  INPUT (for packets destined for the local sockets);
  FORWARD (for packets being routed through a machine) and
  OUTPUT (packets originating from local sockets).
- The ***nat*** table is used when a packet encountered by the router/firewall has to go through network address translation. The nat table consists of three built-in chains:
  PRE-ROUTING - used to change the destination IP address of the incoming packets
  POST-ROUTING - used to change the source IP address of the outgoing packets
  OUTPUT - used to alter and send out the locally generated packets

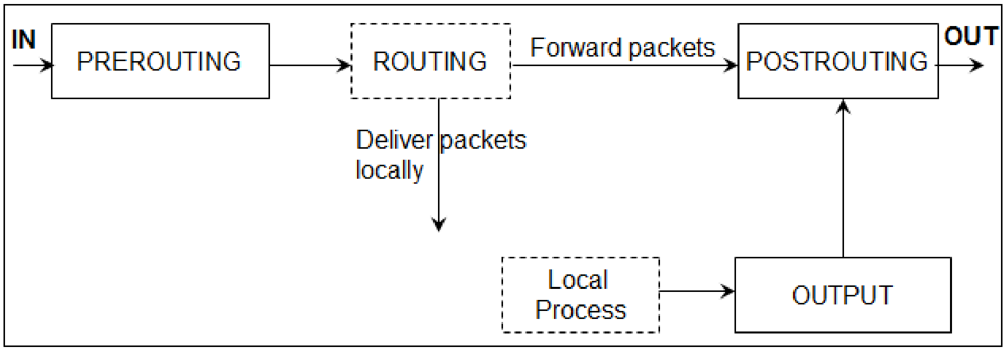


Figure 1: NAT Table

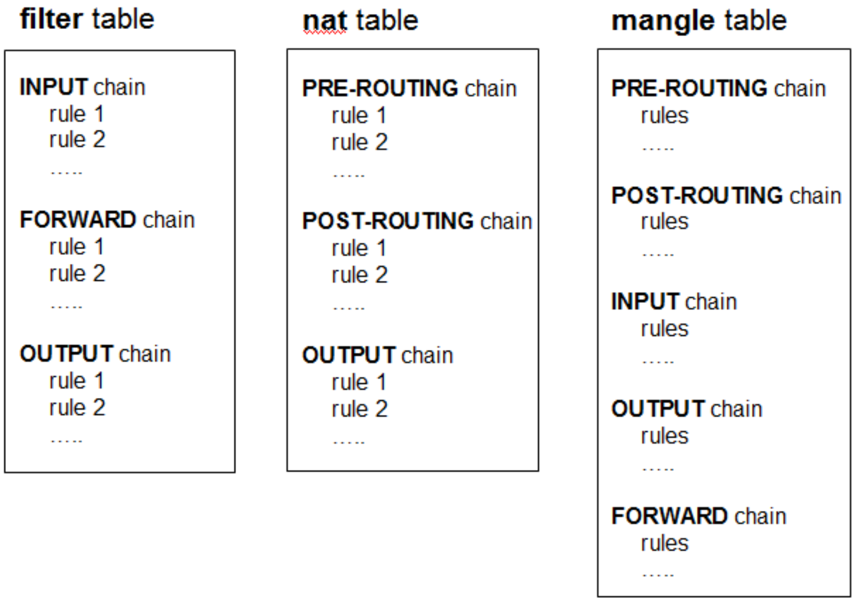


Figure 2: Tables and Chains of IPTables

- The ***mangle*** table is used to do some special alterations to the headers of packets that need some quality of service. Like the nat table, the mangle table has the pre-routing, post-routing and output chains (that have functionalities similar to those in the *nat* table) as well as input and forward chains (that have functionalities similar to those in the *filter* table).

A rule in a chain comprises of criteria and a target action.

Scenarios and IPTables commands

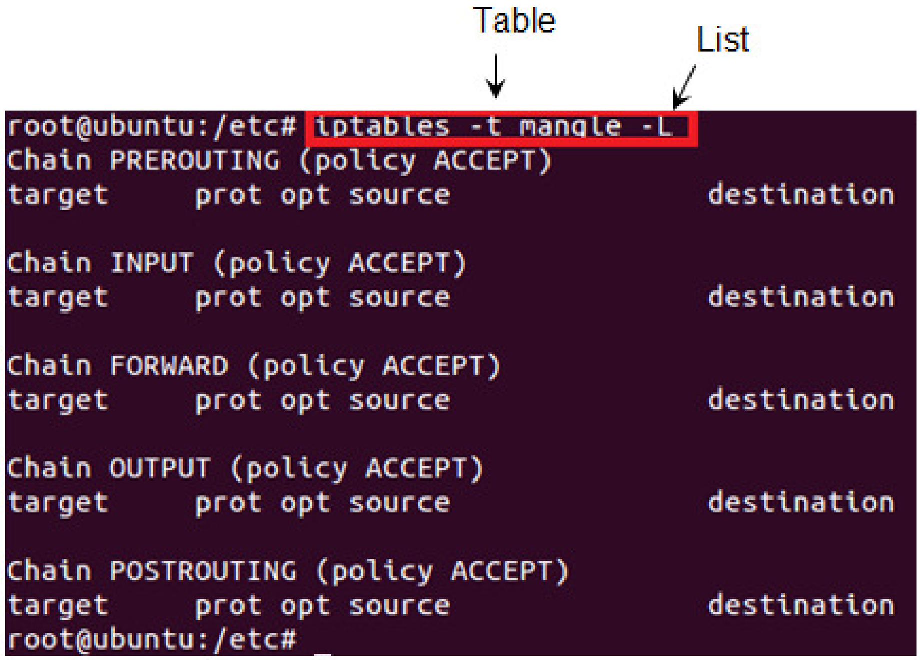
To change the contents or access the IPtables, one needs to have root access. Hence, I would suggest you login as root user. Otherwise, if you want to change/access the contents of IPtables as a regular user, you would have to prefix **sudo** upfront of every command as well as may be asked to enter the root password every time a command is run.

Assumption: Unless otherwise specified, for every scenario in this tutorial, all the chains are assumed to operate under a default-accept policy.

Validation Process: An incoming (or outgoing or transiting) packet is processed by the appropriate chain in the appropriate table (the filter table, by default). If a packet matches to the criteria in the chain, then the packet is subjected to the corresponding target action; otherwise, the packet is validated against the subsequent rules in the chain. If the packet cannot be matched with any of the criteria in the list, the packet is accepted (yes - the default policy for all chains of IPtables is to accept a packet, unless it matches to a criteria because of which the packet needs to be dropped).

S1: To list the contents of the *mangle* table of IPtables

Command: iptables -t mangle -L

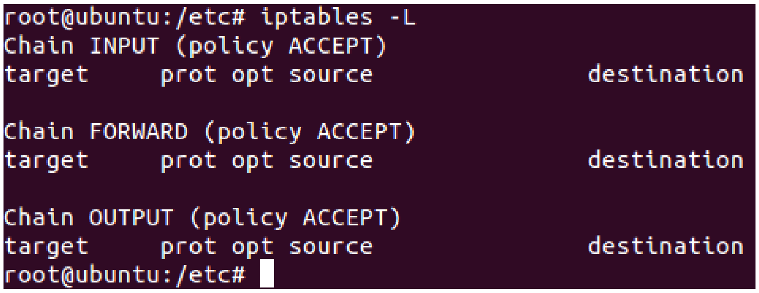


```
root@ubuntu:/etc# iptables -t mangle -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

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

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:/etc# _
```

As we see in the screenshot, the contents of the chains are empty and the default policy is ACCEPT. We will later see how to change this to DROP using the -P option (note it is uppercase 'P' for Policies and lowercase 'p' for ports).

S2: To list the contents of the *filter* table of IPtables

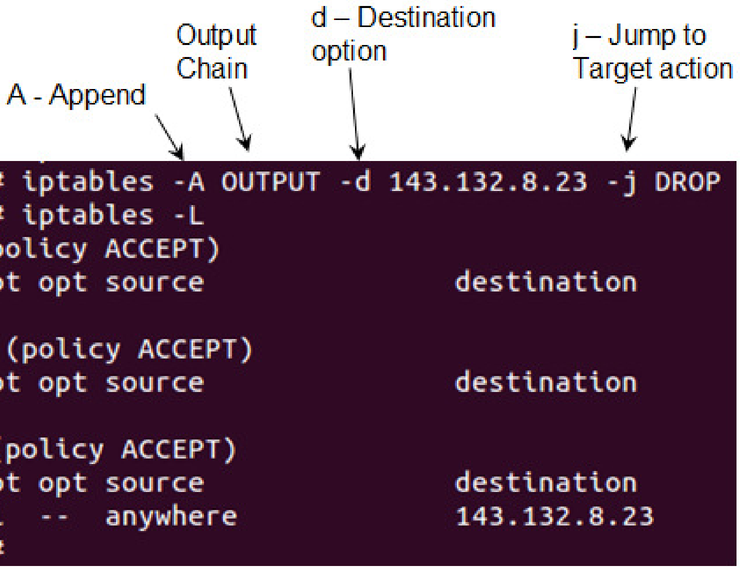
We do not need to use the -t option when we want to access the *filter* table. If we run an iptables command without the -t option, the *filter* table will be processed by default. **Command: iptables -L**



```
root@ubuntu:/etc# iptables -L
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
root@ubuntu:/etc# 
```

S3: To prevent a user on the local machine from visiting the Jackson State University web server whose IP address is 143.132.8.23.

Command: iptables -A OUTPUT -d 143.132.8.23 -j DROP



```
root@ubuntu:~# iptables -A OUTPUT -d 143.132.8.23 -j DROP
root@ubuntu:~# iptables -L
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
DROP       all  --  anywhere             143.132.8.23
root@ubuntu:~#
```

We could open a web browser and try to visit www.google.com; we could visit without any problem. On the other hand, try to visit www.jsums.edu; you will only see a message on the browser telling "connecting to...," but it could not connect eventually.

S4: To delete all the entries in the IP tables/chains.

Command: iptables -F

This command will delete/flush all the entries in the filter iptable. If you want to delete all the entries in the nat table, you need to then run **iptables -t nat -F**.

IMPORTANT NOTE: Note that the flush operation does not reset the default-accept or drop policy of a chain. One has to manually change the default policy of a chain to the intended policy.

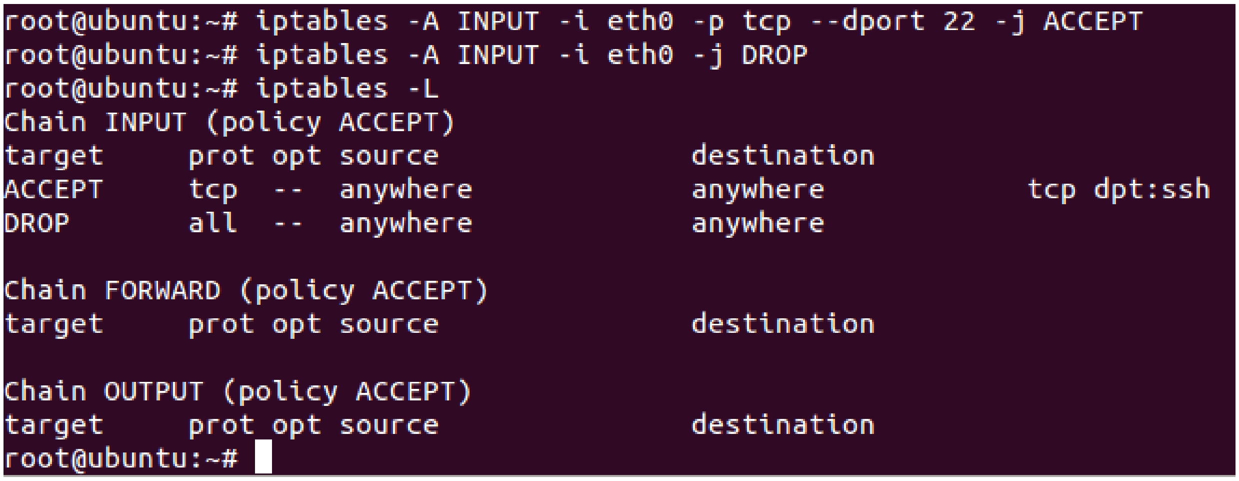
S5: Allow only SSH communications as incoming connection

If the objective is to allow only SSH communications as incoming connections, we could set the firewall to do this through two ways: In the first way, with the default policy being ACCEPT, the two rules are listed in this order: (i) Accept all incoming TCP packets coming to destination port 22 and (ii) Drop all other incoming packets (OR) In the second way, with the default policy changed to DROP, one can just setup a rule to accept all incoming TCP packets to destination port 22.

Method 1:

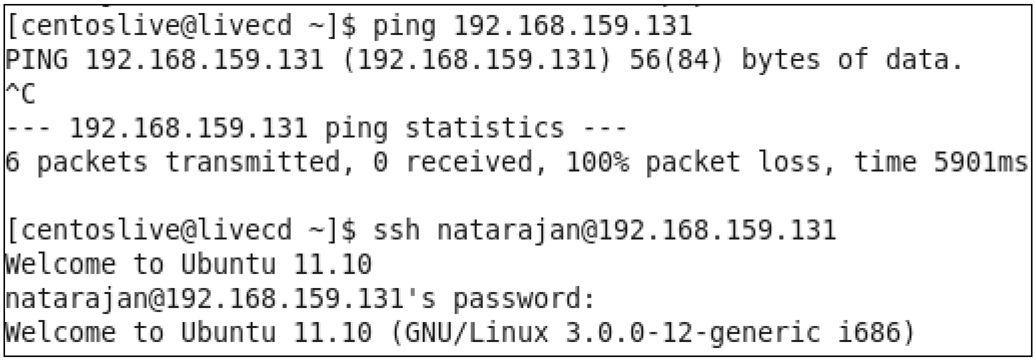
Commands (run in this order): Under a default-accept/allow policy, Once you have specified the rules to accept incoming an packet, it is better to specify a default rule to drop any incoming packets. Since rules are executed in numerical order, one after the other, starting from the first rule, the default rule to drop any incoming packets should be the last rule.

```
iptables  -A  INPUT    -i eth0   -p tcp   --dport 22  -j ACCEPT
iptables  -A  INPUT   -i  eth0   -j DROP
```



```
root@ubuntu:~# iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
root@ubuntu:~# iptables -A INPUT -i eth0 -j DROP
root@ubuntu:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~#
```

One can test the rules from another virtual machine (as shown below) running on the same network.

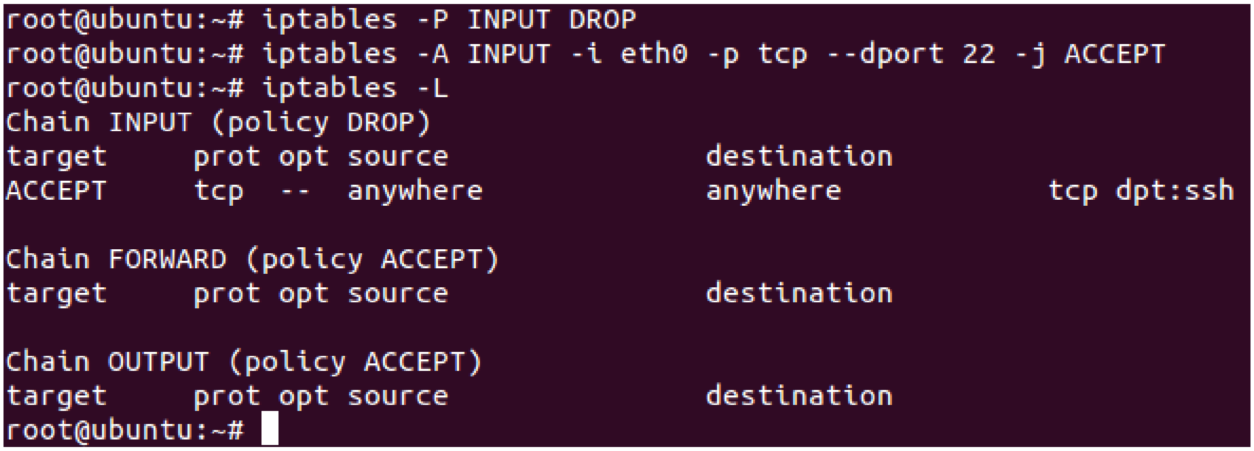


```
[centoslive@livecd ~]$ ping 192.168.159.131
PING 192.168.159.131 (192.168.159.131) 56(84) bytes of data.
^C
--- 192.168.159.131 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5901ms

[centoslive@livecd ~]$ ssh natarajan@192.168.159.131
Welcome to Ubuntu 11.10
natarajan@192.168.159.131's password:
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)
```

Method 2:

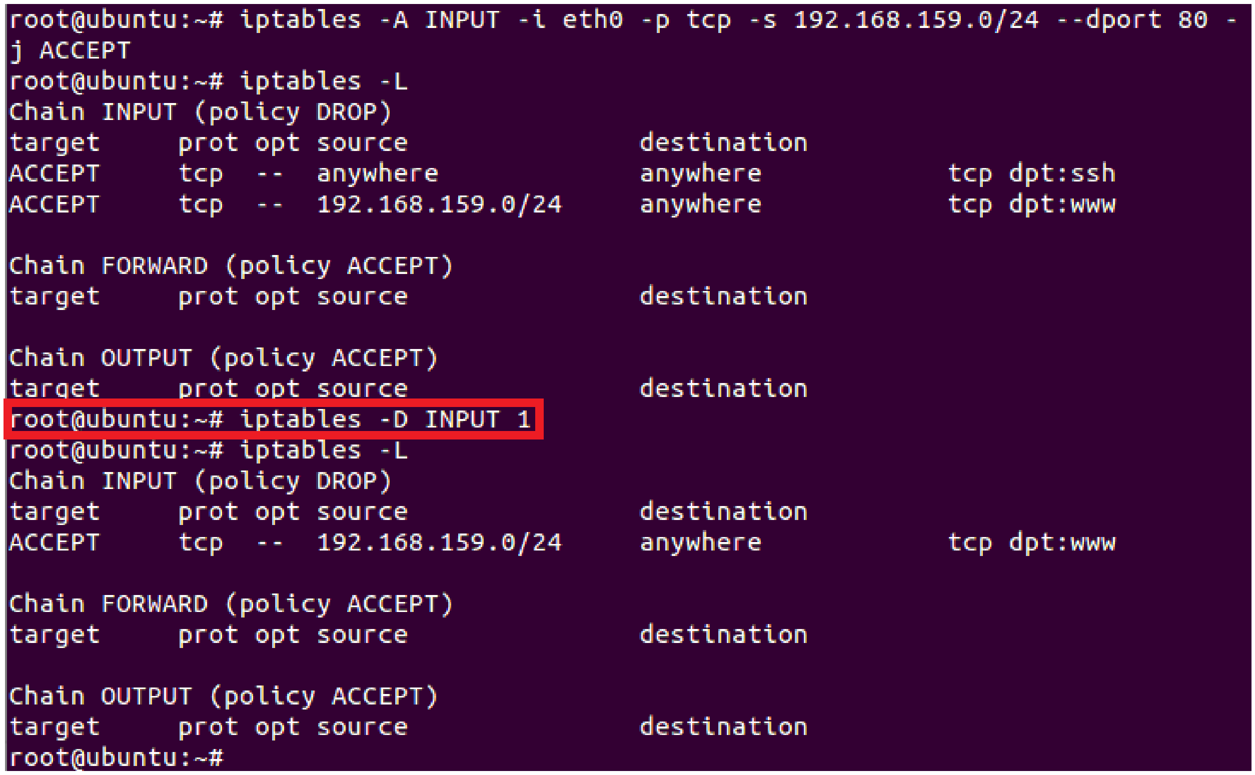
Commands (run in either order should be fine): Note that the uppercase 'P' denotes policy. We are changing the default input policy to DROP. That is, if an incoming packet does not match to any criteria corresponding to the rules in the INPUT chain, the packet will be dropped. This is the default-deny policy.

iptables -P INPUT DROP
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT



```
root@ubuntu:~# iptables -P INPUT DROP
root@ubuntu:~# iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
root@ubuntu:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~# 
```

__S6:__ Lets continue from Method 2 of Scenario 5, where we set the default policy for the INPUT chain to be DROP and configured the firewall to accept only incoming SSH connection requests. Lets first add a rule that would allow only hosts from a particular network (with prefix say 192.168.159.0/24) to send web traffic to the Linux host; all other traffic should be dropped. After configuring the above rule, lets delete the first rule to allow SSH packets.

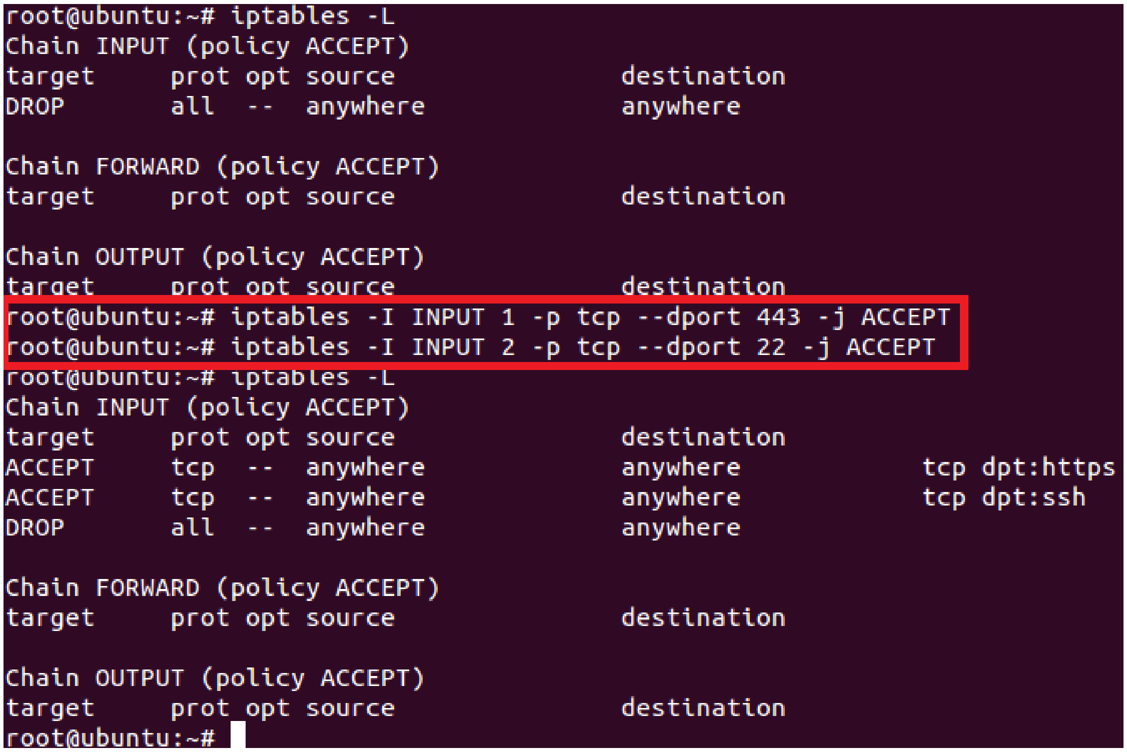


```
root@ubuntu:~# iptables -A INPUT -i eth0 -p tcp -s 192.168.159.0/24 --dport 80 -
j ACCEPT
root@ubuntu:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     tcp  --  192.168.159.0/24     anywhere             tcp dpt:www

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~# iptables -D INPUT 1
root@ubuntu:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.159.0/24     anywhere             tcp dpt:www

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~#
```

__S7:__ Flush the contents of the iptables resulting at the end of Scenario 6. Configure the INPUT chain to default-accept policy (note that this has to be done manually; the flush operation wouldn't do this for us).

Lets say, by mistake, I then configured the firewall with a rule not to accept any incoming traffic. However, I realized later that I want to insert a rule that allowed the firewall to accept any incoming traffic to port 443 (https) and port 22 (ssh).

Commands (in this order): Note that everything following # is considered a comment.
iptables -F
iptables -P INPUT ACCEPT
iptables -A INPUT -j DROP
iptables -L # not needed if you do not want to see the contents of the iptables until now.
iptables -I INPUT 1 -p tcp --dport 443 -j ACCEPT
iptables -I INPUT 2 -p tcp --dport 22 -j ACCEPT

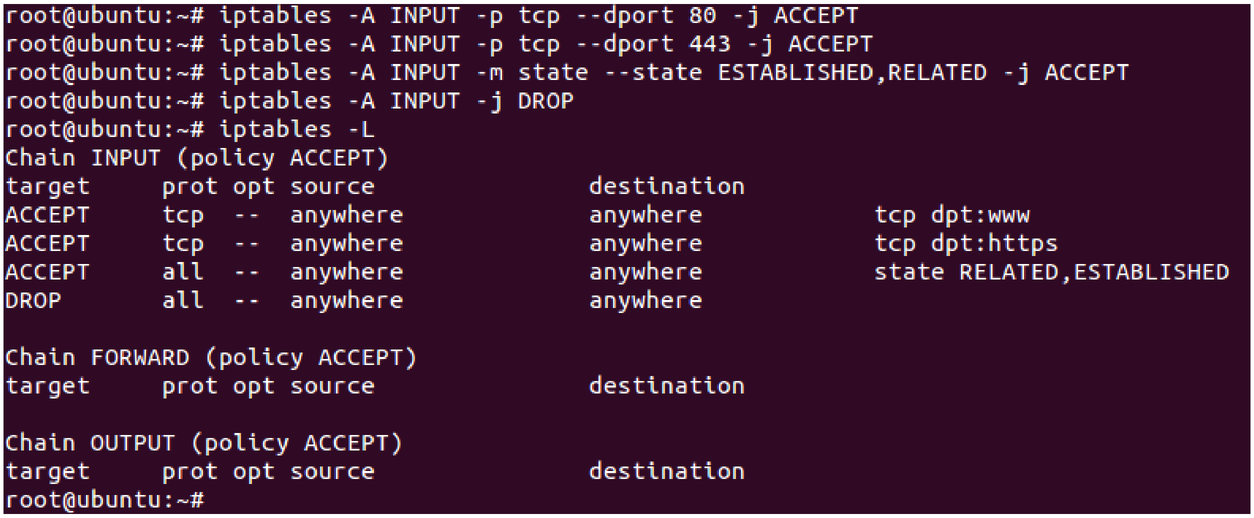


```
root@ubuntu:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~# iptables -I INPUT 1 -p tcp --dport 443 -j ACCEPT
root@ubuntu:~# iptables -I INPUT 2 -p tcp --dport 22 -j ACCEPT
root@ubuntu:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~#
```

<u>**S8:**</u> With the configurations setup in Scenario 7, one can notice that we cannot still visit any website, because we need a domain name resolution to find the IP address of the web server whose domain name/website address is entered in the browser and we do not allow DNS traffic (port 53). Also, websites that use HTTPS need to be setup access for both ports 80 and 443. Websites that use only port 80 cannot be visited either as this port is not setup for ACCEPT in Scenario 7. In Scenario 8, we will basically do an enhanced implementation of Scenario 7 permitting packets from any website and drop all other incoming packets, including SSH, which we can test.

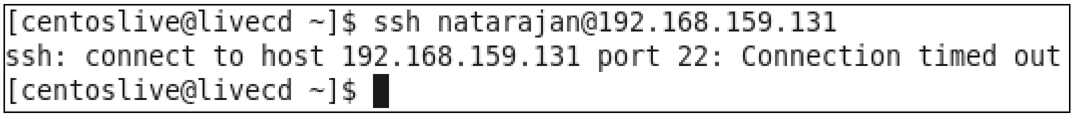
When it comes to permitting web traffic, we do not know what other protocols/ports and the corresponding packets need to be let in. So, it is better to insert a rule that lets packets that are related and/or as part of established sessions need to be permitted in. This could be done using the -m option. Note that -m option could be used in three contexts: to limit the number of times the rule has to match; multiport option (both of which we will see later) and the state of new, existing or related connections. We will the -m option for this scenario in the last context. The syntax to use the -m option in this context is **-m state --state ESTABLISHED,RELATED**. Note that there should not be blank space between words RELATED and ESTABLISHED.

Commands (the rule with the -m option can be either the first one or after the two tcp rules):
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -j DROP



```
root@ubuntu:~# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
root@ubuntu:~# iptables -A INPUT -p tcp --dport 443 -j ACCEPT
root@ubuntu:~# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
root@ubuntu:~# iptables -A INPUT -j DROP
root@ubuntu:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:www
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~#
```

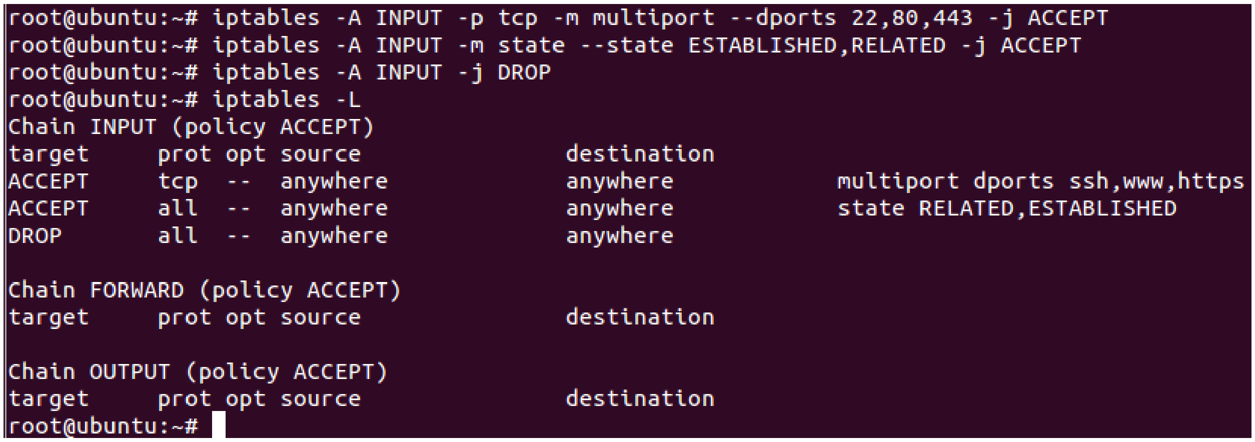
As we can now see, SSH connection to the Ubuntu host (192.168.159.131) is not accepted.



```
[centoslive@livecd ~]$ ssh natarajan@192.168.159.131
ssh: connect to host 192.168.159.131 port 22: Connection timed out
[centoslive@livecd ~]$
```

S9: We will configure the firewall to allow SSH, HTTP and HTTPS traffic and block any other protocol incoming traffic. We will do this using the multiport option.

The syntax is to use **-m multiport --dports 22,80,443** along with the rest of the parameters for the rule as indicated in the screenshot. Note that there should not be any blank space in between the port numbers.



```
root@ubuntu:~# iptables -A INPUT -p tcp -m multiport --dports 22,80,443 -j ACCEPT
root@ubuntu:~# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
root@ubuntu:~# iptables -A INPUT -j DROP
root@ubuntu:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             multiport dports ssh,www,https
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~#
```

Now, SSH traffic is also allowed in.

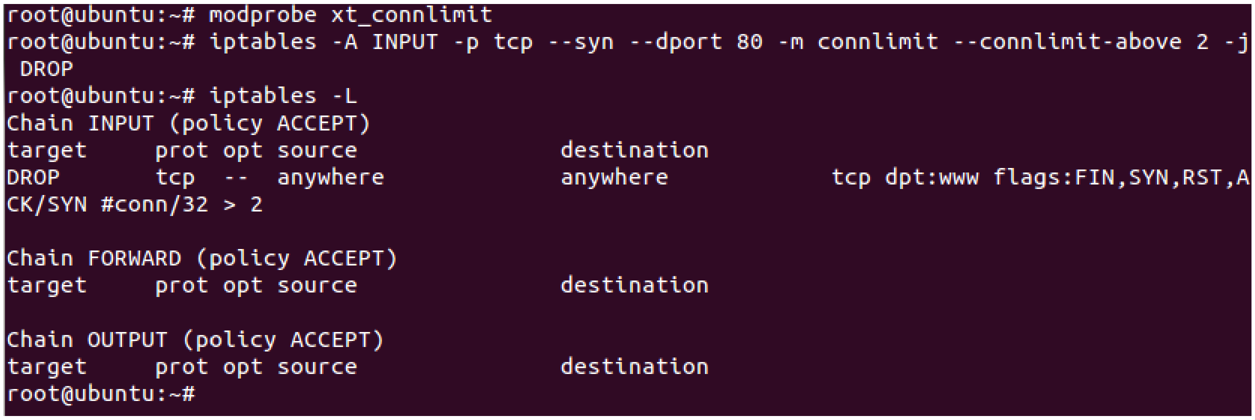
S10: We want to limit the number of active connections to the web server running on port 80 to 3.

We will use the xt_connlimit module to limit the number of connections. To do so, we first add the xt_connlimit module to the Linux kernel using the modprobe program (a built-in program in Linux). We can then run the iptables **command** as follows:

iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 2 -j DROP

where **--syn** indicates that we block the SYN request packets for a web connection
-m connlimit indicates we are using -m option for limiting the number of connections
--connlimit-above is an option to indicate when to take action; in this case, if the number of active connections is more than 2.

It is very important to limit the number of active connections for the servers running on a host/network; this would help to avoid a Denial of Service attack.



```
root@ubuntu:~# modprobe xt_connlimit
root@ubuntu:~# iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 2 -j
 DROP
root@ubuntu:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere            tcp dpt:www flags:FIN,SYN,RST,A
CK/SYN #conn/32 > 2

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:~#
```

S11: Setup iptables in such a way that we block ping testing from outside (i.e., remote machines cannot run a ping test on our machine) and at the same time we are able to ping remote machines.

We will setup our Ubuntu VM (192.168.159.131) to block ping test from other VMs (including our CentOS VM). To block a remote machine from doing a ping test on our machine, we should block the ICMP Echo-Request messages. Accordingly, we run the iptables command as follows:

```
root@ubuntu:/home/natarajan# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
root@ubuntu:/home/natarajan# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere            icmp echo-request

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ubuntu:/home/natarajan#
```

By explicitly specifying for the **echo-request** message to be dropped, we are permitting the **echo-reply** message by default (assuming the INPUT chain is configured by default to accept messages).

You can test the above setting by trying **ping 192.168.159.131** from your CentOS VM to ping to the Ubuntu VM (of IP address 192.168.159.131). You will not be able to see any feedback. On the other hand, you could ping the CentOS VM (find its IP address using the **ifconfig** command) from the Ubuntu VM.

IP Tables Project Exercises (all questions are with respect to the filter table, and are to be executed independent of the other questions, unless otherwise noted. So, remember to flush the iptables after executing each question, unless you are asked to follow up from a previous question):

Run the iptables -L command after setting up the configuration rules for each question.
Include screenshots of configuring the iptables firewall for every question.

Q0) Before you proceed with the questions on iptables, you need to install SSH on the Ubuntu VM in which you will be configuring the iptables. If you have already installed SSH on the VM, you could skip this question. Otherwise, complete the steps indicated in this question. You need to have the SSH server running on your Ubuntu VM to execute the steps in some of the questions in this project.

i) You need to setup root access in the Ubuntu VM. To setup root access, run the command su passwd root on the terminal. Enter the password you setup to login to the Ubuntu VM as a regular user (in my case, the username of the regular user is natarajan). Then, setup a password for the root level access and confirm it.

ii) Login as root using the command su root.

iii) Install the OpenSSH server application on the Ubuntu VM. After the installation is complete, run the netstat –ntlp command to show that the SSH daemon (sshd) is one of the programs actively running on a tcp port listening for incoming connection requests. Identify the port number on which the sshd daemon is running. Include appropriate screenshot(s).

Q1) Set the default policy for the INPUT chain to DROP. The firewall should only allow incoming packets from the network prefix 143.132.0.0/16. The default policy for the OUTPUT chain is ACCEPT. So, the user working on the machine could visit any website like www.google.com. Given the above policy for incoming packets, can the web pages visited by the user be displayed in the browser? Explain.

Q2) Set the default policy for the INPUT chain be DROP and the default policy for the OUTPUT chain be ACCEPT. Configure the INPUT chain to accept all incoming web traffic to port 80 and drop any other incoming traffic. Can you visit the website: www.hotmail.com? Why or why not? If you cannot visit the website, what aspect of this website is preventing you from visiting it, given that your default OUTPUT policy is ACCEPT and the firewall has been configured to accept traffic coming to port 80? Also, if you cannot visit the website, configure the firewall to let you be able to visit websites of such type. What changes/deletions/additions to the rules had to be done to facilitate this?

Q3) The previous question permitted only incoming packets related to web traffic. Do an insertion to the rules in the INPUT chain to permit SSH traffic. Show that you can connect to the SSH server running on the Ubuntu VM by connecting to it from another VM (centos or anything) or from the physical host machine (Windows). Include appropriate screenshots. You can get the IP address of a Linux machine by running the ifconfig command in the terminal. Refer to the screenshots (for example, under scenarios S5, S8) in the tutorial to see how you could SSH to a machine under a particular username.

Q4) Configure your IPtables filter table on your Ubuntu VM such that sessions/packet exchange originating from the Ubuntu VM (as the source) are successful; on the other hand, sessions/packet exchange originating from a remote machine to the Ubuntu VM (as the destination) are not successful. You need to implement this scenario with the minimal number of rules and policy changes, if any. Also, explain why your set of rules and policies implementing the stated scenario will work.

Q5) Configure your IPtables filter table to limit the number of active SSH connections to the Ubuntu VM (hosting the SSH server) is 2. Test the working of this rule by attempting to open three SSH connections, each in separate terminals, from another VM (like a CentOS VM) or from the host machine itself. Show appropriate screenshots.

Q6) Set the default policy of the INPUT and OUTPUT chains of your filter table of iptables is to DROP using an appropriate command (show a screenshot executing the command and the output of the iptables - L command). You could use the Ubuntu VM and CentOS VM in your virtual environment to implement this scenario. Now configure your iptables on the Ubuntu VM to (do parts a and b independently):
(a) Only allow remote machines to ping the local machine and block the local machine from pinging others.
(b) Only allow the local machine to ping the remote machines and block the remote machines from
pinging the local
machine.
(c) Allow ping communication in both directions (from the local machine to remote machine and vice-versa).

Note that you have to use the --icmp-type echo-request and --icmp-type echo-reply options appropriately.

Show appropriate screenshots executing the iptables commands to realize the above for (a), (b) and (c) and the structure of the iptables. Also, capture the successful or unsuccessful execution of the ping command from the local machine and remote machine (in either direction) for each of the three cases (a), (b), (c).