Practical – 5

Understand the concept of firewall and configure the Statefull Packet Inspection(SPI) firewall IPTABLES.

1) Firewall

A Firewall is a network security device that monitors and filters incoming and outgoing network traffic based on an organization's previously established security policies. At its most basic, a firewall is essentially the barrier that sits between a private internal network and the public Internet. A firewall's main purpose is to allow non-threatening traffic in and to keep dangerous traffic out.

Types of Firewalls:-

A firewall can either be software or hardware. Software firewalls are programs installed on each computer, and they regulate network traffic through applications and port numbers. Meanwhile, hardware firewalls are the equipment established between the gateway and your network. Additionally, you call a firewall delivered by a cloud solution as a cloud firewall.

There are multiple types of firewalls based on their traffic filtering methods, structure, and functionality. A few of the types of firewalls are:

- **Packet Filtering**

A packet filtering firewall controls data flow to and from a network. It allows or blocks the data transfer based on the packet's source address, the destination address of the packet, the application protocols to transfer the data, and so on.

- **Proxy Service Firewall**

This type of firewall protects the network by filtering messages at the application layer. For a specific application, a proxy firewall serves as the gateway from one network to another.

- **Stateful Inspection**

Such a firewall permits or blocks network traffic based on state, port, and protocol. Here, it decides filtering based on administrator-defined rules and context.

- **Next-Generation Firewall**

According to Gartner, Inc.'s definition, the next-generation firewall is a deep-packet inspection firewall that adds application-level inspection, intrusion prevention, and information from outside the firewall to go beyond port/protocol inspection and blocking.

- **Unified Threat Management (UTM) Firewall**

A UTM device generally integrates the capabilities of a stateful inspection firewall, intrusion prevention, and antivirus in a loosely linked manner. It may include additional services and, in many cases, cloud management. UTMs are designed to be simple and easy to use.

- **Threat-Focused NGFW**

These firewalls provide advanced threat detection and mitigation. With network and endpoint event correlation, they may detect evasive or suspicious behavior.

Key Uses of Firewalls

- Firewalls can be used in corporate as well as consumer settings.
- Firewalls can incorporate a security information and event management strategy (SIEM) into cybersecurity devices concerning modern organizations and are installed at the network perimeter of organizations to guard against external threats as well as insider threats.
- Firewalls can perform logging and audit functions by identifying patterns and improving rules by updating them to defend the immediate threats.
- Firewalls can be used for a home network, Digital Subscriber Line (DSL), or cable modem having static IP addresses. Firewalls can easily filter traffic and can signal the user about intrusions.
- They are also used for antivirus applications.
- When vendors discover new threats or patches, the firewalls update the rule sets to resolve the vendor issues.
- In-home devices, we can set the restrictions using Hardware/firmware firewalls.

Functions of Firewall

- The most important function of a firewall is that it creates a border between an external network and the guarded network where the firewall inspects all packets (pieces of data for internet transfer) entering and leaving the guarded network. Once the inspection is completed, a firewall can differentiate between benign and malicious packets with the help of a set of pre-configured rules.
- The firewall abides such packets, whether they come in a rule set or not, so that they should not enter into the guarded network.
- This packet form information includes the information source, its destination, and the content. These might differ at every level of the network, and so do the rule sets. Firewalls read these packets and reform them concerning rules to tell the protocol where to send them.

Advantages of Using Firewalls

Now that you have understood the types of firewalls, let us look at the advantages of using firewalls.

- Firewalls play an important role in the companies for security management. Below are some of the important advantages of using firewalls.
- It provides enhanced security and privacy from vulnerable services. It prevents unauthorized users from accessing a private network that is connected to the internet.
- Firewalls provide faster response time and can handle more traffic loads.
- A firewall allows you to easily handle and update the security protocols from a single authorized device.
- It safeguards your network from phishing attacks.

How to Use Firewall Protection?

To keep your network and devices safe, make sure your firewall is set up and maintained correctly. Here are some tips to help you improve your firewall security:

- Constantly update your firewalls as soon as possible: Firmware patches keep your firewall updated against any newly discovered vulnerabilities.
- Use antivirus protection: In addition to firewalls, you need to use antivirus software to protect your system from viruses and other infections.
- Limit accessible ports and host: Limit inbound and outbound connections to a strict whitelist of trusted IP addresses.
- Have active network: To avoid downtime, have active network redundancies. Data backups for network hosts and other critical systems can help you avoid data loss and lost productivity in the case of a disaster.

Application Layer and Proxy Firewalls

Proxy firewalls can protect the application layer by filtering and examining the payload of a packet to distinguish valid requests from malicious code disguised as valid requests for data. Proxy firewalls prevent attacks against web servers from becoming more common at the application layer. Besides, proxy firewalls give security engineers more control over network traffic with a granular approach.

On the other hand, application layer filtering by proxy firewalls enables us to block malware, and recognize the misused amongst various protocols such as Hypertext Transfer Protocol(HTTP), File Transfer Protocol (FTP), certain applications, and domain name system(DNS).

2) IPTables

In linux operating system, the firewalling is taken care of using netfilter. Which is a kernel module that decides what packets are allowed to come in or to go outside.

iptables are just the interface to netfilter. The two might often be thought of as the same thing. A better perspective would be to think of it as a back end and a front end.

Using iptables You get more flexibility regarding the things you want to with a packet.

iptables architecture

iptables consists of different components which are discussed below:

- **chains**: There are 5 chains in iptables and each is responsible for a specific task. These chains are: prerouting, input, forward, output & postrouting. As their name suggests, they're responsible for packets either as soon as they arrive, if they are destined for local socket or just before routing to the outer world. We'll discuss these below.
- **tables**: Again, different tables are responsible for different tasks. The list contains filter, nat, mangle, raw & security. The first two are the most used. Filter is responsible for filtering and restricting the packets to/from our computer. Nat is responsible for Network Address Translation. We'll discuss these terms below as well.
- **targets**: Targets specify where a packet should go. This is decided using either iptables' own targets: ACCEPT, DROP, or RETURN, or it's extensions' target which are 39 at the moment and the most popular ones are DNAT, LOG, MASQUERADE, REJECT, SNAT, TRACE and TTL. Targets are divided into terminating and non-terminating. Which is just what the name suggests. Terminating targets ends rule traversal and the packets will be stopped there, but non-terminating ones touch a packet in some way and the rule traversal will continue afterward.

iptables chains

As mentioned above, each chain is responsible for a specific task.

- **Prerouting**: this chain decides what happens to a packet as soon as it arrives at the network interface. We have different options such as altering the packet (for NAT probably), dropping a packet, or doing nothing at all and letting it slip and be handled elsewhere along the way.
- **Input**: This is one of the popular chains as it almost always contains strict rules to avoid some evil doers on the internet harming our computer. If you want to open/block a port, this is where you'd do it.
- **Forward**: This chain is responsible for packet forwarding. Which is what the name suggests. We may want to treat a computer as a router and this is where some rules might apply to do the job.
- **Output**: This chain is the one responsible for all your web browsing among many others. You can't send a single packet without this chain allowing it. You have a lot of options whether you want to allow a port to communicate or not. It's the best place to limit your outbound traffic if you're not sure what port each application is communicating through. (A small hint: use the command ss -tulpen).

- **Postrouting**: This chain is where packets leave their trace last, before leaving our computer. This is used for routing among many other tasks just to make sure the packets are treated the way we want them to.

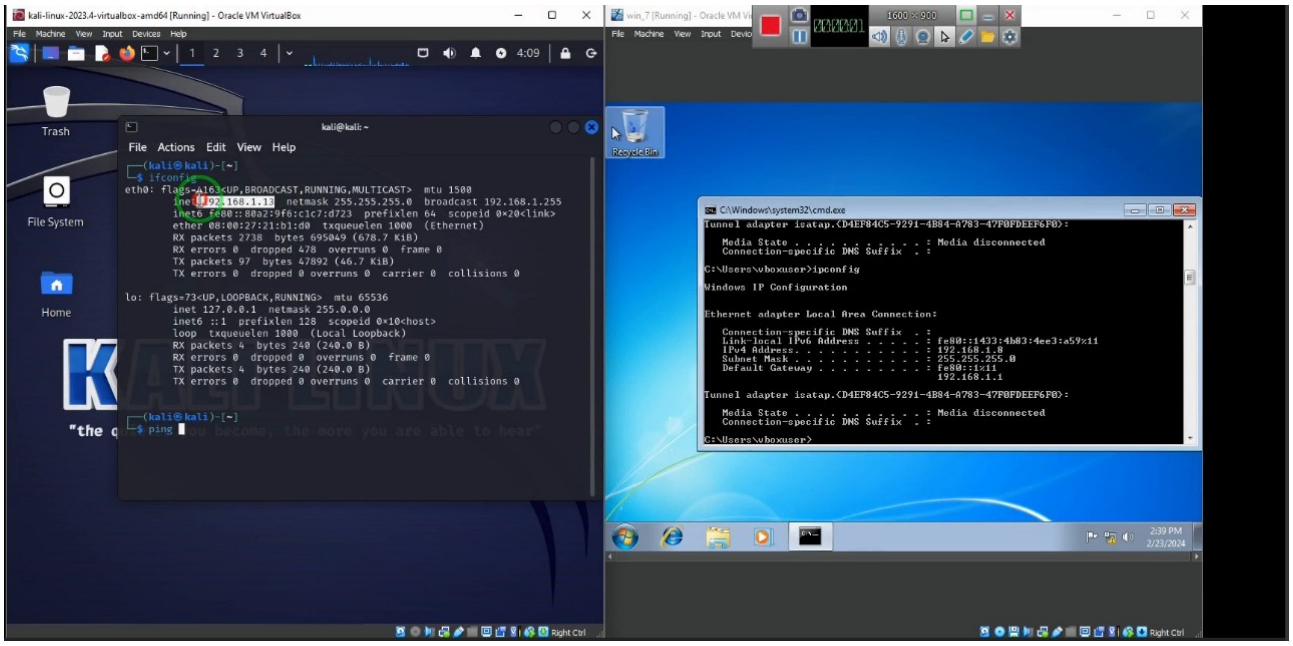
Most of your use cases on a regular basis would be to Input & Output chain. With small modifications to other chains — which are not a day-to-day need most of the time.

iptables tables

Although the title has rhyme to it, you won't need a lot of this item on a day-to-day basis. Your main use case, most of the time is on filter table. Which is the default table and you wouldn't need any switch to activate it. That being said, if you don't specify any table, filter is used by default. So bear that in mind and continue below where I explain each table.

- **Filter**: This is the table most used on a daily basis. Which is why it's the default table. In this table you would decide whether a packet is allowed in/out your computer. If you want to block a port to stop receiving anything, this is your stop.
- **Nat**: This table is the second most popular table and is responsible for creating new connection. Which is shorthand for Network Address Translation. And if you're not familiar with the term don't worry. I'll give you an example below.
- **Mangle**: For specialized packets only. This table is for changing something inside the packet either before coming in or leaving out.
- **Raw**: This table is dealing with the raw packet as the name suggests. Mainly this is for tracking the connection state. We'll see examples of this below when we want to allow success packets from SSH connection.
- **Security**: It is responsible for securing your computer after the filter table. Which consists of SELinux. If you're not familiar with the term, it's a very strong security tool on modern linux distributions.

Step 1:- check ip of connected machines machines

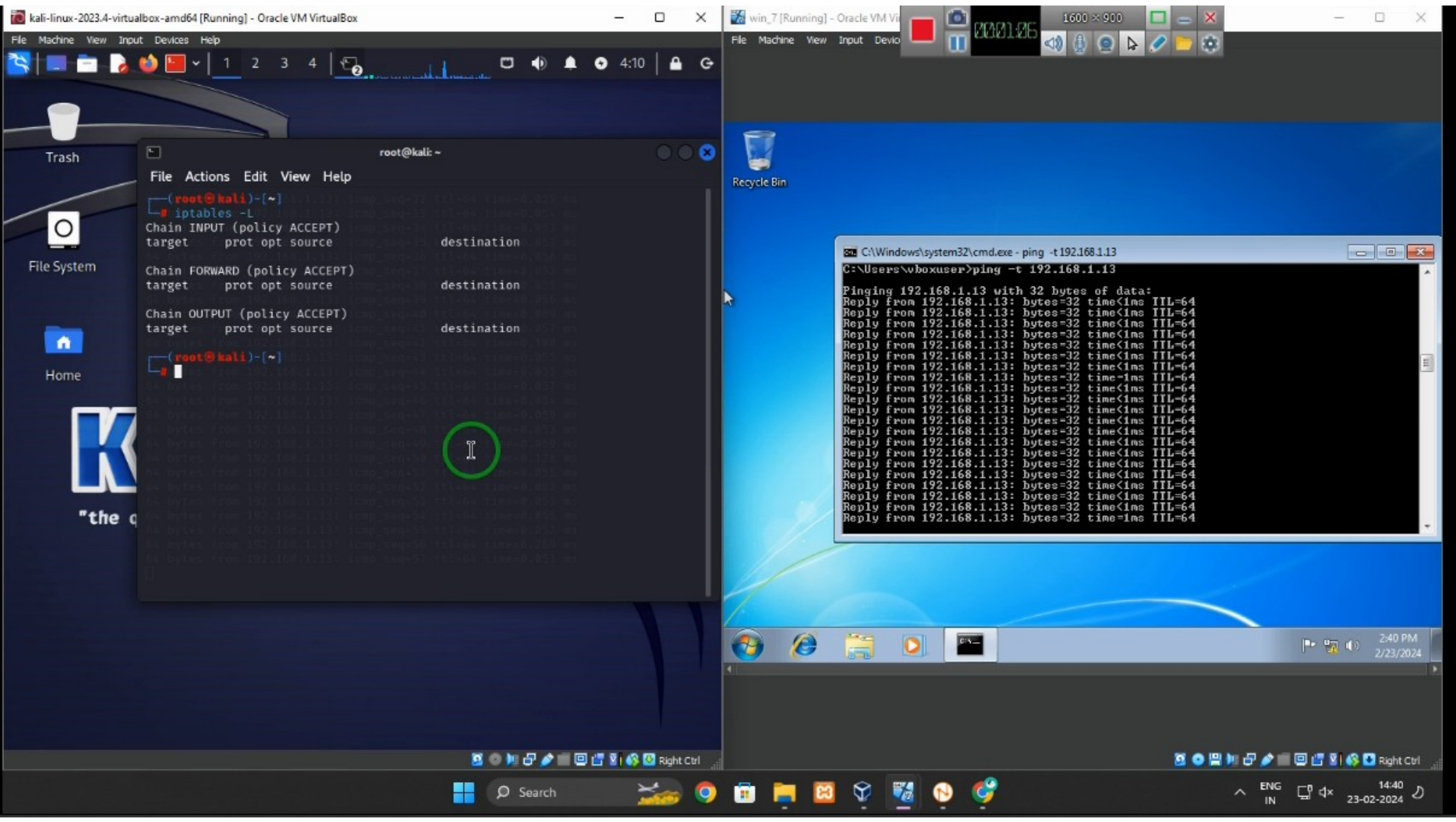


```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.13  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::80a2:9f6:c1c7:d723  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:21:b1:d0  txqueuelen 1000  (Ethernet)
        RX packets 2738  bytes 695049 (678.7 KiB)
        RX errors 0  dropped 478  overruns 0  frame 0
        TX packets 97  bytes 47892 (46.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

┌──(kali㉿kali)-[~]
└─$ ping
```

```
Tunnel adapter isatap.{D4EF84C5-9291-4B84-A783-47F0FDEEF6F0}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\vboxuser>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1433:4b83:4ee3:a59%11
   IPv4 Address. . . . . . . . . . . : 192.168.1.8
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::1%11
                                       192.168.1.1

Tunnel adapter isatap.{D4EF84C5-9291-4B84-A783-47F0FDEEF6F0}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\vboxuser>
```

Step 2 :- ping kali from windows using ping -t 192.168.1.13 & check configuration of iptables in kali using command (ip in ping cmd is of kali machine)

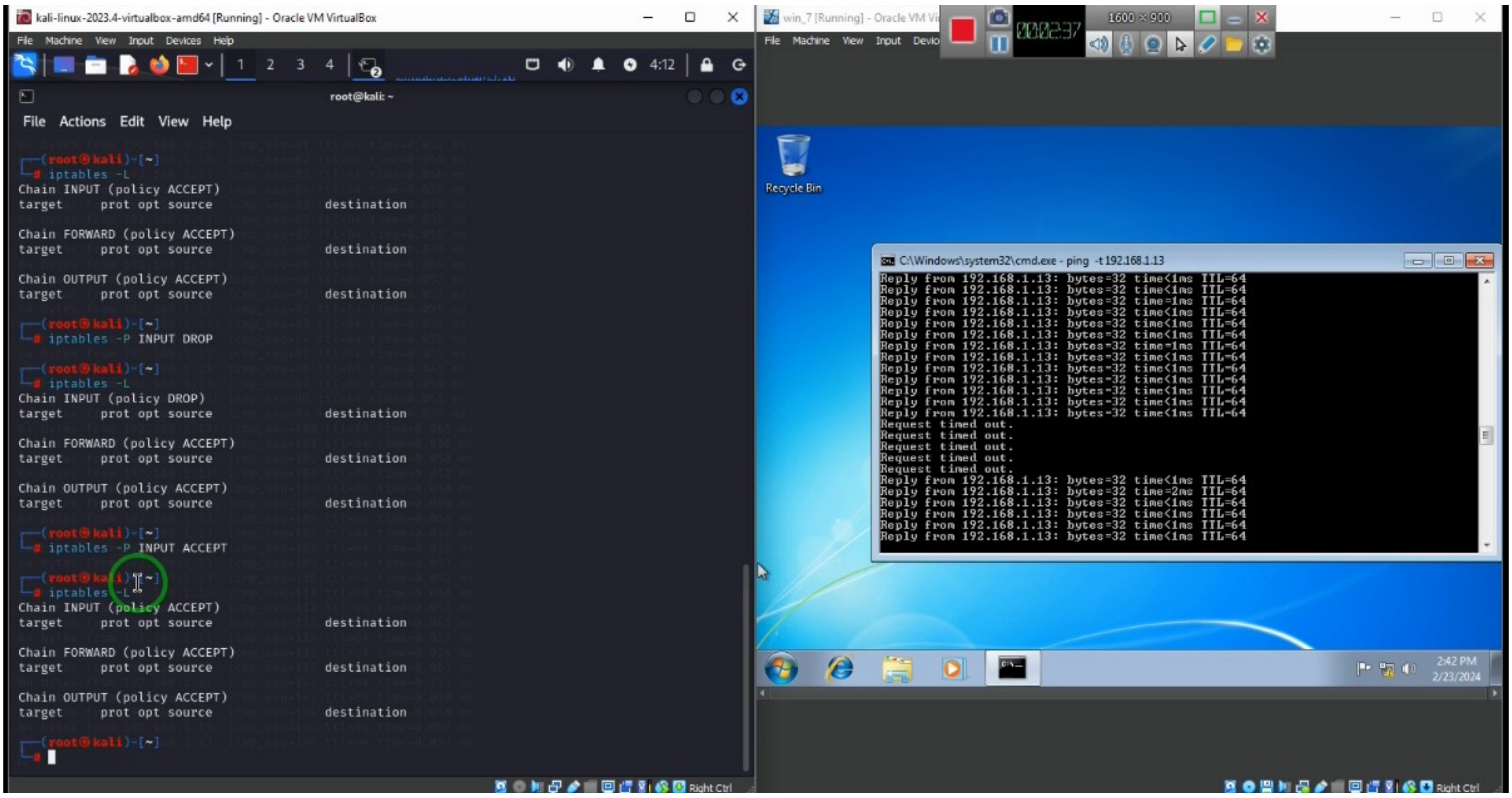
iptables -L



```
┌──(root㉿kali)-[~]
└─# 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㉿kali)-[~]
└─#
```

```
C:\Users\vboxuser>ping -t 192.168.1.13

Pinging 192.168.1.13 with 32 bytes of data:
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
```

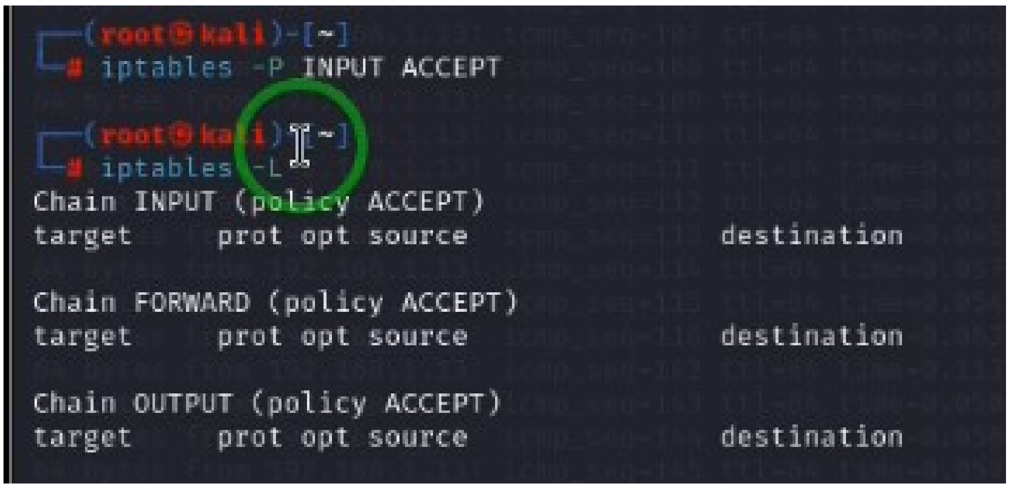
Step 3:- drop incoming packets using command : iptables -P INPUT DROP this will drop ll incoming packets or ping requests from windows


kali-linux-2023.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox
root@kali: ~
File Actions Edit View Help
win_7 [Running] - Oracle VM VirtualBox
Recycle Bin
2:42 PM
2/23/2024

```
┌──(root㉿kali)-[~]
└─# 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㉿kali)-[~]
└─# iptables -P INPUT DROP

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

┌──(root㉿kali)-[~]
└─# iptables -P INPUT ACCEPT

┌──(root㉿kali)-[~]
└─# 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㉿kali)-[~]
└─#
```

```
C:\Windows\system32\cmd.exe - ping -t 192.168.1.13
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=2ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
```

Revert it back to accept rule by replacing DROP in above command to ACCEPT i.e iptables -P INPUT ACCEPT similarl we can change rules of FORWARD and OUTPUT field.

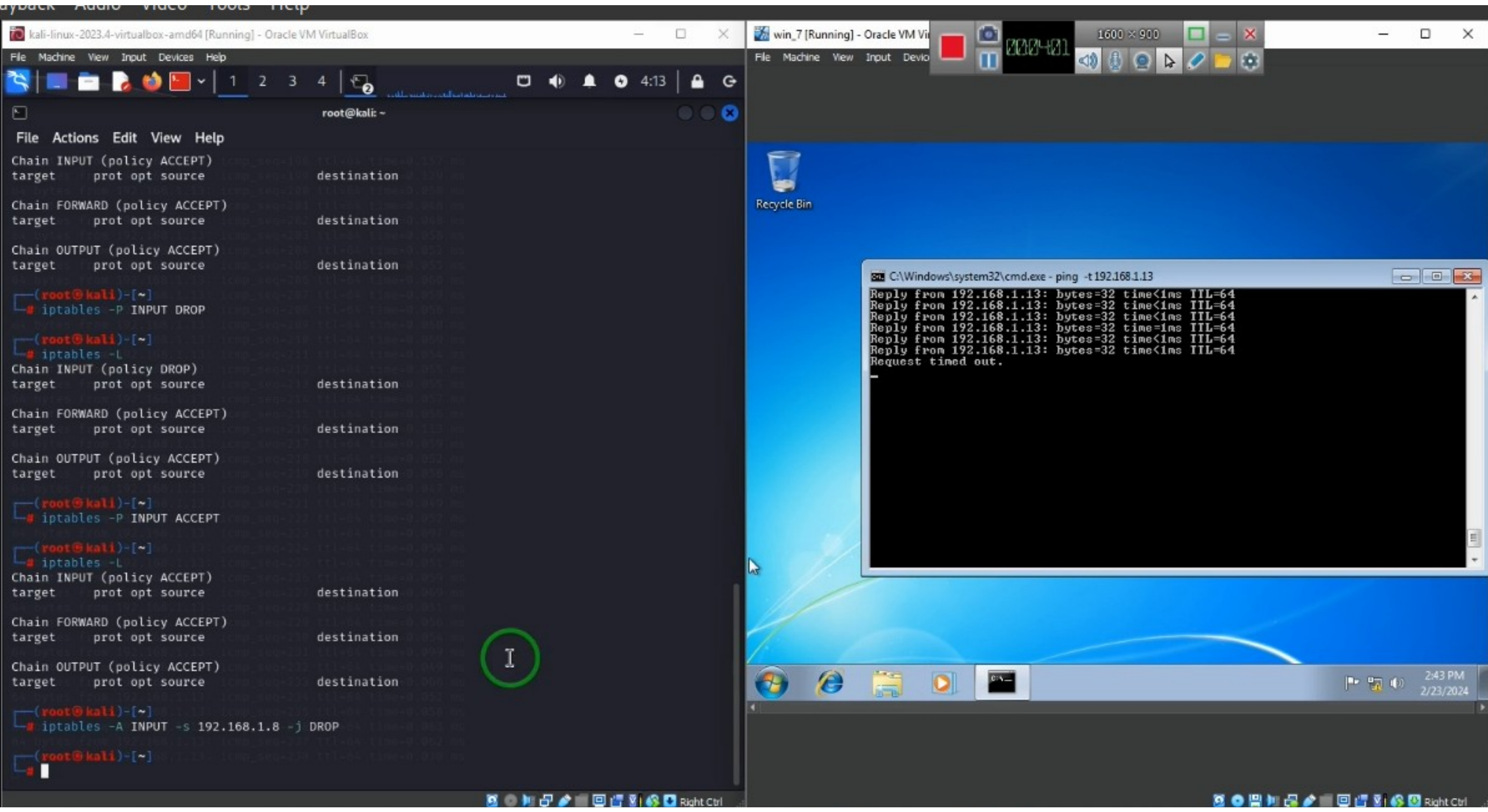


```
┌──(root㉿kali)-[~]
└─# iptables -P INPUT ACCEPT

┌──(root㉿kali)-[~]
└─# 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
```

Step 4 :- now let us drop packets of only windows machine using cmd :

iptables –A INPUT –i 192.168.1.8 –j ACCEPT

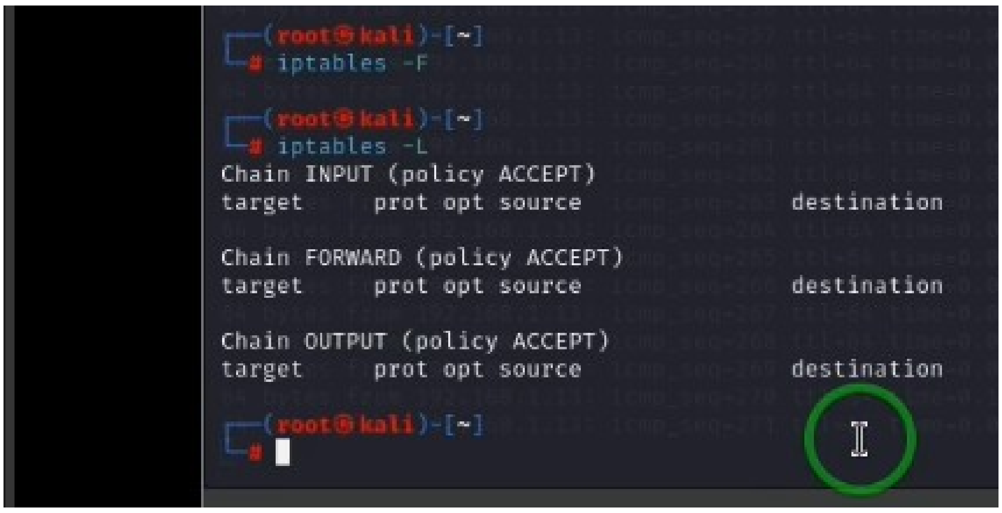
(ip in cmd is of windows machine you can also select entire network by replacing 192.168.1.8 with 192.168.1.8/24)


kali-linux-2023.4-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@kali: ~
File Actions Edit View Help
win_7 [Running] - Oracle VM Vi
Recycle Bin
Right Ctrl

```
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㉿kali)-[~]
└─# iptables -P INPUT DROP

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

┌──(root㉿kali)-[~]
└─# iptables -P INPUT ACCEPT

┌──(root㉿kali)-[~]
└─# 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㉿kali)-[~]
└─# iptables -A INPUT -s 192.168.1.8 -j DROP

┌──(root㉿kali)-[~]
└─#
```

```
C:\Windows\system32\cmd.exe - ping -t 192.168.1.13
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Request timed out.
```

Step 5:- Flushing all rules in iptables it can be done using cmd : iptables -F and conver INPUT policy to drop like in step 3

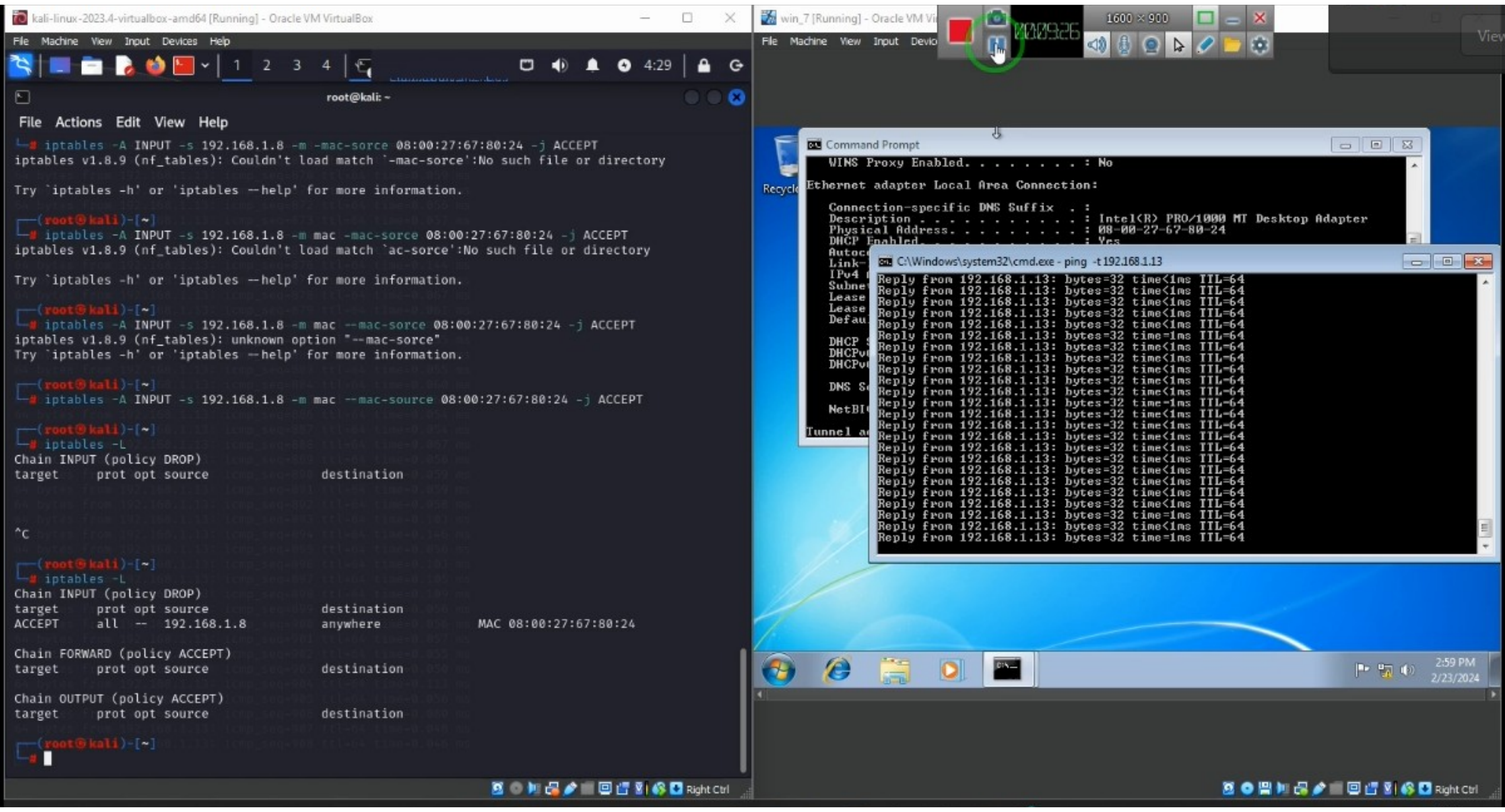


```
┌──(root㉿kali)-[~]
└─# iptables -F

┌──(root㉿kali)-[~]
└─# 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㉿kali)-[~]
└─#
```

Step 6 : Accepting packets from only windows machine with mac address

iptables –A INPUT –s 192.168.1.8 –m mac –mac-source 80:00:27:67:80:24 –j ACCEPT

(80:00:27:67:80:24 is mac address of windows machine) it can also be used with drop rule



```
└─# iptables -A INPUT -s 192.168.1.8 -m -mac-sorce 08:00:27:67:80:24 -j ACCEPT
iptables v1.8.9 (nf_tables): Couldn't load match `-mac-sorce':No such file or directory

Try `iptables -h' or 'iptables --help' for more information.

┌──(root㉿kali)-[~]
└─# iptables -A INPUT -s 192.168.1.8 -m mac -mac-sorce 08:00:27:67:80:24 -j ACCEPT
iptables v1.8.9 (nf_tables): Couldn't load match `ac-sorce':No such file or directory

Try `iptables -h' or 'iptables --help' for more information.

┌──(root㉿kali)-[~]
└─# iptables -A INPUT -s 192.168.1.8 -m mac --mac-sorce 08:00:27:67:80:24 -j ACCEPT
iptables v1.8.9 (nf_tables): unknown option "--mac-sorce"
Try `iptables -h' or 'iptables --help' for more information.

┌──(root㉿kali)-[~]
└─# iptables -A INPUT -s 192.168.1.8 -m mac --mac-source 08:00:27:67:80:24 -j ACCEPT

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination



^C

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  192.168.1.8          anywhere             MAC 08:00:27:67:80:24

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

┌──(root㉿kali)-[~]
└─#
```

```
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
   Physical Address. . . . . . . . . : 08-00-27-67-80-24
   DHCP Enabled. . . . . . . . . . . : Yes
```

C:\Windows\system32\cmd.exe - ping -t 192.168.1.13

```
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
```

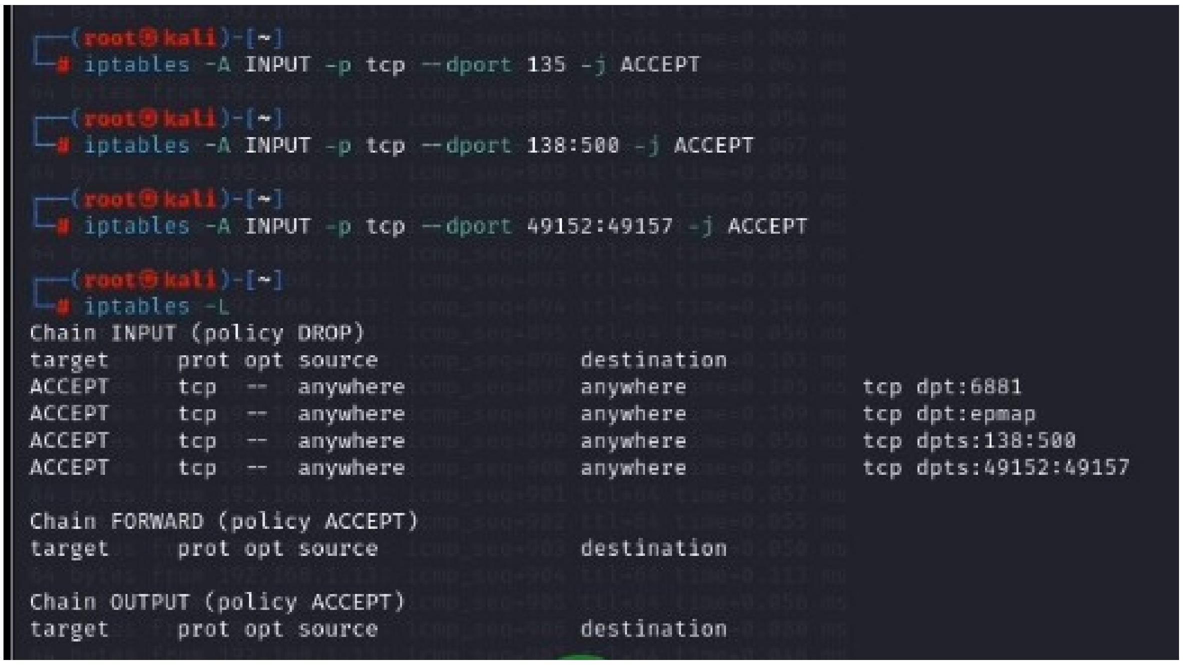
Step 7 :- Port filtering so our machine only accept packets on this ports only can be seen using nmap

Single port

iptables –A INPUT –p tcp - -dport 6881 –j ACCEPT

Port Range

iptables –A INPUT –p tcp - -dport 6881:6890 –j ACCEPT

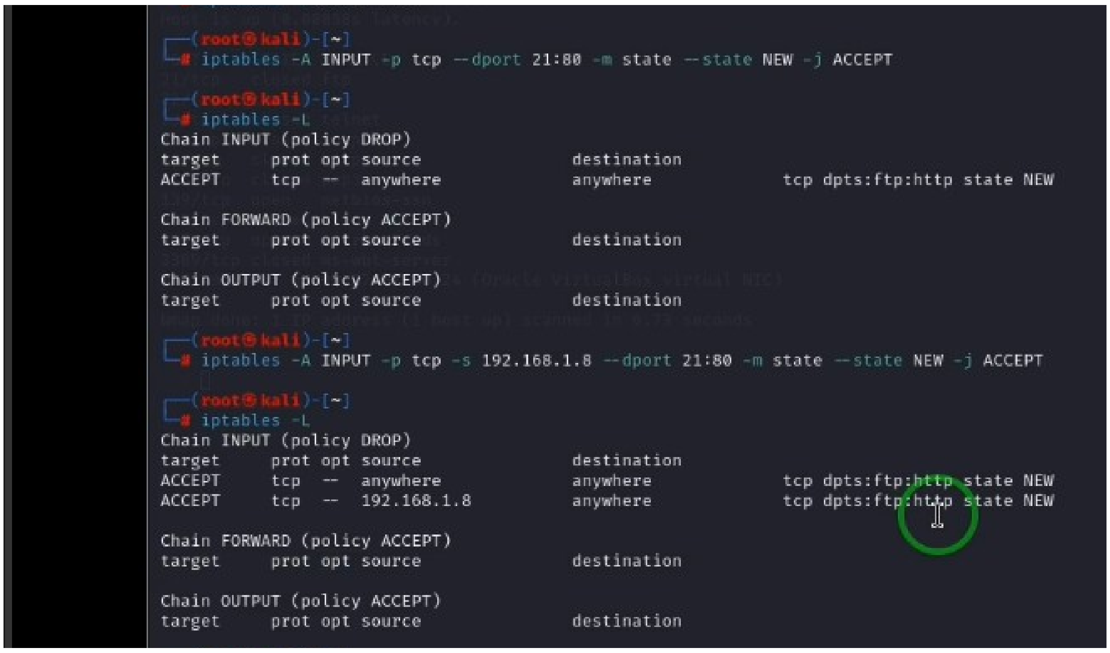


```
┌──(root㉿kali)-[~]
└─# iptables -A INPUT -p tcp --dport 135 -j ACCEPT

┌──(root㉿kali)-[~]
└─# iptables -A INPUT -p tcp --dport 138:500 -j ACCEPT

┌──(root㉿kali)-[~]
└─# iptables -A INPUT -p tcp --dport 49152:49157 -j ACCEPT

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:6881
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:epmap
ACCEPT     tcp  --  anywhere             anywhere             tcp dpts:138:500
ACCEPT     tcp  --  anywhere             anywhere             tcp dpts:49152:49157

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

step 8 :- Filtering packets based on state

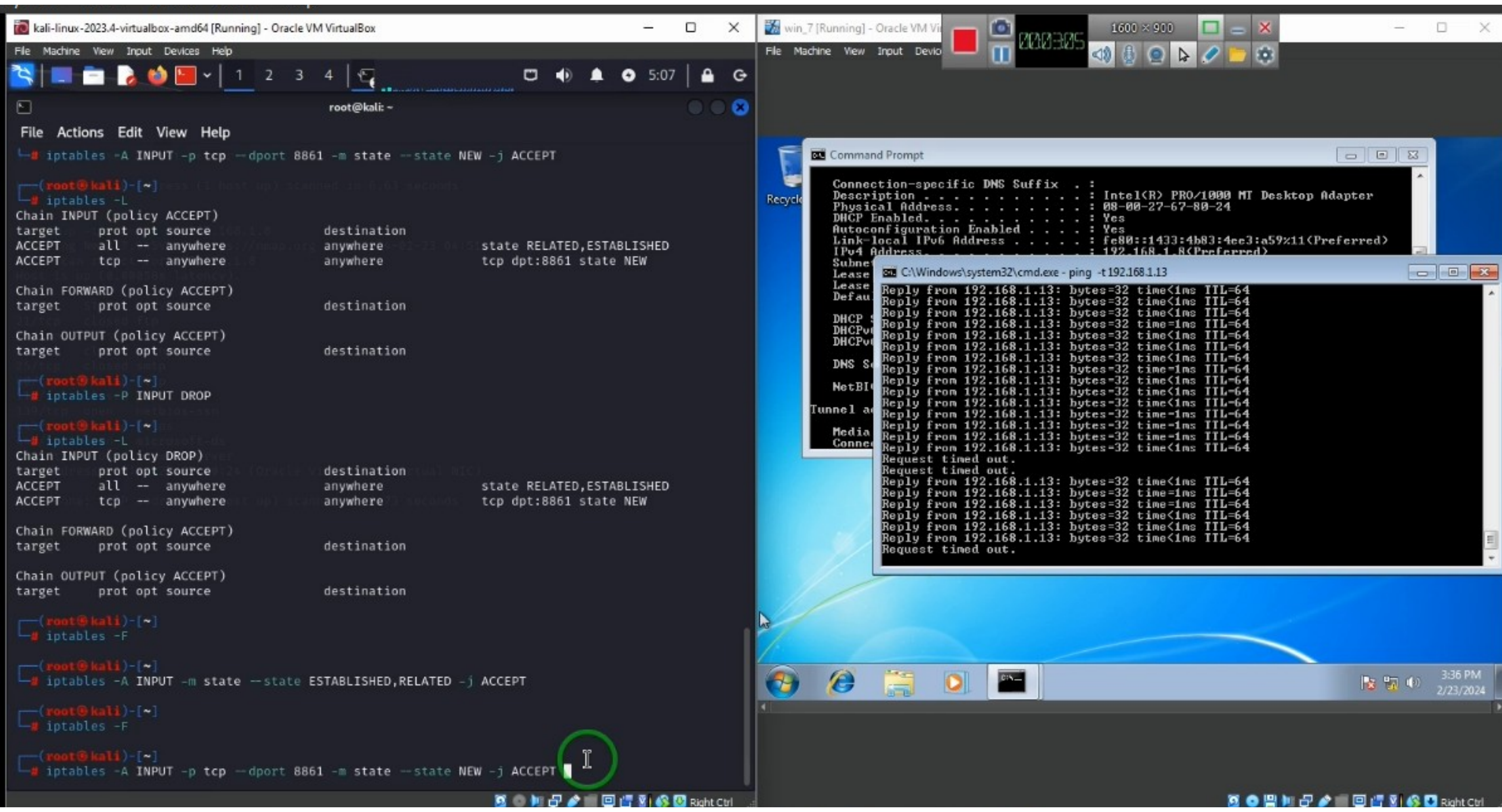
iptables –A INPUT –p tcp –dport 21:80–m state - -state NEW –j ACCEPT

(will accept only new connection on TCP single port or any port range can also be specified)



```
┌──(root㉿kali)-[~]
└─# iptables -A INPUT -p tcp --dport 21:80 -m state --state NEW -j ACCEPT

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpts:ftp:http state NEW

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

┌──(root㉿kali)-[~]
└─# iptables -A INPUT -p tcp -s 192.168.1.8 --dport 21:80 -m state --state NEW -j ACCEPT

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpts:ftp:http state NEW
ACCEPT     tcp  --  192.168.1.8          anywhere             tcp dpts:ftp:http state NEW

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

iptables -A INPUT –m state –state ESTABLISED, RELATED –j ACCEPT



```
└─# iptables -A INPUT -p tcp --dport 8861 -m state --state NEW -j ACCEPT

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:8861 state NEW

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

┌──(root㉿kali)-[~]
└─# iptables -P INPUT DROP

┌──(root㉿kali)-[~]
└─# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:8861 state NEW

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

┌──(root㉿kali)-[~]
└─# iptables -F

┌──(root㉿kali)-[~]
└─# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

┌──(root㉿kali)-[~]
└─# iptables -F

┌──(root㉿kali)-[~]
└─# iptables -A INPUT -p tcp --dport 8861 -m state --state NEW -j ACCEPT
```

```
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
   Physical Address. . . . . . . . . : 08-00-27-67-80-24
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::1433:4b83:4ee3:a59%11(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.1.8(Preferred)
```

```
C:\Windows\system32\cmd.exe - ping -t 192.168.1.13
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Request timed out.
Request timed out.
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time=1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Reply from 192.168.1.13: bytes=32 time<1ms TTL=64
Request timed out.
```

Information that can go further in IPTABLES

-A appends the iptables chain (INPUT,OUTPUT,FORWARED)

-F Flush pervious rule

-h provide list of command structure and summary information

-L lists all the rules or if chain is specified the rules in a chain

-N creates a new user defined chain

-P sets the default policy for particular chain

-p sets the protocol

-x Detects a user defined chain

-Z Zeros the byte and packet counters in all chains for a particular table.