Course of Network Security

UNIVERSITÀ DEGLI STUDI DI TRENTO

Stateless Firewall Lab Report

Group 16

18th May 2016, Trento

AUTHORS:

Gamaliel Bepa
Vincent Ogwara
Noellar Kappa
Teweldebrhan Haile

Submitted To:
Luca Allodi(PHD)

Table of Contents

Topic page

1.Introduction

1.1 Firewall basics :

A Firewall is a network component that filters incoming or outgoing traffic to and from a network. It is placed at the perimeter/border of your network. A firewall controls access to network resources by allowing only traffic defined in the firewall policy and denying all other traffic. A network firewall resides on a network node such that host or router. Its main role is to inspect all the forwarding traffic. based on its configuration, the firewall makes a decision regarding what action like accept or deny in order to perform on a given packet. The firewall configuration is composed by a set of ordered rules. Each rule consists of conditions and an action.

1.2 TCP/IP for Firewalls:

The OSI model is a reference framework for how messages should be transmitted between any two points in a network. It is composed of seven layers, Application, Presentation, Session, Transport , Network, Data-Link & Physical. The seven layers have different functions. The Physical layer is for transmission of bits of data. The Data-Link layer is for physical addressing, flow control and error checking. The Network layer allows end to end delivery and logical addressing on different network segments. The Transport layer allows reliable delivery through sequence numbers and error recovery. Session layer controls conversations or sessions by establishing and maintaining connections. Presentation controls data formats ensuring data is received in an acceptable format and the Application Layer provides services/protocols to applications. Firewalls generally operate on the Network, Transport and Application layer.

TCP traffic is handed over to the network layer as packets, which consist of an ip header, tcp header that contains control information such as source and destination addresses, packet sequence information and the data also known as a payload. While the control information in each packet helps to ensure that its associated data gets delivered properly, the elements it contains also provides firewalls a variety of ways to match packets against firewall rules.

It is important to note that successfully receiving incoming TCP traffic requires the receiver to send an acknowledgement back to the sender. The combination of the control information in the incoming and outgoing packets can be used to determine the connection state of between the sender and receiver.

1.3 Introduction to Stateless Firewall:

There are three different types of firewalls; static packet filters/stateless firewalls, stateful packet filtering and proxies which are application-level gateways & circuit level gateways.
A firewall is stateless if the rules or conditions are based on header information in a packet such as source address,destination address, protocol,source port and destination port. In this case the firewall treats each packet in isolation. Such packet filters operate at the

network layer and function more efficiently because they only look at the header part of packet. A drawback of pure packet filters is that they are stateless; they have no memory of previous packets which makes them vulnerable to spoofing attacks. Such a firewall has no way of knowing if any given packet is part of an existing connection, is trying to establish a new connection, or is just a rogue packet.

2 .Attacks and Firewalls

Firewalls are configured to protect network resources against attacks. Some of the steps taken by hackers to infiltrate your network and launch attacks include operating system footprinting/scanning, enumeration by determining vulnerable exploits, launching the attack by taking advantage of vulnerable exploits and leaving systems open for future exploits.

2.1 Management and Configuration of Firewalls :

Network traffic that traverses a firewall is matched against rules to determine if it should be allowed through or not.

If we have a server with this list of rules that apply to incoming traffic.

1. Accept new and established incoming traffic to the public network interface on port 80 and 443(HTTP and HTTPS web traffic)
2. Drop incoming traffic from IP addresses of the unauthorized user in your office to port 22(SSH)
3. Accept new and established incoming traffic from your office IP range to the private network interface on port 22(SSH)

In each of these examples, the filter is either "accept","reject",or "drop". This specifies the action that the firewall should do in the event that a piece of network traffic matches a rule. *Accept* means to allow the traffic through,*reject* means to block the traffic but reply with an "unreachable" error, and *drop* means to block the traffic and send no reply. The rest of each rule consists of the condition that each packet is matched against for example the port number or source and destination ip addresses. Traffic is matched against a list of firewall rules in a sequence, or chain, from first to last once a rule is matched, the associated action is applied to the network traffic in question. In our example, if an accounting employee attempted to establish an ssh connection to the server they would be rejected based on rule 2, before rule 3 even checked. A system administrator, however, would be accepted because they would match only rule 3.

2.2 Default policy

Default policy caters for traffic that is not matched to any of the rules explicitly defined. Firewall chains will have a default policy specified which either allows traffic unmatched by previous rules or deny that traffic.

Suppose the default policy for the example chain above was set to drop. If any computer outside of your office attempted to establish an ssh connection to the server, the traffic would be dropped because it does not match the conditions of any rules. If the default policy were set to accept, anyone,except the source ip address you give permission, would be able to establish a connection to any open service on your server. This would be an example of a very poorly configured firewall because it only keeps a subset of your employees out.

2.3 Flow of traffic

Traffic can be either incoming or outgoing, a firewall maintains a distinct set of rules for either case. Traffic that originates elsewhere, incoming traffic, is treated differently than outgoing traffic that the server sends. It is typical for a server to allow most outgoing traffic because the server is usually, to itself,trustworthy. Still , the outgoing rule set can be used to prevent unwanted communication in the case that a server is compromised by an attacker or a malicious executable.

In order to maximize the security benefits of firewall, we should identify all of the ways we want other systems to interact with server, create rules that explicitly allow them, then drop all other traffic. Keep in mind that the appropriate outgoing rules must be in place so that a server will allow itself to send outgoing acknowledgements to any appropriate incoming connections. Also, as a server typically needs to initiate its own outgoing traffic for various reasons

3. Iptables:

The implementation for stateless firewall policies and rules is to be configured with the use of iptables. Iptables is a generic table structure for the definition of rulesets pre-installed with the linux operating system._Iptables is included in most Linux distributions by default

Before you start configuring your firewall, you need to make sure that iptables/netfilters is installed on your Linux environment. If it is not there, then you can download it using the following commands :

```
sudo apt-get update
sudo apt-get install iptables
```

iptables is used to *inspect, modify, forward, redirect,* and/or *drop* IPv4 packets. The code for filtering IPv4 packets is already built into the kernel and is organized into a collection of *tables*, each with a specific purpose. The tables are made up of a set of predefined *chains*, and the chains contain rules which are traversed in order. iptables is the user utility which allows you to work with these chains/rules.

3.1 Tables :

iptables contains **five** tables:

1. Raw is used only for configuring packets so that they are exempt from connection tracking.
2. Filter is the default table, and is where all the actions typically associated with a firewall take place.
3. Nat is used for network address (e.g. port forwarding).
4. Mangle is used for specialized packet alterations.
5. Security is used for mandatory access control networking rules

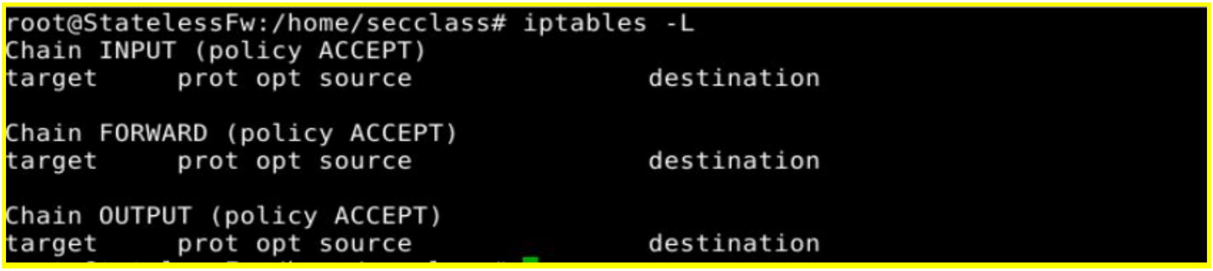
In this case we shall use two of these: **Filter** and **Nat**.

3.2 Chains :

Tables consist of *chains*, which are lists of rules which are followed in order. The default table, filter, contains three built-in chains: INPUT, OUTPUT and FORWARD. Input Chain is for managing packets input to the server. Here we add rules to control remote input connections. The Output Chain controls packets from the server to the outside. Here we add rules to manage local outbound connections. The Forward chain is used to add rules to manage connections from one network interface to another on the same device. The nat table includes PREROUTING, POSTROUTING, and OUTPUT chains. The Prerouting chain translates packets before routing. The Postrouting chain translates packets after routing completes.

If iptables is now installed and running on your machine, it will have a default policy, which is generally set to ACCEPT. To view the rules in your iptables, use the following syntax to list the rules in a chain;-

#iptables -L;



```
root@StatelessFw:/home/secclass# 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
```

It shows that currently the input, forward, and output chain's values are all set to ACCEPT. The general iptables syntax for writing all stateless firewall rules is as show bellow:
Syntax

#iptables < option >< chain >< matching_criteria>< target >

To clear all previous/old rules which are previously set, you need to use this syntax to flush out the old rules in the chain.

```
#iptables –F or #iptables –flush
```

We are now ready to create our custom rules for this lab.

4.Environment Setup:

In this illustration, see Figure 1, our network is made up of two hosts and one server which serves as firewall(where the rules are implemented). The firewall has two interfaces, one acting as the gateway for our two hosts on the common local area network (192.168.1.0/24) and the second interface acting as a gateway for a different external network with a web service locally hosted the same firewall on port 8080 in the 172.16.10/24 network.

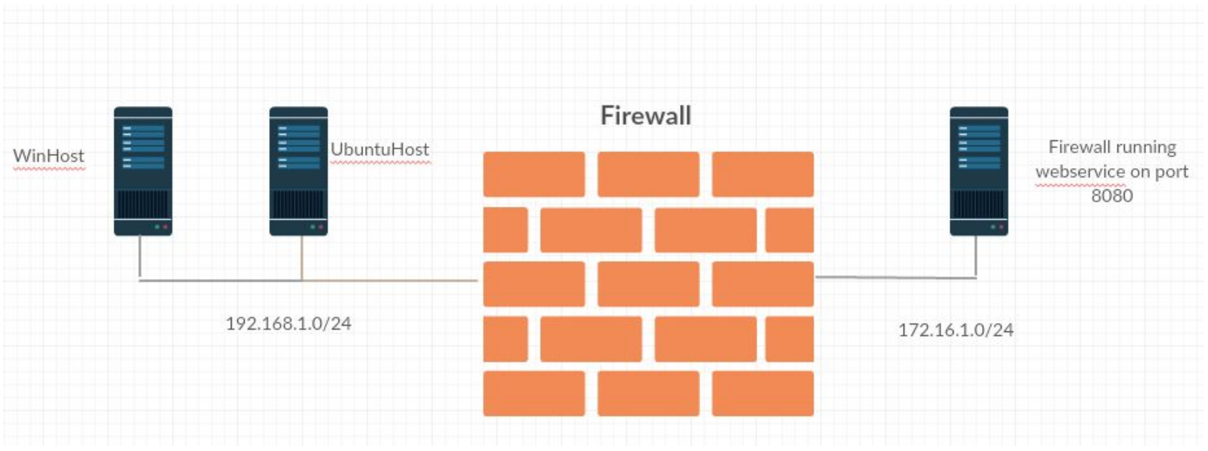


Figure 1 : Network made of 2 hosts and server(Firewall)

4.1 WinHost (windows server 2008)

→ A windows operating Vm used as a host to test the effectiveness of our rules.

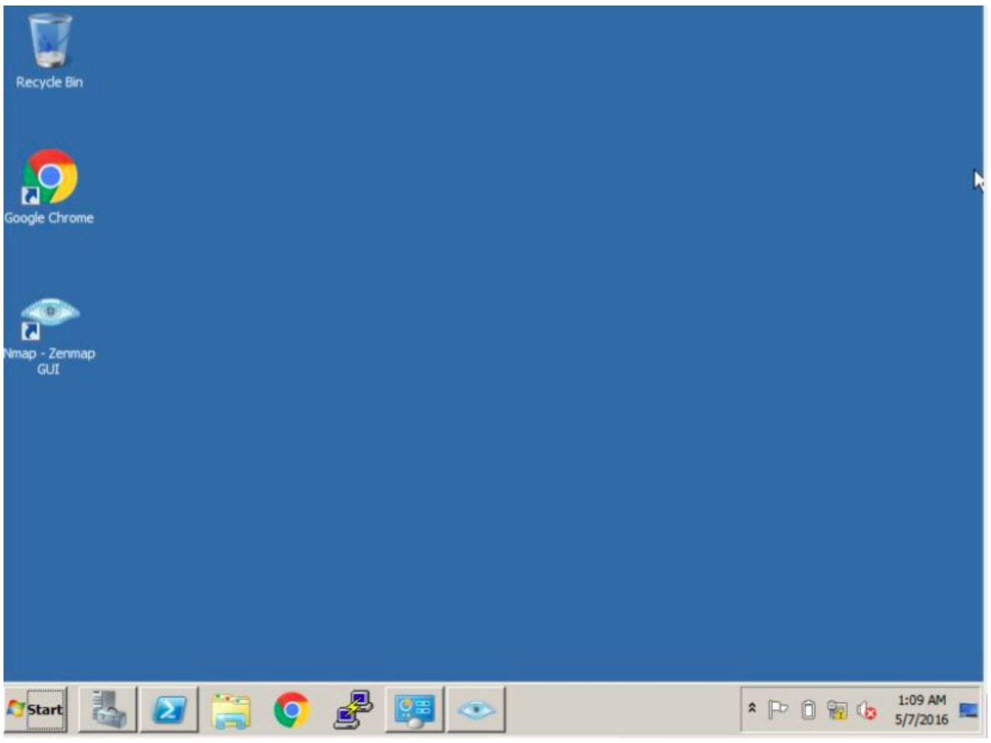


Fig : Winhost Vm

Login to WinHost :

➔ **Windows server 2008**

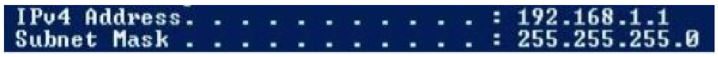
Password : password@1

Check Settings :

- *ipconfig*

➔ Should be *192.168.1.1*

- Use the password "**password@1**" to login on WinHost
- Check settings from *terminal* by typing "*ipconfig*" and ensure ip -> *192.168.1.1*



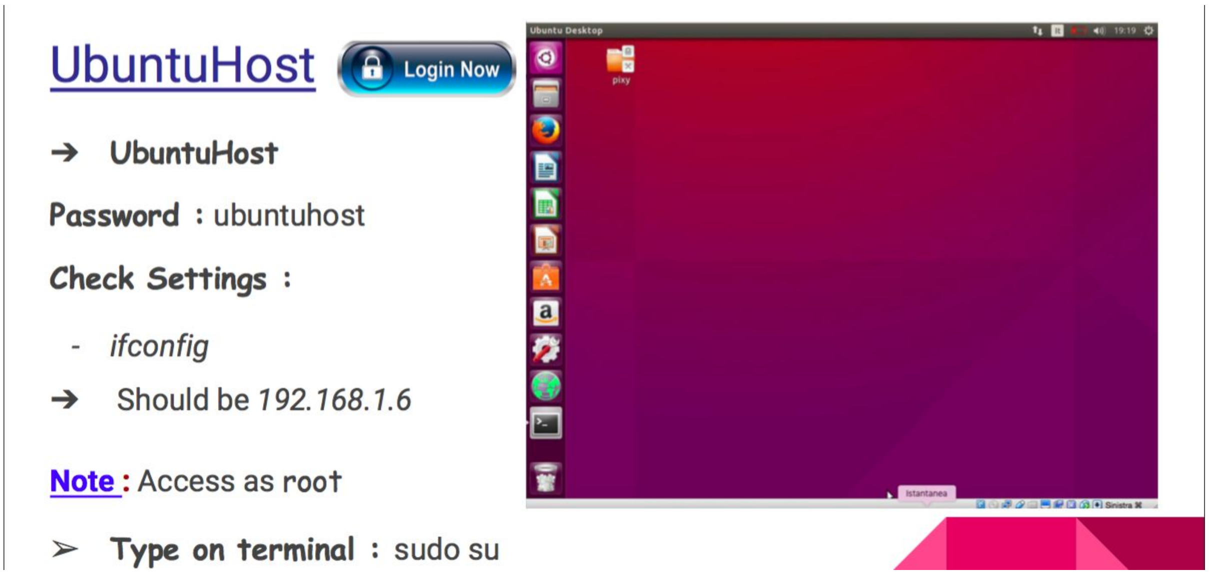
```
IPv4 Address. . . . . . . . . . . : 192.168.1.1
Subnet Mask . . . . . . . . . . . : 255.255.255.0
```

Note:

→ Has *putty* installed which is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including, SSH, Telnet, rlogin, and raw socket connection. It can also connect to a serial port.

4.2 UbuntuHost (ubuntu 15.04)

→ A linux based Vm host used to test and operate across the firewall.



UbuntuHost Login Now

➔ UbuntuHost

Password : ubuntuhost

Check Settings :

- *ifconfig*

➔ Should be *192.168.1.6*

Note : Access as root

➢ Type on terminal : sudo su

Login to UbuntuHost :

- Use the password “**ubuntuhost**” to both login on Vm and access as **root(sudo su)** on *terminal.*
- Check by typing “`ifconfig`” from terminal and ensure the **ip**→ ***192.168.1.6***

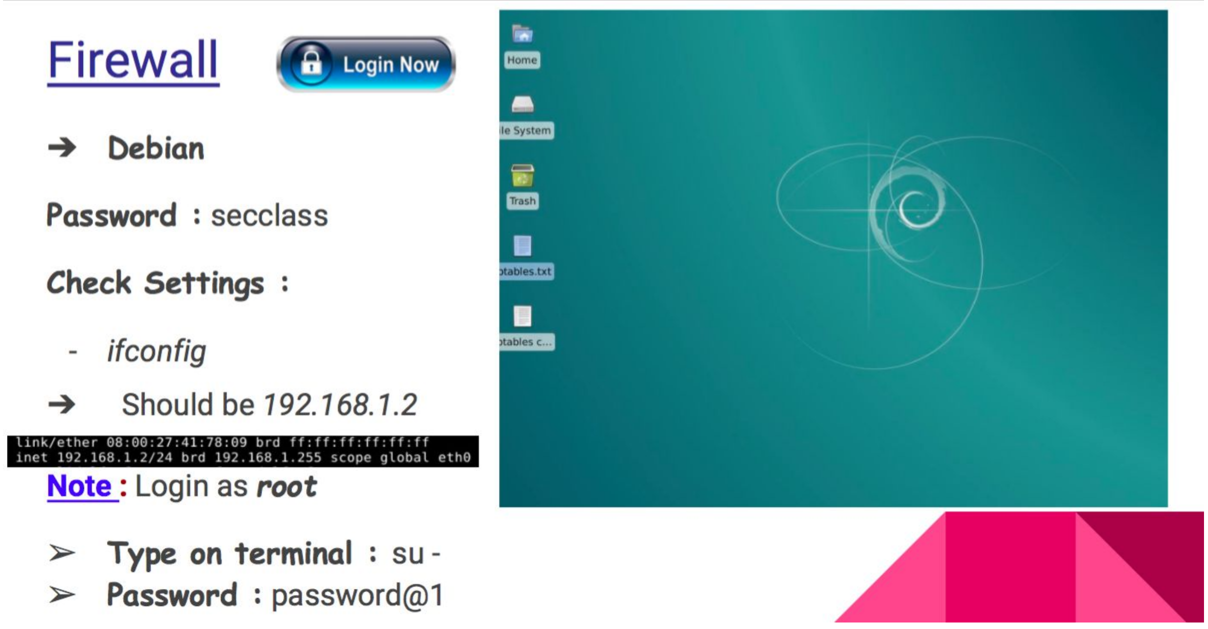
→ A network security tool **Hping3** is installed on this Vm.

Hping3 is a

- command-line oriented TCP/IP packet assembler/analyzer.
- The interface is inspired to the ping(8) unix command, but hping isn't only able to send ICMP echo requests.
- It supports TCP, UDP, ICMP and RAW-IP protocols,
- Has a traceroute mode and has the ability to send files between a covered channel, and many other features.

We shall use it in our case for **firewall testing**.

4.3 Firewall (Debian)


Firewall

Login Now

➔ Debian

Password : secclass

Check Settings :

- ifconfig

➔ Should be *192.168.1.2*

```
link/ether 08:00:27:41:78:09 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.2/24 brd 192.168.1.255 scope global eth0
```

Note : Login as ***root***

➢ **Type on terminal** : su -
➢ **Password** : password@1

Fig : Firewall Vm

⟶ Debian Vm is used as our Firewall where rules are implemented.

⟶ An apache2 server is installed

⟶ Iptables are installed by default. (Basic rules on iptables were covered in previous chapters)

Login to Firewall :

→ Login using the password "**secclass**" and check the ip settings from terminal by typing "**ifconfig**" and should be *ip-> 192.168.1.2*
→ Login as root from *terminal* too using password "password@1"

With all the Vms set, we are set to proceed with the various tasks of our Lab.

5. Tasks and exercise

5.1 Using Stateless Rules To Filter Traffic

Types of Policies

There are two types of policies in these types of firewalls.

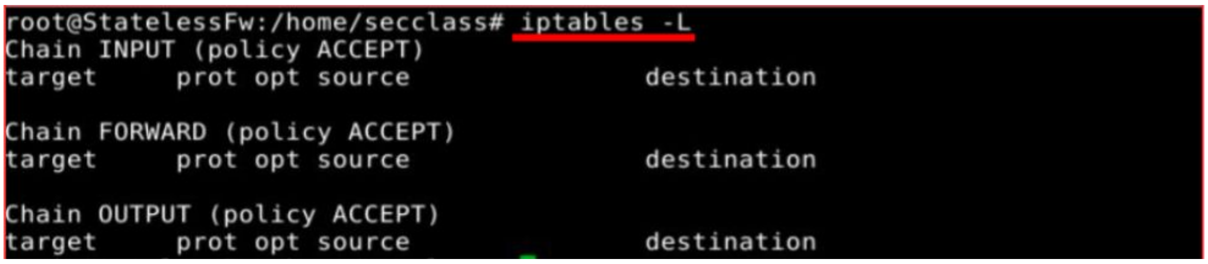
1. **Permissive(Default Allow):**

Generally it allows all packets to pass through the network using relative ports and block just some packets, such as IRC, TELNET, SNMP, etc. If you forget to block something, it can be a vulnerability for the network.

In our first task, we will use the default allow condition. From firewall :

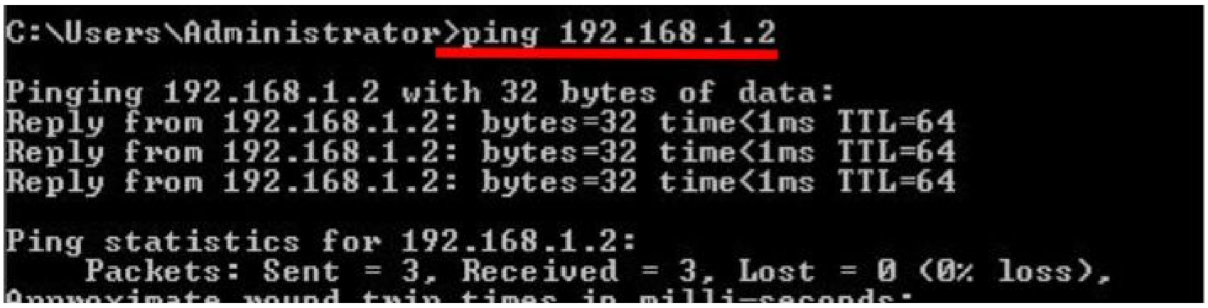
- List the iptables rules and ensure they are all on ***Accept***

> Use ***iptables -L***



```
root@StatelessFw:/home/secclass# 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
```

- As we can observe, all the policies are all set to ***ACCEPT***
- An easy test can be to ping from both **WinHost/UbuntuHost** and ensure every system is reachable : ***ping 192.168.1.2*** .See figure below



```
C:\Users\Administrator>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.2:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
```

Now we can proceed with the second task which is specifically to block all icmp echo packets coming to their server.

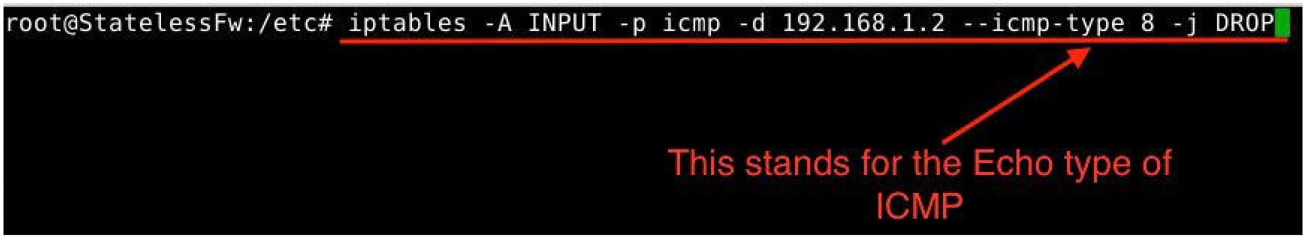
2. Block all ICMP Echo packets coming to the server

The idea is to block all ICMP packets coming to the server. To facilitate our task we shall start :

- Do a continuous ping on **WinHost** : ping 192.168.1.2 -t
- Implement the following rules on the "**Firewall**" :

#iptables -A INPUT -p icmp -d 192.168.1.2 –icmp-type 8 -j DROP

Where **-A (Allow) INPUT -p(protocols) icmp to destination 192.168.1.2 type 8, if packet matchesv jump to (-j) DROP**



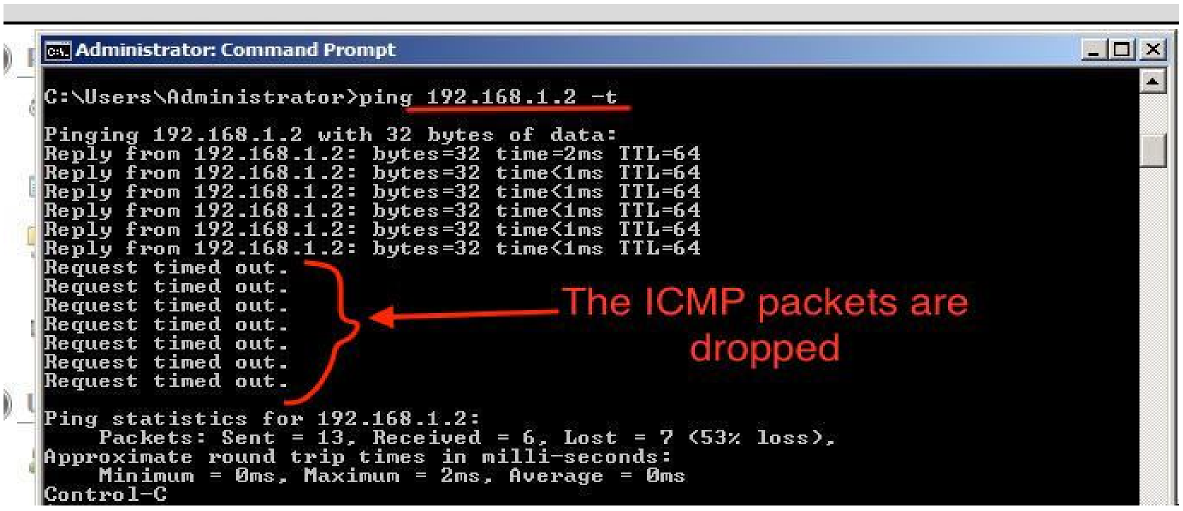
```
root@StatelessFw:/etc# iptables -A INPUT -p icmp -d 192.168.1.2 --icmp-type 8 -j DROP
```

Here we have defined the destination IP address where we want to block/DROP all icmp echo packets.

Icmp has some number types starting from 0 to 41-255 and each number has some meaning, for example number 8 stands for echo. If you want to know more about that, refer to RFC 792.
Here we want to describe some options used in this command which will remain constant throughout our exercise.
Option -A stands for append. Its work is to append the given rule to the relevant rule chain. Here our rule chain is INPUT.
The “-p” option is used to define the protocol such as ICMP,
The “-d” option is used to specify the destination IP address.
The -j option, this specifies the target of the rule; i.e., what to do if the packet matches it.

Testing:

Initially, having performed a continuous ping from one of the host(**WinHost**) terminal, after the rules has been implemented we can observe packets being dropped. See figure below :



```
C:\Users\Administrator>ping 192.168.1.2 -t

Pinging 192.168.1.2 with 32 bytes of data:
Reply from 192.168.1.2: bytes=32 time=2ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.2:
    Packets: Sent = 13, Received = 6, Lost = 7 (53% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 0ms
Control-C
```

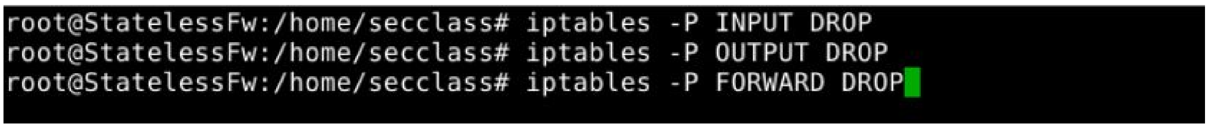
3. Restrictive(Default Deny):

By default it blocks all packets and allows some packets to pass through the network using certain ports and allows SSH, HTTP/s etc. This policy is more *secure*, compared to the permissive one. Here if you forget to allow any port, someone will complain to you and you can allow that port; on the other hand, in permissive policy, if ports are open without you knowing, then no one will complain or tell you.
To implement the default Deny policy, all our primary policies should be set in such a way that all chain policy will be set to DROP in order to drop all the packets initially at the firewall level so that they don’t come inside or go outside the network, and then we will set different specific rules to let them come inside our network, or go out of our network.

- The commands to set default Deny policy are as follows :

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

- On ***Firewall*** implement the previous rules :



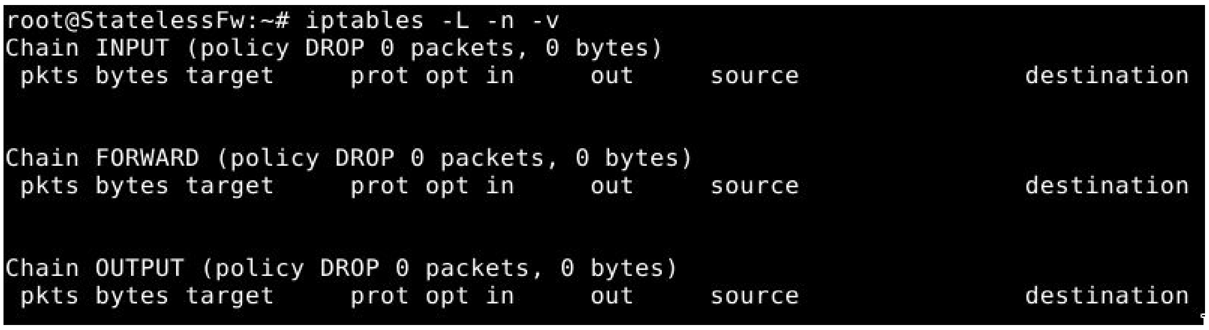
```
root@StatelessFw:/home/secclass# iptables -P INPUT DROP
root@StatelessFw:/home/secclass# iptables -P OUTPUT DROP
root@StatelessFw:/home/secclass# iptables -P FORWARD DROP
```

Where

- Option **-P** stands for *Policy*. Set the policy for the chain to the given target

With this rules, both **INPUT** and **OUTPUT** traffics are blocked.

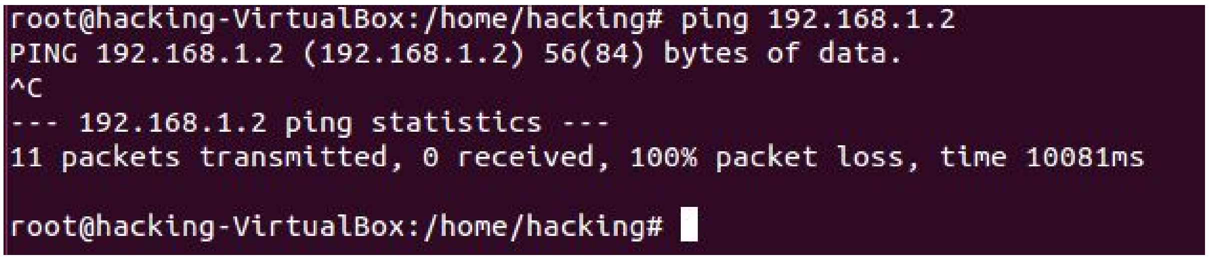
- List the new policies using **iptables -L -n -v**



```
root@StatelessFw:~# iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

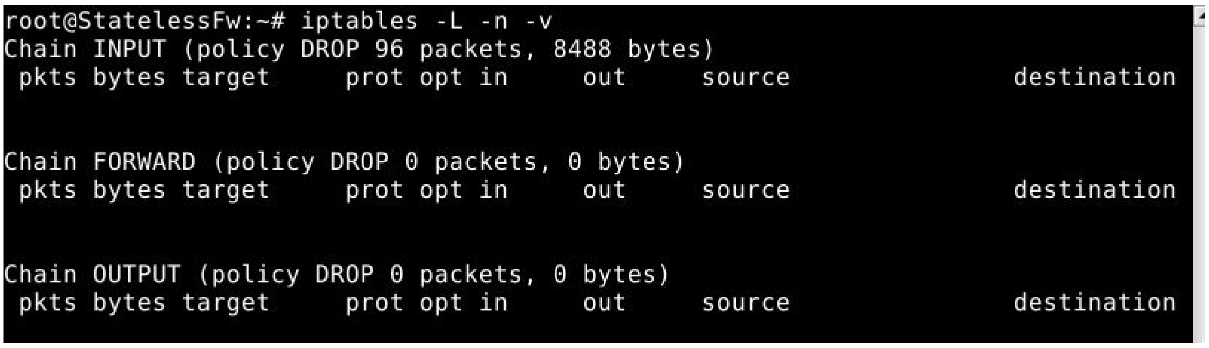
Testing :

- Open *terminal* on the **UbuntuHost** and ping the **Firewall**



```
root@hacking-VirtualBox:/home/hacking# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
^C
--- 192.168.1.2 ping statistics ---
11 packets transmitted, 0 received, 100% packet loss, time 10081ms

root@hacking-VirtualBox:/home/hacking#
```

Note : As expected, there is no traffic, since all are being dropped.
Check traffic on the Firewall using **iptables -L -n -V**



```
root@StatelessFw:~# iptables -L -n -v
Chain INPUT (policy DROP 96 packets, 8488 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

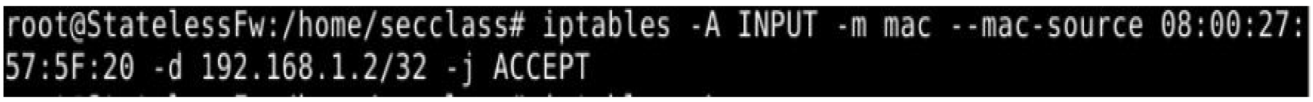
4. Whitelist traffic from the WinHost's mac address

Generally we use IP addresses to allow/deny a client via iptables, but it's not necessary or automatic that each client has static IP on their side. In that case it's hard to open port time to time for their IPs. In this situation we used *MAC* based filtering in iptables as we know that MAC addresses are fixed and can't be changed. MAC addresses are also knowns as physical/hardware address of network interface card. We used MAC based Filtering Iptables rules to whitelist trafics from the WinHost *MAC* address.

Since the current primary policy is default Deny, it means that all the chain's policy value was set to DROP.
We define a policy to allow outgoing traffic from the firewall, we set the output policy value as *ACCEPT*, as there won't be any harm if some packets are going out from our Server.
So here are those terminal commands.

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
```

Then Allow traffic for the **WinHost's** *MAC* address using the following rules

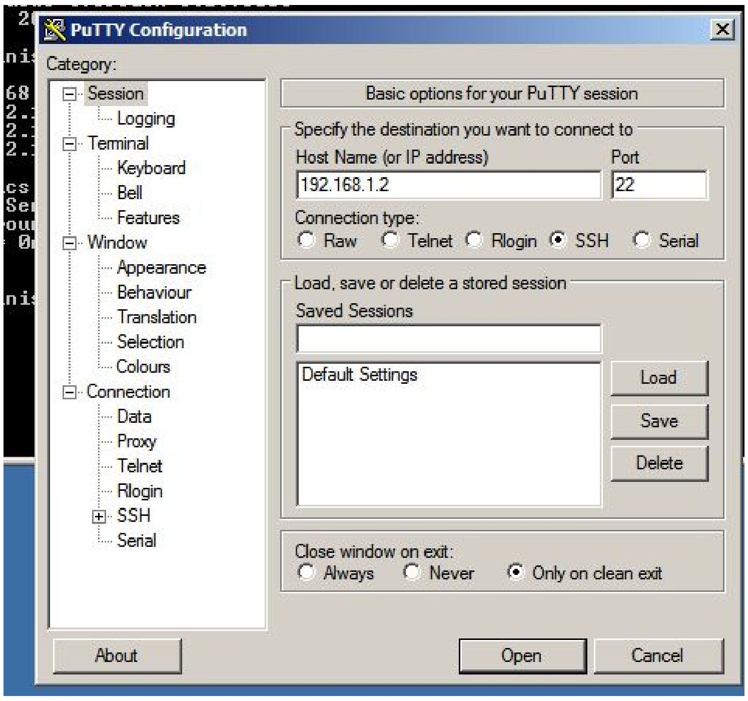


```
root@StatelessFw:/home/secclass# iptables -A INPUT -m mac --mac-source 08:00:27:
57:5F:20 -d 192.168.1.2/32 -j ACCEPT
```

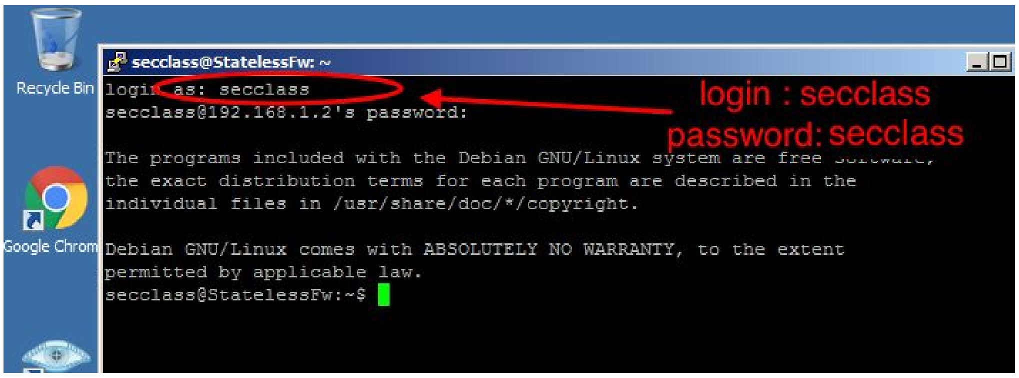
The --mac-source option Match source MAC address. It must be of the form XX:XX:XX:XX:XX:XX. Note that this only makes sense for packets coming from an Ethernet device and entering the PREROUTING, FORWARD or INPUT chains.

Test :

On **WinHost**, Open Putty located on the taskbar and connect to the **Firewall** :

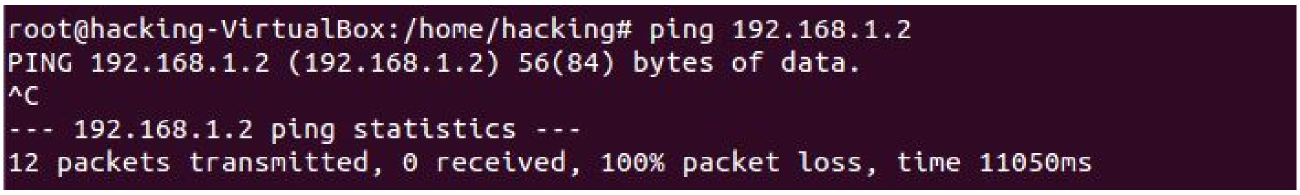


Specify the destination you want to connect to
Host Name (or IP address): 192.168.1.2
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]



```
login as: secclass
secclass@192.168.1.2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
secclass@StatelessFw:~$
```

It is clearly seen the the connection from WinHost to the Firewall is Successful whereas a ping from UbuntuHost whose MAC address was not whitelisted in out rule is not possible as shown below.

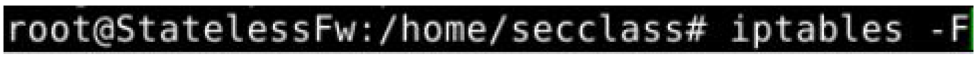


```
root@hacking-VirtualBox:/home/hacking# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
^C
--- 192.168.1.2 ping statistics ---
12 packets transmitted, 0 received, 100% packet loss, time 11050ms
```

Fig : Shows ping to firewall not working

5. Open port 22(SSH) for Specific client

- We begin by flashing the existing iptables rules using



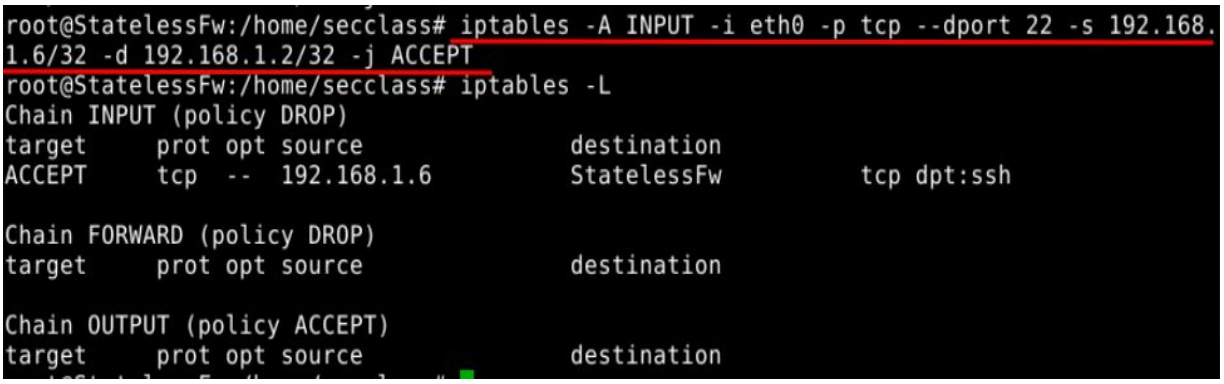
```
root@StatelessFw:/home/secclass# iptables -F
```

- We set the default OUTPUT policy to ACCEPT since a tcp connection works on 3 hand-shake, so opening in INPUT, OUTPUT has to be opened too.

```
#iptables -A OUTPUT ACCEPT
```

- We open the SSH port whose port number is 22 to allow access from **UbuntuHost** whose IP address is 192.168.1.6 using the command :

```
#iptables -A INPUT -i eth0 -p tcp --dport 22 -s 192.168.1.6/32 -d 192.168.1.2 -j ACCEPT
```



```
root@StatelessFw:/home/secclass# iptables -A INPUT -i eth0 -p tcp --dport 22 -s 192.168.
1.6/32 -d 192.168.1.2/32 -j ACCEPT
root@StatelessFw:/home/secclass# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.1.6          StatelessFw          tcp dpt:ssh

Chain FORWARD (policy DROP)
target     prot opt source               destination

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

Fig : Commands to implement on Firewall

Here we have new options which which we want to explain the meaning. The **"-i"** option stands for interface declaration. We had more than one interface, so it is important to set which interface you want to apply these rules. According to our challenge, we wanted to apply these rules on our eth0 interface.

The **"-p"** option is used to define the protocol such as tcp and udp. Here we are defining our SSH port, so we will use the tcp option

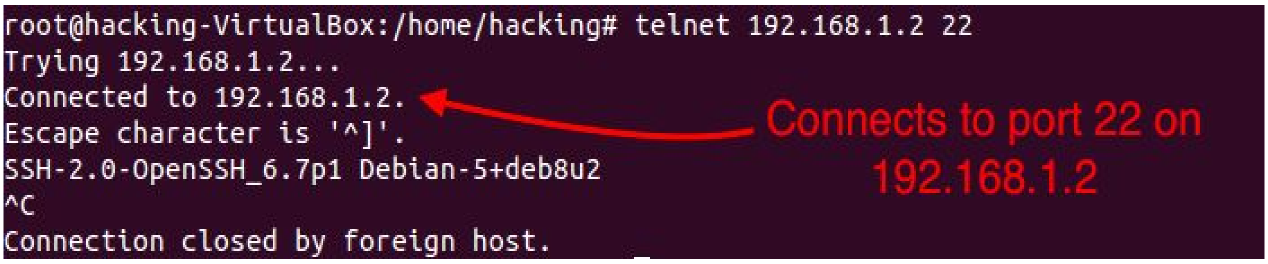
The "-dport" option, which stands for destination port. In our case, we are working on SSH and the port number of SSH is 22. So we defined dport value as 22.

The **"-s"** option is used for the source IP address. We mentioned it in our command.

Testing:

We tested this implementation by Telnet from the **UbuntuHost** using this command

```
# telnet 192.168.1.2 22
```



```
root@hacking-VirtualBox:/home/hacking# telnet 192.168.1.2 22
Trying 192.168.1.2...
Connected to 192.168.1.2.
Escape character is '^]'.
SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u2
^C
Connection closed by foreign host.
```

Fig : Shows Telnet on port 22 of firewall

Exercise: Allow traffic to a different port on the firewall from the **UbuntuHost** or **WinHost.** Ensure there is a service running on this port.
Here the students were ask to perform a similar task as the one before ie. opening a common port on the firewall accessible by the **WinHost.**

Solution: Using the same rule as before and just changing the port number resolves the exercise `#iptables -A INPUT -i eth0 -p tcp --dport 111 -s 192.168.1.6/32 -d 192.168.1.2 -j ACCEPT`
For the testing, execute `telnet 192.168.1.2 111`

5.2 Filtering specific TCP flags

Before we explain and set our firewall rule let's describe a little bit what TCP flags are with brief explanation.

U, URG (Urgent): This flag is used to identify incoming data as urgent. Such incoming segments do not have to wait until the previous segments are consumed by the receiving end but are sent directly and processed immediately.
An Urgent could be used during a stream of data transfer where a host is sending data to an application running on a remote machine. If a problem appears, the host machine needs to abort the data transfer and stop the data processing on the other end. Under normal circumstances, the abort signal will be sent and queued at the remote machine until all previously sent data is processed, however, in this case, we need the abort signal to be processed immediately.

A, ACK(Acknowledgement): This flag is used to acknowledge the successful receipt of packets. This will be happen if you run a packet sniffer while transferring data using the TCP, you will notice that, in most cases, for every packet you send or receive, an Acknowledgement follows. So if you received a packet from a remote host, then your workstation will most probably send one back with the ACK field set to 1.

R,RST(Reset) : The reset flag is used when a segment arrives that is not intended for the current connection. In other words, if you were to send a packet to a host in order to establish a connection, and there was no such service waiting to answer at the remote host, then the host would automatically reject your request and then send you a reply with the RST flag set. This indicates that the remote host has reset the connection.
While this might prove very simple and logical, the truth is that in most cases this 'feature' is used by most hackers in order to scan hosts for 'open' ports. All modern port scanners are able to detect 'open' or 'listening' ports thanks to the 'reset' function.
The method used to detect these ports is very simple: When attempting to scan a remote host, a valid TCP segment is constructed with the SYN flag set (1) and sent to the target host. If there is no service listening for incoming connections on the specific port, then the remote host will reply with ACK and RST flag set (1). If, on the other hand, there is a service listening on the port, the remote host will construct a TCP segment with the ACK flag set (1). This is, of course, part of the standard 3-way handshake.

Once the host scanning for open ports receives this segment, it will complete the 3-way handshake and then terminate it using the FIN that we will see below flag, and mark the specific port as "active".

S, SYN (Synchronization): This flag contains in the TCP flag options is the most well known flag used in TCP communications. As you might be aware, the SYN flag is initially sent when establishing the classical 3 way handshake between two hosts.

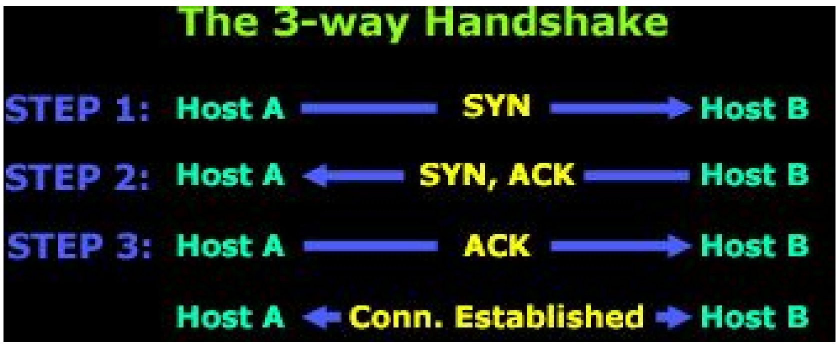

The 3-way Handshake
STEP 1: Host A SYN Host B
STEP 2: Host A SYN, ACK Host B
STEP 3: Host A ACK Host B
Host A Conn. Established Host B

Fig :3 way handshake

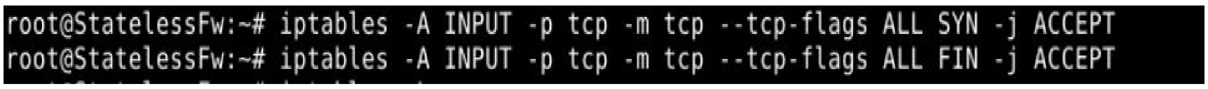
Let's assume we have two hosts A and B as shown in this diagram, Host A needs to download data from Host B using TCP as its transport protocol. The protocol requires the 3-way handshake to take place so a virtual connection can be established by both ends in order to exchange data.
During the 3-way handshake we are able to count a total of 2 SYN flags transmitted, one by each host. As files are exchanged and new connections created, we will see more SYN flags being sent and received.

P, PSH (Push): The Push flag, like the Urgent flag, exists to ensure that the data is given the priority (that it deserves) and is processed at the sending or receiving end. This particular flag is used quite frequently at the beginning and end of a data transfer, affecting the way the data is handled at both ends.

F, FIN (Finished): The final flag available is the FIN flag, standing for the word FINished. This flag is used to tear down the virtual connections created using the previous flag (SYN), so because of this reason, the FIN flag always appears when the last packets are exchanged between a connections.

So for illustration purpose we will choose two flags out of all flags we have mentioned earlier. In this case we will choose only SYN and FIN as accept in firewall by typing the following command in terminal.

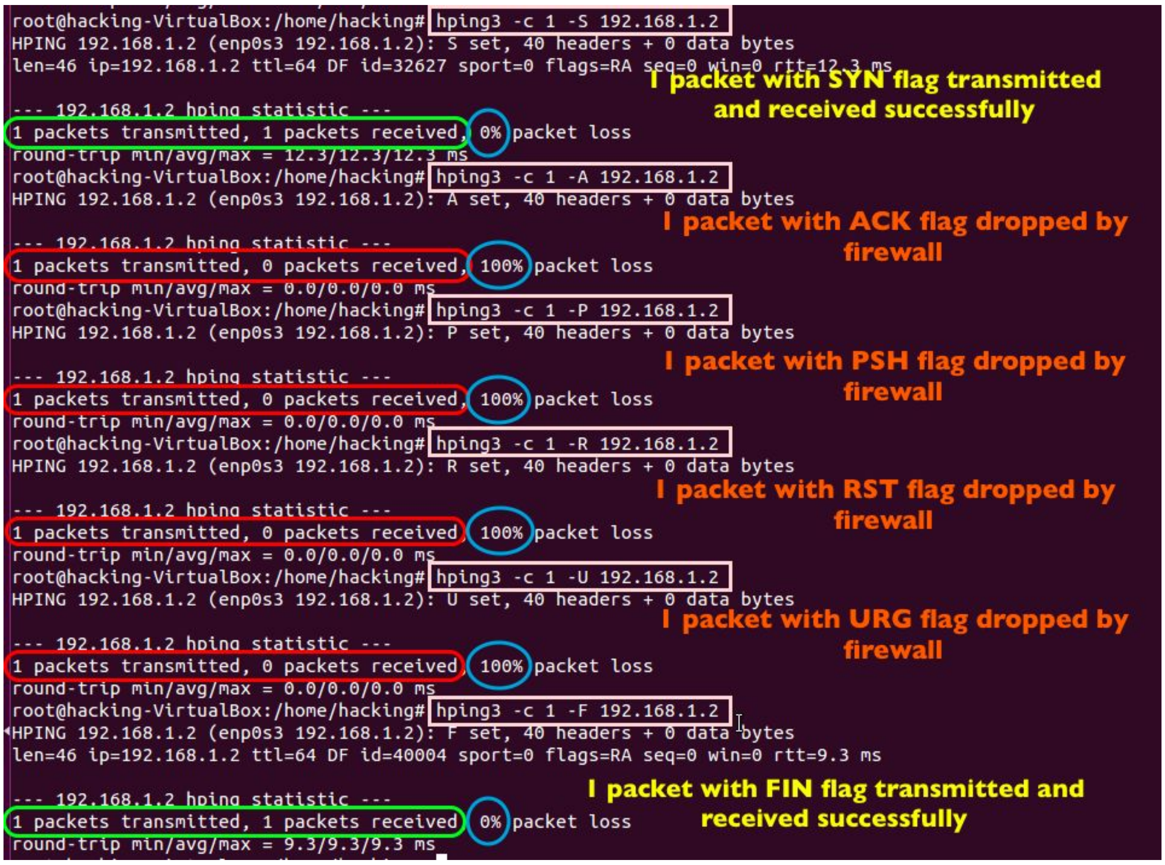
```
# iptables –A INPUT –p tcp –m tcp –tcp-flags ALL SYN –J ACCEPT
# iptables –A INPUT –p tcp –m tcp –tcp-flags ALL FIN –J ACCEPT
```



```
root@StatelessFw:~# iptables -A INPUT -p tcp -m tcp --tcp-flags ALL SYN -j ACCEPT
root@StatelessFw:~# iptables -A INPUT -p tcp -m tcp --tcp-flags ALL FIN -j ACCEPT
```

Where -p stands for match protocol, unlike when -m isn't used, they do not have to be within a range, SYN is flag symbol for synchronization, -J stands for Jump to the specified target chain when the packet matches the current rule.

Testing:
In ubuntu host we have already installed special tool packet analyser,tracer called hping3.so that using this too we can hping to the firewall using the following command
#hping3 -c 1 -S 192.168.1.2
Where -c stand for count, 1 for number of packet,-S is symbol of flag for SYN and lastly ip address of our firewall.
So in this case we keep testing by changing each flag symbol as a result we will see those flag which have been accepted in firewall,will display a description with successfully acceptance meaning 0 packet loss and the rest 100% packet loss



```
root@hacking-VirtualBox:/home/hacking# hping3 -c 1 -S 192.168.1.2
HPING 192.168.1.2 (enp0s3 192.168.1.2): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.2 ttl=64 DF id=32627 sport=0 flags=RA seq=0 win=0 rtt=12.3 ms

--- 192.168.1.2 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 12.3/12.3/12.3 ms
root@hacking-VirtualBox:/home/hacking# hping3 -c 1 -A 192.168.1.2
HPING 192.168.1.2 (enp0s3 192.168.1.2): A set, 40 headers + 0 data bytes

--- 192.168.1.2 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@hacking-VirtualBox:/home/hacking# hping3 -c 1 -P 192.168.1.2
HPING 192.168.1.2 (enp0s3 192.168.1.2): P set, 40 headers + 0 data bytes

--- 192.168.1.2 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@hacking-VirtualBox:/home/hacking# hping3 -c 1 -R 192.168.1.2
HPING 192.168.1.2 (enp0s3 192.168.1.2): R set, 40 headers + 0 data bytes

--- 192.168.1.2 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@hacking-VirtualBox:/home/hacking# hping3 -c 1 -U 192.168.1.2
HPING 192.168.1.2 (enp0s3 192.168.1.2): U set, 40 headers + 0 data bytes

--- 192.168.1.2 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@hacking-VirtualBox:/home/hacking# hping3 -c 1 -F 192.168.1.2
HPING 192.168.1.2 (enp0s3 192.168.1.2): F set, 40 headers + 0 data bytes
len=46 ip=192.168.1.2 ttl=64 DF id=40004 sport=0 flags=RA seq=0 win=0 rtt=9.3 ms

--- 192.168.1.2 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 9.3/9.3/9.3 ms
```

1 packet with SYN flag transmitted and received successfully
1 packet with ACK flag dropped by firewall
1 packet with PSH flag dropped by firewall
1 packet with RST flag dropped by firewall
1 packet with URG flag dropped by firewall
1 packet with FIN flag transmitted and received successfully

Fig : SYN and FIN packets accepted while rest are dropped

Exercise: Exploiting the open port 22 (ssh) from previous task,define default deny OUTPUT policy and allow traffic for tcp flags SYN and ACK. In other words make traffic possible. Test with ssh as per task 1 above

Solution: Use the commands to enable traffic for SYN and ACK on port 22 :
# iptables -A INPUT -p tcp -m tcp -tcp-flags ALL SYN -J ACCEPT
# iptables -A INPUT -p tcp -m tcp -tcp-flags ALL ACK -J ACCEPT

In otherwords, we are permitting 2 important flags for connection establishment ie. SYN and ACK. Note that for a traffic to be possible, packets have to flow to and fro, so having in INPUT packets on port 22, on OUTPUT only the SYN(ok) and ACK packets will be transmitted back.

Testing: Using `#hping3 -c 1 -S 192.168.1.2` and `#hping3 -c 1 -A 192.168.1.2`as in previous task and changing the flags, we can observe that ONLY SYN and ACK packets are forwarded.

5.3 NAT & PORT FORWARDING

Network address translation(NAT) is a general term for mapping ip addresses in order to redirect them to an alternative address. NAT is usually used to increase security, using a single IP address to represent multiple IP addresses. Port forwarding is also used to redirect incoming traffic from a low numbered port to software listening on a higher port. This software can be running as a normal user, which avoids the security risk caused by running as the root user.A host that implements NAT typically has access to two or more networks and is configured to route traffic between them.

Port forwarding also called “port mapping” is the process of forwarding requests for a specific port to another host, network, or port. The network address translator /gateway in this case, the firewall, changes the destination address or destination port of the packet to reach a host within a masqueraded, typically private, network. We will be redirecting traffic from port 8080 to the common http port 80 for this task

The nat table has the following built-in chains:

- **Pre-routing chain**: maps packets when the destination address of the packet needs to be changed before routing
- **Post-routing chain**: maps packets when the source address of the packet needs to be changed after routing
- **Output chain**: maps packets originating from the different firewall interfaces.

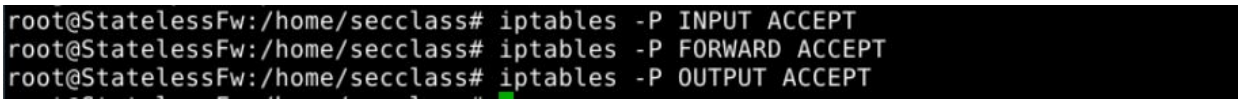
Below is a scheme on how packets are processed by these chains :

```
PACKET IN
    |
PREROUTING--[routing]-->--FORWARD-->--POSTROUTING-->--OUT
 - nat (dst)   |           - filter       - nat (src)
               |                            |
               |                            |
              INPUT                        OUTPUT
              - filter                     - nat (dst)
               |                           - filter
               |                            |
               `----->-----[app]----->------'
```

During port forwarding, the packet entering the firewall is inspected by the rules in the nat table PREROUTING chain to see whether it requires destination modification (DNAT). The packet is then routed by Linux router after leaving the PREROUTING chain.

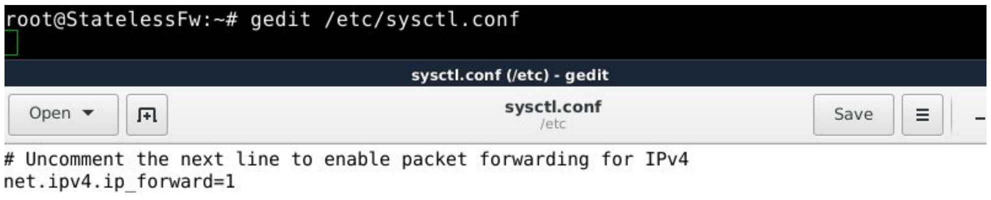
To carry out this task, carry out the following steps;-

- Flush filter tables with **`iptables -F`** **`- flushes the rules in the filter table`**
- Flush nat table **`iptables -t nat -F`** , - t selects the table in this case nat and flushes the rules
- Define all policies to accept traffic in the filter table



```
root@StatelessFw:/home/secclass# iptables -P INPUT ACCEPT
root@StatelessFw:/home/secclass# iptables -P FORWARD ACCEPT
root@StatelessFw:/home/secclass# iptables -P OUTPUT ACCEPT
```

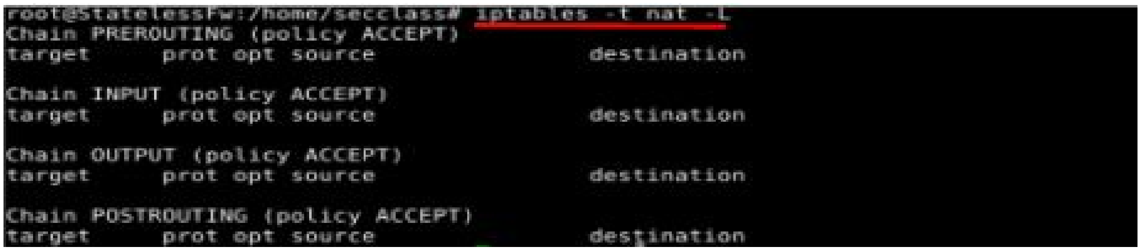
- Uncomment the following line in the sysctl.conf file to enable ip forwarding for IPv4



```
root@StatelessFw:~# gedit /etc/sysctl.conf
```

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

- check iptables **`iptables -L`** by listing the rules in the filter table
- check the nat policies by using **`iptables -t nat -L`** by listing the rules in the nat table



```
root@StatelessFw:/home/secclass# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (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
```

- Insert the following rules to activate port redirection :

Enable linux IP forwarding

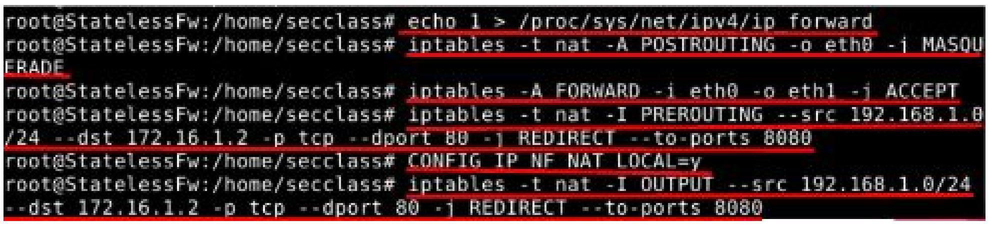
```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

The *MASQUERADE* extension is so that packets which are forwarded to the new destination IP can be properly routed back to the origin of the request.

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Redirect http-Traffic going to Port 80 to port 8080

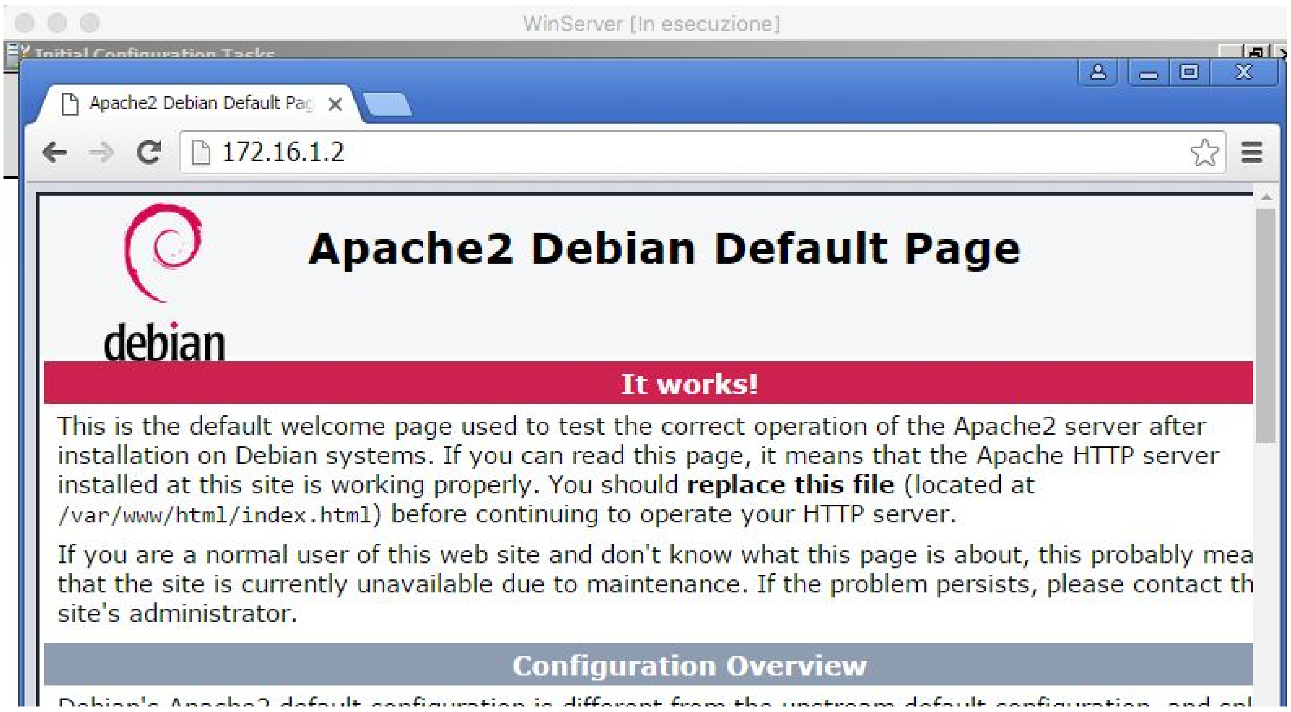
```
#iptables -t nat -I PREROUTING --src 192.168.1.0/24 --dst 172.16.1.2
-p tcp --dport 80 -j REDIRECT -- to-ports 8080

iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT

CONFIG_IP_NF_NAT_LOCAL=y

iptables -t nat -I OUTPUT --src 192.168.1.0/24 --dst 172.16.1.2 -p
tcp --dport 80 -j REDIRECT --to- ports 8080
```



```
root@StatelessFw:/home/secclass# echo 1 > /proc/sys/net/ipv4/ip_forward
root@StatelessFw:/home/secclass# iptables -t nat -A POSTROUTING -o eth0 -j MASQU
ERADE
root@StatelessFw:/home/secclass# iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
root@StatelessFw:/home/secclass# iptables -t nat -I PREROUTING --src 192.168.1.0
/24 --dst 172.16.1.2 -p tcp --dport 80 -j REDIRECT --to-ports 8080
root@StatelessFw:/home/secclass# CONFIG_IP_NF_NAT_LOCAL=y
root@StatelessFw:/home/secclass# iptables -t nat -I OUTPUT --src 192.168.1.0/24
--dst 172.16.1.2 -p tcp --dport 80 -j REDIRECT --to-ports 8080
```

Testing :

- Open a browser on **WinHost** and Type "**172.16.1.2:80**"



WinServer [In esecuzione]

Initial Configuration Tasks

Apache2 Debian Default Pag

172.16.1.2

Apache2 Debian Default Page

debian

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably mea that the site is currently unavailable due to maintenance. If the problem persists, please contact th site's administrator.

Configuration Overview

- We can observe that the web service can be accessed on port 80

6.Conclusion:

By now, you should be comfortable with forwarding ports on a Linux server with iptables. The process involves permitting forwarding at the kernel level, setting up access to allow forwarding of the specific port's traffic between two interfaces on the firewall system, and configuring the NAT rules so that the packets can be routed correctly. This may seem like an unwieldy process, but it also demonstrates the flexibility of the netfilter packet filtering framework and the iptables firewall. This can be used to disguise your private networks topology while permitting service traffic to flow freely through your gateway firewall machine.

Best Practices in firewall implementation

→ Document all firewall rule changes.

While this tip sounds like a no-brainier, firewalls do not have a change management process built into them, so documenting changes has never become a best (or even a standard) practice for many organizations. If a firewall administrator makes a change because of an emergency or some other form of business disruption, chances are he is under the gun to make it happen as quickly as possible, and process goes out the window. But what if this change cancels out a prior policy change, resulting in downtime? This is a fairly common occurrence.

Firewall management products provide a central dashboard that provides full visibility into all firewall rule bases, so all members of the team have a common view and can see who made what change, when they made it and from where. This makes troubleshooting and overall policy management much easier and more efficient.

→ Install all access rules with minimal access rights.

Another common security issue is overly permissive rules. A firewall rule is made up of three fields: source (IP address), destination (network/subnet) and service (application or other destination). In order to ensure there are enough open ports for everyone to access the systems they need, common practice has been to assign a wide range of objects in one or more of those fields. When you allow a wide range of IP addresses to access a large group's networks for the sake of business continuity, these rules become overly permissive, and as a result, insecure. A rule where the service field is 'ANY' opens up 65,535 TCP ports. Did the firewall administrator really mean to open up 65,535 attack vectors for hackers?

→ Verify every firewall change against compliance policies and change requests.

In firewall operations, daily life centers around finding problems, fixing problems and installing new systems. In the cycle of installing new firewall rules to solve problems and enable new products and business units, we often forget that the firewall is also the physical implementation of the corporate security policy. Every rule should be reviewed to understand that it meets the spirit and intent of the security policy and any compliance policies, not just the letter of the law.

→ Remove unused rules from the firewall rule bases when services are decommissioned.

"Rule bloat" is a very common occurrence with firewalls because most operations teams have no process for deleting rules. Business units are great at letting you know they need new rules, but they never let the firewall team know they no longer need a service. Getting into the loop on server and network decommissioning as well as application upgrade cycles is a good start for understanding when rules need to come out. Running reports on unused rules is another step. Hackers like the fact that firewall teams never remove rules. In fact, this is how many compromises occur.

→ Perform a complete firewall review at least twice per year.

If you are a merchant with significant credit card activity, then this one is not just a best practice but a requirement; PCI requirement 1.1.6 calls for reviews at least every six months.

Firewall reviews also are a critical part of the maintenance of your firewall rule base. Your networks and services are not static so your firewall rule base should not be either. As corporate policies evolve and compliance standards change, you need to review how you are enforcing traffic on the firewalls. This is a good place to clean up all those redundant rules that have been replaced by new rules, rules for services no longer used that you were not informed about, and all those temporary exceptions that were added to get projects, acquisitions, mergers and so on finished. The best way to keep bad things from happening is to not create an environment where they can.

References

- https://www.digitalocean.com/community/tutorials/how-to-forward-ports-through-a-linux-gateway-with-iptables
- http://csrc.nist.gov/publications/nistpubs/800-41-Rev1/sp800-41-rev1.pdf
- http://www.networkworld.com/article/2247110/network-security/top-5-best-practices-for-firewall-administrators.html
- https://www.digitalocean.com/community/tutorials/what-is-a-firewall-and-how-does-it-work
- https://wiki.archlinux.org/index.php/Iptables#Installation