Acceptance clause: by completing this project the student accepts that he/she is responsible for what he/she will do and that he/she will not use these tools outside the virtual one created for that purpose. The student also accepts the legal repercussions that some of these actions may be subject to if these actions are carried out on external sites.

Project 4: Firewalls

Level: 4

Difficulty: medium-high

Environment: Linux Machines (VM)

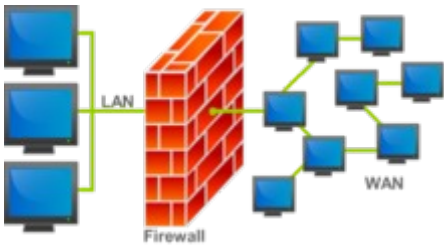
Objective: Install and analyze the indicated tools obtaining information about worker1 (DO NOT USE outside the virtual network).

Task: The student will have to collect information about iptables, and configure it in worker1 to allow access to http but block ssh connections from Router.
Recommended: iptables is installed in major Linux distributions but is recommendable to install iptables-persistent package to store and load the rules in the shutdown/bootup process (**sudo apt install iptables-persistent**).

Recommended: Start with project 2-3 to learn about scanning tools.

In computing, a firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted network and an untrusted network, such as the Internet.

Firewalls are categorized as a network-based or a host-based system. Network-based firewalls can be positioned anywhere within a LAN or WAN. They are either a software appliance running on general-purpose hardware, a hardware appliance running on special-purpose hardware, or a virtual appliance running on a virtual host controlled by a hypervisor. Firewall appliances may also offer non firewall functionality, such as DHCP or VPN services. Host-based firewalls are deployed directly on the host itself to control network traffic or other computing resources. This can be a daemon or service as a part of the operating system or an agent application for protection.



An illustration of a network-based firewall within a network.

Packet filter: The first reported type of network firewall is called a packet filter, which inspect packets transferred between computers. The firewall maintains an access control list which dictates what packets will be looked at and what action should be applied, if any, with the default action set to silent discard. Three basic actions regarding the packet consist of a silent discard, discard with Internet Control Message Protocol or TCP reset response to the sender, and forward to the next hop. Packets may be filtered by source and destination IP addresses, protocol, source and destination ports. The bulk of Internet communication in 20th and early 21st century used either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) in conjunction with well-known ports, enabling firewalls of that era to distinguish between specific types of traffic such as web browsing, remote printing, email transmission, and file transfers.

The most useful firewalls (packets filters) are acts at layer 3 or 4 of OSI communication model.

Application firewall: in 1993 an application firewall known as Firewall Toolkit (FWTK) was released. The key benefit of application layer filtering is that it can understand certain applications and protocols such as File Transfer Protocol (FTP), Domain Name System (DNS), or Hypertext Transfer Protocol (HTTP) (is to say act at layer 7). This allows it to identify unwanted applications or services using a non standard port, or detect if an allowed protocol is being abused. It can also provide unified security management including enforced encrypted DNS and virtual private networking.

Personal Firewall: There are personal firewalls to protect the personal computers from unwanted communications & information gathering (Windows & MacOS includes software for it) but, per example, for Windows there are some efficient/well-considered/ free-software alternatives such as https://personalfirewall.comodo.com/ or https://filehippo.com/download_zonealarm-free-firewall/

iptables

iptables is a user-space utility program that allows a system administrator to configure the IP packet filter rules of the Linux kernel firewall, implemented as different Netfilter modules. The filters are organized in different tables, which contain chains of rules for how to treat network traffic packets. Different kernel modules and programs are currently used for different protocols (i.e. iptables applies to IPv4, ip6tables to IPv6). At this moment a new evolution of iptables is running in major distribution named **nftables** but all distribution as a compatibility package for use iptables under nftables.

iptables requires elevated privileges to operate and must be executed by user root and allows the system administrator to define tables containing chains of rules for the treatment of packets. Each table is associated with a different kind of packet processing. Packets are processed by sequentially traversing the rules in chains. A rule in a chain can cause a go-to or jump to another chain, and this can be repeated to whatever level of nesting is desired. Every network packet arriving at or leaving from the computer traverses at least one chain.

Packet flow paths. Packets start at a given box and will flow along a certain path, depending on the circumstances.

The origin of the packet determines which chain it traverses initially. There are five predefined chains (mapping to the five available Netfilter hooks), though a table may not have all chains. Predefined chains have a policy, for example DROP, which is applied to the packet if it reaches the end of the chain. The system administrator can create as many other chains as desired. These chains have no policy; if a packet reaches the end of the chain it is returned to the chain which called it. A chain may be empty.

- PREROUTING: Packets will enter this chain before a routing decision is made.
- INPUT: Packet is going to be locally delivered. It does not have anything to do with processes having an opened socket; local delivery is controlled by the "local-delivery" routing table: ip route show table local.
- FORWARD: All packets that have been routed and were not for local delivery will traverse this chain.
- OUTPUT: Packets sent from the machine itself will be visiting this chain.
- POSTROUTING: Routing decision has been made. Packets enter this chain just before handing them off to the hardware.

A chain does not exist by itself; it belongs to a table. There are three tables: **nat, filter, and mangle**. Unless preceded by the option -t, an iptables command concerns **the filter table by default**. For example, the command **iptables -L -v -n**, which shows some chains and their rules, is equivalent to **iptables -t filter -L -v -n**. To show chains of table nat, use the command **iptables -t nat -L -v -n**

Each rule in a chain contains the specification of which packets it matches. It may also contain a target (used for extensions) or verdict (one of the built-in decisions). As a packet traverses a chain, each rule in turn is examined. If a rule does not match the packet, the packet is passed to the next rule. If a rule does match the packet, the rule takes the action indicated by the target/verdict, which may result in the packet being allowed to continue along the chain or may not. Matches make up the large part of rulesets, as they contain the conditions packets are tested for. These can happen for about any layer in the OSI model, as with e.g. the --mac-source and -p tcp --dport parameters, and there are also protocol-independent matches, such as -m time.

The packet continues to traverse the chain until either

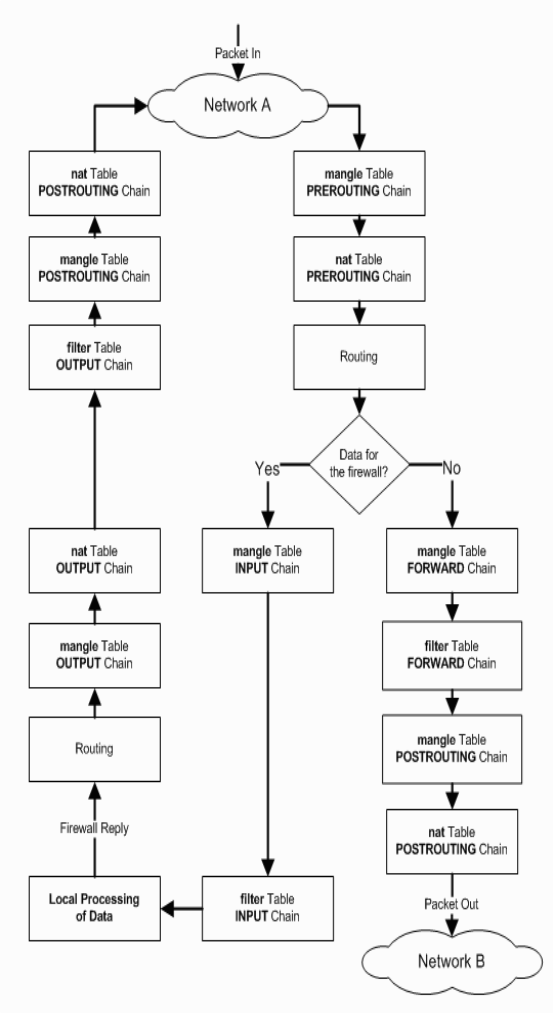
- a rule matches the packet and decides the ultimate fate of the packet, for example by calling one of the ACCEPT or DROP, or a module returning such an ultimate fate; or
- a rule calls the RETURN verdict, in which case processing returns to the calling chain; or
- the end of the chain is reached; traversal either continues in the parent chain (as

if RETURN was used), or the base chain policy, which is an ultimate fate, is used.

- Targets also return a verdict like ACCEPT (NAT modules will do this) or DROP (e.g. the REJECT module), but may also imply CONTINUE (e.g. the LOG module; CONTINUE is an internal name) to continue with the next rule as if no target/verdict was specified at all.

Queue Type	Queue Function	Packet Transformation Chain in Queue	Chain Function
Filter	Packet filtering	`FORWARD`	Filters packets to servers accessible by another NIC on the firewall.
		`INPUT`	Filters packets destined to the firewall.
		`OUTPUT`	Filters packets originating from the firewall
Nat	Network Address Translation	`PREROUTING`	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as **destination NAT** or **DNAT**.
		`POSTROUTING`	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as **source NAT**, or **SNAT**.
		`OUTPUT`	Network address translation for packets generated by the firewall. (Rarely used in SOHO environments)

Mangle	TCP header modification	`PREROUTING`
		`POSTROUTING`
		`OUTPUT`
		`INPUT`
		`FORWARD`
			Modification of the TCP packet quality of service bits before routing occurs. (Rarely used in SOHO environments)



Each firewall rule inspects each IP packet and then tries to identify if it can apply some type of operation to it. Once a target is identified, the packet has to go through for further processing:

ACCEPT : the packet is accepted. iptables ends.

DROP: the package is blocked. iptables ends.

LOG : Packet info is sent to syslog . iptables continues with the next rule.

REJECT : same as drop but sends an error message.

DNAT : used to do destination network address translation. Rewriting of the destination IP address of the packet.

SNAT : used to do source network address translation . Rewriting of the source IP address of the packet.

MASQUERADE : used to do Source Network Address Translation.

Parameters used:

-t <-table-> if the table is not specified filter table is chosen.

-j <target> Jump to target chain when this rule is checked.

-A Append rule to the end of the chain

-F Flush. Delete all the rules in the selected table.

-p <protocol-type> Match protocol (icmp, tcp, udp, and all)

-s <ip-address> Match source IP address

-d <ip-address> Match destination IP address

-i <interface-name> Match "input" interface (packet input)

-o <interface-name> Match "output" interface (packet output)

Examples of rules:

iptables -A INPUT -s 0/0 -i enp0s3 -d 192.168.1.1 -p TCP -j ACCEPT

iptables is configured to accept TCP packet from anywhere using device enp0s3 and destination 192.168.1.1 (0/0 represents anywhere address).

iptables -A FORWARD -s 0/0 -i enp0s3 -d 192.168.1.58 -o enp0s8 -p TCP --sport 1024:65535 --dport 80 -j ACCEPT

Iptables is configured to accept paquetes TCP to be routed when arrives to enp0s3 device and destination 192.168.1.58 on device enp0s8. The source port must be in the range 1024 - 65535 and destination port is 80 (www/http).

iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 3/s -i eth0 -j ACCEPT

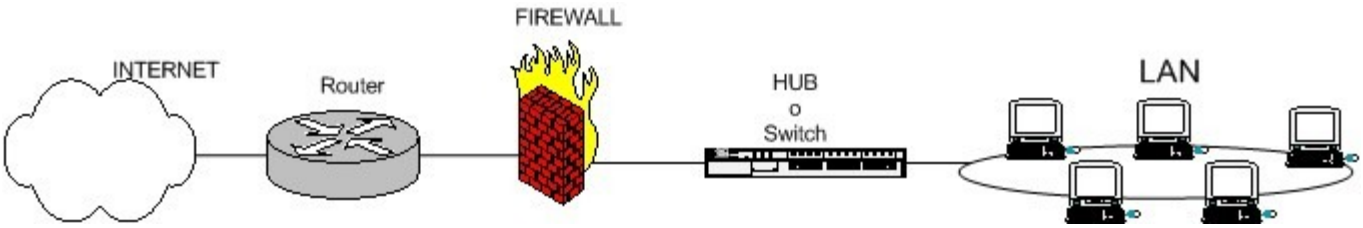
The parameter limit allows us insert a maximum average number of matches to allow per second. In this case 3/second (3/s).

Example of configuration:

```
# Example 1
## FLUSH
iptables –F
iptables –X
```

```
iptables –Z
iptables –t nat –F

## Default policy
iptables –P INPUT ACCEPT
iptables –P OUTPUT ACCEPT
iptables –P FORWARD ACCEPT
iptables –t nat –P PREROUTING ACCEPT
iptables –t nat –P POSTROUTING ACCEPT

## Filter  localhost enabled.
/sbin/iptables –A INPUT –i lo –j ACCEPT
# Our IP Enabled
iptables –A INPUT –s 195.65.34.234 –j ACCEPT
# Friend 1: In to mysql
iptables –A INPUT –s 231.45.134.23 –p tcp ––dport 3306 –j ACCEPT
# Fiend 2: FTP enabled
iptables –A INPUT –s 80.37.45.194 –p tcp –dport 20:21 –j ACCEPT
# Port 80: open web service.
iptables –A INPUT –p tcp ––dport 80 –j ACCEPT
# Remaining ports: closed
iptables –A INPUT –p tcp ––dport 20:21 –j DROP
iptables –A INPUT –p tcp ––dport 3306 –j DROP
iptables –A INPUT –p tcp ––dport 22 –j DROP
iptables –A INPUT –p tcp ––dport 10000 –j DROP
echo " OK . Verify with iptables –L –n"
# End of script


# Example 2
## enp0s3 = External
## enp0s8 = Internal LAN

# Access to the firewall from the LAN
iptables –A INPUT –s 192.168.10.0/24 –i enp0s8–j ACCEPT
# Masking LAN with BIT FORWARDING
iptables –t nat –A POSTROUTING –s 192.168.10.0/24 –o enp0s3 –j MASQUERADE
# Allow forward of packet in the FW
# all machines can out to the FW.
echo 1 > /proc/sys/net/ipv4/ip_forward
## Closed all
# Note: 0.0.0.0/0 = all net
# Close: all knowed ports.
iptables –A INPUT –s 0.0.0.0/0 –p tcp –j DROP
```

iptables –A INPUT –s 0.0.0.0/0 –p udp –j DROP



References:

https://www.geeksforgeeks.org/iptables-command-in-linux-with-examples/
https://webguy.vip/example-of-iptables/
https://www.cyberciti.biz/tips/linux-iptables-examples.html