---->cat /etc/group | grep sudo ////to know all users (super-user) that can use sudo command

###############We don't want you to use the DHCP service of your machine. You've got to configure it to have a static IP and a Netmask in \30

---->sudo apt-get update

---->https://linuxconfig.org/how-to-configure-static-ip-address-on-ubuntu-18-10-cosmic-cuttlefish-linux

--------->vim /etc/netplan/50-cloud-init.yaml

------>sudo netplan --debug apply

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
     dhcp4: no
        addresses: [10.12.254.253/30]
     gateway4: 10.12.254.254
     nameservers:
       addresses: [8.8.8.8]
```

---->https://netplan.io/reference

---------->networkd = deamon to configure your network interface via DHCP

Contenu HTML

---------->The top-level node in a netplan configuration file is a `network:` mapping that

Contenu HTML

contains `version: 2`, and then device definitions grouped by their type, such

Contenu HTML

as `ethernets:`, `wifis:`, or `bridges:`. These are the types that our renderer can understand

and are supported by our backends.

---------->dhcp4 = disable DHCP for IPv4. Off by default

---------->ethernets = interface type

---------->enp0s3 = nom d interface

---------->10.12.254.253/30 (normaly in netmask /30 you have .252 for netmask .255 for broadcast so you have only the choice to chose between .253 and 254(probably gateway))

---------->gateway4 = Set default gateway for IPv4

---------->nameservers : DNS a utiliser

###########You have to change the default port of the SSH service by the one of your choice

---->https://serversforhackers.com/c/configuring-sshd-on-the-server

---->sudo apt-get install ssh

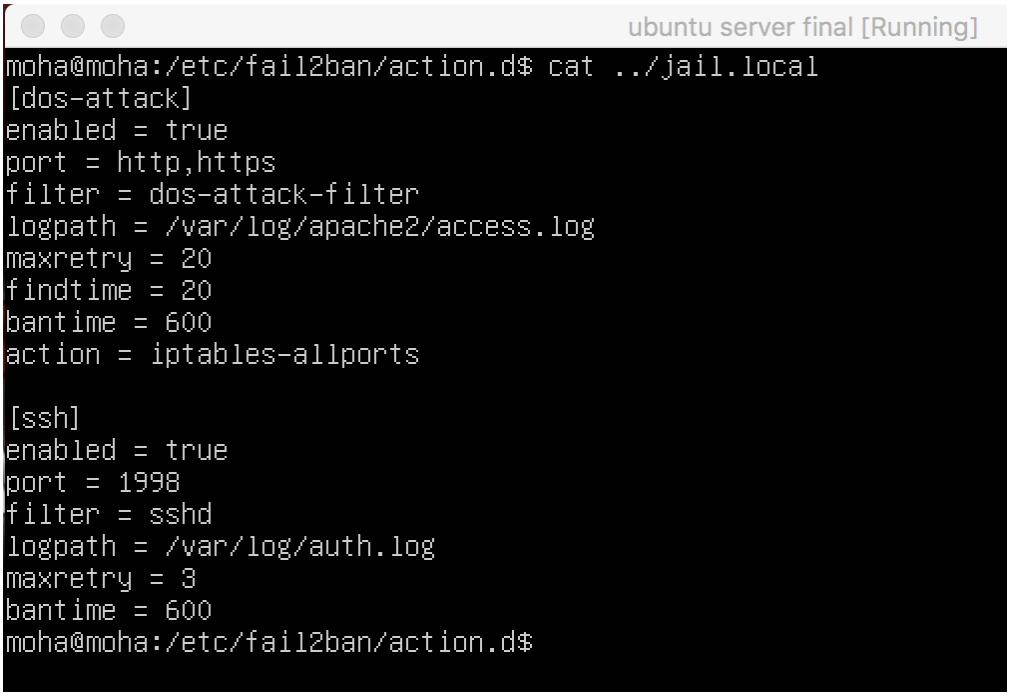
>sudo apt-get install ssh
---->sudo vim /etc/ssh/sshd_config
---->change in the file (#Port 22) to (Port 1998)
########## SSH root access SHOULD NOT be allowed directly, but with a user who can be root
---->in sshd_config change (#PermitRootLogin prohibit-password) to (PermitRootLogin no)
##########SSH access HAS TO be done with publickeys
###before this you have to send you public key (in mac)-->rm -rf .ssh -->ssh-keygen -->ssh-copy-id moha@10.12.254.253 -p 1998 -->ssh moha@10.12.254.254
---->in sshd_config change (#PasswordAuthentication yes) to (PasswordAuthentication no)
---->sudo service ssh restart

##########You have to set the rules of your firewall on your server only with the services used outside the VM
---->if you want to use iptables https://www.grafikart.fr/tutoriels/iptables-694 --> https://drive.google.com/drive/folders/1jx4ibVKbvi7yqn1OFV524-cU_RYf6Yob
-->**Iptables** is used to set up, maintain, and inspect the **tables** of IP **packet filter rules** in the Linux kernel.
->Several different **tables** may be defined.
->Each **table** contains a number of built-in **chains** and may also contain **user-defined chains**.
->Each **chain** is a **list of rules** which can match a set of packets.
->Each rule specifies what to do with a packet that matches.
->This is called a '**target**', which may be a jump to a user-defined chain in the same table
->**target** A firewall rule specifies criteria for a **packet**, and a **target**. If the **packet** does not match, the next rule in the chain is the examined;
->if it does match, then the next rule is specified by the value of the **target**,
->which can be the name of a user-defined chain or one of the special values ACCEPT, DROP, QUEUE, or RETURN. //see man iptables
--> **-t** specifie le tableau a utiliser (filter: est le tableau par defaut)
--> **-F --flush** [chain] : Flush the selected chain (all the [chains] in the table if none is given). This is equivalent to deleting all the rules one by one
--> **-X, --delete-chain** [*chain]* Delete the optional user-defined chain specified
-->**-P, --policy** [*chain target]* -> Set the policy for the chain to the given target
-->**-A, --append** [*chain rule-specification]* Append one or more rules to the end of the selected chain // ألحق
-->**Match Extensions**
->iptables can use extended **packet matching modules**. These are loaded in two ways: implicitly, when -p or --protocol is specified, or with the -m or --match options, followed by the matching module name;
->after these, various extra command line options become available, depending on the specific module. You can specify multiple extended match modules in one line
-->**state**
->This module, when combined with connection tracking, allows access to the connection

tracking state for this packet.
->**--state state**
ESTABLISHED meaning that the packet is associated with a connection which has seen packets in both directions,
RELATED meaning that the packet is starting a new connection, but is associated with an existing connection,
-->**-j, --jump target** ; This specifies the target of the rule; i.e., what to do if the packet matches it. // ip to **target ACCEPT**
-->**-i, --in-interface [!] [name]** Name of an interface via which a packet was received
-->**-o, --out-interface [!] [name]** Name of an interface via which a packet is going to be sent
-->**-p, --protocol [!] protocol**
-->--destination-port,**--dport** [!] port[:port]
---->using ufw :
-->https://www.youtube.com/watch?v=f9-iYQ25K-g&list=PLJW8noa1vP4ZwjQq9qVwo83p7ZbET1BrD&index=7&t=0s
-->sudo apt-get install ufw
-->sudo ufw enable
-->sudo ufw default deny incoming
-->sudo ufw default allow outgoing
-->sudo ufw status verbose
-->sudo ufw limit/allow http --> https://www.cyberciti.biz/faq/howto-limiting-ssh-connections-with-ufw-on-ubuntu-debian/
-->sudo ufw limit/allow https --> limit rule. Currently only IPv4 is supported. With this syntax you can deny connections from an IP address that has attempted to initiate 6 or more connections in the last 30 seconds
-->sudo ufw limit/allow 1998
-->to delete a rule -->sudo ufw status numbered -->sudo ufw delete 5
-->to reset ufw -->sudo ufw reset

##########You have to set a DOS (Denial Of Service Attack) protection on your open ports of your VM
---->install apache2 :
-->https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-18-04-quickstart
-->sudo apt-get install apache2
-->sudo ufw app list
-->sudo ufw allow 'Apache'
-->sudo service apache2 status
--> from a browser log into 10.12.254.253:80 you have to be redirected to a apache2/ubuntu test page
---->fail2ban :
-->https://blog.rapid7.com/2017/02/13/how-to-protect-ssh-and-apache-using-fail2ban-on-ubuntu-linux/

-->sudo apt-get install fail2ban -->sudo vim /etc/fail2ban/jail.local



-->sudo vim /etc/fail2ban/filter.d/dos-attack-filter.conf

-->this filter is the patern find in /var/log/apache2/access.log (^)first of the line (<HOST>)ip address

-->sudo service fail2ban start

-->sudo fail2ban-client status (you should see the jail that u ve created) -> Jail list: dos-attack, ssh, sshd

-->sudo fail2ban-client status ssh/dos-attack (to see if you are banning someone)

-->sudo fail2ban-client set ssh/dos-attack unbanip 10.12.3.12 (unban ip)

-->sudo fail2ban-client unban 10.12.3.12

##########You have to set a protection against scans on your VM's open ports
---->sudo ufw disable
---->https://wiki.debian-fr.xyz/Portsentry
---->to use iptables -->https://drive.google.com/drive/folders/1jx4ibVKbvi7yqn1OFV524-

cU_RYf6Yob
---->sudo ufw enable
---->add this command to a script and crontab it @reboot sudo sh script.sh
-->sudo ufw disable /> sudo service portsentry restart /> sudo ufw enable

##########Create a script that updates all the sources of package, then your packages and which logs the whole in a file named /var/log/update_script.log
---->echo "sudo apt-get update -y >> /var/log/update_script.log" >> /home/moha/update_script.sh
########## Create a scheduled task for this script once a week at 4AM and every time the machine reboots
--->https://crontab.guru/
###reboot //> crontab -e //>@reboot sudo sh /home/moha/update_script.sh
###weelly //>crontab -e //>0 4 * * SUN sudo sh /home/moha/update_script.sh

##########Web Part SSL///this luck a lot of details count on liens
---->https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-18-04

##########Web Part login page
https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-debian-9
---->install apache2
---->create you new site https://www.youtube.com/watch?v=OWNxUVnY3pg&t=575s -->https://drive.google.com/drive/folders/1jx4ibVKbvi7yqn1OFV524-cU_RYf6Yob
---->add a new directory named /var/www/moha.com and chmod +777 it
+++ sudo chown -R $USER:$USER /var/www/moha.com/
---->copy it to the machine by
---------->scp -P98 /Users/msoulaim/Desktop/www/* moha@10.12.254.253:/var/www/moha.com/
---->sudo nano /etc/apache2/sites-available/moha.com.conf
---->add just the first <VirtualHost *:80> without /Redirect/ line

```
ServerName 10.12.254.100
<VirtualHost *:80>
        ServerAdmin mohamed-soulaimani@outlook.com
        ServerAlias www.moha.com
        DocumentRoot /var/www/moha.com/

        Redirect / https://10.12.254.100/

        Errorlog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

<VirtualHost _default_:443>
                ServerAdmin mohamed-soulaimani@outlook.com
```

```
                DocumentRoot /var/www/moha.com/

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                SSLEngine on

                SSLCertificateFile      /etc/ssl/certs/apache-selfsigned.crt
                SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
</VirtualHost>
~
"/etc/apache2/sites-available/moha.com.conf" 25L, 741C
```

---->sudo a2ensite moha.com.conf //to enable your site
---->sudo a2dissite 000-default.conf //to disable the default (debian site)
---->by this your site is the one is working
---->https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-18-04
---->create a SSL -->`sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt`
---->the second <VirtualHost 433> in the last photo is added in this part
---->`sudo a2enmod ssl` to enable SSL
----> now you have to disable apache to listen to the local host
---->sudo vim /etc/apache2/ports.conf

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 10.12.254.100:80

<IfModule ssl_module>
        Listen 10.12.254.100:443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 10.12.254.100:443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
~
~
~
"/etc/apache2/ports.conf" 15L, 362C
```

---->sudo service apache2 restart

####mondatory part
-->scp -P98 /Users/msoulaim/Desktop/www/* moha@10.12.254.253:/var/www/moha.com/

##UFW Portsentry

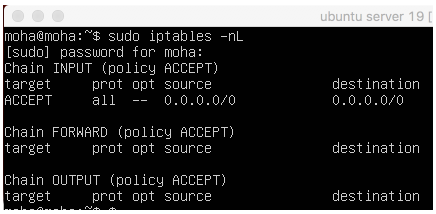
with iptables #thos rules cause problemes sush as cant connect to internet
###iptables -L (list rules) -v (more info)
#######to delete an iptables rule >iptables -S (list all valid command) >sudo iptables -D INPUT -m conntrack --ctstate INVALID -j DROP (delete a specific command)
#######>sudo iptables -L --line-numbers >sudo iptables -D INPUT 3 (delete a specific rule from the list)
>sudo iptables -F ////////Flush (This is equivalent to deleting all the rules one by one.)
>sudo iptables -A INPUT -i lo -p all -j ACCEPT /////**-A** Append one or more rules to the end of the selected chain (chain is a list of rules which can match a set of packets)

/////**filter**: This is the default table. It contains the built-in chains **INPUT** (for packets destined to local sockets)

/////**-i,** Name of an interface via which a packet was received //**lo** is the interface used by -i (loopback) (ip a) for more info

/////**-p,** The protocol of the rule or of the packet to check //**all** here we dont specifie protocol

/////**-j, --jump** This specifies the target of the rule //**Targets** A firewall rule specifies criteria for a packet (*ACCEPT*, *DROP*, *QUEUE*, or *RETURN*)

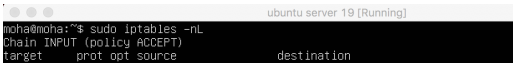


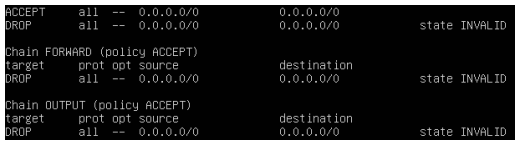

```
# Droping all invalid packets
>iptables -A INPUT -m state --state INVALID -j DROP
>iptables -A FORWARD -m state --state INVALID -j DROP
>iptables -A OUTPUT -m state --state INVALID -j DROP
```

/////**-m** or **--match** options, followed by the matching module name; after these, various extra command line options become available

//3la mafhamt ila dazt had -m katkhalli had l condition kaddoz dima ya3ni makatb9ach tchekiha kolla marra (google traduction)

/////**-m** = means “match” and is used to call Iptable’s extensions like conntrack which is not part of iptable’s core functions
////**state** This module, when combined with connection tracking, allows access to the connection tracking state for this packet. (katb9a metrakia l connection)
////**--state** Where state is a comma separated list of the connection states to match. (katb9a tab3a l connection wach at matcha m3a l condition)
////**Possible states are INVALID** meaning that the packet could not be identified for some reason which includes running out of memory and ICMP errors which don't correspond to any known connection -->https://en.wikipedia.org/wiki/Mangled_packet





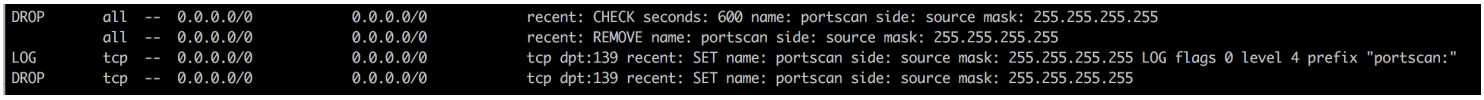

```
# Protecting portscans
# Attacking IP will be locked for 24 hours (3600 x 24 = 86400 Seconds)
iptables -A INPUT -m recent --name portscan --rcheck --seconds 86400 -j DROP
iptables -A FORWARD -m recent --name portscan --rcheck --seconds 86400 -j DROP

# Remove attacking IP after 24 hours
iptables -A INPUT -m recent --name portscan --remove
iptables -A FORWARD -m recent --name portscan --remove

# These rules add scanners to the portscan list, and log the attempt.
iptables -A INPUT -p tcp -m tcp --dport 139 -m recent --name portscan --set -j LOG --log-prefix
"portscan:"
iptables -A INPUT -p tcp -m tcp --dport 139 -m recent --name portscan --set -j DROP

iptables -A FORWARD -p tcp -m tcp --dport 139 -m recent --name portscan --set -j LOG --log-prefix
"portscan:"
iptables -A FORWARD -p tcp -m tcp --dport 139 -m recent --name portscan --set -j DROP
```

////**recent** Allows you to dynamically create a list of IP addresses and then match against that list in a few different ways.
//For example, you can create a 'portscan' list out of people attempting to connect to port 139 on your firewall and then DROP all future packets from them without considering them.
////**--name** Specify the list to use for the commands. 'portscan'
////**--rcheck** Check if the source address of the packet is currently in the list.
////**--seconds** This option must be used in conjunction(match) with one of **--rcheck**. When used, this will narrow(specifie) the match to only happen when the address is in the list and was seen within the last given number of seconds.
////**--remove** Check if the source address of the packet is currently in the list and if so that address will be removed from the list and the rule will return true
////**--set** This will add the source address of the packet to the list. If the source address is already in the list, this will update the existing entry
////**--destination-port**,**--dport**
////**LOG** Turn on kernel logging (تسجيل) of matching packets. with a prefix(اختصار) named "portscan:"
//**--log-prefix "portscan:"**
////**DROP (ترك)**
////**FORWARD** (for packets being routed through the box)




```
# Allow the following ports through from outside
iptables -A INPUT -p tcp -m tcp --dport 25 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
```

```
# Allow ping means ICMP port is open (If you do not want ping replace ACCEPT with REJECT)
iptables -A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT

########this is the block scan command ###Lastly reject All INPUT traffic
iptables -A INPUT -j REJECT
```

####################mail Script
>sudo apt-get install mailutils
>sudo vim crontab_check.sh
|||||||| #!/bin/bash
|||||||| filename="md5_print"
|||||||| if [! -f old_cron]
|||||||| then
|||||||| md5sum /etc/crontab | cut -d ' ' -f 1 > old_cron
|||||||| else
|||||||| old_hash=$(cat old_cron)
|||||||| new_hash=$(md5sum /etc/crontab | cut -d ' ' -f 1)